MEDESS

A METHODOLOGY FOR
DESIGNING
EXPERT SUPPORT SYSTEMS
MEDESS

A METHODOLOGY FOR
DESIGNING
EXPERT SUPPORT SYSTEMS

PROEFSCHRIFT

ter verkrijging van de graad van doctor
aan de Technische Universiteit Delft,
op gezag van de Rector Magnificus,
Prof. drs. P.A. Schenck,
in het openbaar te verdedigen
ten overstaan van een commissie
door het College van Dekanen daartoe aangewezen,
op dinsdag 23 april 1991 te 14.00 uur

door

JIM ADRIAAN VAN WEELDEREN

informatica ingenieur

goingen te Delft
Dit proefschrift is goedgekeurd door de promotor:

Prof. Dr. H.G. Sol

en door de leden van de promotiecommissie:

Prof. Dr. A. Bosman
Prof. Dr. H. Koppelaar
Prof. Dr. Ir. M. Looijen
Prof. Ir. D.H. Wolbers
Dr. Ir. J.H. Blom
To all experts
The study reported in this dissertation was sponsored by KEMA. The screens included in this dissertation are printed with permission of KEMA.

CIP-DATA, KONINKLIJKE BIBLIOTHEEK, DEN HAAG

Weelderen, Jim Adriaan van

MEDESS: a methodology for designing expert support systems /
Jim Adriaan van Weelderen. - Arnhem : KEMA. - Ill.
Thesis Technical University Delft. - With ref.
ISBN 90-353-1027-6
SISO 521 UDC 681.3.02(043.3)
Subject headings: information systems / expert support systems /
design methodologies

Copyright 1991 by J.A. van Weelderen

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording or otherwise, without written permission of the author.
CONTENTS

PREFACE ................................................................. ix

1 BACKGROUND .......................................................... 1
1.1 Expert Support Systems ........................................... 1
1.2 Research questions and research approach ...................... 4

2 THEORIES ON PROBLEM SOLVING .................................... 11
2.1 Problems and problem solving ..................................... 11
2.2 What makes a problem solver an expert? ......................... 16
2.3 Problem solving and errors ...................................... 18
2.4 Conclusions ...................................................... 20

3 ESS DESIGN: CONTEMPORARY APPROACHES ..................... 23
3.1 Framework ....................................................... 23
3.2 Discussion .................................................... 25
3.3 Conclusions .................................................... 36

4 MEDESS ................................................................. 41
4.1 Introduction .................................................... 41
4.2 Way of thinking ................................................. 46
4.3 Way of modelling ............................................... 46
4.4 Way of working ............................................... 50
4.5 Way of control ................................................. 60

5 FIRST CASE: CONFIGURING INSPECTION REGIMES ............ 63
5.1 Introduction .................................................... 63
5.2 Systiological problem ........................................... 64
5.3 Infological problem ............................................. 68
5.4 Datalogical problem ............................................ 73
5.5 Technological problem .......................................... 76
5.6 Discussion ..................................................... 80

6 SECOND CASE: DESIGNING TEST CIRCUITS ..................... 83
6.1 Introduction .................................................... 83
6.2 Systiological problem ........................................... 84
6.3 Infological problem ............................................. 90
6.4 Datalogical problem ............................................ 98
6.5 Technological problem .......................................... 100
6.6 Discussion ..................................................... 104
7 THIRD CASE: PLANNING INSPECTION TRIPS
   7.1 Introduction .................................... 109
   7.2 Systological problem .............................. 110
   7.3 Infological problem ................................ 115
   7.4 Datalogical problem ................................ 122
   7.5 Technological problem ............................. 124
   7.6 Discussion ........................................ 129

8 FOURTH CASE: DIAGNOSING TURBINE CONDITIONS
   8.1 Introduction ..................................... 131
   8.2 Systological problem .............................. 131
   8.3 Infological problem ................................ 135
   8.4 Datalogical problem ................................ 139
   8.5 Technological problem ............................. 141
   8.6 Discussion ........................................ 145

9 EPILOGUE
   9.1 Summary .......................................... 147
   9.2 Conclusions ...................................... 150

INDEX ................................................... 157

SAMENVATTING ......................................... 167

CURRICULUM VITAE .................................... 173
PREFACE

Technology can no longer be considered as a constraining factor in designing effective Expert Support Systems (ESSs). The challenge in mastering the problem of designing an ESS is now to improve our methodological insights. The study reported in this dissertation shows that it is possible to design effective ESSs by fully understanding problem situations before solving them, by considering problem situations not only from the micro-perspective, but also from the meso- and the macro-perspectives and by addressing the systelological and infological design problems as well as the datalogical and technological ones.

This study has been carried out at KEMA, the joint laboratories of the Dutch electricity supply companies. This "knowledge-intensive" organization employs many experts in various engineering disciplines. As an international engineering, certification and research institution, KEMA considers it essential to achieve a high quality level in its services to its clients. It has been recognized that the integration of ESSs into its information systems can be a way to achieve this objective. I am proud to say that KEMA has given me the opportunity to carry out this study and to prove that ESSs can indeed be effective instruments for improving the performance of the experts it employs.

Many persons have contributed in accomplishing this dissertation. First I would like to thank my promotior, Henk Sol. He is a dynamic personality and knows how to create an inspiring tension between science and business. In a constructive atmosphere he has guided me in developing my insights in the sense and nonsense of ESSs. I am indebted to him for putting so much invaluable effort in keeping this study on track and in reading and commenting on various versions of this dissertation. Indeed, I consider it a privilege to have written this dissertation under his supervision.

I would also like to thank Ton Degens. With a positive and enthusiastic attitude to this study he created an atmosphere at KEMA in which I felt free and supported to do all that was necessary to accomplish this dissertation. I am grateful to him for pointing out to me in so many ways the world of difference between "gelijk hebben" and "gelijk krijgen". But most of all I sincerely thank him for the opportunities he has given me to make the past four years some of the most exciting ones in my life.

I thank Wander van der Berg and Anne Haitsma for recommending the students, without whose efforts it would have been impossible to develop and evaluate the ESSs described in this dissertation. In particular I would like to thank Johan Draijer, Gert-Jan Zwijnenberg, Erik Wilbrink, Jos Overbeek, Lex de Wolff, Adri Nijsse, Edwill Jansen, Martin Bloot and Ronald Hetharia who, each of them in his own unique way, have contributed to accomplishing this dissertation within the framework of achieving their M.Sc. degree.
I thank my KEMA colleagues of the department of Electrical Research and Information Technology for their pleasant working environment, in which I enjoyed accomplishing this dissertation. I am indebted to my colleagues Berto Jansen, Martin Kwakkel, Peter van Oirschot, Kim Procé, Ger Roelofs, Henk Roodhardt and Kees van de Water for their daily, stimulating interest. I finally would like to thank Mart Janssen for his patient support in finishing this study and in starting a new form of cooperation.

I thank my Kongsji colleagues at the department of Information Systems in Delft for providing such a stimulating atmosphere, in which it is only natural to talk about writing a dissertation.

Special thanks are also due to the experts who have cooperated in designing the ESSs described in this dissertation. As to Xpection, I would like to thank Ben Kaufman and Jan Klok. Regarding OSS, I am indebted to Piet Knol, Dolf de Loo, Wim van der Linden and Henk te Paske. Concerning IPRESS, my special thanks go to Nico Bos. Finally, I would like to thank Johann Klees, Piet Westra and Raymond Pahladsingh for their cooperation in designing PRESSTO.

I thank Peter van Slingerland, KEMA's English corrector, for his contribution to the improvement of the English text of this dissertation.

In conclusion, I wish to express my love to Els and my parents for giving me nothing but their best. Their affection and support has been invaluable.

Arnhem
January 1991

Jim A. van Weelderen
BACKGROUND

1.1 Expert Support Systems

Without any doubt experts are of great importance to organizations. When it comes to maintaining their competitive position, organizations set themselves to improving the performance of these individuals. The challenge of using computers to master this problem keeps many researchers and practitioners busy. In the past two decades we have witnessed an increasing interest in concepts such as Decision Support System, Knowledge Based System and, more recently, Expert Support System. In this section we will outline the development of these concepts.

In the early 1960s many organizations entered the era of computer-based information systems. Transaction Processing Systems (TPS) formed the backbone of their information systems. A TPS is aimed at gathering, updating and presenting information according to predefined procedures. It achieves a high degree of efficiency in processing large amounts of information. In the late 1960s the concept of a Management Information System (MIS) emerged. A MIS is built upon a TPS and is aimed at aggregating information which is contained in a TPS according to predefined procedures and at reporting such information to managers for review purposes. Today MISs, like TPSs, are incorporated into the information systems of many organizations.

The increasing popularity of timesharing computer systems in the early 1970s stimulated the design of interactive software. Inspired by this technological development, Gorry and Scott Morton (1971) advocated the potential of a new concept called Decision Support System (DSS). In extension to a MIS, a DSS should serve to improve the quality of decisions rather than the efficiency of decision making processes. This philosophy is reflected in their definition of a DSS. They define DSSs as "interactive computer based systems which help decision makers utilize data and models to solve unstructured problems". The terms "unstructured" and "ill-structured" play a central role in most definitions of a DSS. They have their origin in the work of Simon (1960, p. 6). These terms are used to characterize a problem for which "there is no cut-and-dried method for handling the problem because it hasn't arisen before, or because its precise nature and structure are elusive or complex, or because it is so important that it deserves custom-tailored treatment".

Most work on DSSs is concerned with organizational problem solving. As Doukidis (1988, p. 346) puts it: "DSS mainly serve the strategic and management level". Individuals who work at these organizational levels apparently deal mostly with unfamiliar, ill-structured problems.
According to Pfeifer and Lüthi (1985, p. 46) a "DSS must allow the user to confront a problem in a flexible, personal way by providing the ability to manipulate the data and models in a variety of ways while going through the decision making process. The a-priori knowledge about the problem-solving process is sparse - the system merely facilitates the potential usage of methods, models, and data, on an individual basis". This view seems to reflect the general philosophy behind DSSs, i.e. that their users are capable of handling ill-structured problems with a great deal of common sense. In a way they are assumed to be experts. Moreover, this view seems to go hand in hand with personal computing, a concept which came into perspective in the early 1980s.

In this period there also was the introduction of DSS generators. These contain facilities which enable fast handling of data and construction of mainly financial and economic models.

In the 1950s the belief that intelligence is not necessarily a human characteristic stimulated many scientists to explore how computers can be programmed to simulate human cognitive activities. In 1956 McCarthy proposed to call this new scientific discipline Artificial Intelligence (AI). Initially most research in this field was characterized by an intensive search for universal patterns in human thought. However, in the 1970s Feigenbaum (1977, p. 1016) introduced a different concept of an "intelligent" computer application. He argued that "the problem solving power exhibited in an intelligent agent's performance is primarily a consequence of the specialist's knowledge employed by the agent, and only very secondarily related to the generality and power of the inference method employed. Our agents must be knowledge-rich, even if they are methods-poor". Since the late 1970s, the term "Knowledge Based System" (KBS) is used to label computer applications characterized by a strict separation of specific knowledge and generic inference mechanisms. In the early 1980s research into KBSs too was stimulated by the introduction of the concept of personal computing. Along with DSS generators, KBS shells became available. A KBS shell contains generic inference mechanisms and facilities to specify the knowledge that can be used by its inference mechanisms to infer conclusions.

When the specific knowledge of a KBS corresponds with an expert's knowledge, a KBS is in many cases called an Expert System (ES). Waterman (1986, p. 390) defines an ES as "a computer program that uses expert knowledge to attain high levels of performance in a narrow problem area. These programs typically represent knowledge symbolically, examine and explain their reasoning processes, and address problem areas that require years of special training and education for humans to master". In contrast to DSSs, "ES usually support operational activities", see Doukidis (1988, p. 346). This implies that individuals who work at this organizational level are assumed to deal mostly with familiar, structured problems.
Lately efforts have been directed towards integrating the concepts of DSS and KBS into the concept of an "intelligent" DSS. Doukidis (1988, p. 345) notes that "the strong shift in orientation of thinking and practice towards knowledge based DSS is becoming more and more apparent".

Turban and Watkins (1986, p. 124) argue: "While typical DSS support quantitative, mathematical, and computational reasoning, DSS should also be developed to support qualitative analysis based on methodologies such as analogical reasoning, pattern recognition, and content analysis. ES are particularly well suited for these types of methodologies, and thus may provide an important link to the DSS for providing more balance in supporting various types of decision processes". They elaborate upon two ways of integration, i.e. "ES integration into DSS components" and "ES as a separate component in the DSS" and argue that the former will be prevalent. In contrast, Goul et al. (1985, p. 446) work out the second possibility. They argue that "the inclusion of an 'expert component' in a decision support system provides the potential for novel approaches to offering computer-based support to decision makers faced with unstructured problems". They demonstrate an "expert subsystem" which applies "the concept of decision channeling to the intelligence phase of decision making ... by leading the decision maker to pertinent models of reasoning, offering theoretical conclusions, and providing supporting justification for those conclusions. Data gathering can be improved as the decision maker's scope is narrowed or widened as necessary, dictated by the expert subsystem's requests for information".

The concept of an "intelligent" DSS as described by Turban and Watkins (1986) bears a great resemblance to the concept of an Expert Support System (ESS) as put forward by Luconi et al. (1984, p. 1). They argue that "the knowledge that can be feasibly encoded in an ES is not sufficient to make decisions by itself". They propose a shift of focus from designing ESs to designing "ESS that will aid, rather than replace, human decision makers". Since the mid 1980s many researchers and practitioners have picked up the concept of an ESS.

Cook (1986, p. 273) demonstrates an ESS which is designed to "help contracting and program managers identify and evaluate the relevant criteria, subcriteria, and alternatives to be considered in formulating a competition strategy, and to enable these professionals to integrate their experience, judgement, and the facts that are available into a coherent, comprehensive competition strategy".

Zeidner et al. (1986, p. 30) make a distinction between "ESS that are aimed at making the expert more productive by providing him with an environment consisting of an integrated set of tools" and "ES that are aimed at replacing the engineering expert with software that emulates his current behavior and rationale". They demonstrate an ESS which can support software engineers in building ESSs.
Subramaniam (1987, p. 446) argues that "we should focus our attention on designing systems that support expert users rather than replacing them". He also explains how: "ESS help people to solve a much wider class of problems (than ESs do, JvW). This is done by pairing the human with the expert system in such a way that the expert system provides some of the knowledge and reasoning steps, while the human provides over-all problem solving direction as well as specific knowledge not incorporated in the system. Much of this knowledge may be imprecise and will remain below the level of consciousness, to be recalled to the conscious level of the decision maker only when it is triggered by the evolving problem context".

Farah (1988, p. 75) discusses requirements for an ESS that can support an information systems designer in constructing and manipulating various models of information systems. He notes that "the term ESS is used to describe an ES that is used to supplement the knowledge and expertise of the information system's designer rather than replace him".

In brief, recent developments show a trend towards integrating the concepts of a DSS and of a KBS into the concept of an "intelligent" DSS, also called ESS. These developments seem to be guided by a commonly shared view that computers should support an expert in solving problems rather than replace him.

1.2 Research questions and research approach

Technology can no longer be considered as a constraining factor when it comes to designing effective ESSs. On the contrary, technological developments stimulate our imagination in working out even more sophisticated ways to support human problem solving. Still, the mission is and will always be to improve the performance of experts by making appropriate use of Information Technology (IT). According to Sol (1990) concepts such as DSS, KBS and ESS can be categorized under the concept of an "Information System to Support Decision Processes" (ISDP).

Information systems based primarily on TPSs and MISs, are no longer considered sufficient to provide an organization with a competitive advantage. Without any doubt the ESS concept offers a starting point to develop computer applications which can be used to improve the performance of experts and thus to maintain or increase the success of an organization in realizing its objectives. Many organizations are becoming interested in integrating ESSs into their information systems. However, one can question whether their expectations are met in practice. Some authors report successful implementations of the concept. However, their optimism should be set against a number of conflicting findings.
According to Elam (1989, p. 1) "much of the DSS work has been driven by an almost religious belief that the decision making situation would be improved with the introduction of computer-based systems. Some studies have demonstrated that a DSS leads to increased decision quality; others have shown just the opposite".

Benbasat and Todd (1989, p. 17) are very resolute: "The notion that Decision Support Systems (DSS) would improve decision quality has been around since the earliest days of research in the field. While this notion has obvious intuitive appeal, demonstrating the value of DSS in empirical work has been difficult. This gap between normative expectations and empirical findings is puzzling. ... DSS use does not result in improved decision quality because decision makers are not interested in decision quality only ..., because in some cases they have no control over the decision process ..., or because their motivation to perform is limited due to the absence of an obvious relationship between system use and some desirable outcome".

Søl (1990, p. 117) draws a similar conclusion: "Although DSS enjoy an enormous popularity, the results have so far not lived up to expectation".

In a survey, d'Agapeyeff and Hawkins (1987) found that "many large organizations were experimenting with expert systems but that operational applications are still limited and narrowly held".

There is consensus among these authors with regard to the promising potential of the ESS concept. However, they also share the opinion that the design of ESSs is not being put into practice effectively.

Breuker and Wielinga (1986, p. 5.2) note that "the fact that few systems ever reach an operational stage can be seen as an indication that knowledge engineering still is an art rather than a discipline".

Elam (1989, p. 5) gives a possible explanation for similar findings: "Still, after 20 years of research, we can only conclude that we still do not have robust design principles for constructing and evaluating decision support systems".

The same line of thought is put forward by Woods (1987, p. 572). He remarks that "while our ability to build more powerful machine cognitive systems has grown and promulgated rapidly, our ability to understand how to use these capabilities has not kept pace".

Søl (1990, p. 117) is more specific when he remarks that "many applications have been set up to evaluate alternatives without proper problem and remedy definitions. Other applications assume that an increase in the information quantity and quality automatically results in better decisions. This is not necessarily true: the availability of large amounts of data does not guarantee effective decision making". He suggests (p. 120): "To overcome these problems, we clearly need new ways of thinking and new ways of modeling for the development of information systems in general, and for support of decision processes in particular".
These authors suggest that ESS design should be taken up in another way than has been done so far.

In the field of information systems many authors use the term "methodology" to label a way of designing an information system. In accordance with Sol (1987, p. 90), we define a methodology as a body of methods and techniques which can be followed and applied to solve instances of a class of practical problems. A method can be considered as a procedure to be followed and in which techniques can be applied to produce a deliverable. A technique can be considered as an instrument to produce a deliverable. We emphasize that we do not use the term "methodology" in the meaning of the science of methods.

ESS design cannot simply be taken up in a different way than has been done so far. We first need to determine how contemporary ESS design methodologies support ESS design. Therefore, our first research question can be formulated as:

1. How do contemporary ESS design methodologies support ESS design?

Considering the answer to this question as a point of reference, we can identify, work out and test new ways of effectively designing ESSs. We can formulate our second research question as:

2. How can contemporary ESS design methodologies be improved?

In this study the focus is on the analysis and design phases defined in the traditional development life cycle of an information system. No attention is paid to the realization and implementation phases.

We conclude this chapter with a discussion of our research approach. Problem, problem solving and expert are important concepts in this study. Not only is the way how an expert solves a problem of interest, but also the way how an ESS designer can solve the problem of designing an ESS. In chapter 2, we shall determine what a problem is, how humans solve problems and what makes a problem solver an expert.

This study is carried out as follows, see figure 1.1. First we determine how contemporary ESS design methodologies support ESS design. For this purpose we set up a framework to describe an ESS design methodology. This framework is derived from the literature on information systems. In addition we select a sample consisting of ten contemporary, methodological contributions to ESS design. For each example we determine how it fits into the framework. We identify which issues in the framework receive relatively little attention and suggest possible directions for research. The results are reported in chapter 3.
Figure 1.1 Research approach.
With these results in mind, we set up a specific ESS design methodology which incorporates one or more possible improvements as proposed in chapter 3. The methodology is also based on new insights gained from the literature on information systems, the ESS literature and other literature related to e.g. operations research, management science and cognitive psychology. This methodology is termed "MEDESS", an acronym of "Methodology for Designing Expert Support Systems". Its setup is discussed in chapter 4.

To test MEDESS we apply it to four situations in which an organization wishes to solve a problem concerning the performance of an expert and in which that organization is interested in providing the expert involved with an ESS. These cases are carried out at KEMA, the joint laboratories of the Dutch electricity supply companies.

Each case is reported in a separate chapter, i.e. chapters 5, 6, 7 and 8. The first case concerns a problem situation in which two experts design regimes for the future inspection of steam headers and steam pipes used in fossil fuel power plants. The second case is concerned with four experts who design short-current circuits for testing circuit breakers. The third case is focused upon an expert whose job it is to plan trips for the inspection of electrical products and their production processes in factories all over the European continent. The fourth and final case addresses the work of an expert who is responsible for diagnosing the condition of the turbine of a Dutch nuclear power plant.

In the final chapter we evaluate whether the improvements proposed in chapter 3 and incorporated in MEDESS actually hold in the cases studied. The conclusions are based on the extent to which in each of these cases the problems indicated are solved.

References in chapter 1


2

THEORIES ON PROBLEM SOLVING

2.1 Problems and problem solving

In the previous chapter we used terms such as "problem", "problem solving" and "expert" without explaining what we mean by these terms. It is essential to acquire an understanding of the concepts behind these terms because we are interested in the way how an expert solves a problem as well as in the way how an ESS designer can solve the problem of designing an ESS. In this chapter we will discuss what a problem is, how people solve problems and what makes a problem solver an expert. We will point out that an individual's performance is affected e.g. by his limited cognitive abilities. In this respect we will relate the concept of error to human problem solving.

Many descriptions of the concept problem can be found in the literature. According to Monhemius (1984, p. 9) an individual has a problem if he finds himself in a situation in which he experiences a discrepancy between his notion of the desired reality and his perception of the reality which he wishes to eliminate. His problem is solved when he finds that he no longer experiences the discrepancy. In extension to Monhemius, who emphasizes the recognition of a problem, Ackoff (1981, p. 20) addresses the elimination of a problem: "by a problem we mean a situation that satisfies three conditions: First, a decision-making individual or group has alternative courses of action available; second, the choice made can have a significant effect; and third, the decision maker has some doubt as to which alternative should be selected". In this dissertation we shall adopt Monhemius' definition of a problem.

Many views on problem solving can be found in the literature as well. To stay with Monhemius (1984, p. 13), problem solvers perceive different realities than do problem owners. He identifies two major phases within a process of problem handling. In the diagnostic phase a problem solver classifies a problem as a problem of perception, a problem of notion or a problem of reality. In the solving phase he modifies the problem owner's perception of reality, the problem owner's notion of a desired reality or reality itself until the problem owner no longer experiences a discrepancy between his notion of the desired reality and his perception of reality.

According to Simon (1977, p. 40) "decision making comprises four principle phases: finding occasions for making a decision, finding possible courses of action, choosing among courses of action, and evaluating past choices". These phases are named "intelligence", "design", "choice" and "review". He seems to place an emphasis on "courses of action" rather than on actual situations and new, improved, situations.
However, more recently Simon et al. (1987, p. 11) have made a distinction between what they call a "decision" and a "problem": "If the set of alternatives has been identified, requiring evaluation and choice only, we would like to speak of a decision. But when intelligence and/or design activities are still required we would rather speak of a problem".

Describing problem situations accurately is not an easy task for humans. Reality often is too complex to be dealt with directly. Consequently, people tend to reduce the reality they perceive to a level of simplicity that is easy to deal with.

In the fields of management science and of operations research, the complex activity of problem solving is often modelled according to the rational actor model. What is meant by rational is explained by Sage (1981, p. 65): "The idea of rationality originated in the economics literature where micro-economic models of the consumer and the firm assumed complete information and rationality. The rational person is assumed to have identified a set of well-defined objectives and goals and is assumed to be able to express preferences between different states of affairs according to the degree of satisfaction of attaining these objectives and goals. A rational person has identified available alternative courses of action and the possible consequences of each alternative. The rational person makes a consistent choice of alternative actions in order to maximize the expected degree of satisfaction associated with attaining identified objectives and goals". Simon (1973) put forward that humans do not solve problems according to the rational actor model. He introduced the concept of "bounded rationality". It suggests that problem solvers compensate for their limited cognitive abilities by constructing a simplified representation of a problem and behaving rationally within the constraints imposed by this model.

Forced by their limited cognitive abilities, human beings handle complexity by abstracting reality into simplified representations, i.e. models. In a number of contributions problem solving is considered as a process of model construction. Mitroff et al. (1974, p. 47) introduce a "simple whole systems view of the activity of problem solving", see figure 2.1. We quote: "Suppose that every scientific inquiry starts with ... the existence of a problem situation ... The arrow or path from the circle labeled "Conceptual Model" is meant to indicate that the "first phase" of problem solving consists in formulating a conceptual model of the problem situation. The conceptual model sets out in broad terms the definition of the particular problem that will be solved; it specifies the field variables that will be used to define the nature of the problem and the level to which the variables will be treated, for example, whether from a micro or macro point of view. Once a conceptual model of the problem has been formed, a scientific or formal model can be formed (the term 'scientific model' is used to label a model which often is referred to as 'empirical model', JvW), and from this a solution, if one is possible, can be derived. If the solution is then "fed back" to the problem for the purpose of taking action
on it, we have the case of implementation. In this way, any one of a number of iterations can be performed on the problem. ..., the arrow or path from "Reality" to the "Scientific Model" corresponds to the "degree of correspondence" between "reality" and the model in the traditional meaning of the notion of correspondence". The verticle arrow in the middle of the diagram is intended to illustrate their point that science is losing its grip on reality and is becoming a goal in itself: "The vertical path between the "Conceptual Model" and the "Solution", we have labeled "feedback in the narrow sense" ... this cycle (the loop from II to III to IV and back to II again, JvW) was all there was to problem solving - that the goal of problem solving is merely to iterate on deriving better scientific solutions to bigger and better scientific models. This is what we mean by "feedback in the narrow sense". These forms of activity never get back to "reality" to question their initial starting assumptions".

![Diagram](image)

**Figure 2.1** Problem solving according to Mitroff et al. (1974).
Determining what the problem actually is in a particular situation is considered an essential activity in a process of problem solving. As Ackoff (1974, p. 8) puts it: "We fail more often because we solve the wrong problem than because we get the wrong solution to the right problem".

This vision can also be identified in the contribution of Mitroff et al. (1974). They make an explicit distinction between an activity of describing a problem and an activity of solving a problem. In addition, an activity of conceptualizing a problem is introduced. Mitroff and Featheringham (1974, p. 383) consider this activity crucial in a process of problem solving. They show how a problem's nature changes by applying different inquiring systems. They argue that the choice of the right inquiring system is essential when it comes to solving a problem. Their work is based on the premise that "the initial representation or conceptualization of a problem is so crucial to its subsequent treatment ... that the most important as well as most difficult issue underlying the subject of problem solving is precisely "the problem of how to represent problems".".

In our view, a shortcoming in the contribution of Mitroff et al. (1974) is that it does not address the introduction of new concepts during the construction of a solution, thus ignoring the creativity of a problem solver. This shortcoming may be eliminated by using the context provided by Bosman and Sol (1985). They make a distinction on the one hand between conceptual and empirical models and on the other hand between descriptive and prescriptive models. They explain (p. 82) that conceptual models "define a problem using a context in the form of a paradigm or world-view and a construct paradigm for the specification. In this specification variables, parameters and relationships between these variables are used. Conceptual models are data-void, they do not directly refer to a problem in reality. If a problem in reality has to be solved, the conceptual model has to be transformed into an empirical model. This model or models define reality through: a specification of the relevant variables, a specification of the relationships between the relevant variables and a numerical specification of the parameters". A model can be used in a descriptive or prescriptive way. In the former case a model represents something as it is. In the latter case a model represents something as it should be. Bosman and Sol integrate the two distinctions and define four kinds of models which can be constructed in a process of problem solving, i.e. descriptive conceptual, descriptive empirical, prescriptive conceptual and prescriptive empirical models.

By placing the activities which we identified in the contribution of Mitroff et al. (1974) in the context provided by Bosman and Sol (1985), we can construct a scheme that can be used to describe how human beings solve problems, see figure 2.2.
Figure 2.2 A scheme to describe human problem solving.

The scheme starts off with the existence of a problem situation. A problem owner experiences a discrepancy between his perception of reality and his notion of desired reality and wishes to eliminate this discrepancy.

The problem solver first sets himself to finding the problem of the problem owner. He then enters the descriptive modelling phase, in which he creates an understanding of the problem situation. This is an iterative process. The problem solver observes the problem situation. He defines artefacts which, in his opinion, are relevant to the description of the problem situation. He may represent these artefacts in terms of object types. Subsequently, he specifies these object types into object instances which he identifies in the problem situation. It is possible that he identifies unfamiliar object instances which can be relevant to the specification of the problem situation, but were unforeseen initially. The problem solver classifies them into object types. The phase of descriptive modelling is finished when the empirical model corresponds with the problem situation.

The problem solver then proceeds with finding a solution to the problem of the problem owner. He enters the phase of prescriptive modelling, in which he designs a picture of a new situation. Like descriptive modelling, prescriptive modelling is an iterative process. It involves carrying out operations upon the descriptive conceptual model. An operation can be e.g. the incorporation of a new object type. Once the problem solver has constructed a prescriptive conceptual model, he specifies it into a
prescriptive empirical model. It is possible that he thereby identifies unfamiliar object instances. They can be relevant to the specification of a new situation, but were unforeseen initially. The problem solver classifies them into object types. The phase of prescriptive modelling is finished when the evaluation of the differences between the descriptive empirical model and the prescriptive empirical model demonstrates that the implementation of the prescriptive empirical model yields the elimination of the discrepancy experienced by the problem owner.

The problem solver then enters the phase of implementation. He implements the prescriptive empirical model by creating or selecting and carrying out one or more operations upon the problem situation until the new situation corresponds with the prescriptive empirical model, i.e. until the problem owner no longer experiences the discrepancy specified.

2.2 What makes a problem solver an expert?

So far we have described what a problem is and how it is solved by a human being. Many authors distinguish between "structured", "semi-structured" and "ill-structured" problems. However, the attribute "structuredness" can cause confusion. It applies, at a particular moment in time, to the degree of novelty of a problem to a problem solver in the sense that he does or does not know how to solve it. This difference has much to do with the fact whether a problem solver is an expert or a novice. As we are interested in using Information Technology (IT) to support an expert in solving a problem, it may be fruitful to gain a better understanding of how experts specifically solve problems.

The term "expert" stems from the Latin word "expertus". It can be translated as "he who has experienced", suggesting that an individual who has solved many instances of a particular class of problems is considered for this problem class to be an expert. How an expert solves a problem is determined by his great problem solving experience.

According to Waern (1989, p. 64) "anyone skilled in the performance of a particular cognitive task will already have acquired the ability to find the correct representation of the task and to direct an efficient search in the problem space".

Woods (1986, p. 165) puts it as follows: "One characteristic of human expertise is the ability to quickly recognize what is potentially important or promising and to progressively focus in onto the critical data for the current context. This ability is related to the fact that human experts are able to see problems in terms of meaningful high level structures rather than individual data elements".
Simon (1973, p. 83) draws a similar conclusion: "On the basis of the research on chessplayers, what appears to distinguish an expert from a novice is not only that the former has a great quantity and variety of information, but that his perceptual experience enables him to detect familiar patterns in the situations that confront him, and by recognizing these patterns, to retrieve speedily a considerable amount of relevant information from long-term memory. It is this perceptual experience that permits the chessmaster to play, and usually win, many simultaneous games against weaker opponents, taking only a few seconds for each move".

When we apply these findings to the scheme of problem solving which we adopted in the previous section, it can be argued that an expert has at his disposal a conceptual model of both an actual and a new, associated, situation and that he uses these models to specify an empirical model of both an actual and a new situation.

Now that we have an indication of what makes a problem solver an expert, we can direct our attention towards finding out how a problem solver becomes one. Kolb (1984) and Rasmussen (1982) both express a view which can be useful in answering this question. We will conclude this section with a discussion of Kolb’s contribution. Rasmussen’s contribution is discussed in the next section.

The concept of experiential learning is applicable to the model of problem solving which we adopted in the previous section. The processes of describing actual situations and designing new, improved ones both correspond with a process of experiential learning as put forward by e.g. Kolb (1984). In his work learning is conceived as a cyclic process consisting of four learning modes, i.e. "concrete experience", "reflective observation", "abstract conceptualization" and "active experimentation", see figure 2.3. Kolb explains (p. 40): "In this model, concrete experience-abstract conceptualization and active experimentation-reflective observation are two distinct dimensions, each representing two dialectically opposed adaptive orientations. ... Knowledge results from the combination of grasping experience and transforming it".

Kolb distinguishes between two opposed ways of experiencing, i.e. "through reliance on conceptual interpretation and symbolic representation" called "comprehension" and "through reliance on the tangible, felt qualities of immediate experience" called "apprehension". He also distinguishes between two opposed ways of transforming experience into knowledge, i.e. "through internal reflection" called "intension" and "through active external manipulation of the world" called "extension". Experience can be transformed into four types of knowledge. "Divergent knowledge" results from transforming apprehended experience through intension. "Assimilative knowledge" results from transforming comprehended experience through intension. "Convergent knowledge" results from transforming comprehended experience through extension. Finally, "accommodative knowledge" results from transforming apprehended experience through extension.
We conclude that an individual’s learning style changes with the number of instances of a particular class of problems he solves. As to an expert, experience is no longer grasped through comprehension but rather through apprehension and no longer transformed into knowledge via intension but rather via extension.

2.3 Problem solving and errors

We will finish this chapter with a discussion of a concept that seems to affect the way humans solve problems, i.e. the concept of error. People tend to generate errors when solving problems. According to Rouse (1985, p. 620) errors are generated "in the sense that one or more humans' actions were inappropriate for the context and situation in which they occurred". Heslinga (1988, p. 18) extends this definition: "A human error is the non-performance or incorrect performance of a desired activity, provided that adequate conditions for correct performance are present".
Reason (1986) distinguishes between three types of errors, i.e. "slips", "lapses" and "mistakes". Slips are unconscious errors in executing correct intentions. Lapses occur when inferences that have proven valid in the past, yield wrong outcomes in the present situation. Mistakes are errors in the sense that inappropriate intentions are formed. In the latter case a problem solver's conceptual model of a problem is incorrect.

A human being not only tends to generate errors when solving a problem. He also tends to "monitor" this process. Monitoring a process of problem solving is a specific form of coordination. It involves looking for, identifying and recovering generated errors. As Reason (1986, p. 258) explains: "Routine action ... comprises segments of preprogrammed behavioural sequences interspersed with attentional checks upon progress, carried out either consciously or preconsciously. These checks involve bringing the higher levels of the cognitive system momentarily into the control loop and seek to establish (a) whether the actions are running according to plan, and (b) at a more complex level, whether the plan is still adequate to achieve the goal".

Rasmussen's (1982, p. 316) view on human error mechanisms distinguishes between three levels of problem solving behaviour. He explains: "In the skill based domain, including automated, more or less subconscious routines, performance is controlled by stored patterns of behavior in a time-space domain. Errors are related to variability of force, space or time coordination". Thus, we can state that errors generated on the skill-based level can be classified as slips. Instances of this type of error are most likely to be generated by experts, not by novices.

The rule-based level "includes performance in familiar situations controlled by stored rules for coordination of subroutines, and errors are typically related to mechanisms like wrong classification or recognition of situations, erroneous associations to tasks, or to memory slips in recall of procedures. Since rule-based behavior is used to control skill-based subroutines, the error mechanisms related to skill-based routines are always active ..., and the immediate criteria for errors deal with whether the relevant rules are recalled and followed correctly or not". Thus, we can say that errors generated on the rule-based level can be classified as lapses. Instances of this type of error are generated by experts and novices alike.

The knowledge-based level "is called upon in unique, unfamiliar situations for which actions must be planned from an analysis and decision based on knowledge of the functional, physical properties of the system and of the various goals. ..., errors in this domain can only be defined in relation to the goal of the task and generic error mechanisms can only be defined from very detailed studies based on verbal protocols which can supply data on the actual data process". Thus, we can state that errors generated on the knowledge-based level can be classified as mistakes. Instances of this type of error are most likely to be generated by novices, not by experts.
The latter level of problem solving behaviour corresponds with Waern's (1989, p. 44) view on problem solving: "Problem solving occurs when there is a goal to be reached, when the method for reaching the goal is not yet known, and when attempts to reach the goal are being made. This definition distinguishes problem solving from routine performance (when the method is known), from free association (when there is no explicit goal) and from wishful thinking (which implies no concrete efforts to achieve a goal)."

2.4 Conclusions

In this study concepts such as problem, problem solving and expert play an important role. Both experts and ESS designers solve problems. We discussed what a problem is and how to describe the way people solve problems. Using the views of Kolb and Rasmussen, we determined what makes a problem solver an expert. It can be concluded that errors are generated due to selective processing of information. Attention is paid to the wrong features. The right features are overlooked.

An expert's performance is determined by his problem solving behaviour which is shaped by his limited cognitive abilities and by his experience in solving instances of a particular class of problems. It is also determined by the way he coordinates his problem solving processes. These observations can provide guidance in determining how experts solve problems and how their performance can be improved by making appropriate use of IT.

References in chapter 2


3

ESS DESIGN: CONTEMPORARY APPROACHES

3.1 Framework

In the previous chapter we put forward what a problem is, how people solve problems and what makes a problem solver an expert. These findings form an essential basis, for instance to describe how an ESS can be designed.

In chapter 1 we recorded a number of statements telling us that the design of ESSs is not being put into practice effectively and that it should be taken up in another way than has been done so far. These statements, however, are quite global and do not address in some detail how an ESS should be designed. In this chapter our objective therefore is to describe how contemporary ESS design methodologies support ESS design. We first need to introduce a number of concepts in terms of which we can describe an ESS design methodology. For this purpose we introduce a framework derived from Sol (1990). He distinguishes between a way of thinking, a way of modelling, a way of working and a way of control.

Way of thinking

The way of thinking underlying a methodology covers amongst others the perspectives from which a problem situation is considered. Sol (1990) distinguishes between three perspectives i.e. the micro-, the meso- and the macro-perspective.

From the micro-perspective, the objective is to improve the performance of an expert. Attention is paid to identifying which problems an expert solves to fulfil his function in an organization, to identifying how he solves these problems and to designing IT applications that can support him in solving these problems.

From the meso-perspective, improving the performance of an organization is focal. Attention is paid to identifying within an organization how problem solving processes, which can be identified from the micro-perspective, are coordinated and to designing IT applications that can support the coordination of these processes.

From the macro-perspective, the objective is to improve the performance of two or more cooperating organizations. Attention is paid to identifying how the cooperation between two or more organizations is coordinated and to designing IT applications which can support the coordination of such cooperation.

Although it may be expected that a methodology explicates the perspectives it addresses, Seligmann et al. (1988) note that *the philosophy behind a methodology is often overshadowed by or implicit in the techniques and methods embedded in that methodology.*
Way of modelling

The problem an ESS designer faces, i.e. improving the performance of an expert by making appropriate use of IT, can be considered a complex problem. Dividing it into sub-problems can help to reduce its complexity. This is to say that one can apply a specific way of modelling. Like Welke (1977, p. 150) and Sol (1982, p. 9) we shall adopt a subdivision into the systological, infological, datalogical and technological problems.

It should be noted that our explanation of these problems differs from the ones given by Welke and Sol. The systological problem is addressed when a designer considers e.g. the problems an expert solves and an expert’s thereby achieved performance. The infological problem is addressed when a designer considers which information is processed by an expert to solve a problem. The datalogical problem is addressed when a designer considers how information is processed and grouped, without taking into account which technology is used to achieve this. Finally, the technological problem is addressed when a designer considers how information is processed, thereby taking into account which technology is applied to achieve this.

Way of working

In a way of working, Sol and IJpelaar (1983) distinguish between four types of phasing, i.e. a direct acceptance of requirements and a linear, an iterative or an incremental phasing. Each type of phasing represents an approach which a designer can follow to acquire a complete and accurate specification of an intended ESS.

A way of working also covers the methods followed and the techniques applied to construct and represent models. Many authors use the term "prototyping" to indicate how they conduct an ESS design process. Protoyping is often associated to an incremental phasing in which a designer follows certain methods and applies certain techniques to construct various "working" models of (parts of) an ESS.

Way of control

A way of control addresses the balance between the efficiency and the effectiveness of an ESS design process. According to Sol (1989) "the control of a project requires consideration of time, money and manpower in order to produce a good quality product. The organization of the project and the information concerning the progression of the project are fundamental". A way of control and a way of working interact mutually. As Dennis et al. (1987, p. 31) put it: "The problem in developing relatively large, complex information systems is the need to incorporate flexibility in information requirements determination while at the same time maintaining control of the overall development process to ensure that the system is developed efficiently".

24
Summarizing, in contemporary, methodological contributions on ESS design we try to identify a way of thinking, a way of modelling, a way of working and a way of control. As to a way of thinking, we are particularly interested in determining whether the micro-, meso- and macro-perspectives are addressed. Regarding a way of modelling, we intend to find out whether a designer is supported in modelling the systiological, infological, datalogical and technological problems. As to a way of working, we look for phasings, methods and techniques. Finally, regarding a way of control, we wish to know whether a designer is supported in managing a project.

3.2 Discussion

In this section we will use our framework to discuss a sample consisting of ten contemporary, methodological contributions to ESS design. The first five contributions are derived from the DSS literature. The other five contributions are derived from the KBS literature. We note that we have found no contributions which explicitly focus on ESS design. However, there is evidence indicating that ESSs are designed in the same way as DSSs and KBSs. Turban and Watkins (1986, p. 131) argue that "several approaches to the design process for DSS can be borrowed from the construction of ES and vice versa".

DSS

The first contribution that we take into consideration is "A Framework for the Development of Decision Support Systems". It was reported by Sprague (1986).

We start by shaping a picture of Sprague's way of thinking. He states that "the ultimate objective must be viewed in terms of the ability of information systems to support the improved performance of people in organizations". We can conclude that he considers a problem situation from the micro- and the meso-perspective when he argues that "the ultimate objective" can be achieved by providing an expert with a DSS that supports "decisions which are interdependent as well as those that are independent". In the former case an expert "makes part of a decision which is passed on to someone else". In the latter case he "has full responsibility and authority to make a complete and implementable decision". In addition, like Gorry and Scott Morton, Sprague defines DSSs as "interactive computer based systems, which help decision makers utilize data and models to solve unstructured problems". In a DSS-generator, Sprague identifies a user-interface, a database management system and a model management system. In line with the paradigm of bounded rationality, Sprague notes that DSSs should support its user(s) in solving instances of unfamiliar problem classes in such a way that "all phases of the decision making process" are addressed.
DISCUSSION

Sprague puts forward that it is not appropriate to incorporate a linear or iterative phasing in the design process of a DSS as a DSS should support experts in solving instances of unfamiliar problem classes. He argues that "the traditional approaches for analysis and design have proven inadequate because of the rapidity of change in the conditions which decision makers face. Designers literally 'cannot get to first base' because no one can define in advance what the functional requirements of the system should be". Therefore, he suggests an incremental phasing: "The most important four steps in the typical systems development process - analysis, design, construction, implementation - are combined into a single step which is iteratively repeated ... with each cycle, capabilities are added to, or deleted from, the Specific DSS from those available in the DSS Generator". According to Sprague, a DSS design process starts off when "the manager and builder agree on a small but significant subproblem, then design and develop an initial system to support the decision making which it requires". It is noted here that "the manager is actually the iterative designer of the system; the systems analyst is merely the catalyst between the manager and the system, implementing the required changes and modifications". From this point of view, it looks as if an ESS designer is placed into a passive role, accepting and working out the wishes of an expert.

The second contribution considered in this discussion is "An Approach for Designing Decision Support Systems". It was reported by Carlson (1983).

As to what we may consider a way of thinking, Carlson suggests to improve an expert's performance by enhancing his cognitive abilities. He identifies "displaced cost" and "added value" as the potential benefits of a DSS. Displaced cost "results from reduced costs for data gathering, computation, and data presentation in support of decision making". Added value results from a "change in the decision-making process, such as considering more alternatives". Carlson too takes the position that a DSS should support its user in solving instances of unfamiliar problem classes and suggests that it should support the phases Intelligence, Design and Choice. A DSS should provide "1. representations as the context for system use; 2. operations on the representations to support Intelligence, Design, and Choice activities; 3. a variety of memory aids to support use of the representations and operations; and 4. aids for controlling the representations, operations, and memories". We can conclude that Carlson too adopts the definition of a DSS as given by Gorry and Scott Morton and that he considers a problem situation from the micro-perspective.
Regarding a possible way of modelling, Carlson seems to conceptualize the infological problem. In the Intelligence phase, he distinguishes between the following types of operations: "gather data, identify objects, diagnose problem, validate data and structure problem". In the Design phase, he distinguishes between "gather data, manipulate data, quantify objectives, generate reports, generate alternatives and assign risks or values to alternatives". In the Choice phase, he distinguishes between "generate statistics on alternatives, simulate results of alternatives, explain alternatives, choose among alternatives and explain choice". In addition, Carlson seems to conceptualize the datalogical problem. He distinguishes between several types of memory aids: "1. A database ..., 2. views (aggregations and subsets) of the extracted database, 3. workspaces for displaying the representations and for preserving intermediate results as they are produced by the operations, 4. libraries for saving workspace contents for later use, ..., 6. triggers to remind a decision maker that certain operations may need to be performed".

The third contribution considered in this discussion is "Meta-Design Considerations in Building DSS". It was reported by Moore and Chang (1983).

The function which Moore and Chang assign to a DSS is to support its user in solving instances of unfamiliar classes of problems, a way of thinking which we also identified in the contributions of Sprague and Carlson. They define a DSS as "an extensible system with intrinsic capability to support ad hoc data extraction, analysis, consolidation, and reduction, as well as decision-modelling activities". Like Carlson, they too seem to consider a problem situation from the micro-perspective.

In this contribution we can also identify that one should incorporate an incremental phasing in the design process of a DSS: "Although the systems development life cycle approach, with its iterative modification, has proven to be the single most useful approach to developing transaction systems and most MIS-oriented systems, we hold that its focus and purpose are inadequate for the task of building and implementing successful DSS". They come up with four design considerations: (1) "system/problem migration", a concept that "highlights the difficulty of predefining system specifications when learning and DSS development occur simultaneously", (2) "evolving system capabilities", i.e. "easy definition of new operators, data constructs, or reports by the user without redesign or even reprogramming", (3) "soft" versus "hard" capabilities", which involves "the development of a core of generalized, but possibly inefficient capabilities in early versions of the system, which can subsequently be specialized into "hardened", easy-to-use, efficient system operators" to enable the decision maker to reveal "individual preferences for aids through use of the system", and (4) "weak" versus "strong" design", i.e. little or much interference in how decisions are made.
The fourth contribution considered in this discussion is "A Framework for Systems Analysis for Decision Support Systems". It was reported by Bahl and Hunt (1984).

Bahl and Hunt were triggered to write this contribution by their observation that "little attention has been paid to the systems analysis phase, even though it has been recognized as a pivotal part of the DSS development process". As in all contributions discussed so far, they depart from the assumption that a DSS should support its users in solving instances of unfamiliar problem classes. Problem situations seem to be considered from the micro- and the meso-perspective. This conclusion can be drawn when considering what we can regard a way of modelling.

"Events" and "actors" are major concepts: "Events tend to involve a unitary activity engaged by a unitary set of actors for a limited period of time in a single physical location". An event may "dominate the attention" of one or more actors. A process of decision making is considered a sequence of events. It is divided into a "pre-decision period", a "decision-period" and a "post-decision period". Taking Simon's model of decision making into account, the pre-decision period corresponds with the Intelligence and Design phase, the decision period corresponds with the Choice phase and the post-decision period corresponds with what lately is often referred to as the Implementation phase. Emphasis is put on modelling the first and second period. As to the first period, Bahl and Hunt focus on describing an organization's properties as its "cognitive", "social-emotional" and "material resources". In addition, they focus both on describing conditions which can give rise to making a decision and on determining the time it takes a decision maker to make a decision. In our view these concepts relate to the systological problem, since they can be considered relevant in expressing why an ESS should be incorporated in the information system of an organization. As to the second period, Bahl and Hunt focus on describing a decision-maker's "familiarity of the subject", whether he "focuses on an existing repertoire of alternatives, or on designing new responses to situations". In addition, they focus on describing "evaluation criteria" and "selection mechanisms". These concepts can be regarded as part of the datalogical problem, since they can be considered relevant in expressing how an expert solves a problem.

The fifth and final "DSS contribution" considered in this discussion was reported by Keen and Scott Morton (1978) in "Decision Support Systems: an Organizational Perspective".

Concerning what we may consider a way of thinking, Keen and Scott Morton focus on improving a manager's performance by enhancing his "decision making ability". They too depart from the assumption that a DSS should support a manager in solving instances of unfamiliar classes of problems. A DSS is regarded as "more a service than a product" because "the problem can only partially be structured, and since managers
grow in their understanding and needs over time, a DSS must constantly grow and evolve as the user adapts and learns". Problems seem to be considered from the micro-perspective. They identify four "levels of support", i.e. "access to facts or information retrieval", "the addition of filters and pattern recognition ability to this data retrieval", "simple computations, comparisons and projections" and "useful models to the manager".

Regarding a possible way of working, Keen and Scott Morton suggest to carry out a "predesign cycle" prior to the design stage. They argue that "decision support requires a detailed understanding of decisionmaking in organizations. A descriptive framework provides the basis for prescriptive design; that is, to "improve" a decision process, one must first define and analyze it". This issue is overlooked by many designers: "Historically, computer specialists have tended to either pick the technology first and then look for a suitable application or to accept the client's definition of the problem". The predesign cycle consists of an "entry" stage and a "decision analysis" stage. They are carried out in an iterative way. The first stage centres around questions such as "1. Which alternative(s) offers the most improvement to the existing decision process? What are the likely economic payoffs? What is the cost of the organizational or behavioral gains? 2. How difficult will it be to implement this degree of change? Are we ready to commit the dollars, time, prestige, and support necessary?". In the second stage major decisions are identified and described as they are carried out and as they should be carried out. Keen and Scott Morton note that "in some instances, descriptive analysis will suggest that only a small subset of the decisions involved justify the use of a DSS, and the normative analysis will imply no payoff from such support. Some of the key decisions will be highly structured and thus best dealt with through batch computer systems or analytic models". In the design phase a DSS designer should translate design objectives into a user-interface, a set of "initial" routines and a database management system. The DSS should be checked for its robustness, modified after preliminary usage and assessed in relation to design objectives. Keen and Scott Morton note that "the managers' opinions and understanding of the decision situation are a major input to the design process", thereby proposing close participation of the future user of a DSS in its design process.

KBS

The sixth contribution considered in this discussion was reported by Waterman (1986) in "A Guide to Expert Systems".

As to what we may consider a way of thinking, we recall Waterman's definition of an ES as quoted in chapter 1. The "artificial expertise" embedded in an ES can be regarded as consistent, permanent, inexpensive to operate and portable as opposed to an expert's expertise. This is one of the major reasons why Waterman argues that an ES can be incorporated in an organization's information system. A relationship
between how an expert solves problems and how an ES should support him is difficult to identify. These observations seem to indicate that problems are considered from the meso-perspective rather than from the micro-perspective. Waterman apparently departs from the assumption that an expert shows restricted rational behaviour when he solves a problem. Artificial expertise is considered to consist of heuristics which an expert employs to solve instances of familiar problem classes.

Waterman departs from the assumption that a KBS consists of a knowledge base and an inference engine. He discusses a number of concepts which can be used to express the knowledge with which the knowledge base of a KBS can be furnished. He distinguishes between a rule, a frame and a semantic net. In our view, these concepts comprise a conceptual model of the datalogical problem. They serve to express how an expert or ES solves a problem.

Waterman suggests to incorporate an incremental phasing in the design process of an ES. He identifies five phases: "Identification", "Conceptualization", "Formalization", "Implementation" and "Testing". The first phase involves "identifying the problem itself ..., the participants in the development process ..., the required resources ..., and the goals or objectives of building the expert system". It looks as if an actual situation is not taken into account as a reference for designing an ES. During the conceptualization phase "the knowledge engineer and expert decide what concepts, relations, and control mechanisms are needed to describe problem solving in the domain". The next phase, formalization, seems to cover the same issue. It involves "expressing key concepts and relations in some formal way, usually within a framework suggested by an expert system building language". In fact, the concepts used to express how an ES should solve problems correspond with the ones used to express how an expert solves problems. In the implementation phase the designer "turns the formalized knowledge into a working computer program". Testing involves "evaluating the performance and utility of the prototype program and revising it as necessary".

The seventh contribution considered in this discussion was reported by Harmon et al. (1988) in "Expert Systems: Tools and Applications". It is "synthesized from interviews with many people who have fielded commercial applications".

The ES concept is defined in much the same way as Waterman does. However, Harmon et al. come up with a more explicit view on why an ES can be incorporated into the information system of an organization. They suggest that an ES can preserve and distribute the expertise of one or more experts. However, an ES can also contribute to making better, more consistent and faster decisions. The former observation indicates that a problem situation is considered from the meso-perspective. The latter one indicates that it is considered from the micro-perspective.
Like Waterman, Harmon et al. distinguish between concepts such as a rule, a frame and a semantic net as elements of a conceptual model of the datalogical problem. They too depart from the assumption that a KBS consists of a knowledge base and an inference engine.

Harmon et al. identify five phases in an ES development process: "Front end analysis", "Task analysis", "Prototype development", "System development" and "Field testing". Front end analysis, "involves all the questions you should ask before beginning an expert systems project in the first place". It involves "describing the overall objective of a project", i.e. researching ESs or solving an organizational problem, "identifying a problem/opportunity", organizing the development and implementation process, "analyzing costs and benefits", determining the return on investment and "establishing benchmarks for comparison". Task analysis involves "a detailed look at exactly what the expert does, with an eye to defining the precise portion of the task that will be useful to encode in an expert system". It is suggested to decompose a task into smaller ones. Once a task has been decomposed, one can "determine which steps or parts are hard and which are easy". What is meant by "hard" and "easy" is left unexplained. Emphasis should be placed on "hard" tasks. The designer "begins a detailed analysis of the knowledge and the inference processes actually used by the expert". In the third phase, prototype development, the designer continues to establish "concepts representing knowledge by describing the key concepts of the problem domain, interrelationships in the problem domain, and the flow of information needed to describe the problem-solving process (i.e. forward or backward chaining)". He should also select "appropriate software and hardware for the target application" and "the formats used for knowledge representation". Subsequently, he "starts to identify and document the reasoning processes of the human expert". In addition, the performance criteria which should be met by the prototype are specified. "A small version of the expert system to demonstrate the overall feasibility of the proposed system" is developed, tested with case studies and "adjustments are made to the performance and capabilities of a complete expert system". System development involves "expanding the knowledge base", "tailoring the user interface" and "monitoring the system's performance" and comparing it with the established benchmarks. The final phase, field testing, involves testing the ES "in the user environment". It is noted that "the prototype development, system development, and testing processes start all over whenever modifications are made to the system".

Harmon et al. discuss how a designer can plan the development process of an ES (concerning time, salaries and costs) and how he can calculate an ES's return on investment. These issues relate to a way of control.
The eighth contribution considered in this discussion is "Developing Knowledge-Based Systems: Reorganizing the System Development Life Cycle". It was reported by Weitzel and Kerschberg (1989).

According to Weitzel and Kerschberg, KBSs "depend more on access to knowledge than on computational techniques. They are particularly applicable when the problem cannot be solved with an algorithm". This way of thinking seems to correspond with the view that a DSS should support its user in solving instances of unfamiliar problem classes. Problem situations are considered from the micro-perspective. Weitzel and Kerschberg were triggered to write this contribution by their observation that there is a lack of detailed guidelines for KBS development. In this contribution they outline the "Knowledge-Based-System Development Life Cycle (KBSDLIC)". It is "a prototyping methodology for KBSs that uses expert system shells and programming environments". The function of a prototype in the KBS development process is regarded as a design specification.

Concerning a possible way of modelling, Weitzel and Kerschberg suggest to express the information which an expert processes in terms of e.g. entities, attributes and relationships. This observation applies to the infological problem. In addition, it is suggested to express how an expert processes information as "production rules" and "pseudo-code". This applies to the datalogical problem. As to the technological problem, they state that expert system shells should be used to develop prototype KBSs.

The following "processes" are defined in KBSDLIC: (1) "Define Problem and Assess Feasibility", (2) "Identify Subproblems", (3) "Identify and Define Conceptual Structure", (4) "Conceptual Design", (5) "Detail Design", (6) "Code", (7) "Test Reasoning", (8) "Test Knowledge" and (9) "Validation". These processes are "activated", "deactivated" and "reactivated", which indicates an iterative phasing. In the first phase, a distinction should be made between a "business problem" and a "knowledge problem". Solving the business problem involves defining the desired effects on an organization of incorporating a KBS into its information system. Solving the knowledge problem involves determining which classes of problems should be taken into consideration. In addition, the "technical" and "economic" feasibilities of a project should be addressed. In the second phase, identify subproblems, "the problem should be broken down into workable subproblems", because "small problems make for small prototypes that are easier to build than large ones". In the third phase, identify and define conceptual structure, "knowledge engineers search for concepts (to be represented as entities, attributes and relationships, JvW) that characterize the expert's thinking about the problem". In the fourth phase, conceptual design, "the knowledge engineers select a knowledge representation: first-order logic, procedural representations, semantic networks, ..., frames, ..., and database structures". In the next phase, detail design, a designer should "identify propositions for logic; write descriptions and pseudo-code for procedures;
draw network diagrams for semantic networks; write English language rules for production rules; ...; and identify and name table entries for data tables". The sixth phase, coding, "translates the detail design into the language of the knowledge engineering tool. This includes entering the knowledge representation into the tool's knowledge-base". The next phase, test reasoning, can be considered as an extension to the coding phase. Code is tested with regard to programming bugs. As "correct code does not mean correct knowledge", the eighth phase, test knowledge, "attempts to detect invalid and ambiguous knowledge ... It is detected when the system violates the expert's expectations". The last phase, validation, seems merely an extension to the previous two phases: "the main task is the detection of conditions missed earlier. This requires using a large sample of real cases".

In the mid 1980s Breuker and Wielinga (1986) identified "an increasing need for a methodological approach to the development of knowledge-based systems". Under their supervision much effort has been put into the development of KADS, an acronym of "Knowledge Acquisition Documentation Structuring". Schreiber et al. (1988) argue that "the objective of a knowledge based system is to solve problems". They explain that in KADS the development process of a KBS is considered "a mapping of a model of the problem solving process in the KBS domain onto a model of the KBS to be developed". A KBS too provides a mapping of problems onto solutions. It is of little value to determine whether a problem situation is considered from the micro-, the meso- or the macro-perspective. The focal issue in this methodology is not to improve the performance of an expert, organization or group of organizations but to model problem solving processes.

Concerning a possible way of modelling, in KADS a distinction is made between four "layers" of knowledge which in our view seem to apply to the infological problem. The first, "domain", layer "contains the static knowledge of the domain: domain concepts, relations and complex structures". The second layer is the "inference layer". It builds upon the domain layer and "states what inferences can be made: not how or when they are made. The 'when' is specified at the next, task layer. The 'vocabulary' for the inference layer consists of three elements: knowledge sources, metaclasses and dependencies between these. A structure of these elements is called an inference structure. The inference structure provides a flow of data view; the task structure a flow of control view superimposed on this inference structure". A "knowledge source" can be viewed as "abstracted operations on abstracted concepts". A "metaclass" stands for a set of domain concepts and can be considered the parameters of a knowledge source. The fourth layer is the strategic layer "in which knowledge resides which allows the system to make plans - i.e. create a task structure -, control and monitor the execution of tasks, diagnose when something goes wrong and find repairs for impasses".
Figure 3.1 Phasing and abstraction in KADS, see Schreiber et al. (1988).

KADS also seems to provide concepts to support the construction of models of infological and datalogical designs. As to modelling an infological design, KADS identifies "functions" like "problem solving", "data I/O" and "data storage". To model a datalogical design, KADS provides "methods" such as "hierarchical classification" and "table lookup". Methods consist of "design elements" such as "rules" and "procedures". A KBS is assumed to consist of a knowledge base, a database and an inference engine.

Regarding a way of working, KADS identifies two major phases in the development process of a KBS: analysis and design. These phases are carried out in a linear way. In these phases modelling takes place at five levels of abstraction: at a "linguistic", "conceptual", "epistemological", "logical" and "implementation" level, see figure 3.1.

The designer starts off by collecting data. Data "consist of raw input data from the real world: text books, interviews, thinking aloud protocols, etcetera". Data reside under the "linguistic" level of abstraction. In these data a designer should identify and classify concepts which are distinguished in the above-mentioned four-layer model. The resulting model is called a "conceptual model". KADS provides a designer with means to construct such a model, i.e. with "interpretation models", see e.g. Breuker et al. (1987). They explain: "The role of an interpretation model is a template structure that allows for (some) top down refinement instead of bottom up construction of the conceptual model in the Analysis stage. An interpretation model is an intermediary
between "what to look for in the data" and the conceptual model that is to be constructed". The "epistemological" level is the highest level of abstraction. At this level, the analysis phase stops and the design phase starts off: "M1 is a model of the problem solving process at both the conceptual and the epistemological level. It is a model of the domain expertise ... it is completely neutral with respect to implementation formalisms. M2 - i.e. the 'design' model - is a model at the same level of abstraction, but of the artifact rather than of the real world. It is, in fact, a high level system design. In Schreiber et al. (1987) it is explained that "in the design phase the output of the analysis phase ... is decomposed into a set of functional blocks", called a "functional description". It "specifies the internal and external behaviour of the artifact". Appropriate formalisms for which realized implementations are known, i.e. "methods", are selected and assembled. Schreiber et al. explain: "Methods are selected to realize the desired behaviour specified in functional blocks. This part is viewed as being a crucial step in KBS design. The elements required to realize a particular method are called design elements. Methods will be mostly (but not necessarily) AI methods". Then "M2 can be transformed into a detailed system design (M3) and subsequently into actual code without major decisions having to be made". Schreiber et al. name this activity "physical description": "The choice of an environment is also part of the physical description. An environment is defined as an off-the-shelf software system with a predefined set of methods and design elements". KADS provides a designer with methods and techniques to construct and represent Data, M1, M2 and M3.

"Structured Knowledge Engineering" (SKE) is a KBS development methodology which has its origins in KADS, see e.g. SKE (1987). It is the last contribution considered in this discussion.

The developers of SKE put forward that in a business environment the development process of a KBS should be clarified and controlled. SKE contains the same way of modelling we identified in KADS.

As opposed to methodologies which can be characterized as "rapid prototyping", it is claimed that SKE is a "structured methodology". In SKE the following phases are distinguished: (1) "Strategic Knowledge- and Information Planning", (2) "Initiation", (3) "Knowledge Acquisition", (4) "Technical Design", (5) "Building", (6) "Testing and Acceptation" and (7) "Maintenance". It is suggested to carry out these phases in a linear way. The first phase is aimed at identifying one or more eligible application domains for KBS and at selecting one on the basis of "financial-economical", "commercial" and "general knowledge technological" feasibility studies. This is achieved by making an analysis of the actual "organizational structure", "processes", "knowledge domains" and "information systems". In the second phase, Initiation, a more detailed "knowledge technological" feasibility study is carried out upon the application domain
selected in the previous phase. A plan is made containing the further steps to be taken in the project. The third phase, Knowledge Acquisition, can be considered similar to the Analysis phase in KADS. It is divided into three sub-phases, i.e. "Orientation", "Problem Identification" and "Problem Analysis". In the "orientation" sub-phase, a designer should determine an expert's organizational function, the kind of problems he solves and should make a glossary of domain specific concepts. In the "problem identification" sub-phase, one should carry out a more detailed functional analysis, make a decomposition of an expert's task into sub-tasks and identify structures of domain concepts. In the last "problem analysis" sub-phase, a designer should construct a "conceptual model" of an expert's problem solving process, viz. by means of an interpretation model. In addition, he should describe communication aspects, both between the system and its user and between the system and other existing systems. In the "technical design" phase, a designer should specify knowledge bases, describe the inference strategy of an expert as it should be carried out by a KBS, describe the relations between the KBS and existing databases and the required interfaces, the required help facilities, screen layouts, uncertainty factors, software tools and hardware. In the "implementation" phase, the designer should develop software, software documentation, an installation manual and a user manual. In the testing and acceptance phase, the system is tested for correctness and completeness of its functionality and for its code. The last phase, maintenance, is tentative. SKE provides a designer with the same methods and techniques as KADS does.

3.3 Conclusions

In the previous section we discussed a sample consisting of ten contemporary contributions to ESS design. The contributions discussed are derived from the DSS and KBS literature. We have found no contributions which explicitly address the problem of ESS design. However, there is evidence indicating that ESSs are designed in the same way as DSSs and KBSs.

In the DSS literature much attention is paid to a way of thinking. As Sol (1987, p. 203) puts it: "Many writers seem to approach DSS as a philosophy to seek a useful complementary between technological tools and human judgement and discretion". Problem situations are considered from the micro- and the meso-perspective. The macro-perspective is not addressed. Little attention is paid to understanding a problem situation. Systematic efforts to complement a way of thinking with a way of modelling, a way of working and a way of control are not reported, although we should note that attention is focused on deciding which phasing should be incorporated into a DSS design process. In many contributions an iterative or
incremental phasing is proposed. However, it can be questioned whether this type of phasing is applied with care. As Sol (1987, p. 206) puts it: "A prototyping or incremental approach is not necessarily converging and is no guarantee for an effective DSS".

In the KBS literature much attention is paid to a way of modelling and a way of working. A way of thinking and a way of control receive little or no attention. Problem situations are considered mainly from the micro-perspective. As in the DSS literature, little attention is paid to understanding a problem situation. As to a way of modelling, the systological and infological problems are not modelled in any of the contributions considered. A possible explanation can be found in the fact that most methodologies are strongly influenced by a technology push rather than by an application pull. Woods (1987, p. 572) argues that "the problem of providing effective decision support hinges on how the designer decides what will be useful in a particular application, that is, a problem-driven rather than technology-driven approach where requirements and bottlenecks in cognitive task performance drive the development of tools to support the human problem-solver". Relatively much attention is paid to modelling the datalogical and technological problems. Regarding a way of working, much attention is paid to suggesting which kind of phasing should be incorporated into the design process of a KBS. There is relatively little interest in providing a designer with methods to construct models.

We find it striking that in all contributions discussed little attention is paid to a way of control. Taking into account that no attention is paid to modelling the systological problem, we can conclude that a few ESSs have been developed in a business environment. As Ivanov (1988, p. 93) puts it: "Most ESS prototypes disregard many economical and social aspects which appear only too late with the full-scaleness of ESS production and usage ... It becomes a 'heuristic' muddling-through with scanty reference to the classical problems of the hypothetical-deductive, inductive-deductive method, and statistics, and of the problem of synthesis between realism and idealism. In practice this is reflected by the rare literature on cost-effectiveness of ESS, and on the implementation from prototype into production".

To conclude this chapter, we propose to relate the effectiveness of an ESS to the way the ignored issues are addressed. One can open new paths for research by incorporating them into the design process of an ESS. As will be demonstrated in the next chapter, our contribution lies within a systematic effort in postulating a way of thinking and in complementing this way of thinking with a way of modelling, a way of working and a way of control.
References in chapter 3


Schreiber, G., Breuker, J., Bredeweg, B. and Wielinga, B., "Towards a Design Methodology for KBS", Deliverable D8, Esprit Project 1098, VF Memo 97, Knowledge Acquisition in Formal Domains, Department of Social Science Informatics, University of Amsterdam, Amsterdam, 1987.


4

MEDESS

4.1 Introduction

In the previous chapter we suggested a framework to describe how contemporary ESS design methodologies support ESS design. In this framework attention is paid to the way of thinking underlying a methodology and to the manner in which a way of thinking is complemented by a way of modelling, a way of working and a way of control.

We observed that problem situations are not frequently considered from the meso- and the macro-perspective, that actual situations are not often taken into consideration as a reference for designing and evaluating ESSs, that the systelological and infological problems are not frequently addressed, that few methods and techniques are provided to design ESSs and that a way of control receives little or no attention.

We hypothesize that ESS design can be put into practice effectively by using a methodology that addresses the neglected issues. To test this hypothesis we set out a way of thinking and complement it by a way of modelling, a way of working and a way of control. The resulting methodology is called "MEDESS", an acronym of "Methodology for Designing Expert Support Systems".

4.2 Way of thinking

We adopt the Information Paradigm as a starting point to set the boundaries of a problem situation. The Information Paradigm is described by Brussaard and Tas (1980, p. 822): "Each set of interesting phenomena (i.e. also each dynamic system such as an organization) can be abstracted to a real system and an information system which determines the behaviour of the real system. The information paradigm (IP) includes the "recursion principle", which stipulates that the information paradigm also holds for the sub-systems, i.e. for all RS and IS within the system (organization) considered and that it does so at all levels (e.g. hierarchical levels in an organization)."

We suggested that an ESS designer can consider a problem situation from the micro-, the meso- and the macro-perspective. Bots (1989, p. 10) remarks that "these three perspectives on the improvement of organizational performance can be related directly to information systems when we adopt the information paradigm". From each perspective an ESS designer faces the problem of improving the performance of an IS/RS combination.
In the remainder of this section, we will discuss with the Information Paradigm in mind how the boundaries of a problem situation can be set when it is considered from the micro-, the meso- and the macro-perspective.

**Micro-perspective**

We follow the paradigm of bounded rationality when it comes to describing how an expert solves a problem. As opposed to the paradigm of substantive rationality, which assumes that a human being possesses unlimited cognitive abilities, the paradigm of bounded rationality puts an emphasis on the "procedure" which a person follows to solve a problem. We call such a procedure a *task*.

A task is associated with a class of problems. Executing a task involves solving an instance of a problem class. Executing a task corresponds with understanding and changing the behaviour of a part of reality. With the Information Paradigm in mind, we think of a task as an IS/RS combination, see figure 4.1. The IS contains and processes images of the RS. The arrows that come to and depart from the IS and RS and that can be read from left to right represent information flows and real flows respectively which can exist between tasks. In chapter 2 we pointed out that a problem solver coordinates the process of solving a problem. As put forward by Brusgaard and Tas, the Information Paradigm includes the "recursion principle". We apply this principle to the IS of a task. In this way we can make a distinction between coordinated and coordinating tasks. The RS of the IS consists of *sub-tasks*. The IS of the IS consists of *coordination tasks*. Executing a coordination task involves processing images of e.g. sequences in which sub-tasks should be executed.

![Figure 4.1 Conceptual model of a task.](image-url)
An expert fulfills a function in an organization. He solves instances of one or more problem classes. His objective is to do this according to an established standard. From the micro-perspective we express a problem situation in terms of an expert's objective and performance in achieving his objective. More specifically, we distinguish between his efficiency and/or effectiveness. Efficiency is the valuation of the amount of means which an expert uses to execute a task. Effectiveness is the valuation of the extent to which an expert solves a problem by executing a task.

In chapter 2 we observed that an expert's learning style is directed towards generating accommodative knowledge. He does not always have to conceptualize a problem. An expert may have an appropriate conceptual model at his disposal. In this line of thought we should not be primarily concerned with designing ESSs that can support an expert in conceptualizing problem situations. Rather, we will concentrate on designing ESSs that can support an expert in specifying problem situations.

Designing an ESS involves assigning an expert and an intended ESS to the tasks which so far have been executed by the expert only. An ESS can support an expert in executing both sub-tasks and coordination tasks.

As to sub-tasks, we distinguish between tasks which an expert executes on the skill-based level and tasks which he executes on the rule-based level.

On the skill-based level an expert directly associates a solution with a problem. The execution of such a task can take an expert relatively little time. When an expert's performance is defined as his time efficiency, there may be little value in eliciting, formalizing and automating the execution of a description of how he executes a task. An ESS can merely support an expert in recording problems and inferred solutions.

On the rule-based level an expert selects and carries out a procedure to arrive at a solution to a specific problem. The execution of such a task can take an expert substantial amounts of time. When an expert's performance is defined as his time efficiency, we may direct our attention to eliciting, formalizing and automating the execution of a description of how he executes a task. Such support has improvement of time efficiency as its main objective. However, its spin-off can be improvement of effectiveness as an expert cannot generate slips anymore. As Rouse (1985, p. 623) puts it, this type of support can result in "decreased workload due to a reduced need for humans to monitor their own performance as closely as would be necessary without aiding". It should be kept in mind that such support can result in automatic generation of lapses. This is often regarded as one of the pitfalls of automating the execution of a task. As Sage (1981) puts it: "We should be rather cautious however in the apparently reasonable inference that we learn correctly from experience ... by no means do people always improve their judgment and decision making ability on the basis of increased experience".
An ESS cannot execute an expert's coordination tasks. It should be applied to support an expert rather than to replace him. However, an expert can instruct an ESS in which sequence it should execute the tasks which an expert used to execute on the rule-based level.

An expert can generate slips and lapses during the execution of his tasks. If he does not detect an error, effectiveness decreases. If he detects an error and recovers it, efficiency decreases. To detect and recover errors, an expert monitors the execution of his tasks. We consider monitoring as a specific form of coordination. Support can be achieved by automatically monitoring the execution of tasks which an expert executes on the skill-based level. As Norman (1981, p. 11) puts it: "For a slip to be detected, the monitoring function mechanism must be made aware of the discrepancy between intention and act".

**Meso-perspective**

An organization fulfills a primary function in its environment, e.g. it provides a particular service. An organization also fulfills a secondary function for the individuals employed by it, e.g. to meet their needs of being acknowledged for their merits. Problem situations are often due to developments taking place in the environment in which an organization fulfills its primary function and/or inside the boundaries within which an organization fulfills its secondary function. These developments correspond with the varying urges of organizations and individuals to expand, consolidate or even reduce the scope of their position or development. The interests of organizations and individuals have their origin in these urges. The will to meet these interests can lead to tensions, i.e. problem situations, see e.g. In't Veld (1983).

We think of an organization as an IS/RS combination, see figure 4.2. The arrows that come to and depart from the IS and RS and that can be read from left to right represent information flows and real flows respectively which can exist between organizations. The RS of an organization can be expressed in terms of problem classes, individuals and tasks relating individuals to problem classes. The IS of an organization coordinates them. It processes images of individuals, problem classes, tasks and sequences in which tasks should be executed.

From the meso-perspective we express a problem situation in terms of an organization's objective and performance in achieving its objective. More specifically we distinguish between its efficiency and/or effectiveness. It can be considered as a complex function of the performances which characterize the tasks taken into account from the micro-perspective.
Designing an ESS can be placed in a broader context of reconsidering an organization's objective and performance. Within this context one may review existing and specify new problem classes, individuals and tasks relating individuals to problem classes. One may also reconsider in which sequences tasks should be executed. Once these issues are resolved, the design of an ESS can come into view.

**Macro-perspective**

An inter-organizational system can be considered as a system of two or more cooperating organizations that recognize that their performances in achieving their separate objectives can be improved by addressing their joint performance in achieving a common objective.

From the macro-perspective we express a problem situation in terms of the common objective of an inter-organizational system and its performance in achieving this objective.

One can disregard the boundaries between two or more organizations and consider them as one organization. The performance of an inter-organizational system can also be a complex function of the performances which characterize the tasks taken into account from the micro-perspective.
As is the case from the meso-perspective, the design of an ESS can be placed in a broader context of reconsidering the common objective of an inter-organizational system and its performance. Within this context one may review existing and specify new problem classes, individuals and tasks relating individuals to problem classes. One may also reconsider in which sequences tasks should be executed. Once these issues are resolved, the design of an ESS can come into view. However, special attention should be paid to achieving consensus among the organizations involved as to which organization owns which ESS.

4.3 Way of modelling

Systeological problem

When an ESS designer addresses the systeological problem he describes a problem situation from an organizational perspective. To represent the concepts which we consider essential when considering a problem situation from this perspective, we adopt the Entity-Attribute-Relationship description form.

The first concept which we introduce is one of an organization. An organization is a purpose-oriented system. It has an interest. To meet its interest an organization tries to achieve a number of objectives according to some established standard. One can distinguish between primary and secondary objectives. An organization's performance in achieving its objectives is focal.

type Organization
[ attributes
  - Interest
  - Objective
  - Performance
 ]

We also introduce the concept of a problem class. In an organization instances of one or more problem classes are solved to achieve its objectives.

type Problem class
[ attributes
  - Organization
 ]
Subsequently, we introduce the concept of an \textit{individual}. In an organization one or more individuals are employed to solve problems. Like an organization, an individual, too, can have an interest. This may very well be a personal one. To meet his interest, an individual also tries to achieve a number of objectives according to some established standard. His \textit{performance} in achieving his objectives is focal.

\begin{verbatim}
entity type Individual
[ attributes
  - Organization
  - Interest
  - Objective
  - Performance
]
\end{verbatim}

We introduce a M:N relationship between a \textit{problem class} and an \textit{individual}. Instances of a problem class can be solved by one or more individuals and an individual can solve instances of one or more problem classes. Along with this relationship we finally introduce the concept of a \textit{task}. The objective of a task is associated with the objective of an organization. The performance of a task in achieving an associated objective is focal.

\begin{verbatim}
entity type Task
[ attributes
  - Problem class
  - Individual
  - Objective
  - Performance
]
\end{verbatim}

Summarizing, we have introduced four simple concepts to describe the systelogical problem, viz. an organization, a problem class, an individual and a task. These concepts are essential in describing a problem situation from an organizational perspective and in specifying the added value of an ESS to that organization. They look fairly obvious. Nevertheless, as we pointed out in the previous chapter, these concepts and the relationships between them are often overlooked.
Infological problem

In addressing the infological problem an ESS designer describes the tasks which an expert or ESS executes, the information which is processed in executing a task and the sequence in which these tasks are executed. To describe a task we basically follow the line set out by Bots (1989, p. 40). A task is a 4-tuple \( t = (I_{in}(t), I_{out}(t), B(t), S(t)) \). \( I_{in}(t) \) represents the information needed to execute \( t \) and \( I_{out}(t) \) represents the information resulting from the execution of \( t \). \( B(t) \) represents the problem solving behaviour, i.e. skill-based or rule-based, shown by an expert while executing \( t \). \( S(t) \), the support structure of an ESS associated with \( t \), represents a collection of tasks which an ESS can execute to support or to automate the execution of \( t \).

To describe the input information and the output information of a task, i.e. the RS of a task, we adopt the Entity-Attribute-Relationship description form.

The introduced definition of a task, which emphasizes the static aspects of a task, is not sufficient to describe a task in terms of its sub-tasks. Therefore, in extension to this definition, we introduce the concept of a task precedence structure. The precedence structure of \( t \) is a directed graph \( (ST, E_p) \). \( ST = \{st_1, ..., st_n\} \) is the collection of all sub-tasks which are executed to solve an instance of a problem class with which \( t \) is associated. \( E_p = \{(st_i, st_j) | I_{out}(st_i) \cap I_{in}(st_j) \neq \phi\} \). \( E_p \) is the subset of \( ST \times ST \) including a precedence relationship, i.e. an edge \( e_{ip} \) leading from \( st_i \) to \( st_j \) if and only if the output part of \( st_i \) overlaps with the input part of \( st_j \).

As we put forward in MEDESS' way of thinking we also wish to describe a task's dynamics in terms of its coordination tasks. Therefore we introduce the concept of a task coordination structure. The coordination structure of \( t \) is a directed graph \( (ST \times CT, E_c) \). \( ST = \{st_1, ..., st_n\} \) is the collection of all sub-tasks which are executed to solve an instance of a problem class with which \( t \) is associated. \( CT = \{ct_1, ..., ct_m\} \) is the collection of all coordination tasks which are executed to determine which sub-task in \( ST \) or coordination task in \( CT \) should be executed next. \( E_c = \{(x, y) | x \in ST \cup CT \land y \in ST \cup CT \land x \neq y\} \). \( E_c \) is the subset of \( ST \times CT \times ST \cup CT \) including a coordination relationship, i.e. an edge \( e_{ip} \) leading from \( x_i \) to \( y_p \). There is one important restriction to the elements of \( E_c \). It cannot contain tuples \( (x, y) \) whereby both \( x \in ST \) and \( y \in ST \).

The support structure \( S(t) = \{s_1(t), ..., s_n(t)\} \) of an ESS associated with \( t \) is a collection of tasks which an ESS can execute to support an expert in executing \( t \) on the skill-based level or to automate the execution of \( t \) when an expert used to do this on the rule-based level. We call such a task a function so as to distinguish between a task executed by an expert and a task executed by an ESS.
Datalogical problem

In addressing the datalogical problem an ESS designer describes how information is grouped and how an expert or an ESS executes a task without considering which technology is used to achieve it.

To describe how information is grouped, we adopt the relational model, which is discussed in e.g. Van der Ende (1987). A relation $R$ is considered an object type $R$ with properties $V_1$, ..., $V_n$, which can be denoted as $R(V_1, ..., V_n)$. The key $S$ of a relation $R$ is a subset of attributes of $R$ with two properties: (1) In each tuple of $R$ the value of $S$ unambiguously determines that tuple, and (2) No attribute of $S$ can be omitted without eliminating the first property.

To describe how grouped information can be operated upon, one can apply relational algebra or Structured Query Language (SQL). An operation has one or two relations as an operand and produces a new relation. Relational algebra as well as the semantics of SQL are discussed in e.g. Van der Ende (1987).

To describe how an expert or an ESS executes a task we adopt Structured English. It can be considered as a limited subset of the natural English language. The basics of Structured English are discussed in e.g. Ziya Aktas (1987).

Technological problem

In addressing the technological problem an ESS designer describes the technical implementation of an ESS. The hardware which an ESS designer chooses to implement an ESS on and the software which he selects to program an ESS in provide the context in which he can describe the technical implementation of an ESS.

An ESS can be used by executing one or more programs. For a discussion of the concepts in terms of which a program can be described we refer to e.g. Wirth (1976), who discusses the semantics of the Pascal programming language. Basically a distinction is made between data structures and algorithms.

As e.g. Sprague (1986), we identify three major elements in the technical implementation of an ESS, i.e. a database management system, a model management system and a user interface.

A description of the database management system of an ESS can contain descriptions of record types each representing a relation that comprises the database and descriptions of algorithms that represent the operations on these relations.

A description of the model management system of an ESS can contain descriptions of algorithms that prescribe how an ESS can support an expert in coordinating the execution of the tasks to which it is assigned. As to the tasks which an expert executes
on the skill-based level, it can contain descriptions of algorithms that prescribe how an ESS can support an expert in recording problems and solutions. Regarding the tasks which an expert executes on the rule-based level, it can contain descriptions of algorithms that prescribe how an ESS should execute a task.

By user interface we refer to the sub-system of an ESS that supports an expert and an ESS in communicating with each other. The issue of designing a user interface appears to receive little attention. According to Dos Santos and Holsapple (1989, p. 1) DSSs "tend to be developed on an ad hoc basis, without the benefit of a formal framework for design. Consequently, the interface is often one of the weakest aspects of a DSS in the sense of being unnecessarily restrictive and inflexible".

Therefore, we introduce the concept of a task interface. It enables an expert and an ESS to communicate with each other during the execution of a sub-task or a coordination task. The user interface of an ESS can consist of one or more task interfaces. We describe a task interface mainly in terms of windows, lists and forms.

A window is a rectangle which demarcates part of a screen. It can contain a header and a footer. These attributes serve to provide a window with a meaningful description. A window can be used e.g. to focus an expert’s attention on and provide him with the information which is the input information or the output information of a coordination task or a sub-task.

A list is a sequence of items. It can also be attributed with a header. When a list serves as a "pick list" and contains many items, it can be provided with a scroll bar. A list can be used e.g. to enable an expert to execute a coordination task by offering him the opportunity to select from it the task which an ESS should execute next. A list is then often called a menu.

A form is also a sequence of items. In a form an item consists of a label and a part that can represent a value. When an item consists of a label and more than one part that can represent a value, we speak of a table. A form, too, can be provided with a header. It can be used e.g. to enable an expert to specify the contents of a new record.

A description of the user interface of an ESS can also contain descriptions of procedures that prescribe how an ESS should generate a task interface.

4.4 Way of working

Phasing

In the design process of an ESS we distinguish between two major phases, i.e. an understanding phase and a design phase. In the understanding phase attention is directed to conceptualizing and specifying an actual situation. In the design phase attention is directed to conceptualizing and specifying a new situation.
A problem situation can be considered from the micro-, the meso- and the macro-perspective. From each perspective the systelological, infological, datalogical and technological problems can be modelled iteratively. Welke (1977, p. 150) remarks that "in fact any initially chosen sequence is satisfactory so long as it is iterative, all three (i.e. four, JvW) perspectives are included and alternatives do not become solutions until all perspectives are accounted for".

The phasing which we propose in MEDESS is represented in figure 4.3. It can be interpreted as follows. On the left we identify the reality which is of interest to an ESS designer, i.e. an IS and a RS, in which the IS controls the RS. He starts by creating a systelological, infological, datalogical and technological description. This can be an iterative learning process, in which a review of e.g. the systelological description can result in a review of the infological description etc. Once an ESS designer finds that he fully understands the problem situation, he continues by creating a systelological, infological, datalogical and technological design. This, too, can be an iterative learning process, in which a review of e.g. the systelological design can result in a review of the infological design etc. On the right we identify a new situation, i.e. an IS and a RS, in which the IS controls the RS as well. This situation occurs when the systelological, infological, datalogical and technological designs are implemented.

![Figure 4.3 Phasing in MEDESS.](image-url)
Methods

A way of working does not only apply to the phasing but also to the methods an ESS designer can employ to construct models. Here we will propose a number of methods to construct an empirical description of the systelographical, infological, datalogical and technological problems. In MEDESS' way of modelling an ESS designer can look for concepts which he can use to construct a conceptual description of these problems.

Systelological description

Figure 4.4 represents a possible method for constructing a systelological description. One can start by defining (part of) the organization that employs the expert of interest. One can continue by determining and assigning suitable values to its objective and its performance. This can have much to do with the developments taking place in the environment in which an organization fulfils its primary function and/or inside the boundaries within which an organization fulfils its secondary function. As put forward earlier, these developments correspond with the continuously changing urges of organizations and individuals to expand, consolidate or even reduce the scope of their position or development. The interests of an organization or individual have their origin in these urges.

One can establish an organization's objectives by identifying its interests and those of the organizations and/or individuals associated with it. Subsequently, one can identify which wishes they have with respect to each other in order to meet their interests. Finally, one can translate these wishes into primary and secondary objectives. As to an organization's performance, one can distinguish between its efficiency and effectiveness in achieving these objectives.

Considering the organization at hand, one can continue by iteratively defining appropriate problem classes, individuals and tasks and by determining and assigning suitable values to the objective and performance of a task.

One should choose an appropriate level of aggregation on which one can identify instances of the problem classes solved to fulfil an organization's functions. An appropriate level of aggregation can be one on which a distinction can be made between instances of problem classes solved by separate individuals.

One can derive the objective of a task by identifying the desired contribution which an organization has associated with this task in order to achieve its objectives. As to the performance of a task, one can also distinguish between the efficiency and effectiveness in achieving its objective.
Figure 4.4 Construction of a systological description.

Systological design

Figure 4.5 represents a possible method for constructing a systological design. One can start by adapting an organization’s interests and subsequently its objectives and/or performance to the assessed developments taking place in the environment in which an organization fulfills its primary function and/or inside the boundaries within which an organization fulfills its secondary function. These developments concern an organization’s urge to expand, consolidate or reduce the scope of its position.

Considering the organization defined and the question whether it intends to consolidate, expand or reduce the scope of its position, one can iteratively define, review and remove problem classes, individuals and tasks in such a way that an organization’s desired performance in achieving its (new) objectives can be realized. For instance, when an organization intends to expand its position an ESS designer will primarily be concerned with defining new problem classes, individuals and tasks. Reviewing a task involves determining and assigning newly desired values to its objectives and/or performance.
Figure 4.5 Construction of a systelological design.

The methods which we suggested for constructing a systelological description and a systelological design can be followed when considering a problem situation from the micro- and the meso-perspective. These methods cannot directly be followed when considering a problem situation from the macro-perspective. One should then extend these methods with an extra iteration in which one can explicitly address the definition and review of an organization and in which one can explicitly address the definition and review of relationships between organizations and problem classes and between organizations and individuals.
Infological description

Describing a task involves iteratively defining its precedence structure, its RS and, when necessary, its coordination structure. Figure 4.6 represents one iteration.

The definition of a task precedence structure involves iteratively decomposing a task into appropriate sub-tasks and precedence relationships between them. The first coordination task in this process should be executed to determine whether, in view of the sub-tasks and precedence relationships defined so far, it is necessary to define a new sub-task or precedence relationship. One can continue by decomposing a task until a level of aggregation has been reached on which one can only define sub-tasks of which the output information corresponds with the value of one attribute of one entity in the RS of the task. The second coordination task should be executed to determine whether the task precedence structure is complete, e.g. when the input information of the first sub-task and the output information of the last sub-task correspond with the input information and output information of the task.

Defining the RS of a task involves iteratively defining appropriate entities and relationships between them. The first coordination task in this process should be executed to determine whether, in view of the entities and relationships defined so far, it is necessary to define a new entity or relationship. This can be done in two dimensions of abstraction i.e. the generalization-specialization or "is a" dimension and the aggregation-decomposition or "is part of" dimension, see e.g. Ter Bekke (1988). The second coordination task should be executed to determine whether the RS is complete, e.g. when the input information and the output information of a task can be mapped onto its RS.

The definition of a task coordination structure involves iteratively decomposing a task into appropriate sub-tasks, coordination tasks and coordination relationships between them. The first coordination task in this process should be executed to determine whether, in view of the sub-tasks, coordination tasks and coordination relationships defined so far, it is necessary to define a new sub-task, coordination task or coordination relationship. The sub-tasks that comprise the precedence structure of a task should also be found back in its coordination structure. Both a sub-task and a coordination task should be preceded by a coordination task when their execution is not unconditionally. The second coordination task should be executed to determine whether the task coordination structure is complete, e.g. when the coordination relationship that points at the first sub-task or coordination task and the one that points from the last sub-task or coordination task correspond with the ones pointing at and from the task.
Figure 4.6 Construction of an infological description.
Infological design

Reviewing a task involves reviewing its associated task precedence structure, its RS and its associated task coordination structure. Although it may be necessary to review a task this thoroughly, we expect that the construction of an infological design will primarily involve defining the support structure of an ESS associated to the task. Figure 4.7 represents one iteration.

Reviewing a task precedence structure involves iteratively defining, reviewing and removing sub-tasks and defining and removing precedence relationships between them. Reviewing the RS of a task involves iteratively defining, reviewing and removing entities and defining and removing relationships between them. Reviewing a task coordination structure involves iteratively defining and removing sub-tasks, coordination tasks and coordination relationships between them.

The definition of a support structure involves iteratively defining appropriate support functions and automated functions. Basically, we follow the line that a task which an expert executes on the skill-based level should be associated with a support function that for instance can provide an expert with the input information needed to execute the task. In addition we suggest that a task which an expert executes on the rule-based level should be associated with an automated function.

Datalogical description/design

One does not always have to distinguish between the construction of a datalogical description and design. We are not primarily concerned with improving the way in which an expert executes his tasks. We are more concerned with designing how an ESS can execute the functions taken up in the support structure of a task and aggregated under a database management system or a model management system.

Regarding a database management system, the design of a database involves transforming the entities and relationships which constitute the RS of a task into a set of appropriate relations. It is an iterative process of defining, removing and reviewing relations again in two dimensions of abstraction, viz. the generalization-specialization or "is a" dimension and the aggregation-decomposition or "is part of" dimension. How to design operations, in relational algebra or in SQL, which can be executed to retrieve, store and eliminate tuples of a relation, is demonstrated by e.g. Van der Ende (1987).

Concerning a model management system, the design of how an ESS should execute an automated function and a support function primarily involves eliciting and describing how an expert executes a task.
Figure 4.7 Construction of an infological design.
How an expert executes a task on the rule-based level can be elicited and described by following one of the many methods described in the literature. When we rely on verbal information, we can interview an expert by introspection, self report and review, see e.g. Breuker and Wielinga (1984). To choose an appropriate interviewing technique one can take into account whether one wants to derive a procedure directly, i.e. by introspection, or whether one wants to obtain examples of how an expert executes a task, i.e. by self report, and configure a general procedure on the basis of these examples.

How an expert executes a task on the skill-based level can be elicited and described when an expert can explain how he executes this task on the rule-based level. This can also be done when the ranges of problems, solutions and associations between them are limited, that is to say when one can easily derive a "heuristic" decision table.

Technological description/design

One does not always have to distinguish between the construction of a technological description and design. We are primarily concerned with the formulation of a technological design.

An ESS designer can start by choosing the hardware on which an ESS can be implemented and the software in which it can be programmed. Today it is fairly obvious on which hardware an ESS can be implemented, i.e. on a micro-computer or workstation. The software in which an ESS can be programmed, however, is less obvious. One can choose a 3GL such as Turbo Pascal, a 4GL such as DBase IV plus, a DSS generator such as Personal Wizard, an ES shell such as Goldworks or an AI language such as Prolog. The choice of appropriate software can depend for instance on whether the database management system of an ESS will contain many relations and operations on these relations, whether the model base in an ESS will contain many mathematical models or rather many "symbolic" models, whether the user interface of an ESS will contain many graphical task interfaces.

The choice of hardware and software sets the context in which an ESS designer can describe the technical implementation of an ESS. One can proceed by designing a database management system, a model management system and a user interface. These elements can be designed for example according to the guidelines presented by Sommerville (1982). Engineering the software of these three elements of an ESS can involve transforming the datalogical designs of a database management system and of a model management system into data structures and algorithms and subsequently into source code.
Designing a database management system involves transforming the relations which constitute the database of an ESS into record types. It also involves transforming the operations which can be executed on these relations into procedures and/or functions.

Designing a model management system involves transforming the models which constitute the functions in a support structure into procedures and/or functions.

Designing a user interface involves designing task interfaces and designing procedures and/or functions that can generate and manage these interfaces. As to designing a task interface, one may apply a form to support an expert for instance in recording a problem or an inferred solution. One may also apply a form to support an ESS in presenting to an expert an inferred solution. One may apply a menu to support an expert in instructing an ESS which task to execute next.

4.5 Way of control

Where MEDESS' way of working is concerned with supporting an ESS designer in constructing accurate and complete descriptions of the systelogical, infological, datalogical and technological problems, its way of control is concerned with supporting an ESS designer in constructing these descriptions efficiently. MEDESS' way of working and way of control mutually interact. This is the reason why Seligmann et al. (1989) suggest to put them together under a way of organizing.

Sol (1990) identifies two major issues in a way of control, i.e. the organization of a project and its progression. It should be noted that these issues are not specific for MEDESS. However, they should be addressed in any ESS design methodology.

The organization of a project concerns the planning of (1) which tasks should be executed to construct descriptions of the systelogical, infological, datalogical and technological problems and in which sequence which they should be executed, (2) the time, manpower and equipment required to execute these tasks, (3) the initiation and duration of the project and (4) the budget of the project.

The progression of a project concerns the planning and execution of control tasks in the fixed sequence of tasks. One can execute a control task on the basis of results recorded in e.g. project documents. One can assess, on the basis of a project's stated duration and budget, whether the project is still on schedule and whether the above-mentioned tasks are still executed in an appropriate sequence. For instance, once the systelogical problem has been described, one should be able to determine whether it is worthwhile to continue a project. This control task can be executed on the basis of the projected Return On Investment (ROI) period, i.e. the estimated total investment divided by estimated annual savings of an ESS. The longer the ROI period of an ESS, the less worthwhile it is to continue the design, realization and implementation of the ESS. We note that we will leave the issue of control tasks unaddressed.
References in chapter 4


Breuker, J. and Wielinga, B., "Techniques for Knowledge Elicitation and Analysis", Report 1.5 in ESPRIT project 12, University of Amsterdam, Amsterdam, 1984.


5

FIRST CASE: CONFIGURING INSPECTION REGIMES

5.1 Introduction

One of KEMA’s functions in the Dutch electricity supply sector is to advise power utilities how to maintain fossil fuel power plants. Two consultants, employed by the Mechanical and Metallurgical Testing department (WMK) fulfil this function. They are primarily concerned with advising power utilities how to maintain steam pipes and steam headers.

In the past decade, the technical lives of many power plants have enlarged. Power utilities are urged to inspect the conditions of these power plants and to determine whether life extension is feasible. In addition to creep, fatigue is more and more considered a serious life determinant of steam-containing components in fossil fuel power plants. When this development persists, WMK may receive more requests for consultancy.

In 1987 the Joint Research Centre (JRC) of the Commission of the European Communities in Petten informed KEMA of its intention to launch a project to develop an “ES for forecasting the residual life of structural components and planning optimum maintenance actions”. The project was intended to address pressurized equipment and vessels. Amongst other organizations in the international electricity supply sector, KEMA was interested in participating in the project. JRC arranged a meeting amongst possible participants. One of the consultants attended this meeting. A JRC representative explained that JRC would develop the ES according to the development methodology described by Waterman (1986). The consultants were invited to bring in their knowledge.

When KEMA was informed that commencement of the project would be postponed, it withdrew and launched an internal project with similar objectives. We were invited to design an ESS which could support the consultants in executing their tasks.

This chapter is structured as follows. In separate sections we will describe the systeological, infological, datalogical and technological problems. In the final section we will evaluate whether the discrepancies identified have been eliminated.
5.2 Systelogical problem

*Actual situation*

Advising a power utility on how to maintain a component involves executing the following tasks, see the task precedence structure in figure 5.1. The consultants start by making an inventory of the critical parts comprising a component. They receive from a power utility representative the information they need to execute this task. For each part consultant A calculates its usage factor. If consultant B finds that a calculated usage factor is critical he may decide to acquire additional information about a part's condition. In such a case he places it on an inspection list and subsequently he inspects it on location. Finally, consultant B determines when it should be inspected again.

The consultants handle about twelve requests for advice per year, see table 5.1. This table contains mainly numbers which we could trace. The numbers in brackets represent the real amount of requests handled by the consultants. The time it takes them to handle a request has increased in 1987 by 13% and in 1988 by 83%. It can be expected that the number of requests which WMK accepts per year remains the same, but that the volume of a request will increase. If WMK continues to handle all requests accepted and to employ the two consultants to fulfil the function, the time efficiency of the consultants has to increase.

The consultants estimate that it takes two weeks to make an inventory of critical parts. This estimate should be accompanied by two observations. First, they handle several requests at once. The task is executed discretely. Second, the information they receive from power utility representatives is not always complete. Sometimes the consultants need additional information. Assuming that they handle two requests at once, we divided their estimate by two and came up with 80 hours per request.

Consultant B cannot estimate how much time it takes him make an inventory of comparable parts. As he does not frequently execute this task, we decided not to determine the time.

<table>
<thead>
<tr>
<th>Hours</th>
<th>1986</th>
<th>1987</th>
<th>1988</th>
</tr>
</thead>
<tbody>
<tr>
<td>Consultancy</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Requests</td>
<td>5 (11)</td>
<td>10 (13)</td>
<td>9 (12)</td>
</tr>
<tr>
<td>Hours/Request</td>
<td>53</td>
<td>60</td>
<td>113</td>
</tr>
</tbody>
</table>

Table 5.1 Hours spent on consultancy in 1986, 1987 and 1988.
Figure 5.1 Task precedence structure: inspecting a component.
Consultant B estimates that it takes him two days, i.e. sixteen hours, to inspect parts on location. The task has a rather physical nature. Therefore, we take over his estimate.

The consultants cannot estimate the time it takes them to calculate a part's usage factor, to configure an inspection list and to configure an inspection regime. We carried out an experiment to establish these times, see Bloot (1990).

We constructed the following experimental setup. Consultant A calculates a usage factor of ten parts in three sessions. The first session concerns four parts, in which he takes into account that a part suffers from creep. The second session concerns four parts, in which he takes into account that a part suffers from fatigue. The final session concerns two parts, in which he takes into account that a part suffers from both phenomena. The information which consultant A needs to execute this task concerns artificial pressure and temperature design values for creep calculations and artificial cycles for fatigue calculations.

In a fourth session, consultant B determines for each part whether it should be placed on an inspection list. The information which consultant B needs to execute the task concerns a part's usage factor and artificial maintenance history. In a final session, consultant B determines for each part when it should be inspected again. The information which he needs to execute the task concerns a part's usage factor, artificial maintenance history and artificial inspection results, provided that it has been placed on the inspection list.

We recorded the times when a consultant started and when he finished executing a task. From these times we derived the average time it takes the consultants to execute a task. The results are summarized in table 5.2. In this table $\Sigma t$ represents the time it takes to execute a task accumulated from all parts taken into consideration. $N$ represents the number of times a task is executed.

<table>
<thead>
<tr>
<th>Task</th>
<th>Calculate Usage Factor</th>
<th>Configure Inspection List</th>
<th>Configure Inspection Regime</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\Sigma t$</td>
<td>356.00</td>
<td>72.00</td>
<td>44.00</td>
</tr>
<tr>
<td>N</td>
<td>12</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>Min</td>
<td>14.00</td>
<td>2.00</td>
<td>3.00</td>
</tr>
<tr>
<td>Max</td>
<td>51.00</td>
<td>12.00</td>
<td>7.00</td>
</tr>
<tr>
<td>$\mu^*$</td>
<td>29.67</td>
<td>7.20</td>
<td>4.40</td>
</tr>
<tr>
<td>$\sigma$</td>
<td>129.15</td>
<td>8.62</td>
<td>1.82</td>
</tr>
</tbody>
</table>

Table 5.2 Times established by experimentation.
<table>
<thead>
<tr>
<th>Task</th>
<th>Hours/request</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Make Inventory of Parts</td>
<td>80</td>
<td>63</td>
</tr>
<tr>
<td>Calculate Usage Factor</td>
<td>25</td>
<td>20</td>
</tr>
<tr>
<td>Configure Inspection List</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>Inspect Part</td>
<td>16</td>
<td>13</td>
</tr>
<tr>
<td>Configure Inspection Regime</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Total</td>
<td>126</td>
<td>100</td>
</tr>
</tbody>
</table>

Table 5.3 Absolute and relative amount of time needed to execute a task.

In 1988 a request concerned about eight components. For each component the consultants considered two or three parts. Table 5.3 shows the time required by each task in the handling a request. We note that the total amount of 126 hours is thirteen hours more than 113 hours as pointed out in table 5.1. A possible explanation for this difference can be that since 1989 consultant A has been taking into account that a part can also suffer from fatigue. Another explanation can be that the estimate of the time it takes the consultants to make an inventory of critical parts is not accurate.

Regarding the fatigue calculations, it struck us that consultant A used a logarithmic graph instead of a formula to determine the number of cycles until crack initiation. The former option takes less time but affects the accuracy of the number of cycles until crack initiation. We determined that this error ranges between 10% and 25%.

**New situation**

A possible scenario is that the amount of consultancy which WMK accepts per year will increase by 100%. If no measures are taken, the consultants will need more than 1000 hours extra to fulfil their function.

The consultants spend 63% of their time making inventories of critical parts. They spend an additional 20% of their time calculating usage factors. We conclude that a required efficiency increase should primarily concern the execution of these two tasks. As to the former task we propose to reduce the amount of component information which is exchanged between the consultants and representatives of power utilities and which is not subject to change.
5.3 Infological problem

*Actual situation*

The task precedence structure in figure 5.1 represents the way the consultants advise a power utility with respect to the maintenance of a component.

The consultants start by making an inventory of the critical parts in a component. They execute this task in a fairly systematic way, i.e. on the rule-based level. In the design drawings of a component the consultants specially pay attention to bends, welds and openings. For each part identified they record design values such as the material of which it is constructed, its geometry and values of pressures and temperatures it can withstand. The consultants also record a part’s maintenance history, i.e. inspections and repairs carried out in the past. At a client’s request they can take into account a part’s operational history, i.e. pressures and temperatures recorded during its operation, and compress it into a *collective*. Further on in this section we shall discuss this issue in more detail.

For each part consultant A calculates a usage factor. He executes this task on the rule-based level. Consultant A deals with *creep* and *fatigue* separately.

Creep is the time-dependent plastic deformation in the micro structure of a material activated by high temperatures and stresses. It usually emerges during stationary operation of a unit. The models published by the Dutch regulatory authority "Stoomwezen" are applied to calculate a part’s usage factor. It is the quotient of its number of operational hours and its expected life expressed in number of hours until crack initiation.

Consultant A calculates a part’s expected life for creep in three steps. First he calculates the strength reduction coefficients of each of its openings. The smallest coefficient is considered representative of the part. Subsequently, he calculates the stress upon its walls. Finally, he determines its expected life by interpolation in a creep strength table related to its constituent material. Consultant A monitors himself by executing each calculation twice.

At a client’s request consultant A bases his calculations on design values, maximum recorded operational values or all recorded operational values. In the latter case a part’s operational history should be compressed into a *collective*. Regarding creep, a collective is a set of pressure-temperature classes in which each class contains the time that the part has been in operation under the conditions defined by that class. For each class consultant A follows the latter two steps described to calculate a usage factor fraction. He calculates a part’s usage factor by adding all fractions.
Fatigue is caused by stress ranges which originate from internal pressure changes and temperature differences over a part's wall. One can distinguish between low-cycle fatigue and high-cycle fatigue. Low-cycle fatigue is the macro-plastic deformation of the micro structure of a material caused by a relatively low number of large stress ranges. High-cycle fatigue is the micro-plastic deformation of the micro structure of a material caused by a relatively large number of small stress ranges. The models published by the German regulatory authority "Deutschen Dampfkessel Ausschuß" (DDA) are applied to calculate a part's usage factor. Its usage factor is the quotient of its number of operational cycles and its expected life expressed in number of cycles until crack initiation.

Consultant A calculates a part's expected life for fatigue in three steps. First, he defines one or more cycle classes which correspond with cold, warm and hot starts and stops. Second, for each cycle class he calculates a part's stress ranges on the basis of its warming up and cooling down velocities. Finally, he calculates for each cycle class the number of cycles until crack initiation. Again consultant A monitors himself by executing each calculation twice.

At a client's request consultant A bases his calculations on design values or all recorded operational values. In the latter case a part's operational history should be compressed into a collective. Regarding fatigue, a collective is a set of stress range-determining temperature classes in which each class contains the number of cycles that the part has completed under the conditions characterizing that class. For each temperature class consultant A in the operational history counts instances of the cycle classes defined in the first step. For each temperature class and for each cycle class he follows the latter two steps described to calculate a usage factor fraction. For each cycle class he calculates a part's usage factor by adding all fractions in all temperature classes all fractions.

If consultant B feels that he should acquire additional information about a part's condition, he places it on an inspection list. In executing this task he takes into account a part's usage factor and maintenance history. He may also try to recollect information about comparable parts operating under analogue conditions in other units or power plants, during which process he recalls that something was wrong with these parts. Consultant B executes this task on the skill-based level.

Inspecting a part on location involves identifying surface cracks near the critical areas of a part. For instance, near welds consultant B distinguishes between a finely grained and a coarsely grained zone. In these zones he qualifies the micro structure of a part's constituent material. Creep emerges at the edge of the finely grained zone and the basic material. Micro structures are classified according to a classification ranging from class 0, i.e. undamaged, to class 5, i.e. complete crack. Cracks can also be caused by manufacturing errors.
To decide when a part should be inspected again, consultant B takes into account a part's usage factor, maintenance history, inspection results and a component's economic life. He executes this task on the skill-based level. However, he explained how he can execute it on the rule-based level, see the task precedence structure in figure 5.2. He classifies a part's condition as critical, less critical or not critical. With a part's condition in mind he determines whether it should be inspected again within one, two or four years and records this advice in an inspection regime.

To classify a part's condition, consultant B proceeds as follows. He classifies the condition of each area on the basis of crack information and micro structure information. Of these two classifications he selects the most critical one. Subsequently he determines of all area classifications the most critical area classification. He also classifies a part's condition on the basis of its usage factor. Of this classification and of the most critical area classification he then selects the most critical one.

Figure 5.2 Task precedence structure: determining a part's next inspection.
A model of the RS of the tasks is represented in figure 5.3. The major entity type is a fossil fuel power plant. Most plants consist of two, three or four units. In most units the consultants are concerned with a boiler and a steam pipe system. In these systems, the components with which the consultants are concerned are steam pipes and steam headers. They divide a component into critical parts which may consist of one or more critical areas such as a weld, a group of openings and a separate opening. Consultant B takes into account when and to what extent welds and openings have been inspected and repaired when it comes to placing a part on an inspection list. A part can be placed on one or more inspection lists whereas an inspection list can contain one or more parts. Similarly, a part can be inspected more than once whereas an inspection regime can concern one or more parts.

Figure 5.3 Model of the RS of the tasks.
New situation

We call the intended ESS "Xpection", an integration of "eXpert" and "insPECTION". The support structure of Xpection is shown in figure 5.4. This structure represents the functions which are embedded in the second version of Xpection. In the following discussion of this support structure we depart from the assumption that Xpection is provided with a database that can contain information about the RS of the tasks which the consultants execute.

The support structure of the task of making an inventory of critical parts contains functions that support the consultants in editing, storing, retrieving and deleting information about plants, units, systems, components, parts, welds, openings, inspections and repairs. It also contains functions that support consultant A in compressing a part's operational history into collectives.

![Diagram of the Xpection structure]

Figure 5.4 Support structure of Xpection.

72
The task of calculating a part's usage factor is executed by consultant A on the rule-based level. Xpection contains functions that can take over the execution of this task. Xpection also contains functions to support consultant A in editing, storing, retrieving and deleting tables which contain material properties such as creep strengths, thermal conductivities and linear expansions.

The task of configuring an inspection list is executed by consultant B on the skill-based level. Xpection contains functions to support him in editing, storing, retrieving, reporting and deleting information about inspection lists. The edit function supports him in specifying whether a part should be inspected on location. This is achieved by retrieving comparable parts and inferring on the basis of the consultant's knowledge whether a part should be inspected on location.

The task of configuring an inspection regime is executed by consultant B on the skill-based level as well. Xpection contains functions to support consultant B in editing, storing, retrieving, reporting and deleting information about inspection regimes. The edit function supports him in specifying when a part should be inspected again. This is also achieved by inferring on the basis of the consultant's knowledge when a part should be inspected again.

5.4 Datalogical problem

In this section we are primarily concerned with designing a new situation. There is little need to improve how the consultants execute their tasks. However, consultant A considers "fatigue" usage factors which are calculated according to the rainfall method to be more accurate than the ones which are calculated according to the less laborious method which he currently follows. Therefore Xpection will calculate "fatigue" usage factors according to the rainfall method.

Xpection's database is designed as a set of relations such as component, part, weld, combined opening, opening, creep table, fatigue table, welding inspection, welding repair, opening inspection, opening repair, inspection list, inspection regime, part in list and part in regime, see e.g. Bloot (1990). The operations which can be carried out on these relations are fairly simple i.e., projection and restriction. For instance, parts present in the database can be retrieved by consecutively specifying the plant, unit, system and component they are associated with. In relational algebra this sequence of operations may be represented as:

\[
\begin{align*}
\text{Plant}[\text{Plant_ID}], \\
\text{Unit}[\text{Plant_ID} \rightarrow \text{P}] [\text{Unit_ID}], \\
\text{System}[\text{Plant_ID} \rightarrow \text{P}, \text{Unit_ID} \rightarrow \text{U}] [\text{System_ID}], \\
\text{Component}[\text{Plant_ID} \rightarrow \text{P}, \ldots, \text{System_ID} \rightarrow \text{S}] [\text{Component_ID}], \\
\text{Part}[\text{Plant_ID} \rightarrow \text{P}, \ldots, \text{Component_ID} \rightarrow \text{C}] [\text{Part_ID}] 
\end{align*}
\]
Xpection can support consultant B in configuring an inspection list and an inspection regime by executing descriptions of how he executes these tasks. Here we consider the design of the function of editing an inspection regime.

In an interview, which we carried out by introspection, consultant B explained how he classifies a part's condition and how he determines when a part should be inspected again. Of this interview we consider the following chunk:

<table>
<thead>
<tr>
<th>Chunk</th>
<th>Interview 3, No. 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Begin</td>
<td>Page 2 &quot;Yes, that is correct ...&quot;</td>
</tr>
<tr>
<td>End</td>
<td>Page 3 &quot;... you can immediately say it is class 3.&quot;</td>
</tr>
<tr>
<td>Contents</td>
<td>Classification of a part's condition using a part's usage factor and crack qualification.</td>
</tr>
<tr>
<td>Trace</td>
<td>usage factor 0.6, 0.8, 0.9, ..., crack, stable, non-stable, ..., critical, less critical, critical, ..., grinding, repair, class 1, 2, 3.</td>
</tr>
</tbody>
</table>

The two interview fragments in this chunk which we will discuss describe how consultant B classifies an area's condition on the basis of crack information and how he classifies a part's condition on the basis of its usage factor. We note that a condition is classified as either critical (1), less critical (2) or not critical (3).

**Fragment 1:** "So you have an inspection. Then there are two possibilities, you have found a crack or not. If you have found a crack, you determine whether it is a stable one or not. Then what action do you take? You grind it and if necessary you repair it. So you remove the crack anyway. When you have found a stable crack and you can remove it by grinding only, I would say that it is not critical and I advise to inspect again in four years or not at all, at least when the cause of this crack has been removed. So I put it in class 3. But if the crack was caused by operation or if it was an old one, which has grown since the last inspection, then I would say it is critical. Then I put it in class 1".

From this fragment we derived two rules describing how consultant B classifies a part's condition on the basis of crack information:

if Area.Inspection.CrackDetected
and Area.Inspection.CrackStable
and Area.Inspection.CrackGround
and Area.Inspection.WallThicknessOK
then Area.Class = 3

if Area.Inspection.CrackDetected
and not Area.Inspection.CrackStable
then Area.Class = 1

74
Fragment 2: "When the usage factor of a part reaches the value of 0.6 and nothing has been found with inspection, then I put it in class 3 if it is a simple part and in class 2 if it is a complex part. But when the usage factor reaches the value of 0.8 or even 0.9 it will be put in class 2 anyway".

From this fragment we derived three rules describing how consultant B classifies a part's condition on the basis of its usage factor:

\[
\begin{align*}
\text{if } & \text{ Part.UsageFactor} < 0.6 \\
\text{and } & \text{ Part.StrengthReductionCoefficient} = 1 \text{ (Not a complex part)} \\
\text{then } & \text{ Part.Class} = 3
\end{align*}
\]

\[
\begin{align*}
\text{if } & \text{ Part.UsageFactor} < 0.6 \\
\text{and } & \text{ Part.StrengthReductionCoefficient} < 1 \text{ (Complex part)} \\
\text{then } & \text{ Part.Class} = 2
\end{align*}
\]

\[
\begin{align*}
\text{if } & \text{ Part.UsageFactor} > 0.6 \\
\text{and } & \text{ Part.UsageFactor} < 0.8 \\
\text{then } & \text{ Part.Class} = 2
\end{align*}
\]

We also derived four rules describing how consultant B determines when a part should be inspected again on the basis of its condition:

\[
\begin{align*}
\text{if } & \text{ Part.Class} = 1 \\
\text{then } & \text{ PartInRegime.InspectAgain} = 1 \text{ year}
\end{align*}
\]

\[
\begin{align*}
\text{if } & \text{ Part.Class} = 2 \\
\text{then } & \text{ PartInRegime.InspectAgain} = 2 \text{ year}
\end{align*}
\]

\[
\begin{align*}
\text{if } & \text{ Part.Class} = 3 \\
\text{then } & \text{ PartInRegime.InspectAgain} = 4 \text{ year}
\end{align*}
\]

\[
\begin{align*}
\text{if } & \text{ Part.Class} = 4 \\
\text{then } & \text{ PartInRegime.InspectAgain} = \text{No}
\end{align*}
\]

We ultimately derived 26 rules which we integrated into an algorithm which Xpection can execute to determine when a part should be inspected again, see Weghorst (1988) and Van Rikxoort (1990).
5.5 Technological problem

Xpection can be characterized by flexible user-interaction, fast database access, fast calculations and simple inferences. We implemented Xpection on a micro-computer (Intel 8088 8 MHz processor, 20 Mb hard disk, color EGA). Although ESSs are often developed by means of a DSS generator or an ES shell, we programmed Xpection in Turbo Pascal 5.0, extended with the Turbo Pascal Database Toolbox, both manufactured by Borland International.

The user interface of Xpection consists of a menu and a fixed window. The menu is adapted to the sequence in which the consultants execute their tasks and supports them in selecting one of Xpection’s main functions.

To support communication between the consultants and Xpection when making an inventory of critical parts, we designed the task interface shown in figure 5.5. At their request Xpection generates a list of all plants present in its database. The consultants can scroll through this list and select the plant on which they want to retrieve more information. Xpection then generates a list of all units in the plant selected. To make an inventory of parts, the consultants can continue until they have selected a component. Xpection then generates a list of all parts in the component selected.

In Van Weelder and Sol (1990) we discuss a task interface that supports communication between consultant A and Xpection when compressing a part’s operational history into a collective.

![Diagram](image)

Figure 5.5 Task interface for making inventory of critical parts.
To support communication between consultant A and Xpection when calculating a part’s "fatigue" usage factor, we designed the task interface depicted in figure 5.6. At his request Xpection can calculate a part’s usage factor on the basis of its operational history according to the rainfall method.

The task interface consists of a form in which consultant A can identify which part is under consideration and a window which contains the information relevant to the calculation. This window consists of a form in which consultant A can identify which opening is under consideration, i.e. nozzle 1 44, of which material it is constructed and which period in the operational history of the opening is under consideration. It also consists of a table in which consultant A can identify the results of the calculation. Regarding the columns in this table, $N_r$ stands for the identification of a cycle type defined. $P_{min}$, $P_{max}$, $T_{min}$ and $T_{max}$ are respectively the minimum and maximum pressures and temperatures in a cycle type, $\theta_p$ and $\theta_m$ represent the temperature ranges over the part’s wall during start or stop respectively of the unit to which it belongs, $n$ stands for the number of occurrences of a cycle type, $dP = P_{max} - P_{min}$. $T_{det}$ is the determining temperature in a cycle type, $d\theta = |\theta_p| + |\theta_m|$, $u$ represents the usage factor fraction for a cycle type and finally $n \cdot u$ the total usage factor fraction for a cycle type. Finally, the window consists of a form in which consultant A can identify the part’s usage factor over the period under consideration as well as over all periods.

<table>
<thead>
<tr>
<th>Collective</th>
<th>Table</th>
<th>Calculation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Plant : Plant 2</td>
<td>Component : Steam header 17</td>
<td></td>
</tr>
<tr>
<td>Unit : Unit 2</td>
<td>Part : C1202</td>
<td></td>
</tr>
<tr>
<td>System : Boiler</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Opening : Nozzle 1 44</th>
<th>Period: 80-01-01</th>
</tr>
</thead>
<tbody>
<tr>
<td>Material : 13 CrMo 44</td>
<td></td>
</tr>
<tr>
<td>90-01-01</td>
<td></td>
</tr>
<tr>
<td>Id</td>
<td>Pmin</td>
</tr>
<tr>
<td>----</td>
<td>------</td>
</tr>
<tr>
<td>1</td>
<td>5.00</td>
</tr>
<tr>
<td>2</td>
<td>0.00</td>
</tr>
<tr>
<td>3</td>
<td>0.00</td>
</tr>
</tbody>
</table>

Usage factor over this period: 0.0396
Usage factor over all periods: 0.0396

Figure 5.6 Task interface for calculating "fatigue" usage factor.
To support communication between consultant B and Xpection when configuring an inspection list, we designed the task interface shown in figure 5.7. At his request Xpection can iteratively retrieve from its database the parts in a component and infer whether they should be inspected on location. Xpection supports consultant B in specifying his decision by offering him the possibility either to acknowledge its decision or to overrule it.

The task interface consists of a form in which consultant B can identify which part is under consideration and a window which contains information relevant to the task. This window consists of two smaller windows and a menu. The left window contains a form in which consultant B can identify the information about a part which he considers relevant. It also contains a list of two possible decisions that can be made in executing the task, in which the decision made by Xpection is highlighted. The right window consists of a form in which consultant B can identify Xpection's arguments in making its decision. The menu supports consultant B in retrieving a part's geometry and maintenance history as well as information about comparable parts.

To support communication between consultant B and Xpection when configuring an inspection regime, we designed the task interface depicted in figure 5.8. At his request Xpection can iteratively retrieve from its database the parts in a component and infer when they should be inspected again. Xpection supports consultant B in specifying his decision by offering him the possibility either to acknowledge its decision or to overrule it.
The task interface consists of a form in which consultant B can identify which part is under consideration and a window which contains the information relevant to the task. This window consists of two smaller windows. The left window contains a form in which consultant B can identify the information about a part which he considers relevant. It also contains a list of four possible decisions that can be made in executing the task, in which the decision which is made by Xpection is highlighted. The right window consists of list of a part’s critical areas. Consultant B can scroll through this list to identify Xpection’s arguments in making its decision and to identify Xpection’s warnings indicating e.g. whether a part should have been repaired while no repair has been specified.

Three versions of Xpection, i.e. its code as well as the data structures and the structure charts which are the blueprints of this code, are documented in De Pater and Van Walsum (1988), Meyberg and Rennings (1988), Amesz and Noordzij (1989), Van Heijningen et al. (1989) and Van der Hoeven and Put (1990). The code represents some 30,000 statements.
A net effort of 30 man-months was spent on designing Xpection over a period of two years by the consultants, one graduate student and eleven under graduate students of Delft University of Technology, two under graduate students of University of Twente and ourselves. The numbers of man-months spent on addressing the systelogical, infological, datalogical and technological problems amount to four, four, six, and sixteen respectively. The latter number may be interpreted as twelve because four man-months were spent on transforming Xpection 1.0 into a more fashionable Xpection 2.0. For a discussion of Xpection 1.0 we refer to Van Weelderen and Sol (1988).

5.6 Discussion

To evaluate whether Xpection is a means to improve the performance of the consultants we will determine the time it takes the consultants to execute their tasks when supported by Xpection and compare these times with those determined in section 5.2.

Xpection can support the consultants in retrieving and reporting information about the parts in a component. The information which will continue to be exchanged between the consultants and representatives of power utilities are operational histories. Xpection can support the consultants in compressing operational histories into collectives. Therefore it can be argued that the time it takes the consultants to execute the task can be reduced by 50%. We can conclude that the consultants will spend 80*0.50 = 40 hours per request on making an inventory of critical parts.

We carried out an experiment similar to the one carried out in section 5.2 to establish the time it takes the consultants to calculate a usage factor, to configure an inspection list and to configure an inspection regime, see Bloot (1990). Prior to carrying out the experiment, we gave the consultants the opportunity to "find their way about" in Xpection in order to avoid that the results would be biased due to a learning effect. The results are summarized in table 5.5. Table 5.6 shows the time required by each task in handling a request.

We carried out a t-test to determine whether the differences are due to coincidence or may be credited to Xpection's support.

It takes consultant A 1.33 minutes instead of 29.67 minutes to calculate a usage factor. Here \( t = (\mu_o - \mu_r)/\sqrt{(\sigma_o^2 + \sigma_r^2)} = 2.29 \). This value implies a 97.86% confidence level that the observed difference is no coincidence but can be credited to Xpection's support. Per request this difference implies a \((29.67 - 1.33) \times 50 = 23.6\) hours' saving. The achieved efficiency improvement is \((\mu_o/\mu_r) \times 100\% = 2231\%\).
It takes consultant B 1.20 minutes instead of 7.20 minutes to determine whether a part should be inspected on location. Here $t = 2.02$. This value implies a 95.45% confidence level that the observed difference can be credited to Xpection's support. Per request this difference implies a 2.5 hours' saving. The achieved efficiency improvement is 600%.

It takes consultant B 1.30 minutes instead of 4.40 minutes to determine when a part should be inspected again. Here $t = 2.17$. This value implies a 97.22% confidence level that the observed difference can be credited to Xpection's support. Per request this difference implies a 1.3 hours' saving. The achieved efficiency improvement is 340%.

Summarizing, we observed that Xpection can support the consultants in handling twice the amount of work they currently do. We also observed that Xpection can support consultant B in calculating a usage factor more accurately.

Table 5.6 Absolute and relative amount of time needed to execute each task.
REFERENCES IN CHAPTER 5

References in chapter 5


SECOND CASE: DESIGNING TEST CIRCUITS

6.1 Introduction

Electricity is transported and distributed via electricity networks. Circuit breakers are crucial elements in such networks. Their function is to switch on and switch off parts of a network under both normal and disturbed conditions. High-voltage equipment is required to have a high degree of reliability. A network failure can have a great impact on our daily lives. Consumers of circuit breakers, such as electricity distribution utilities, purchase circuit breakers only when tests have shown that they have a high degree of reliability. Therefore, manufacturers of circuit breakers test prototypes extensively before taking them into production.

One of KEMA's functions in the international electricity supply sector is to test whether high-voltage equipment can withstand extremely high short-circuit currents. In KEMA's high-voltage laboratory, the "De Zoeten Laboratory" (DZL), such tests are carried out. Testing a circuit breaker involves incorporating it into an appropriate test circuit, switching it on at the moment a short-circuit current is being generated, timing how long it takes the circuit breaker to switch off the short-circuit current and observing whether it has been damaged. At DZL four "observers" are concerned with the testing of high-voltage equipment.

In 1987 we were introduced to the manager who is responsible for maintaining the information system of DZL. One element of this information system is an information system for managing circuit designs. He informed us that the observers had set up a task force for exploring new ways to improve the quality of DZL's services to its clients. One of the suggestions of this task force is to improve the quality of test circuits by automating the current information system for managing circuit designs.

Providing the observers with a computer-based information system for managing circuit designs will not necessarily solve the problem situation. We considered this problem situation an interesting case for testing MEDESS. We offered to design an ESS which could support the observers in executing their tasks. DZL accepted this offer on condition that this ESS would contain a computer-based information system for managing circuit designs.

As in the previous chapter we will describe the systological, infological, datalogical and technological problems. In the final section we will evaluate whether the discrepancies identified have been eliminated.
6.2 Systelogical problem

Actual situation

Testing circuit breakers for their capability to withstand extremely high short-circuit currents involves executing the following tasks, see the task precedence structure in figure 6.1. Preparing the tests requested in an order placed by a client involves designing associated circuits. It also involves designing a test plan, i.e. a specification of the sequence in which the tests should be carried out. Carrying out a test involves implementing its associated circuit, generating a short-circuit current, determining whether the circuit shows an appropriate behaviour, determining whether the circuit breaker operates properly and observing whether it has been damaged in any way.

Figure 6.1 Task precedence structure: testing a circuit breaker.

DZL tests an average of 120 circuit breakers per year, see table 6.1. Most circuit breakers tested are High-Voltage (HV) circuit breakers. Most shifts concern the testing of Medium-Voltage (MV) equipment. In the near future DZL will take over the testing of MV equipment from another KEMA laboratory. In this case we will focus on the testing of MV circuit breakers.

A client wants to carry out as many tests as possible in the shifts he has purchased. DZL wants to make sure that the time it takes to carry out the requested tests does not exceed the time reserved for the purchased shifts. In this context we put forward the following observations.

When a client has modified the design of a prototype that failed to operate properly in a previous test, he can decide to test it again. It may happen that this test is coordinated by another observer than the one who coordinated the first test and that this second observer designs a different circuit. As a result, the conditions under which the modified prototype is tested can differ and a client may be annoyed.

When "shooting" a circuit, a circuit's behaviour may not correspond with the behaviour specified in its design. This discrepancy may be due to an error generated during the design of the circuit. An observer then designs a new circuit and implements it. This causes a delay in the duration of the test. Additional costs due to such a delay are for DZL's account, not for that of its client. In designing a new circuit, an observer cannot communicate with a client.

When a circuit breaker does not withstand its rated short-circuit current, a client can request an observer to reduce the rated short-circuit current of a circuit breaker. In that case an observer should design a new circuit for each test which should be carried out on the basis of the new ratings. Additional costs as a consequence of such a delay are for the client's account not for that of DZL.
SYSTELOGICAL PROBLEM

We recognize three issues which may affect the quality of DZL's services to its customers, i.e. the time it takes to design a circuit, the errors made in this process and the degree of consensus between the observers on how a circuit should be designed. We carried out an experiment to establish their values.

We constructed the following experimental setup, see Jansen (1990). We selected four frequently tested MV circuit breakers. These are rated 15 kV - 31.5 kA, 24 kV - 25 kA, 36 kV - 25 kA and 40.5 kV - 31.5 kA. We also selected five frequently applied IEC duties according to which the circuit breakers have been tested. These are Basic 1, Basic 2, Basic 3, Basic 4 & 5 and Single Phase. In this way we configured a sample of 20 tests. For each test each observer designs an associated circuit. The tests are handed to him one by one in random order. This way an observer does not have to concern himself with determining an optimum sequence in which the tests should be carried out. We recorded the times when an observer started and when he finished executing a task. From these times we derived the average time it takes the observers to execute a task. We also recorded whether an observer designs (d) a new circuit or copies (c) or modifies (m) an existing circuit retrieved from DZL's files.

The results are summarized in table 6.2. In this table $\Sigma t$ represents the time it takes to execute the task accumulated over all items. N represents the number of times a task is executed. It struck us that observers 1 and 3 in some cases retrieved a circuit from DZL’ files for copying or modification purposes.

A way to specify the errors an observer makes in designing a circuit is to compare that circuit with an "objective" standard. Buchanan and Shortliffe (1984) and Nieuwenhuis (1988) propose to use a gold standard. However, we cannot rely on an independent expert who can design an objective standard. To acquire 20 gold standard circuits, we configured four different pairs of observers, with each observer participating in two pairs. Each pair designed five of the 20 gold standard circuits.

<table>
<thead>
<tr>
<th>Aggr. over</th>
<th>$\Sigma t$</th>
<th>$\mu$</th>
<th>$\sigma^2$</th>
<th>min</th>
<th>max</th>
<th>N</th>
<th>d</th>
<th>c</th>
<th>m</th>
</tr>
</thead>
<tbody>
<tr>
<td>Observer 1</td>
<td>206.0</td>
<td>10.3</td>
<td>15.1</td>
<td>4.0</td>
<td>20.0</td>
<td>20</td>
<td>13</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>Observer 2</td>
<td>180.0</td>
<td>9.0</td>
<td>8.1</td>
<td>4.0</td>
<td>15.0</td>
<td>20</td>
<td>20</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Observer 3</td>
<td>141.0</td>
<td>7.0</td>
<td>6.4</td>
<td>3.0</td>
<td>14.0</td>
<td>20</td>
<td>10</td>
<td>3</td>
<td>7</td>
</tr>
<tr>
<td>Observer 4</td>
<td>156.0</td>
<td>7.8</td>
<td>4.9</td>
<td>5.0</td>
<td>13.0</td>
<td>20</td>
<td>20</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Basic 1</td>
<td>144.0</td>
<td>9.0</td>
<td>10.2</td>
<td>6.0</td>
<td>15.0</td>
<td>16</td>
<td>12</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>Basic 2</td>
<td>135.0</td>
<td>8.4</td>
<td>8.7</td>
<td>4.0</td>
<td>15.0</td>
<td>16</td>
<td>14</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>Basic 3</td>
<td>156.0</td>
<td>9.7</td>
<td>18.2</td>
<td>5.0</td>
<td>20.0</td>
<td>16</td>
<td>12</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Basic 4/5</td>
<td>128.0</td>
<td>8.0</td>
<td>6.4</td>
<td>3.0</td>
<td>11.0</td>
<td>16</td>
<td>12</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Single Phase</td>
<td>120.0</td>
<td>7.5</td>
<td>4.9</td>
<td>4.0</td>
<td>12.0</td>
<td>16</td>
<td>13</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

| Total      | 683.0    | 8.5  | 9.8     | 3.0 | 20.0 | 80 | 63 | 6 | 11 |

Table 6.2 Times established by experimentation.

86
Designing a circuit for testing a MV circuit breaker involves filling in the contents of 68 squares. This number corresponds with the number of squares on a circuit card, such as the one depicted in figure 6.2, which an observer can fill in. For each square we record whether its contents correspond with the contents of the associated gold standard square.

If associated squares are both filled in (f/f), we determine whether their contents correspond (=). If their contents do not correspond, we determine whether the procedures to derive the contents correspond (= inf.). If these procedures correspond, we determine whether the fact that the contents of the fields do not correspond is due to an error or due to a continuation of a corresponding procedure on the basis of wrong information (error).

The results are summarized in table 6.3. All columns give percentages, except the final one. This column gives sample sizes. It struck us that the cards on which the circuit and the gold standard circuit are recorded all contain squares which have been filled in (f) but also squares which have been left blank (~f).

We separately consider the squares filled in to specify a circuit's supply and the ones which are filled in to specify a circuit's Transient Recovery Voltage (TRV) control, see table 6.4 and table 6.5.

As to designing a circuit's supply, the contents of 85.3% of the squares have been filled in and can be compared. Multiplying the percentages in the f/f columns by 100/85.3, we can conclude that in 2% of the cases the observers make mistakes and lapses. In 14% of the cases the observers make slips and do not recover them. In 84% of the cases the observers do not make errors and when they do, they recover them.

As to designing a circuit's TRV control, the contents of only 24.5% of the squares have been filled in and can be compared. Multiplying the percentages in the f/f columns by 100/24.5, we can conclude that in 11% of the cases the observers make mistakes and lapses. In 18% of the cases the observers make slips and do not recover them. In 71% of the cases the observers do not make errors and when they do, they recover them. Unlike the conclusion we drew with regard to designing a circuit's supply, this conclusion should be considered highly speculative.

A possible way to specify the degree of consensus between the observers as to how a circuit should be designed is to carry out a mutual comparison of the contents of all corresponding squares. Such an experiment can only be carried out under artificial conditions. The observers may not make errors and they may not neglect to fill in the contents of a square.
## Systeologisch PROBLEM

**Figure 6.2 Circuit card.**

<table>
<thead>
<tr>
<th>Circuit no.</th>
<th>Spec. data</th>
<th>B.O. type</th>
<th>No. ap.</th>
<th>kV</th>
<th>Num. stand.</th>
<th>st</th>
<th>Fase factor</th>
<th>L.</th>
<th>Schakelpool</th>
<th>Zelfindicateur</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Transformer schakeling**

<table>
<thead>
<tr>
<th>Spec. data</th>
<th>Transformer</th>
<th>B.O. type</th>
<th>No. ap.</th>
<th>kV</th>
<th>Num. stand.</th>
<th>st</th>
<th>Fase factor</th>
<th>L.</th>
<th>Schakelpool</th>
<th>Zelfindicateur</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Series-condensor C1** (of)

<table>
<thead>
<tr>
<th>Spec. data</th>
<th>Transformer</th>
<th>B.O. type</th>
<th>No. ap.</th>
<th>kV</th>
<th>Num. stand.</th>
<th>st</th>
<th>Fase factor</th>
<th>L.</th>
<th>Schakelpool</th>
<th>Zelfindicateur</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Onderstroom 1/2** (of)

<table>
<thead>
<tr>
<th>Spec. data</th>
<th>Transformer</th>
<th>B.O. type</th>
<th>No. ap.</th>
<th>kV</th>
<th>Num. stand.</th>
<th>st</th>
<th>Fase factor</th>
<th>L.</th>
<th>Schakelpool</th>
<th>Zelfindicateur</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Onderstroom 1/2** (of)

<table>
<thead>
<tr>
<th>Spec. data</th>
<th>Transformer</th>
<th>B.O. type</th>
<th>No. ap.</th>
<th>kV</th>
<th>Num. stand.</th>
<th>st</th>
<th>Fase factor</th>
<th>L.</th>
<th>Schakelpool</th>
<th>Zelfindicateur</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Par. / Serie ind.** (of)

<table>
<thead>
<tr>
<th>Spec. data</th>
<th>Transformer</th>
<th>B.O. type</th>
<th>No. ap.</th>
<th>kV</th>
<th>Num. stand.</th>
<th>st</th>
<th>Fase factor</th>
<th>L.</th>
<th>Schakelpool</th>
<th>Zelfindicateur</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Par. / Serie cap.** (of)

<table>
<thead>
<tr>
<th>Spec. data</th>
<th>Transformer</th>
<th>B.O. type</th>
<th>No. ap.</th>
<th>kV</th>
<th>Num. stand.</th>
<th>st</th>
<th>Fase factor</th>
<th>L.</th>
<th>Schakelpool</th>
<th>Zelfindicateur</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Balansvoorstand** (of)

<table>
<thead>
<tr>
<th>Spec. data</th>
<th>Transformer</th>
<th>B.O. type</th>
<th>No. ap.</th>
<th>kV</th>
<th>Num. stand.</th>
<th>st</th>
<th>Fase factor</th>
<th>L.</th>
<th>Schakelpool</th>
<th>Zelfindicateur</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Generatorschakeling**

<table>
<thead>
<tr>
<th>Spec. data</th>
<th>Transformer</th>
<th>B.O. type</th>
<th>No. ap.</th>
<th>kV</th>
<th>Num. stand.</th>
<th>st</th>
<th>Fase factor</th>
<th>L.</th>
<th>Schakelpool</th>
<th>Zelfindicateur</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

**Note:** The document contains a table and diagrams related to electrical circuit configurations and specifications. The text is in Dutch and includes various electrical parameters and component relationships. The layout is complex, with columns for different specifications such as circuit numbers, transformer types, kV ratings, and other electrical values. The table structure is horizontal and includes numeric and alphabetical data.
### Table 6.3 Errors established by experimentation (supply & TRV).

<table>
<thead>
<tr>
<th>Observer</th>
<th>$f/f$</th>
<th>$f/-\ell$</th>
<th>$-\ell/-\ell$</th>
<th>N</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.2</td>
<td>12.5</td>
<td>21.0</td>
<td>49.2</td>
</tr>
<tr>
<td>2</td>
<td>2.2</td>
<td>10.7</td>
<td>23.8</td>
<td>49.8</td>
</tr>
<tr>
<td>3</td>
<td>1.0</td>
<td>14.3</td>
<td>18.0</td>
<td>52.5</td>
</tr>
<tr>
<td>4</td>
<td>2.0</td>
<td>11.2</td>
<td>20.2</td>
<td>51.8</td>
</tr>
<tr>
<td>Basic 1</td>
<td>0.4</td>
<td>11.5</td>
<td>16.3</td>
<td>54.8</td>
</tr>
<tr>
<td>Basic 2</td>
<td>1.3</td>
<td>11.5</td>
<td>20.6</td>
<td>52.3</td>
</tr>
<tr>
<td>Basic 3</td>
<td>2.3</td>
<td>13.8</td>
<td>22.9</td>
<td>45.4</td>
</tr>
<tr>
<td>Basic 4/5</td>
<td>2.3</td>
<td>9.4</td>
<td>21.7</td>
<td>52.1</td>
</tr>
<tr>
<td>Single Phase</td>
<td>1.7</td>
<td>14.8</td>
<td>22.3</td>
<td>49.6</td>
</tr>
<tr>
<td>Total</td>
<td>1.6</td>
<td>12.2</td>
<td>20.8</td>
<td>50.8</td>
</tr>
</tbody>
</table>

### Table 6.4 Errors established by experimentation (supply).

| Observer  | 3.4   | 4.9       | 4.9          | 13.7 | 28.0 | 45.1 | 760 |
| 2         | 2.2   | 3.4       | 13.6         | 7.5  | 23.9 | 49.3 | 760 |
| 3         | 2.9   | 5.8       | 6.7          | 8.0  | 24.7 | 51.8 | 760 |
| 4         | 2.0   | 3.3       | 9.3          | 6.4  | 31.6 | 47.4 | 760 |
| Basic 1   | 3.8   | 2.3       | 3.8          | 8.7  | 23.5 | 57.9 | 608 |
| Basic 2   | 2.6   | 5.6       | 5.1          | 11.6 | 29.8 | 42.2 | 608 |
| Basic 3   | 2.3   | 5.4       | 8.2          | 11.6 | 29.8 | 42.2 | 608 |
| Basic 4/5 | 2.0   | 4.1       | 12.0         | 10.5 | 28.9 | 42.2 | 608 |
| Single Phase | 2.5 | 4.3     | 14.0         | 5.9  | 31.1 | 42.3 | 608 |
| Total     | 2.6   | 4.3       | 8.6          | 8.9  | 27.1 | 48.4 | 3040 |

### Table 6.5 Errors established by experimentation (TRV).

89
**New situation**

A MV circuit breaker is often tested according to eight duties. The observers design a circuit for testing a MV circuit breaker more than 350 times a year. When we extrapolate this number over all equipment tested, the observers execute the task of designing a circuit more than 1750 times a year. As DZL expects to test more equipment in the near future, it can be fruitful to reduce the time it takes the observers to design a circuit. However, to improve the quality of DZL's services to its clients it is more important to reduce the amount of errors made in designing circuits.

**6.3 Informological problem**

**Actual situation**

The task precedence structure in figure 6.1 represents the way the observers test a circuit breaker.

An observer starts by preparing the tests requested by a client in an order. It involves designing associated circuits and determining the sequence in which they should be implemented. This sequence is recorded in a test plan. Carrying out a test involves implementing its associated circuit, generating a short-circuit current, determining whether the circuit shows an appropriate behaviour, determining whether the circuit breaker operates properly and observing whether it has been damaged in any way.

When an observer prepares a test he considers a circuit breaker's rated voltage, rated short-circuit current, rated TRV and number of poles. If a circuit breaker has one pole, he takes into account that pole's first pole to clear factor, number of interrupters and grading. He also considers the duty according to which a circuit breaker should be tested. A duty is a specification of a rated operating sequence at a certain percentage of the rated short-circuit current and of a rated TRV. Circuit breakers are often tested according to the duties provided by the International Electrotechnical Commission (IEC) and by the American National Standards Institute (ANSI). IEC duties concern the testing of equipment used in European networks with a frequency of 50 Hz, whereas ANSI duties concern the testing of equipment used in American networks with a frequency of 60 Hz.

Figure 6.4 represents a model of the RS of the task of preparing a test. A company can place an order and/or be the manufacturer of a circuit breaker. An observer handles an order which may concern one or more tests. A test concerns a circuit breaker and a duty. For a test an observer designs one or more associated circuits.
Figure 6.3 Circuit for testing a circuit breaker.

Figure 6.3 describes a 3-phase circuit. A circuit basically consists of a supply, a TRV control and a circuit breaker (CB). A circuit’s supply, which generates high voltages and short-circuit currents, consists of a generator section (G), a transformer section (T) and a reactance coil section (X). The generator section consists of one or more generators. The transformer section consists of one or two transformer groups. The reactance coil section consists of two or three reactance coils. A circuit’s TRV control (Rp, Rs and Cp/s) consists of one or more capacitors, zero, one or several resistance groups and one or several coils. Note that a test is focused on a circuit breaker’s first pole to clear. In the case of a 3-phase (of 120° each) test of a 3-pole circuit breaker, the two other poles remain conducting.

As indicated in the previous section, two observers do not always design a circuit from scratch, see the task coordination structure in figure 6.5. In some cases they retrieve from DZL’s files one or more circuits which were previously implemented to test the same circuit breaker or a comparable one according to the same duty (note the dotted line in the task precedence structure in figure 6.1.). They may copy or modify such a circuit.

In the remainder of the present discussion of this case we will direct our attention to the task of designing a circuit. Besides in preparing a test an observer may also execute the task of designing a circuit while carrying out a test, i.e. when a circuit shows an inappropriate behaviour or when a client requests him to alter the conditions under which a circuit breaker should be tested. Designing a circuit involves designing a circuit’s supply and TRV control, see the task coordination structure in figure 6.6.
Figure 6.4 Model of the RS of the task of preparing a test.
Figure 6.5 Task coordination structure: designing a circuit.

Regarding a circuit's supply, an observer starts by specifying its required behaviour in terms of its line voltage, phase to neutral voltage, short-circuit current, rate of exchange of current, impedance, mutual inductance and power. It involves calculating formulas. An observer executes this sub-task on the rule-based level.

To configure a circuit's supply, an observer first considers its transformer section. He determines with which transformer connection the transformer(s) can best be provided and looks up the associated transformer ratio in a table. He also determines whether the circuit's earthing should be located in the neutral of the transformer(s) or in the short-circuit point behind the circuit breaker. An observer executes this sub-task mostly on the skill-based level.

Subsequently, an observer considers a circuit's generator section. He calculates the voltage and short-circuit current which should be generated by this section and its impedance. To meet these requirements, he determines how many generators the section needs and which generator connection the generator(s) can best be provided with. Finally, he calculates for safety reasons the maximum short-circuit current which each generator may generate in case no load is incorporated into the circuit. An observer executes this sub-task on the skill-based and the rule-based level.
Figure 6.6 Task coordination structure: designing a new circuit.
The transformers with which DZL is equipped are organized in two groups, each consisting of three 14/72 kV transformers and one 14/12 kV transformer. Returning to the transformer section, an observer determines whether the required voltage transformation should be achieved by means of one or two transformer groups. He executes this sub-task on the skill-based level.

Finally, an observer considers the circuit's reactance coil section. The devices in a circuit have reactances when it is implemented. They effect that the short-circuit current with which a circuit breaker is tested is lower than the one generated by the generator section. DZL is equipped with three reactance coils. By using these coils an observer is able to correct this undesired effect. An observer calculates for each device in a circuit its reactance. The reactances of the coils are determined by these reactances. For each reactance coil he looks up in a table an appropriate coil connection. An observer executes this sub-task on the rule-based level.

Once an observer has determined the coil reactances he may assess whether the supply he designed will meet its requirements when the circuit is implemented. He determines the reactance which in reality is related to the selected coil connection and executes most sub-tasks in reverse order. When the ultimate and the initial line voltage differ too much, an observer may redesign the circuit's transformer and/or generator and/or coil section. The observers sometimes are reluctant to redesign a circuit's supply section because it involves much extra work.

So far we have discussed how an observer designs a circuit's supply. Before we discuss how he designs a circuit's TRV control, we will explain why it should be incorporated into a circuit.

During the time interval between the moment that a circuit breaker breaks a short-circuit current in a real network and the moment that the network's voltage recovers at the poles of the circuit breaker to a steady state voltage, this voltage shows transient behaviour. This behaviour is referred to as the network's TRV. It can influence the interrupting performance of a circuit breaker. The rated TRV of a circuit breaker constitutes the limit of the prospective TRV of circuits which a circuit breaker is capable of breaking in the event of a short-circuit in a real network. A TRV can be represented by a wave form. A circuit's TRV control should simulate the TRV which can originate when the circuit breaker breaks a short-circuit current in a real network.

Regarding a circuit's TRV control, an observer starts by specifying the characteristics of the circuit breaker's rated TRV which it should simulate. This behaviour is expressed in terms of a TRV peak voltage (Uc), the time to reach Uc, the associated TRV frequency and the time delay before the TRV originates after the circuit breaker has broken a short-circuit current. An observer also specifies the amplitude
factor between $U_c$ and the steady state voltage which the circuit breaker's first pole to clear should break. It involves calculating formulas. An observer executes this sub-task on the rule-based level.

When a circuit, which is correct on mathematical grounds, is implemented, it frequently happens that it cannot properly simulate a TRV. This phenomenon, which is inherent to the DZL installation, is called depression. The amplitude of the voltage generated by the generators initially is less than the required amplitude of the steady state voltage. As a consequence, $U_c$ cannot be generated. To correct this undesired behaviour of the DZL installation an observer increases the amplitude factor in such a way that $U_c$ is generated. The required behaviour is complemented by a calculation of the impedance and capacity of the circuit's TRV control. An observer executes this sub-task on the skill-based and the rule-based level.

To configure a circuit's TRV control, an observer determines whether resistors should be incorporated and whether they should be connected serially to or in parallel with its capacitors. He calculates the required resistances and capacities of these devices. He executes this sub-task on the skill-based and the rule-based level.

An observer may assess whether the TRV control will meet its requirements when the circuit is implemented, e.g. when the capacity of the circuit's transformer section turns out to be larger than the circuit's total capacity or when the specified resistances cannot be realized with the available resistors. He then respectively increases the circuit's total capacity or selects a set of resistors with resistances approximating the ones specified and executes most sub-tasks in reverse order. If the initial and ultimate $U_c$ differ too much, an observer may redesign the circuit and start by reconsidering the circuit's TRV control or even its supply. The observers sometimes are reluctant to redesign a circuit's TRV control because it involves much extra work.

**New situation**

A slight modification has been made in the task precedence structure and task coordination structures discussed. As to correcting the depression from which a circuit's supply suffers, the observers multiply the amplitude factor with a depression factor, i.e. the reciprocal of a depression factor percentage which may range between 60% and 75%. One of the reasons why they re-specify a circuit's supply after having specified its TRV control is that they do not take into account the desired depression factor percentage in specifying the number of generators needed to generate the required $U_c$. The observers specify the desired depression factor percentage as part of the required TRV behaviour. Consequently, it sometimes happens that the specified number of generators results in too much depression. Once the observers recognized this rather inefficient way of working, they agreed to execute the sub-task of specifying the desired depression factor percentage prior to the task of specifying a circuit's transformer section and generator section.
We call the intended ESS "OSS", an acronym of "Observer Support System". Figure 6.7 gives the support structure of OSS. This structure represents the functions embedded in the second version of OSS. In the following discussion of this support structure we depart from the assumption that OSS is provided with a database that can contain information about the RS of the tasks which the observers execute.

Figure 6.7 Support structure of OSS.
INFOLOGICAL PROBLEM

OSS contains functions that support the observers in editing, storing, retrieving and deleting information about companies, orders, circuit breakers, tests and circuits, see Zwijnenberg (1989b). We will discuss the function of editing a circuit in more detail.

The task of designing a circuit's supply is executed by the observers on the skill-based and the rule-based level. OSS contains functions that execute calculations and that support an observer in specifying the transformer, generator and reactance coil sections of a circuit's supply both in forward and in backward mode.

The task of designing a circuit's TRV control is executed by the observers on the skill-based and the rule-based level as well. OSS contains functions that execute calculations and that support an observer in specifying a TRV circuit both in forward and in backward mode.

Regarding both tasks, the execution of a sub-task which does not involve the calculation of one or more formulas, but rather a skill-based choice, is supported as follows. Before an observer executes such a task, OSS automatically and iteratively executes the sub-tasks which an observer executes to design a circuit's supply and/or TRV control. OSS proceeds until it comes up with an alternative that yields an acceptable difference between a supply's rated voltage and a mathematically assessed voltage, or between a TRV control's rated Uc and a mathematically assessed Uc.

6.4 Datalogical problem

In this section we are primarily concerned with designing a new situation. There is little to improve in how the observers design a circuit.

The database of OSS is designed as a set of relations such as company, order, object, test and circuit. The operations which can be carried out on these relations are fairly simple, i.e. projection and restriction. For instance, circuits present in the database can be retrieved by consecutively specifying the order and the test they are associated with. In relational algebra this sequence of operations may be represented as:

Order[Order_ID],
Test[Order_ID = O?] [Test_ID],
Circuit[Order_ID = O?, Item_ID = I?] [Circuit_ID]

OSS can support an observer in designing a circuit by executing descriptions of how he executes this task. We will describe how OSS supports an observer in editing the transformer connection which the transformer(s) can best be provided with.

98
We recall that the input information of the task consists of one of the two possible generator connections (represented as * or ▲) with which the generators have been provided, the rated line voltage of the circuit’s supply and the desired load factor of the transformer(s). The output information of the task is one out of eight possible transformer connections and an associated transformer ratio.

In a self report session we identified that an observer executes this sub-task on the skill-based level. At our request he explained how he executes the task on the rule-based level. He looks up a transformer connection in a table via a generator connection entry and a transformer ratio entry. With each generator connection four transformer connections are associated and thus four transformer ratios. Expressed in Structured English the observer proceeds as follows:

```perl
get Transformer.Connection which is associated with the specified Generator.Connection and which has the lowest Transformer.Ratio;

while Transformer.Ratio < Supply.LineVoltage/(14*Transformer.Loadfactor) do
    get next Transformer.Connection which is associated with the specified Generator.Connection;
```

We will also describe how OSS supports an observer in editing whether resistors, which are incorporated into a circuit’s TRV control, should be connected serially to or in parallel with its capacitors. The input information of the task consists of the rated phase to neutral voltage of the circuit’s supply, the circuit capacity and the capacity of the transformer section.

We identified that an observer executes this sub-task on the skill-based level. At our request he again explained how he can execute the task on the rule-based level. He proceeds as follows:

```perl

if Supply.PhaseToNeutralVoltage <= 36 kV
    then TRVControl.Damping = parallel
else if TRVControl.ExtraCapacity <= 10% of TRVControl.TotalCapacity
    then TRVControl.Damping = parallel
    else TRVControl.Damping = serial;
```

We ultimately derived thirteen descriptions which OSS can execute to support an observer in specifying a skill-based choice.
6.5 Technological problem

OSS can be characterized by flexible user-interaction, fast database access, fast calculations and simple inferences. We implemented OSS on a micro-computer (Intel 80286 12 MHz processor, 40 Mb hard disk, color EGA). For reasons similar to the first case we programmed OSS in Turbo Pascal 5.0, extended with the Turbo Pascal Database toolbox. For a discussion of our arguments we refer to Van Weelderen and Sol (1989).

The user interface of OSS consists of a menu and a fixed window. The menu is adapted to the sequence in which the observers execute their tasks and supports them in selecting one of the main functions of OSS.

To support communication between the observers and OSS during the execution of the task of retrieving an order, a test and a circuit, we designed the task interface depicted in figure 6.8. At their request OSS generates a list of all companies present in its database. The observers can scroll through this list and select the company of which they want to retrieve an order. OSS then generates a list of all orders ever placed by that company. The observers can scroll through this list and select the order which they want to take into consideration. The task interface consists of a form in which the observers can identify which order is under consideration, which observer handled it and when this was done. Subsequently, OSS generates a list of all tests requested in the order selected. The observers can scroll through this list and select the test which they want to take into consideration. The observers can also identify from the form which test is taken into consideration. Next, OSS generates a list of all circuits designed to carry out the test selected. The observers can scroll through this list and select the circuit which they want to take into consideration.

To support communication between the observers and OSS during the execution of the task of designing a circuit we designed the task interface illustrated in figures 6.9, 6.10, 6.11 and 6.12.

The task interface consists of five forms. The four forms on the left concern a circuit's supply behaviour, transformer section, generator section and reactance coil section. The form on the right concerns a circuit's TRV-control. Except for the form associated with a circuit's transformer section, all forms contain two columns, i.e. "wish" and "calc". The values in the former column describe a circuit as it is specified in forward mode, whereas the values in the latter column describe a circuit as it is specified in backward mode.
Figure 6.8 Task interface for retrieving orders, tests and circuits.

Figure 6.9 Task interface for designing a circuit.
OSS can calculate a supply's required behaviour. The observers can identify this behaviour in the "wish" column of the "supply" form. We continue with the specification of the generator section. OSS can infer with how many generators the required supply should be generated. OSS can support an observer in specifying his decision by offering him the possibility either to acknowledge its decision or to overrule it.

Once an observer has determined the coil reactances, OSS can support him in assessing whether the supply he designed will meet its requirements when the circuit is implemented. OSS calculates the reactances which are in reality associated with the specified coil connections in reverse order, i.e. in backward mode. An observer can identify these values in a form which is added to the task interface shown in figure 6.10. Subsequently, OSS executes the remaining tasks in backward mode. An observer can then compare the ultimate line voltage with the initial one. At his request OSS can support him in re-executing the task of specifying a circuit's supply, note the F9 function key.

OSS can support the observers in executing the task of designing a circuit's TRV circuit in a similar fashion to the way it supports them in executing the task of designing a circuit's supply.

<table>
<thead>
<tr>
<th>TRV</th>
<th>wish</th>
<th>calc</th>
</tr>
</thead>
<tbody>
<tr>
<td>Uc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>t3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ud</td>
<td></td>
<td></td>
</tr>
<tr>
<td>L</td>
<td></td>
<td></td>
</tr>
<tr>
<td>P</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TRANSFORMERS</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connect</td>
<td>14/36</td>
</tr>
<tr>
<td>t</td>
<td>2.57 s</td>
</tr>
<tr>
<td>Earthing</td>
<td>Hall</td>
</tr>
</tbody>
</table>

| GENERATORS | |
| Connect    | 4 |
| Nr          | 2 |
| U           | 4.67 kV |
| Ito           | 61.71 kA |
| Ical         | 30.86 kA |

<table>
<thead>
<tr>
<th>REACTORS</th>
<th>wish</th>
<th>calc</th>
</tr>
</thead>
<tbody>
<tr>
<td>X/Group</td>
<td>44</td>
<td>46</td>
</tr>
<tr>
<td>HS-rail</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Transfer</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>17.5kV</td>
<td>28</td>
<td>30</td>
</tr>
<tr>
<td>Verz</td>
<td>95</td>
<td>95</td>
</tr>
<tr>
<td>Gen</td>
<td>22</td>
<td>22</td>
</tr>
<tr>
<td>Coil</td>
<td>25</td>
<td>25</td>
</tr>
<tr>
<td>Connect</td>
<td>910</td>
<td>910</td>
</tr>
</tbody>
</table>

12,00 kV 40,00 kA Basic 3

Circuit Design

Figure 6.10 Task interface for designing a circuit.
OSS can calculate the first part of the TRV’s required behaviour, i.e. Uc to Af(inh). Once OSS has supported an observer in specifying whether a TRV circuit should be provided with parallel or serial damp resistors, it can calculate the latter part of the TRV’s required behaviour, i.e. α(p/s) to Rs/p. The capacity of the circuit’s transformer section may turn out to be greater than the circuit’s total capacity, i.e. $C_{extra} = C_{total} - C_{trafo} < 0$. OSS then supports an observer in specifying an improved value of $C_{extra}$ by offering him the possibility to acknowledge its decision, i.e. either to increase the circuit’s total capacity by specifying that the TRV circuit does not have to be provided with an extra capacitor ($C_{extra} = 0$) or to overrule it by specifying the capacity of the capacitor with which the circuit’s TRV circuit should be provided ($C_{extra} > 0$). Note the dotted field in the “calc” column of the “TRV” form.

Once an observer has determined whether a circuit’s TRV control should be provided with an extra capacitor, OSS can support him in assessing whether the TRV control he designed will meet its requirements when the circuit is implemented.

OSS calculates the resistances which are in reality associated with the selected resistors in reverse order. Subsequently, OSS executes the remaining tasks in backward mode. An observer can then compare the ultimate Uc with the initial one. At his request OSS can support him in re-executing the task of specifying a circuit’s supply or in re-executing the task of specifying a circuit’s TRV-control. Note the F9 and F10 function keys in figure 6.12.
## Figure 6.12 Task interface for designing a circuit.

Two versions of OSS, i.e. its code as well as the data structures and the structure charts which are the blueprints of this code, are documented in Zwijnenberg (1989a) and Deurwaarder (1989). The code represents some 30,000 statements.

A net effort of 24 man-months was spent on designing OSS over a period of one and a half years by the observers, two graduate students of University of Twente, one under graduate student of Delft University of Technology and ourselves. The numbers of man-months spent on addressing the systelological, infological, datalogical and technological problems are three, three, five, and ten respectively.

### 6.6 Discussion

To determine whether OSS is a means to improve the performance of the observers, we will determine the time it takes them to design a circuit and the errors they make in this process when supported by OSS and compare these values with the ones determined in section 6.2.
Table 6.7 Times established by experimentation.

We carried out an experiment similar to the one carried out in section 6.2 to establish the time it takes the observers to design a circuit and the errors they make in this process. Prior to carrying out the experiment, we gave the observers the opportunity to "find their way about" in OSS in order to avoid that the results would be biased due to a learning effect. The results are summarized in table 6.7. None of the observers has retrieved and (partially) copied a circuit which was previously implemented to test the same circuit breaker or a comparable one according to the same duty and which is stored in the database of OSS. We can conclude that the overall efficiency improvement achieved is \((\mu_p/\mu_r)*100\% = 327\%\).

As to the times it takes the observers to design a circuit, we carried out a t-test to determine whether the observed differences are due to coincidence or may be credited to the support of OSS. The results are summarized in table 6.8. In this table "cl" stands for confidence level, "D" for descriptive and "P" for prescriptive.

<table>
<thead>
<tr>
<th>Aggr. over</th>
<th>(\mu_0)</th>
<th>(\sigma_0^2)</th>
<th>(\mu_p)</th>
<th>(\sigma_p^2)</th>
<th>(t)</th>
<th>cl</th>
</tr>
</thead>
<tbody>
<tr>
<td>Observer 1</td>
<td>10.3</td>
<td>15.1</td>
<td>1.5</td>
<td>1.4</td>
<td>2.17</td>
<td>97.22</td>
</tr>
<tr>
<td>Observer 2</td>
<td>9.0</td>
<td>8.1</td>
<td>2.7</td>
<td>2.6</td>
<td>1.93</td>
<td>94.26</td>
</tr>
<tr>
<td>Observer 3</td>
<td>7.0</td>
<td>6.4</td>
<td>3.2</td>
<td>3.2</td>
<td>1.36</td>
<td>83.85</td>
</tr>
<tr>
<td>Observer 4</td>
<td>7.8</td>
<td>4.9</td>
<td>3.4</td>
<td>10.9</td>
<td>1.11</td>
<td>72.87</td>
</tr>
<tr>
<td>Basic 1</td>
<td>9.0</td>
<td>10.2</td>
<td>3.0</td>
<td>6.4</td>
<td>1.47</td>
<td>86.64</td>
</tr>
<tr>
<td>Basic 2</td>
<td>8.4</td>
<td>8.7</td>
<td>2.0</td>
<td>1.1</td>
<td>2.04</td>
<td>95.45</td>
</tr>
<tr>
<td>Basic 3</td>
<td>9.7</td>
<td>18.2</td>
<td>2.9</td>
<td>9.7</td>
<td>1.29</td>
<td>80.64</td>
</tr>
<tr>
<td>Basic 4/5</td>
<td>8.0</td>
<td>6.4</td>
<td>2.6</td>
<td>3.6</td>
<td>1.71</td>
<td>91.09</td>
</tr>
<tr>
<td>Single Phase</td>
<td>7.5</td>
<td>4.9</td>
<td>2.6</td>
<td>3.5</td>
<td>1.69</td>
<td>91.09</td>
</tr>
<tr>
<td>Total</td>
<td>8.5</td>
<td>9.8</td>
<td>2.6</td>
<td>5.0</td>
<td>1.53</td>
<td>86.64</td>
</tr>
</tbody>
</table>

Table 6.8 Results of the t-test.
We can conclude that the decrease in the time it takes observer 1 and observer 2 to execute the task "circuit design" is no coincidence, but can be credited to the support of OSS. As to observer 3 and observer 4, the results are less convincing.

It should be noted that in the experiment the observers frequently made use of the option to redesign or optimize a circuit's supply and TRV control. As one of the observers stated: "It doesn't hurt anymore to work out a number of alternatives". This statement points out that an ESS should be provided with functions that enable an expert to experiment, i.e. to generate alternatives.

To determine the number of errors the observers make in designing a circuit with the support of OSS we followed an approach similar to the one followed in section 6.2. We note that with the support of OSS designing a circuit for testing a MV circuit breaker involves filling in the contents of 93 squares. We also note that circuits cannot contain squares which are left blank. The results are summarized in table 6.9.

From the results in table 6.9 we conclude that the observers execute the task without making slips. The contents of approximately 2% of all squares were derived at in a different way than the ones of the associated gold standard square. Such errors are lapses or mistakes. The results which concern the design of a circuit's TRV control are particularly noteworthy. In the first experiment it happened that the percentage of fields left blank exceeded 50%. This percentage was of course reduced to 0%. Considering the fourth and fifth column in tables 6.3, 6.4, 6.5, 6.9, 6.10 and 6.11, we can conclude that the overall achieved improvement of effectiveness is \((28.7+69.3)/(27.2+53.4)*100\% \approx 121\%\). For the supply section this improvement is 117% and for the TRV control section 136%.

Summarizing, we observed that OSS can support the observers in designing a circuit three times faster and with a lower error rate, i.e. 2.1% instead of 19.6%. It can be argued that the degree of consensus between the observers on how a circuit should be designed has been increased.
### Table 6.9 Errors established by experimentation (supply & TRV).

<table>
<thead>
<tr>
<th>Aggr. over</th>
<th>( f/f )</th>
<th>( N )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Observer 1</td>
<td>2.2</td>
<td>30.1</td>
</tr>
<tr>
<td>Observer 2</td>
<td>2.3</td>
<td>32.4</td>
</tr>
<tr>
<td>Observer 3</td>
<td>1.3</td>
<td>20.6</td>
</tr>
<tr>
<td>Observer 4</td>
<td>2.5</td>
<td>31.5</td>
</tr>
<tr>
<td>Basic 1</td>
<td>1.5</td>
<td>23.8</td>
</tr>
<tr>
<td>Basic 2</td>
<td>1.1</td>
<td>18.7</td>
</tr>
<tr>
<td>Basic 3</td>
<td>2.6</td>
<td>33.7</td>
</tr>
<tr>
<td>Basic 4/5</td>
<td>2.8</td>
<td>34.0</td>
</tr>
<tr>
<td>Single Phase</td>
<td>2.4</td>
<td>33.1</td>
</tr>
<tr>
<td>Total</td>
<td>2.1</td>
<td>28.7</td>
</tr>
</tbody>
</table>

### Table 6.10 Errors established by experimentation (supply).

<table>
<thead>
<tr>
<th>Aggr. over</th>
<th>( f/f )</th>
<th>( N )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Observer 1</td>
<td>1.4</td>
<td>19.9</td>
</tr>
<tr>
<td>Observer 2</td>
<td>1.7</td>
<td>22.5</td>
</tr>
<tr>
<td>Observer 3</td>
<td>1.0</td>
<td>15.1</td>
</tr>
<tr>
<td>Observer 4</td>
<td>1.8</td>
<td>24.7</td>
</tr>
<tr>
<td>Basic 1</td>
<td>1.5</td>
<td>22.4</td>
</tr>
<tr>
<td>Basic 2</td>
<td>1.6</td>
<td>20.3</td>
</tr>
<tr>
<td>Basic 3</td>
<td>2.0</td>
<td>28.1</td>
</tr>
<tr>
<td>Basic 4/5</td>
<td>1.5</td>
<td>19.2</td>
</tr>
<tr>
<td>Single Phase</td>
<td>0.7</td>
<td>12.6</td>
</tr>
<tr>
<td>Total</td>
<td>1.4</td>
<td>20.5</td>
</tr>
</tbody>
</table>

### Table 6.11 Errors established by experimentation (TRV).

<table>
<thead>
<tr>
<th>Aggr. over</th>
<th>( f/f )</th>
<th>( N )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Observer 1</td>
<td>3.2</td>
<td>42.4</td>
</tr>
<tr>
<td>Observer 2</td>
<td>3.0</td>
<td>44.6</td>
</tr>
<tr>
<td>Observer 3</td>
<td>1.8</td>
<td>27.4</td>
</tr>
<tr>
<td>Observer 4</td>
<td>3.3</td>
<td>39.6</td>
</tr>
<tr>
<td>Basic 1</td>
<td>1.5</td>
<td>25.4</td>
</tr>
<tr>
<td>Basic 2</td>
<td>0.6</td>
<td>16.7</td>
</tr>
<tr>
<td>Basic 3</td>
<td>3.4</td>
<td>40.5</td>
</tr>
<tr>
<td>Basic 4/5</td>
<td>4.3</td>
<td>51.9</td>
</tr>
<tr>
<td>Single Phase</td>
<td>4.3</td>
<td>58.0</td>
</tr>
<tr>
<td>Total</td>
<td>2.8</td>
<td>38.5</td>
</tr>
</tbody>
</table>
References in chapter 6

THIRD CASE: PLANNING INSPECTION TRIPS

7.1 Introduction

In Canada electrical equipment, i.e. industrial control equipment such as switches, business equipment such as micro-computers and household equipment such as washing machines, should meet requirements concerning their electrical safety which are set by the Canadian Standards Association (CSA).

KEMA is commissioned by CSA to evaluate according to CSA standards the suitability of the production facilities of all electrical products which are manufactured on the European continent and which are intended to be exported to Canada. Such an evaluation is called an Initial Factory Evaluation (IFE). When a product is certified, it is provided with the CSA monogram which guarantees its electrical safety. KEMA is also commissioned by CSA to protect the CSA monogram of a certified product by inspecting its electrical safety without prior notice to its manufacturer. Most products should be inspected twice a year. Some products should be inspected three, four or even six times a year. Both IFEs and inspections are carried out by a number of inspectors who are employed by the Foreign Inspections department (LTI).

In 1989 we were informed that the department was considering the purchase of a routing program. By means of a graphical user interface it can support its users in interactively specifying a route and in determining its distance and travelling time.

We arranged a meeting with the persons who were involved in the purchase of the program, i.e. the manager of the department and the inspection coordinator, called "planner" from now on. The planner is responsible for planning CSA inspection trips. The manager and the planner had a strong feeling that better plans could be made by using the routing program. They explained that the routing program could support the planner in planning inspections faster and in a more flexible way. They found it difficult to describe their problem in more detail.

Regarding the considerable amount of money involved in the purchase of the routing program, we argued that it could be fruitful to understand the problem situation prior to solving it. We considered this problem situation an interesting case for testing MEDESS and offered to design an ESS which could support the planner in executing his tasks. The manager and the planner accepted this offer and decided not to purchase the routing program.

As in the previous cases we will discuss the systological, infological, datalogical and technological problems. In the final section we will evaluate whether the discrepancies identified have been eliminated.
7.2 Systeological problem

Actual situation

Planning inspection trips involves executing three tasks, i.e. managing the IFE and inspection administration, making a new plan and reviewing a trip shortly before an inspector will make that trip.

The planner is supported by an inspection administrator and by a computer-based information system in managing the IFE and inspection administration. It involves carrying out mutations received from CSA and recording the findings of the inspectors who have returned from a trip. In this case we are primarily concerned with the latter two tasks.

Twice a year the planner makes a new plan. In December and in June he requests the inspection administrator to provide him with a list of IFEs and inspections which should be carried out over the following period of six months, i.e. from January to June or from July to December. Approximately 2000 IFEs and inspections should be carried out over such a period. The manager of the department put forward that this number increases by about 5% a year. IFEs and inspections are carried out on trips which the inspectors make through Europe. A trip can be considered a list of IFEs and inspections which an inspector should carry out over a period of two weeks in one of the three bi-monthly terms into which a plan is divided.

Making a new plan involves executing the following tasks, see the task precedence structure in figure 7.1. The planner starts by executing a preparation task which involves grouping the inspections to be carried out into "regular" or "high-frequency" calls, counting these calls and filling in these numbers in preprinted maps of Europe such as the one shown in figure 7.2. He continues by configuring as many trips as necessary to carry out all IFEs and inspections requested by CSA. Finally, the planner distributes the trips configured over the three terms in a plan.

Once the planner has made a new plan, he hands it over to the group managers of the department. In mutual arrangement with the inspectors in his group a group manager determines which inspector makes which trips and when.

Shortly before an inspector makes a trip, the planner reviews the composition of that trip. He takes into account the up-to-date status of the IFEs and inspections to be carried out. At his request the inspection administrator provides him with this information.
Figure 7.1 Task precedence structure: making a new plan.

It generally takes an inspector two weeks to make a trip. An inspector is responsible for planning the sequence in which he will carry out the IFEs and inspections assigned to his trip. When an inspector returns from a trip, he reports his findings via his group manager to the inspection administrator. These findings include e.g. which inspections he has carried out and the time it took him to do this. The inspection administrator records this information and informs the planner.

The manager of the department and the planner feel that it takes too much time to make plans and to review trips. The planner estimates that it takes him two weeks to make a new plan. However, this estimate may not be accurate because in June 1989 it took him five days to make a new plan. To gain a better understanding of this issue we carried out an experiment to establish more accurately the times it takes the planner to execute the preparation task and the tasks of configuring a trip and of distributing trips over a term.
Figure 7.2 Map of part of Europe as originally used by the planner.
We designed the following experimental setup, see Nijssse (1989). We requested the planner to make three, rather simple, plans. Such a plan concerned one region with relatively many inspection locations, i.e. Germany, and one region with relatively few inspection locations, i.e. Scandinavia. Each plan is made on a separate day. The sets of inspections which should be planned are random selections of 70% of the inspections which had to be planned in the second half of 1989.

We recorded the times it took the planner to execute the preparation task, to configure the trips necessary to carry out all inspections provided and to distribute all trips configured over the three terms in a plan. We also recorded the number of trips configured. The results are summarized in table 7.1. Note that it takes the planner more time to configure a trip in Scandinavia than in Germany.

The results may be considered rather flattering for the planner. However, it should be noted that the experiment was carried out in his spare time. He was able to make the plans without being interrupted. We may also expect flattering results with regard to the task of reviewing a trip. The planner estimates that it takes him less than half an hour to execute this task.

When we depart from the assumption that a new plan contains about 45 trips, i.e. the number of trips made in 1989, we may conclude that the problem situation cannot be characterized as one of low time-efficiency.

Regarding the effectiveness of a plan we identified that the department manager urges the planner to assign the IFEs and inspections to a minimum amount of trips and the inspectors to carry out IFEs and inspections with appropriate care. The inspectors urge the planner to assign a reasonable amount of calls to a trip in such a way that a trip can be made in two weeks without too much travelling overhead, i.e. "spare time driving".

<table>
<thead>
<tr>
<th>Region</th>
<th>Preparation</th>
<th>Configuration</th>
<th>Assignment</th>
<th>Total</th>
<th>Trips</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scan. 1</td>
<td>11</td>
<td>14</td>
<td>3</td>
<td>28</td>
<td>5</td>
</tr>
<tr>
<td>Scan. 2</td>
<td>14</td>
<td>22</td>
<td>4</td>
<td>40</td>
<td>5</td>
</tr>
<tr>
<td>Scan. 3</td>
<td>9</td>
<td>15</td>
<td>2</td>
<td>26</td>
<td>6</td>
</tr>
<tr>
<td>Scan. μ</td>
<td>11.3</td>
<td>17</td>
<td>3</td>
<td>31.3</td>
<td>5.3</td>
</tr>
<tr>
<td>Germ. 1</td>
<td>27</td>
<td>30</td>
<td>5</td>
<td>62</td>
<td>14</td>
</tr>
<tr>
<td>Germ. 2</td>
<td>28</td>
<td>29</td>
<td>4</td>
<td>61</td>
<td>15</td>
</tr>
<tr>
<td>Germ. 3</td>
<td>23</td>
<td>18</td>
<td>8</td>
<td>49</td>
<td>14</td>
</tr>
<tr>
<td>Germ. μ</td>
<td>26</td>
<td>25.7</td>
<td>5.7</td>
<td>57.3</td>
<td>14.3</td>
</tr>
<tr>
<td>Total μ</td>
<td>18.7</td>
<td>21.3</td>
<td>8.7</td>
<td>44.3</td>
<td>9.8</td>
</tr>
</tbody>
</table>

Table 7.1 Times established by experimentation.
The effectiveness of a plan can be related to the percentage of inspections planned carried out by the inspectors. In 1988 this percentage was 97%, which may be considered good.

The effectiveness of a plan can also be related to the number of trips needed to carry out all IFEs and inspections requested by CSA. Analysis of the reports of the inspections carried out in 1988 pointed out that 50% of the trips were planned well, that 40% of the trips took one or two days less to be carried out and that the rest of the trips took one or two days more to be carried out. Theoretically, the department might need one, two or even three trips less to carry out all the IFEs and inspections requested by CSA. From this point of view, the effectiveness of a plan is good as well.

However, it should be noted that when the planner makes a new plan, he cannot anticipate which inspector will make which trip. Depending on the willingness of the inspectors to drive in their spare time, a trip can be too empty, planned well or too full. The inspectors complain that if they try to carry out an inspection with appropriate care, it means they have to travel during a considerable part of their spare time. Their problem becomes even worse when one or more new inspections or inspections which could not be carried out on an earlier trip are added to their schedule.

To gain a better understanding of this issue we requested three experienced inspectors to assess the plans which the planner made in the experiment. Nijssen (1989) concludes on the basis of these assessments that the planner configures trips in low-density regions rather optimistically, i.e. too full. It should be noted that the setup of this experiment as well as its results are highly speculative. It struck us that the results have led to sharp discussions between the inspectors and the planner. In these discussions the planner put forward that he feels constrained by the division of Europe into fixed areas. It is difficult to revise this division and adapt it to the planner's latest insights due to the setup of the current computer-based information system for managing IFEs and inspections. The planner also put forward that he is often reluctant to consider the topography of an area while he configures a trip, because it takes him too much time to retrieve and consult detailed maps of an area. Furthermore, the planner stated that he is reluctant to configure trips by adding separate calls to a trip instead of adding all calls which are located in an area. He argues that it takes him too much time.

**New situation**

The department manager expects that the number of IFEs and inspections which CSA requests his department to carry out will continue to increase by about 5% a year. Therefore, it can be fruitful to reduce the time it takes the planner to execute the preparation task and the task of reviewing a trip.
Regarding the complaints of the inspectors, we identified three possibilities to fulfil their wishes. We proposed to assign an ESS to the tasks of supporting the planner in revising the division of Europe into areas, of providing him with the topography of an area and of supporting him in adding separate calls to a trip, all in an efficient and convenient way.

7.3 Infological problem

Actual situation

Planning inspection trips involves executing three tasks, i.e. managing the IFE and inspection administration, making a new plan and reviewing a trip shortly before an inspector will make that trip. In this case we are primarily concerned with the latter two tasks. The task precedence structure in figure 7.1 represents the way the planner makes a new plan.

The planner executes the first task on the rule-based level. At his request the inspection administrator provides the planner with a list of IFEs and inspections which are to be carried out in the following period of six months. The list is sorted per area and each area is sorted per manufacturer. The planner starts by grouping inspections into "regular" and "high-frequency" calls. A call is a visit to one manufacturer at one location during which one or more inspections are carried out. After all, it can occur that a manufacturer exports more than one type of product to Canada. A regular call, or R-call for short, contains inspections which are to be carried out only twice per year, whereas a high-frequency call, or HF-call for short, also contains one or more inspections which are to be carried three times per year or more. The planner has divided Europe into areas of equal size with respect to the number of calls they generally contain. For each area the planner counts the R-calls and HF-calls and fills in these numbers in preprinted maps of parts of Europe.

The planner continues by configuring as many trips as necessary to carry out all IFEs and inspections requested by CSA. Basically, he configures a trip by grouping the calls in (parts of) neighbouring areas, until the number of calls which should be carried out on that trip corresponds with a number suitable to the characteristics of the region and which represents two weeks of work. HF-calls which for some reason cannot be assigned to a trip are assigned to the S-trip. The S-trip is a kind of "parking" list of HF-calls which already have been assigned to one or more trips, but which still have to be assigned to one or more other trips. The planner often tries to combine IFEs which take about a week to carry out with one or more HF-calls in the S-trip.
Finally, the planner distributes the trips configured over the three bi-monthly terms in a plan. In his view, an optimum distribution of trips in the first six months is 20%, 40%, 40% and in the second six months this is 0%, 50%, 50%. These distributions relate to the non-optimum road conditions in winter and the non-optimum presence of personnel in, or closure of, factories in summer. The planner executes this task on the skill-based and the rule-based level.

Figure 7.3 shows a graphical representation of the RS of the task of planning inspection trips. A plan can consist of one or more trips. CSA can request KEMA to carry out one or more IFEs and one or more inspections at a manufacturer’s production facility. A call can consist of one or more inspections. An inspection can be assigned to one or more trips and during one trip one or more inspections can be carried out. An IFE can be assigned to one trip and in a trip one IFE can be carried out. Europe consists of one or more countries, cities, roads, lakes and fairways.

Figure 7.3 Representation of the RS of the task of planning inspection trips.
We will continue by discussing the tasks of configuring a trip and of reviewing one. The planner executes the first task within the context of making a new plan. It should be noted that when making a plan, the planner works concentrically from the outer regions of Europe to Arnhem, where KEMA is situated. For example, when the planner considers Southern Europe, he separately considers the smaller regions (1) Portugal and Spain, (2) Malta, Southern Italy and Central Italy and when these two regions are planned, (3) Northern Spain, Northern Italy and Southern France.

To compose a new trip, the planner proceeds as follows, see the task coordination structure in figure 7.4. The planner executes the sub-tasks in this structure mostly on the skill-based level. It should be noted that he spends most of his time retrieving and getting an overview of the information needed.

The planner starts by defining a new trip, assigning a code to it and determining a trip minimum and a trip maximum between which two values the number of calls to be carried out on that trip may range.

The planner tries to identify an appropriate area, i.e. an area that meets the requirement that the sum of the number of calls which the planner has already assigned to the trip and the number of calls to be carried out in the selected area is smaller than the pre-specified trip minimum.

When the planner has identified an appropriate area, he assigns to the trip the calls which should be carried out in this area. When the planner cannot identify an appropriate area, he selects a less appropriate one in which too many calls are located. He may split the area up into two new areas in such a way that at least one area becomes an appropriate one. The planner assigns the calls which should be carried out in this area to the trip. Although the planner feels that he can configure trips that meet the requirements of the inspectors more accurately, he often is reluctant to execute this part of the task because it generates extra administrative work. He then settles for a less appropriate area. HF-calls which are assigned to the trip are also assigned to the S-trip. The planner repeats the execution of this part of the task when the trip minimum has not yet been achieved and when he finds that the trip should contain more inspections.

Otherwise he continues by determining whether the S-trip contains any HF-calls to be carried out during the trips he has configured in the neighbourhood of the "current" trip. If so, he selects such a call and assigns it to the trip. He repeats the execution of this part of the task until the S-trip does not contain another interesting call or until the pre-specified trip maximum is achieved.

The planner proceeds by determining whether the S-trip contains any HF-calls to be carried out along the approach route to the destined areas of the trip. If so, he selects such a call and assigns it to the trip. Again he repeats the execution of this part of the task until the S-trip does not contain another interesting call or until the pre-specified trip maximum is achieved.
Figure 7.4 Task coordination structure: making a new plan.
To review a trip the planner proceeds as follows, see the task coordination structure in figure 7.5. The planner executes the sub-tasks in this structure mostly on the skill-based level. It should be noted that he spends most of his time retrieving and getting an overview of the information needed.

The planner determines whether he should remove calls from the trip. He then deals with calls containing inspections which CSA has requested not to carry out anymore or of which CSA has requested to decrease their frequency.

The planner continues by determining whether he should assign extra calls to the trip. He then deals with calls containing new inspections which CSA has requested to carry out or calls of which CSA has requested to increase their frequency or which have not been carried out during trips already made.

Subsequently, the planner estimates whether the trip involves a workload which can be handled in two weeks by the inspector who is assigned to the trip. He estimates the time it takes an inspector to travel between the inspection locations as well as the times it generally takes to carry out the inspections. To make the first estimate, the planner sometimes takes into account a detailed description of the topography of the areas in which the trip will be made. However, most of the time he feels reluctant to do this for reasons we have already put forward. To make the latter estimate, he takes into account his own experience or of the other inspectors as recorded by the inspection administrator.

If the workload of a trip is too low the planner examines whether another trip still has to be made in the neighbourhood of the one reviewed and if so, whether that trip involves a workload which cannot be handled in two weeks by the inspector assigned to it. In such a case, the planner removes one or more calls from that trip and assigns them to the one reviewed. This process may be repeated until the workload of the trip reviewed is right. Otherwise the planner may have to consider modifying part of the plan, i.e. the composition of trips.

If the workload of a trip is too high the planner examines whether another trip still has to be made in the neighbourhood of the one reviewed and if so, whether that trip involves a workload which can easily be handled in two weeks by the inspector assigned to it. In such a case, the planner removes one or more calls from the trip reviewed and assigns them to the other trip. This process may be repeated until the workload of the trip reviewed is right. Otherwise the planner may again have to consider modifying part of the plan.
Figure 7.5 Task coordination structure: reviewing a trip.
New situation

The tasks the planner and inspection administrator execute concern the processing of information about practically the same RS. We designed two support structures representing the blueprints of two applications sharing one database containing information about areas, manufacturers, IFEs and inspections, plans and trips.

The first support structure supports the inspection administrator in retrieving, editing, storing and deleting information about manufacturers, IFEs, inspections, trips and plans. It also supports him in generating overviews of the database. In this case we are primarily concerned with the second support structure.

The second support structure contains functions that support the planner in executing his tasks. We call the intended ESS "IPESS", an acronym of "Inspection Planning Expert Support System". The support structure of IPESS is represented in figure 7.6. This structure represents the functions embedded in the second version of IPESS. In the following discussion of this support structure we depart from the assumption that the database can contain information about the RS of the tasks which the planner executes.

The support structure of IPESS contains functions that support the planner in editing, storing, retrieving and deleting information about plans, trips, areas, inspections, IFEs and calls. It also contains functions that support the planner in retrieving information about the topography of Europe.

The planner executes the preparation task on the rule-based level. To take over the execution of this task IPESS contains functions that can automatically group inspections into R-calls and HF-calls, count the R-calls and HF-calls located in each area and present these numbers to the planner.

The planner executes the tasks of configuring a trip and of reviewing a trip on the skill-based level. IPESS contains functions that can support the planner in splitting an area up into two smaller ones or in joining two areas to create a larger one. As this function involves storing areas in and deleting areas from the database it resides under the function area management. IPESS also contains functions that support the planner in specifying which topographical information, e.g. cities, roads, lakes and countries, he wants to take into consideration and in retrieving the specified information. In addition, it contains functions that support the planner in specifying which region of Europe he wants to take into consideration. These functions reside under the general function topography management. Furthermore, IPESS contains functions that support the planner in selecting one or more calls and in assigning or removing one or more calls to or from a trip. These functions reside under the general function trip configuration.

The planner executes the task of distributing trips over terms on the skill-based level. IPESS contains functions that support the planner in specifying during which term a trip should be made.

121
Figure 7.6 Support structure of IPESS.

### 7.4 Datalogical problem

Unlike in the previous two cases, we distinguish between an actual and a new situation. The inflexible setup of the current computer-based information system for managing the IFE and inspection administration is claimed to be one of the major reasons why the planner cannot configure trips as well as he would like to. We do not intend to improve how the planner executes his tasks. However, as IPESS will take over the execution of the preparation task, a description of how the planner executes this task should be incorporated into IPESS’ datalogical design.

**Actual situation**

The database of the current inspection management system is organized as a sorted sequence of records. Each record is called a "file". The unique identification of a file is provided by CSA and is called a "file number". Each file relates to a specific inspection.
The database has been developed on the basis of a data model which consists of 77 attributes describing e.g. the submitter of the inspection, the type of product to which the inspection relates, the manufacturer of the product, the area in which the manufacturer is located and the type of inspection involved. It does not correspond with the RS shown in figure 7.3. For instance, one of the attributes of a file is denoted "area code", suggesting that there exists a relationship between an area and an inspection, a manufacturer or a submitter. The existence of this relationship is one of the reasons why the planner is reluctant to split areas up into smaller ones or to shape them in a way that allows a more convenient way of configuring a trip.

New situation

The database of IPESS is designed as a set of relations such as Inspection, Trip and TripContainsInspection:

- Inspection(Inspection_ID, ..., Manufacturer, Frequency, Average Time)
- Trip(Plan_ID, Trip_ID, ..., period_ID, inspector_ID)
- TripContainsInspection(Plan_ID, Trip_ID, Inspection_ID)

This set of relations also consists of relations representing topographical information such as City, AreaBorderSegment, CountryBorderSegment and HighwaySegment:

- City(Name, Location)
- AreaBorderSegment(Area_ID, FirstLocation, SecondLocation)
- CountryBorderSegment(FirstLocation, SecondLocation)
- HighwaySegment(FirstLocation, SecondLocation)

IPESS can group inspections into calls and can count the R-calls and HF-calls located in each area. Therefore, it can execute an algorithm which we designed and which shows correspondence with how the planner executes these tasks, see Klettersteeg et al. (1989). In Structured English this algorithm may be represented as follows:

```plaintext
Execute until End-Of-File(InspectionFile)
  Get Next Inspection;
  if NewLocation(CallFile, Inspection.Location) then Create Call;
      Call.Location := Inspection.Location;
      Call.Frequency := Inspection.Frequency;
      Add Call to CallFile;
  else Get Call where Call.Location = Inspection.Location;
      if Call.Frequency < Inspection.Frequency then Call.Frequency := Inspection.Frequency;
```

123
Execute until End-Of-File(CallFile)
  Get Next Call;
  Get Area where Area.ID = AreaContainsCall(Call.Location);
  Add 1 to Area.Calls;
  if Call.Frequency > 2
    then Add 1 to Area.HF-Calls
  else Add 1 to Area.R-Calls;

It should be noted that we designed - out of curiosity - a function that can automatically configure a trip. A description of a limited part of the planner's expertise on these issues and a discussion of how IPESS can execute the tasks on the basis of this expertise can be found in Hetharia (1990). One of the major conclusions which can be drawn from the results in this work is that the effort put into eliciting, formalizing and automating the execution of the task of configuring a trip does not pay back.

7.5 Technological problem

IPESS can be characterized by flexible and graphical user-interaction, fast database access and fast calculations. The current computer-based information system for managing the IFE and inspection administration is implemented on a mini-computer. We implemented the new system and IPESS on one micro-computer (Intel 80386 16MHz processor, 40 Mb hard disk, color VGA). We programmed both applications in Turbo Pascal 5.0, extended with the Turbo Pascal Database Toolbox.

The topographical information which the planner takes into account when configuring a trip can be found in an atlas. He suggested to fill the database with information recorded in a well-known atlas of the world. We selected the pages which describe the topography of the European continent. Of each page we made an A3-size black & white copy. Of each copy we produced a bit-image stored in a pixel file by means of a scanner. Out of each pixel file we produced topographical information which we stored in the database. To retrieve and display a pixel file on a screen we used the WordPerfect 5.0 text editor. To produce topographical information and to store it in the database we developed a memory-resident program in Turbo Pascal 5.0 which we could execute while working with the text editor. Using a mouse we can specify a city by clicking at a certain location and entering its name. We can specify a segment by entering the type of the segment, e.g. a country boarder, and by clicking at a first location and at a second location. The program transforms the screen coordinates of a location into the coordinates of the used atlas by means of a projection formula.
The planner can use a keyboard and a mouse to communicate with IPESS. Each task interface consists of four windows, see figure 7.7. The main window serves as a graphical interface. It enables the planner and IPESS to exchange topographical information. The window below the main window enables the planner and IPESS to exchange alpha-numerical information. The upper right window serves as a menu interface. IPESS, unlike Xpection and OSS, has one-dimensional menus instead of two-dimensional ones. Finally, the lower right window enables IPESS to present to the planner the identifications of the plan and the trip which the planner is working on and the number of calls which the planner has assigned to a trip.

The task interface in figure 7.7 is the one with which IPESS starts a session. It is an overview of Europe as it is divided into areas. The main menu of IPESS contains five items, viz. *plan, trip, term, area* and *screen*. Via this menu the planner can request IPESS to support him in retrieving, editing, deleting and printing a plan, a trip and a distribution of trips over the three terms in a plan. IPESS can also support the planner in managing areas and other topographical information.

![Figure 7.7 Task interface for supporting coordination.](image-url)
When the planner selects the *screen* function, he enters a sub-menu which contains three items, viz. *select*, *build* and *settings*. The *settings* function supports the planner in specifying which topographical information, e.g. cities, roads, fairways, lakes, trips, (all, assigned and non-assigned) IFEs and inspections, he wants to take into consideration. The *select* function supports the planner in specifying which part of Europe he wants to take into consideration. When the planner selects this function he enters another sub-menu through which he can request IPESS to support him in specifying whether he wants to select Europe or a part of Europe, i.e. by zooming in, note the rectangle in figure 7.8. The *build* function generates a topographical overview of the specified part of Europe according to the settings specified.

When the planner selects the trip function he enters a sub-menu which contains five items, viz. *create*, *delete*, *select*, *assign* and *remove*. The *assign* and *remove* functions support the planner in assigning and removing to or from the current trip an IFE or an inspection or all inspections located in an area.

![Figure 7.8 Task interface for configuring a trip.](image)

126
Figure 7.9 illustrates a situation in which the planner is working on a plan for the second half of 1990. He is configuring a trip to Denmark. He has already assigned the inspections located in two areas to the trip. They are grouped into 20 calls. He has decided to assign to the trip all inspections located in an additional area. He can use a mouse to select an area. Note the "mouse indicator" West of Denmark.

Figure 7.9 also illustrates how IPESS supports the planner in determining whether the annual frequency with which an inspection should be carried out is achieved. An inspection which meets this requirement is represented as an open dotted circle. An inspection which does not meet this requirement is represented as a filled circle.

When the planner selects the area function, he enters a sub-menu which contains the items split and join. The split function supports the planner in splitting an area up into two smaller ones, see figure 7.10. Once the planner has acknowledged that the specified division should be effected, IPESS deletes the old area from and stores the new ones in the database.
Two versions of IPESS, i.e. its code as well as the data structures and the structure charts which are the blueprints of this code, are documented in Kleersteeg et al. (1989), Kien et al. (1990) and Vriend (1990). The code represents some 25,000 statements.

A net effort of 24 man-months was spent on designing IPESS over a period of one and a half years by the planner, two graduate students and six under graduate students of Delft University of Technology, one under graduate student of University of Twente and ourselves. The numbers of man-months spent on addressing the systelological, infological, datalogical and technological problems are three, four, four, and thirteen respectively.

Figure 7.10 Task interface for splitting up an area.
7.6 Discussion

Here we shall discuss whether IPESS is a means to eliminate the discrepancies identified. We pointed out that there is hardly any reason to assume that the problem situation is one of low time-efficiency. Rather, the problem situation should be centered around the effectiveness of the plans made.

In an experiment we determined the time it takes the planner to execute the tasks of grouping inspections into calls and configuring a trip with the support of IPESS, to check whether the planner will at least maintain his current efficiency. The setup of the experiment is comparable to the one described in section 7.2. It takes the planner an average of 13 minutes to plan Scandinavia and an average of 41 minutes to plan Germany, see Hetharia (1990). These results point out that the planner’s time-efficiency can increase by 30% with the support of IPESS.

In the same experiment we observed that the planner is no longer reluctant to take topographical information into account when configuring a trip. Rather, he stated that he is "keen on assessing the time required to drive from highway exits over national roads to the production facilities", see the task interface shown in figure 7.11.

![Figure 7.11 Task interface for configuring a trip.](image)
In the experiment we also observed that the planner is no longer reluctant to split areas up into smaller ones when configuring a trip. He put forward that he intended to "revise areas more in line with highways". The planner put forward that IPESS can support him in reviewing a trip very well, because it "in notime can provide me with a graphical overview of the current status of the IFEs and inspections which should be carried out".

The planner also indicated an unexpected but very handy property of IPESS, i.e. that it can make a graphical hardcopy of a specified part of Europe. An inspector can be provided with such a hardcopy to plan the sequence in which he will carry out the inspections assigned to his trip.

Summarizing, we observed that IPESS can support the planner in planning inspection trips faster, in a more flexible way but also more effectively.

References in chapter 7


FOURTH CASE: DIAGNOSING TURBINE CONDITIONS

8.1 Introduction

In 1968 the first Dutch nuclear power plant, a 60 MWe boiling water reactor with natural circulation, was put into operation. The research plant was designed by General Electric Inc. in co-operation with KEMA. The reactor is equipped with a single turbine generator installation, simply referred to as turbine.

In 1988 the mechanical maintenance superintendent announced his intention to retire in 1989. He confronted the plant with a problem. His expertise had proven to be a valuable asset and it was considered a loss that it would no longer be available to the plant as a resource in fulfilling the function of mechanical maintenance.

We were informed that the plant intended to launch a project to preserve the superintendent’s expertise and to make it available to his successor by means of an ES. We considered the problem situation an interesting case for testing MEDESS, as the superintendent’s successor too is or soon will be an expert. We offered the plant to design an ESS which can support the superintendent’s successor in executing his tasks. The supervisor of the project accepted our offer.

As in the previous cases we will discuss the systelological, infological, datalogical and technological problems. In the final section we will evaluate whether the discrepancies identified have been eliminated.

8.2 Systelological problem

Actual situation

When we initiated the project the superintendent had less than one year to go until his retirement. We reduced the problem by focusing on a specific part of his function, i.e. turbine maintenance. The turbine was initially designed to operate for 100,000 hours but it already has operated for 150,000 hours. In addition, the authorities have approved the dismantling of the plant being postponed until 2003. Until then the plant anticipates maintenance efforts for the turbine to grow due to increasing wear.

The tasks which the superintendent executes to maintain the turbine concern three specific types of maintenance, i.e. preventive, predictive and corrective maintenance.
Preventive maintenance involves an annual inspection of the turbine according to a plan which is set up by the superintendent's staff and which he approves. When the plant is taken out of operation, maintenance personnel carry out the inspection and record their findings. The superintendent evaluates these findings and determines whether corrective actions must be undertaken. For instance, when a bearing shows signs of erosion, he may decide to replace it.

Predictive maintenance involves diagnosing the turbine's condition by means of vibration analysis, predicting which long-term failures could appear and determining which actions must be undertaken to prevent these failures. In 1976 predictive maintenance was taken up in the maintenance programme. In that year a failure in the gearbox resulted in a damage claim of nearly NLG 2 million. There was strong evidence that it could have been prevented if vibration analysis had been carried out periodically. Since this failure, the superintendent four times a year requests a department of KEMA called "Performance Testing and Reliability Analysis of Production Installations" (MAP) to record vibrations, to transform these vibrations into spectra by means of Fourier transformations, to compare these spectra with the ones obtained in the preceding recording session and to make a preliminary diagnosis of the turbine's condition. The superintendent interprets the spectra as well. MAP charges the plant about NLG 20,000 a year for its services. As the plant will stay in operation for another fifteen years, the superintendent's successor will diagnose the turbine's condition at least 60 times.

Corrective maintenance is initiated by the operators on duty. When an operator discovers a failure, he requests a mechanic to inspect it. If a failure concerns a small defect, the mechanic corrects it at once. Otherwise the operator classifies it and instantly reports it to the superintendent. This classification ranges from very urgent, i.e. the failure must be corrected at once, to not urgent, i.e. it can be corrected when the turbine is taken out of operation for preventive maintenance. The superintendent then decides when the failure must be corrected.

Once a year the superintendent evaluates the failures which have occurred frequently during that year and determines which corrections must be carried out to prevent these failures during future operation of the turbine. When the superintendent finds that a failure is due to a design error of a part, he may decide to modify the design of that part, or to manufacture a new part and to replace the old one. Design modifications can also be requested by the authorities.
<table>
<thead>
<tr>
<th>Type of Maintenance</th>
<th>Task</th>
<th>Level of Problem Solving</th>
<th>Future Problem</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Expert</td>
<td>Successor</td>
</tr>
<tr>
<td>Preventive Maintenance</td>
<td>Inspect Plan Evaluate Result(s)</td>
<td>RB</td>
<td>RB</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SB/RB/KB</td>
<td>RB/KB</td>
</tr>
<tr>
<td>Predictive Maintenance</td>
<td>Diagnose Condition</td>
<td>RB</td>
<td>KB</td>
</tr>
<tr>
<td></td>
<td>Diagnose Gearbox Condition</td>
<td>RB</td>
<td>KB</td>
</tr>
<tr>
<td></td>
<td>(not worked out)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Diagnose Axle Condition</td>
<td>RB</td>
<td>KB</td>
</tr>
<tr>
<td></td>
<td>Det. Overall Levels</td>
<td>SB</td>
<td>SB</td>
</tr>
<tr>
<td></td>
<td>Det. Peaks &amp; Amplitudes</td>
<td>SB</td>
<td>SB</td>
</tr>
<tr>
<td></td>
<td>Det. Disturbances</td>
<td>RB</td>
<td>RB</td>
</tr>
<tr>
<td></td>
<td>Gen. Possible Condition</td>
<td>SB/RB</td>
<td>RB/KB</td>
</tr>
<tr>
<td></td>
<td>Ver. Possible Condition</td>
<td>RB</td>
<td>RB</td>
</tr>
<tr>
<td></td>
<td>Determine Action(s)</td>
<td>SB/RB</td>
<td>RB/KB</td>
</tr>
<tr>
<td>Corrective Maintenance</td>
<td>Evaluate Result(s)</td>
<td>RB</td>
<td>RB</td>
</tr>
<tr>
<td></td>
<td>Determine Action(s)</td>
<td>RB/KB</td>
<td>RB/KB</td>
</tr>
</tbody>
</table>

Table 8.1 Classification of the superintendent's and his successor's behaviour.

In table 8.1 we have summarized the behaviour which the superintendent shows and the behaviour which we expect his successor will initially show in executing the tasks concerning preventive, predictive and corrective maintenance. The tasks in which the successor shows knowledge-based (KB) behaviour while the superintendent shows rule-based (RB) or skill-based (SB) behaviour may initially give rise to difficulties, once the superintendent is retired. We may expect that his successor will try to develop a procedure to execute these tasks preferably as efficiently and effectively as the superintendent currently does. This line of thought has induced us to concentrate on the task of diagnosing the turbine's condition.

With the Information Paradigm in mind, we divide the problem situation into an IS consisting of the superintendent and MAP and into a RS consisting of the turbine, see figure 8.1. At his request MAP records vibrations on several locations and in several directions, transforms these vibrations into spectra, compares these spectra with the ones obtained in the preceding recording session and produces a preliminary diagnosis. This information is recorded in a report which is sent back to the superintendent. He also receives process information about the turbine recorded by datalogging equipment. The superintendent can control the turbine, e.g. by repairing or replacing a part.
New situation

Once the superintendent is retired, his successor will continue to request MAP’s services and to interpret the spectra received. It is possible to design an ESS that can support the successor in diagnosing the turbine’s condition for instance by making the superintendent’s expertise available to him. We expect that this way the successor will initially be able to hypothesize more conditions and anticipate more failures and that he will therefore be able to be more certain of the accuracy of his diagnoses.

The insights derived from table 8.1 can help us identify how an ESS can support the successor as a novice, but also as an expert. In the next section we will also concentrate on the latter issue.

For the plant it may be fruitful to find out whether there is a more time- and cost-efficient way to record vibrations, to transform these vibrations into spectra and to generate difference spectra on the basis of these spectra and the spectra obtained from the preceding recording session. IT applications are available that can automatically record vibrations and transform these vibrations into spectra. PRISM2, which is manufactured by Palomar Technology International, is an example of such an IT application. It consists of vibration recording equipment and a program which can transform recorded vibrations into spectra. Integration of PRISM2, which the plant can purchase for NLG 45,000, into a new IS can save the plant more than NLG 255,000 until its dismantling. It will then no longer be necessary to request MAP’s services. Departing from the assumption that the plant will purchase PRISM2, we have included it in the design of the new IS, see figure 8.2.
8.3 Infological problem

Actual situation

The superintendent diagnoses the turbine’s condition by looking at two distinct sections of the turbine, i.e. the gearbox and the axle. Here we focus on the task of diagnosing the condition of the axle. The superintendent executes this task on the basis of eleven pairs of an absolute and a difference spectrum. Each pair is associated with a recording location and a recording direction. A recording location is situated on a bearing of the axle, see figure 8.3.
INFOLOGICAL PROBLEM

SEC. 8.3

For each recording location the superintendent diagnoses the condition of the axle, see the task precedence structure in figure 8.4. He uses difference-spectra to determine whether the overall vibration levels of the bearing have increased. In such a case he determines the peak frequencies which have caused these levels to increase as well as their amplitudes. The superintendent compares the amplitudes with their associated Verein Deutscher Ingenieure (VDI) 2056 standard ratings and determines their degree of disturbance, i.e. good, acceptable or not acceptable. He generates hypothetical axle conditions on the basis of the peak frequencies and their associated disturbances. The superintendent verifies these hypothesized conditions on the basis of process information. An example of how he does this is given in section 8.4.

Figure 8.4 Task precedence structure: diagnosing the condition of the axle.
New situation

We call the intended ESS "PRESSTO" an acronym of "Prototype ESS voor Turbine Onderhoud". Figure 8.5 represents the support structure of PRESSTO. In the following discussion of this support structure we depart from the assumption that PRESSTO is provided with a database that can contain information about the RS of the task of diagnosing the turbine’s condition.

The support structure of PRESSTO contains functions that support the successor in storing and deleting spectra provided by PRISM2 and recorded at several recording locations and in several recording directions.

We may expect that the successor will execute the first three tasks included in the task precedence structure in figure 8.4 on the skill-based level. The first task involves determining whether the overall vibration levels of a bearing have increased. The second task involves determining the peak frequencies which have caused these levels to increase as well as their amplitudes. Finally, the third task involves determining the degree of disturbance in the amplitudes of the peak frequencies. To support the successor in executing these tasks, PRESSTO contains functions to retrieve absolute spectra and to generate difference spectra on the basis of these absolute spectra. It also contains functions to generate trends in two or more absolute spectra or difference spectra.

Figure 8.5 Support structure of PRESSTO.
INFOLOGICAL PROBLEM

We expect that the successor will ultimately execute the task of generating hypothetical conditions on both the skill-based and rule-based level. The above-mentioned functions can support the successor in executing this task on the skill-based level. However, PRESSTO also contains functions that can automatically diagnose the turbine's condition on the basis of the superintendent's expertise. Therefore PRESSTO contains additional functions to support the successor in storing, retrieving and reviewing conditions diagnosed.

8.4 Datalogical problem

We shall first consider the design of the function that can support the successor in diagnosing the turbine's condition by executing a description of how the superintendent executes the task. We did four interviews to obtain this description.

Fortunately, the superintendent had documented in reports and notes the failures and critical symptoms corrected under his supervision. In the first two interviews, carried out by self report, he separately discussed each of his experiences in diagnosing these failures and critical symptoms and in taking corrective actions.

In the last two interviews, carried out by introspection, the superintendent discussed his expectations which conditions may occur during future operation of the turbine, which symptoms may be identified in this and how these hypothetical conditions can be corrected.

In the second interview the superintendent explained e.g. how he diagnoses the condition "parallel misalignment". Of this interview we consider a chunk also discussed in Van Weelderden and Sol (1990):

<table>
<thead>
<tr>
<th>Chunk</th>
<th>Interview 2, No. 9</th>
</tr>
</thead>
<tbody>
<tr>
<td>Begin</td>
<td>Page 7 &quot;Diagnosis of vibrations ...&quot;</td>
</tr>
<tr>
<td>End</td>
<td>Page 16 &quot;... at an early stage.&quot;</td>
</tr>
<tr>
<td>Contents:</td>
<td>Diagnosis of condition &quot;parallel misalignment&quot;</td>
</tr>
<tr>
<td>Trace</td>
<td>..., parallel misalignment, ..., increase 50 Hz vibration level, axle frequency, phase shift, ..., oil temperature, ..., run down time, ..., pit, harmonic.</td>
</tr>
</tbody>
</table>

The interview fragments in this chunk which we will discuss describe how the superintendent diagnoses a possible condition of the axle, i.e. "parallel misalignment", on the basis of spectra derived from vibrations recorded on the bearing behind the medium pressure turbine.
Fragment 1: "So if you have parallel misalignment, what’s the symptom, you then directly get an increase of the 50 Hz vibration level. It crackles up, depending of course upon your misalignment. It does not happen at once, it builds up slowly."

Fragment 2: "Well, the first one is an increase of the 50 Hz vibration level. You directly get a harmonic of 2 times the 50, twice the axle frequency. It has a phase shift of 180°."

Fragment 3: "Parallel misalignment of the rotors causes extra bearing load of the higher bearings. The rotor says, sorry guys I am going to carry you up, believe me it will happen, he carries him up and then this bearing gets less load and this bearing gets more load. You could look at the oil temperatures. It is difficult but it is possible."

Fragment 4: "If you, this could also be a symptom, so if you have a machine with such a parallel misalignment, then the run down time could be shorter."

From these fragments we derived a rule which describes how the superintendent diagnoses the condition of the axle on the basis of spectra derived from vibrations recorded on the bearing behind the medium pressure turbine.

if Bearing. 50Hz_Level increased
and Bearing.100Hz_Level increased
and Bearing.OilTemperature increased
and Turbine.RunDownTime decreased
then Axle.Condition = parallel misalignment

We divided the four interviews into several chunks, each concerning one of the eleven vibration recording locations on the turbine. For each recording location we identified symptoms, conditions and actions. This way we derived 53 rules. For an extensive discussion of these rules we refer to Draijer (1989).

PRESSTO's database is designed as a set of two relations. The first relation represents the spectra retrieved from PRISM2:

Spectrum (Date, Location, Direction, Range, Value1, ..., Value400, Level)

The key of this relation is its recording date, its recording location, its recording direction and its recording range e.g. 02-11-88, Bearing HP (i.e. High Pressure Turbine) Front, horizontal, 200 Hz. The attribute "level" is the average value of the 400 recorded amplitudes.
The second relation concerns the conditions which PRESSTO diagnoses and stores in its database:

Condition (Date, Rule ID, Status)

The key of this relation is the recording date of the vibrations on the basis of which PRESSTO has diagnosed the turbine's condition and the identification of the rule which PRESSTO has evaluated. The attribute "status" can be assigned the values true, false or unknown.

PRESSTO diagnoses the turbine's condition by evaluating each rule consecutively, starting with rule 1, which concerns the oil pump in the gearbox and ending with rule 53, which concerns the bearing in front of the exciter. This way PRESSTO adds 53 conditions to its database.

8.5 Technological problem

PRESSTO can be characterized by flexible user-interaction, fast database access, fast calculations and simple inferences. The successor already had at his disposal a micro computer (Intel 80386 25 MHz processor, 80 Mb hard disk, color EGA) to carry out his work. We implemented PRESSTO on this machine. We programmed PRESSTO in Turbo Pascal 5.0, extended with the Turbo Pascal Database Toolbox.

The user interface of PRESSTO consists of a menu and a fixed window. The menu is adapted to the sequence in which the successor will diagnose the turbine's condition and supports him in selecting one of PRESSTO's main functions.

For a particular recording session PRESSTO can provide the successor with an overview of the spectra that are present in its database. It generates a list of dates of recording sessions of which it contains spectra. The successor can scroll through this list and select the recording session on which he wants to be informed. PRESSTO then generates the overview. To support communication between the successor and PRESSTO we designed the task interface shown in figure 8.6.

The task interface consists of a window in which one can identify a table. The first two columns of this table show at which recording location and in which recording direction a spectrum has been recorded. The last five columns indicate in which frequency range a spectrum has been recorded.
Figure 8.6 Task interface for viewing available spectra.

We designed the task interfaces in figure 8.7 and in figure 8.8 to support communication between the successor and PRESSTO while executing the first four tasks included in the task precedence structure in figure 8.4.

When the successor wants to consider a spectrum, PRESSTO can support him in specifying the type of spectrum (absolute or difference), the recording location and recording direction, the frequency range and one or two recording dates. The number of recording dates specified depends on whether the successor wants to consider an absolute spectrum or a difference spectrum respectively. The task interface in figure 8.7 contains a window in which the successor can identify the spectrum PRESSTO generated. PRESSTO can support him in specifying whether he wants to alter the frequency range and amplitude range. Note the menu with the function keys.

When the successor wants to consider a trend in two or more spectra, PRESSTO can support him in specifying the same issues we just described except for the recording dates. Now PRESSTO can support him in specifying the recording dates of the first spectrum and the last spectrum to be considered. The task interface in figure 8.8 contains a window in which the successor can identify the trend PRESSTO generated. PRESSTO can support him in going through the frequency range by enabling him to press the "up" and "down" keys. Note the "frequency indicator" on the right which keeps track of the successor's position in the frequency range. PRESSTO can support him in specifying whether he wants to alter the frequency and amplitude range. Note the menu with the function keys.
Figure 8.7 Task interface for analyzing a spectrum.

Figure 8.8 Task interface for analyzing a trend in two or more spectra.
At his request PRESSTO supports the successor in consecutively evaluating the 53 rules which we extracted from the superintendent by indicating for each rule which of its premises are true. To support communication between the successor and PRESSTO we designed the task interface shown in figure 8.9. It consists of a window in which one can identify two forms and one list. The form on the left describes the premises of a rule. The form on the right describes whether the premise of a rule is true, false or unknown. The items in the list represent the conclusions of the rule when all premisses are true.

For a particular recording session and for a particular recording location PRESSTO can provide the successor with an overview of the associated rules of that recording location. For each rule PRESSTO indicates whether its premises are true, false or unknown, i.e. whether there was not enough information to evaluate its premises. To support communication between the successor and PRESSTO we designed the task interface depicted in figure 8.10. It consists of a form indicating the identification of a rule and the determination of whether all premises of the rule are true, false or whether there was not enough information to evaluate all premises of the rule. PRESSTO can support the successor in specifying whether he wants to select another recording location or consider a specific rule. Note the menu.
One version of PRESSTO, i.e. its code as well as the data structures and the structure charts which are the blueprints of this code, are documented in Jonker and Koelewijn (1989). The code represents some 15,000 statements.

A net effort of twelve man-months was spent on designing PRESSTO over a period of eight months by the superintendent, his successor, one graduate student and two under graduate students of Delft University of Technology and ourselves. The numbers of man-months spent on addressing the systelological, infological, datalogical and technological problems are three, three, three and three respectively.

8.6 Discussion

When we completed PRESSSTO, the superintendent was just about to retire. Until the plant has purchased PRISM2, the successor will depend on the services of MAP for the recording and transformation of vibration data into spectra.

The superintendent and his successor consider PRESSSTO a tool with which they can efficiently retrieve, generate and analyse spectra. They also consider it a tool with which they can generate check lists of possible conditions which might otherwise be overlooked in diagnosing the turbine's condition. The successor considers it as a great advantage that, "when I have both PRESSSTO and PRISM2 at my disposal, I can diagnose the turbine's condition as often as I please without incurring extra costs".
Not only the availability of the superintendent's expertise makes the successor more confident that he can execute the task successfully but also the availability of a spin-off of our design approach, i.e. the procedure which the superintendent follows to diagnose the turbine's condition. The successor will need to spend little time developing such a procedure.

PRESSTO can be an effective means to solve the problem of the plant. Nevertheless, there are limitations to the ESS concept presented by PRESSTO. It diagnoses conditions on the basis of experiences and expectations. However, to what extent can experiences and expectations be generalized? The complete range of potential turbine conditions may be infinite. The range of conditions which the superintendent can generate is only a small subset. The benefit of this project in supporting the superintendent's successor in executing his future task must be derived initially from the provision of a procedure to execute the task and from a limited check list of possible conditions, and ultimately from the provision of a tool to retrieve and analyze spectra as often as may be necessary.

References in chapter 8

9

EPILOGUE

9.1 Summary

In this dissertation we have addressed the issue of improving the performance of an expert by means of IT. We observed a trend towards integrating the concepts of a DSS and a KBS into the concept of an "intelligent" DSS, also called ESS. This development appears to be guided by a commonly shared view that computers should support an expert in solving problems, rather than replace him. Many organizations are becoming interested in integrating ESSs into their information systems. However, one can question whether their expectations are met in practice. We observed that a number of authors acknowledge the promising potential of the ESS concept but that they also agree that ESS design is not being put into practice effectively.

These observations inspired us to question how ESS design methodologies support ESS design. To answer this question, we set up a framework to describe a methodology. We selected a sample consisting of ten contemporary, methodological contributions on ESS design. For each example we determined how it fits into the framework. We tried to identify a way of thinking, a way of modelling, a way of working and a way of control. As to a way of thinking, we were interested in whether a problem situation is considered from the micro-, the meso- and the macro-perspective. We were also interested in whether attention is paid to creating an understanding of an actual situation prior to designing a new one. In a way of modelling, we considered whether a designer is supported in describing the systelogical, infological, datalogical and technological problems. In a way of working, we looked for phasings, methods and techniques. Finally, regarding a way of control, we considered whether a designer is supported in managing an ESS design project.

We observed that problem situations are not frequently considered from the meso- and the macro-perspective, that actual situations are not often taken into consideration as a reference for designing and evaluating ESSs, that the systelogical and infological problems are not frequently addressed, that few methods and techniques are provided to design ESSs and that a way of control receives little or no attention. On the basis of these observations we hypothesized that ESS design can be put into practice effectively by using a methodology that addresses the issues ignored. To test this hypothesis, we set out a way of thinking and complemented it by a way of modelling, a way of working and a way of control. The resulting methodology is called "MEDESS", an acronym of "Methodology for Designing Expert Support Systems".

147
In the way of thinking behind MEDESS, the Information Paradigm is adopted to set the boundaries of a problem situation concerning the performance of an expert. Another essential paradigm in this way of thinking is the paradigm of bounded rationality. It is suggested that a problem situation should be considered not only from the micro-perspective, but also from the meso- and the macro-perspectives. Finally, understanding a problem situation should receive as much attention as designing a new situation.

From the micro-perspective, we express a problem situation in terms of an expert's objective and performance in achieving his objective. The concept of a task plays here an important role. We distinguish between sub-tasks and coordination tasks. On the skill-based level an expert directly associates a solution to a problem. An ESS can support an expert in recording problems and solutions. On the rule-based level an expert selects and carries out a procedure to arrive at a solution to a problem. An ESS can take over the execution of this class of tasks. In addition, an expert can instruct an ESS in which sequence it should execute the tasks which an expert used to execute on the rule-based level. Finally, an ESS can support an expert in monitoring the execution of tasks which he executes on the skill-based level.

From the meso-perspective, we express a problem situation in terms of an organization's objective and performance in achieving its objective. Designing an ESS can be placed in a broader context of reconsidering an organization's objective and performance. Within this context one may review existing and specify new problem classes, individuals and tasks relating individuals to problem classes. One may also reconsider in which sequences tasks should be executed. Once these issues are resolved, the design of an ESS can come into view.

From the macro-perspective, we express a problem situation in terms of the common objective of two or more organizations and their joint performance in achieving this objective. One can disregard the boundaries between these organizations and consider them as one organization. The design of an ESS, then, can be taken up in much the same way as from the meso-perspective. Special attention should be paid to achieving consensus among the organizations involved as to which organization owns which ESS.

In a way of modelling we distinguish between the systelogical, infological, datalogical and technological problems.

We describe the systelogical problem in terms of organizations, problem classes, individuals and tasks. These concepts are essential in describing a problem situation from an organizational perspective and in specifying the added value of an ESS. They look fairly obvious. Nevertheless, these concepts and the relationships between them are often overlooked.
We describe the infological problem in terms of tasks. A task is a 4-tuple $t$ consisting of the information needed to execute $t$, the information resulting from the execution of $t$, the problem solving behaviour, i.e. skill-based or rule-based, shown by an expert while executing $t$ and, finally, the support structure of an ESS associated with $t$ which represents a collection of functions that an ESS can execute to support or to automate the execution of $t$. We work out a task into a precedence structure, an ER-model of its RS, a coordination structure. The first two elements describe the static aspects of a task, while the last element describes its dynamic aspects.

When we address the datalogical problem, we focus on describing how information is or should be grouped and processed, without taking into account with which technology this is or can be achieved. The major concepts in this problem are a database management system and a model management system.

When we describe the technological problem, we are primarily concerned with designing the technical implementation of a database management system, a model management system and a user interface. The user interface of an ESS can consist of one or more task interfaces, each enabling an expert and an ESS to communicate with each other during the execution of a sub-task or a coordination task.

In a way of working we have specified a phasing in which we distinguish between an understanding phase and a design phase. These phases can be characterized as learning processes in which one can iteratively construct and review systelological, infological, datalogical and technological descriptions and designs. To guide these processes, we have specified methods to construct systelological and infological descriptions and designs. As to the datalogical and technological problems, we follow methods put forward in the literature.

Regarding a way of control, we suggested that a satisfactory balance should be achieved between the effectiveness and the efficiency of a process of ESS design. Therefore, we distinguish between the organization of a project and the progression of a project.

To test MEDESS, we applied it in four instances in which KEMA wished to solve a problem concerning the performance of an expert and in which it was interested in providing the expert involved with an ESS.

The first case concerned two consultants who are employed by the WMK department and who configure regimes for the future inspection of steam headers and steam pipes used in fossil fuel power plants. The objective of WMK is to handle all accepted requests for advice. However, the amount of work involved in handling a request was increasing considerably. The time efficiency of the consultants had to be improved. We designed an ESS that can support the consultants in handling twice the amount of work they currently do.
The second case concerned four observers who are employed by the DZL department and who design short-current circuits for testing circuit breakers. The objective of DZL is to maintain its competitive position, e.g. by improving the quality of its services to its customers. The number of errors made in designing circuits had to be reduced and the degree of consensus between the observers as to how a circuit should be designed had to be enhanced. In addition, the time efficiency of the observers had to be improved. We designed an ESS that can support the observers in designing circuits three times faster and with hardly any errors.

The third case concerned a planner who is employed by the LTI department and who plans trips for the inspection of electrical products and related production facilities all over the European continent. The objective of LTI is to maintain a satisfactory work climate within the department. The complaints of the inspectors regarding the travelling overhead of the trips they make had to be addressed. We designed an ESS that can support the planner in planning trips that involve less travelling overhead than is currently the case and that can help him in planning these trips in a more efficient and flexible way.

The fourth case concerned the successor of the mechanical maintenance superintendent of a Dutch nuclear power plant. The objective of the plant is to keep track of the condition of its equipment as accurately as possible. We identified that the successor was likely to face difficulties in diagnosing the condition of the turbine due to a lack of specific experience. We designed an ESS that can support the successor in diagnosing the turbine’s condition more cost efficiently than his predecessor had done and more accurately than if he had not had the expertise of his predecessor at his disposal.

9.2 Conclusions

Considering our experiences derived from these four cases we can draw the following conclusions concerning MEDESS’ way of thinking, way of modelling, way of working and way of control.

To improve a way of thinking, we suggested that the problem of improving the performance of an expert should be considered not only from the micro-perspective, but that it should also be placed in a broader context of improving the performance of an organization or the performance of an inter-organizational system.

In two cases it turned out to be profitable to consider a problem situation from the macro-perspective. In the first case we advised the WMK department and the power utilities with which it cooperates, to assign the task of retrieving a part’s maintenance history to the consultants instead of to power utility representatives. In addition we
enabled the consultants to compress operational histories in a more efficient and flexible way. In the fourth case we advised the nuclear power plant to assign the tasks of recording the turbine's vibrations and of transforming them into spectra to the successor of the mechanical maintenance superintendent instead of to a representative of the MAP department. We also enabled the plant to execute these tasks in a more cost efficient way.

In all cases it turned out to be fruitful to consider a problem situation from the meso-perspective. In the first case we enabled the WMK department to continue to handle all requests accepted without employing more consultants. In the second case we helped the DZL department to improve the quality of its services to its clients. In the third case we supported the LTI department in maintaining a satisfactory work climate. In the fourth case, finally, we helped the nuclear power plant to keep track of the turbine's condition as accurately as possible.

In all cases we considered a problem situation from the micro-perspective. In the first case we improved the time efficiency of the consultants. In the second case we improved the time efficiency of the observers and reduced their error rate. In the third case we improved the planner's time efficiency and reduced his reluctance to consult topographical information and to split up areas in a more appropriate way. In the fourth case, finally, we enabled the successor to "pick up" the execution of the task of diagnosing the turbine's condition in a minimum learning time and to achieve a similar effectiveness in diagnosing the turbine's condition as his predecessor had achieved.

We conclude that by considering a problem situation not only from the micro-perspective, but also from the meso- and the macro-perspective, we could identify more interests associated to an expert's function. By addressing these interests and the wishes as to how these interests can be served, we could identify more possibilities to improve the performance of an expert.

To improve a way of modelling, we suggested that the systological and infological problems should be addressed in addition to the datalogical and technological ones.

In all cases the conceptual description of the systological problem, which we put forward in MEDESS' way of modelling, was a simple but powerful means to construct a description of objectives and performances of an inter-organizational system, an organization or a task. Most problem owners initially expressed their problems in terms of technical solutions. By addressing the systological problem we could stimulate them to consider their problems from an organizational perspective and to come to the essence of their problems. These problems had much to do with the performance of an organization in achieving its primary or secondary objectives. Note that only in the third case we addressed a secondary objective of the LTI department.
We could also specify the added value of an ESS to an organization. This is exactly what is needed to create a context in which ESS design should be taken up. An ESS should merely serve as an instrument in meeting the interests of an organization and the individuals it employs. Designing an ESS should not become a goal in itself.

In all cases the conceptual description of the infological problem, too, has turned out to be a powerful means to describe a task, much in line with the paradigm of bounded rationality that emphasizes the procedural aspects of the way in which an expert solves a problem. In this respect the concept of a task coordination structure has proven to be very useful. In two cases we constructed a task coordination structure to shed light on the determinants and opportunities for improving an expert's efficiency and effectiveness. In the second case a task coordination structure supported us in identifying why the observers make errors. It appeared that they are sometimes reluctant to re-execute the task of designing a circuit in reverse order. In the third case a task coordination structure supported us in identifying why the planner plans inspection trips with travelling overhead. It appeared for instance that he is often reluctant to split up areas into more appropriate ones.

Rasmussen's view to distinguish between a skill-based, rule-based and knowledge-based level of problem solving behaviour was easy to put into practice and guided us in deciding with what type of function a task can best be supported.

To improve a way of working, we put forward a phasing and a number of methods to describe the systelogical and infological problems.

We suggested a phasing in which we distinguish between an understanding phase and a design phase. This phasing appeared to be useful in describing the systelogical and infological problems. As to the datalogical and technological problems, we were primarily concerned with constructing designs.

Regarding the methods we proposed to describe the systelogical problem, we can say that the method for constructing a description has been followed extensively. The method for constructing a design has been followed less extensively since it focused primarily on reviewing the performance of an organization and/or a task rather than on a reconsideration of a problem class and/or an individual. Note that in the first and in the fourth case we re-assigned a number of tasks to other individuals. Assessing the organizational effects of an ESS appeared to be possible in the first and second case. In the third case this was rather difficult. We had to deal with meeting an organization's secondary objective rather than its primary objective. Personal interests are very subjective and so are personal objectives. Measuring them can lead to highly speculative results. In addition, time was not on our side in waiting and being notified of the effects of IPESS. In the fourth case this was rather difficult as well, again because time was not on our side. If the superintendent had had more time to participate in the project, it might have been possible to carry out an
experiment to compare the diagnoses of both the superintendent and his successor made on the basis of the same spectra and process information. Unfortunately, we neither had time to carry out an experiment to assess the effect of PRESSTO on the effectiveness of the diagnoses made by the successor. Nevertheless, concerning the primary objectives of an organization, we conclude that, by constructing both a systeological description and a systeological design, one is able to define reference points to design and assess the added value of an ESS.

Regarding the methods we proposed to describe the infological problem, we can say that the method for constructing a description has been followed extensively. The method for constructing a design has been followed less extensively since it primarily focused on designing support structures rather than on reviewing task precedence structures, the RSs of tasks and task coordination structures. However, it should be noted that in the second case all observers design circuits from scratch and no longer rely on circuits implemented previously. In the third case the planner can now also join areas. We conclude that the methods proposed are complete and support us in shedding light on the determinants and opportunities to enhance an expert’s limited cognitive abilities so as to meet the requirements stated in a systeological design.

Concerning the elicitation of descriptions of how an expert executes a task, we relied on introspection in the first and in the third case and on self report in the second case. In the fourth case we elicited the superintendent’s expectations by introspection and his experiences by self report. We can say that the choice of introspection or self report depended on whether we had indications that the procedure which an expert followed to execute a task was rather formalized or not.

As to a way of control, we addressed the issues of project organization and project progression. In table 9.1 we have listed for each project the duration expressed in years, the number of man-months spent on the various design problems, the number of ESS versions designed, an estimate of the costs involved expressed in NLG and of the Return On Investment (ROI) period expressed in years.

A cost estimate is calculated by considering a man-month as 150 man-hours and by charging NLG 150 for a man-hour spent on addressing the systeological and infological problems and NLG 100 for a man-hour spent on addressing the datalogical and technological problems. These rates may be considered as representative for the ones currently demanded by professionals in the field of information systems design.

Regarding the first case, we noted in chapter 5 that we designed a more fashionable version 2.0 of Xpection. Therefore we carried out a correction with respect to the duration of the project, i.e. one and a half years instead of two, and with respect to the number of man-months spent on addressing the technological problem, i.e. twelve man-months instead of sixteen. Costs spent on the purchase of hardware and software are relatively negligible.
The annual savings of the ESSs designed are estimated as follows. As Xpection can support the consultants in handling twice the amount of work they currently do, i.e. 1000 hours extra, we estimate the annual savings which Xpection can bring about as NLG 175 * 1000 hours = NLG 175,000. We assume that a consultant costs the WMK department NLG 175 per hour.

OSS supports the observers in designing a circuit three times faster than they currently do, i.e. in three minutes instead of in nine minutes. They execute this task about 1750 times a year. Therefore OSS can save the DZL department approximately 175 hours, i.e. NLG 175 * 175 ≈ NLG 30,000 per year. It is more difficult to estimate the savings which OSS can bring about due to the reduced error rate of the observers in designing a circuit. Assuming that one hour of a shift costs DZL NLG 6000 and that 10% of the average of 120 shifts per year will take one hour extra to carry out due to one or more design errors, we may conclude that OSS can bring about an additional saving of NLG 6000 * 12 = NLG 72,000 per year.

The savings of IPESS are even harder to estimate as IPESS is designed primarily to improve the performance of an organization in achieving a secondary objective. Its merits are determined by personal judgements which are very subjective.

PRESSTO supports the successor in diagnosing the turbine's condition more cost efficiently. He no longer has to request the MAP department for its services, which implies a saving of at least NLG 20,000 per year. It is more difficult to estimate the savings which PRESSTO can bring about due to the reduced chance for the successor to interpret spectra in an inappropriate way. Nevertheless, assuming that the occurrence of one failure, e.g. the one that occurred in 1976, may be prevented due to the support of PRESSTO until the plant's dismantling in 2003, we may conclude that PRESSTO can bring about an additional saving of NLG 100,000 per year.

In this study the duration of a project varied between eight months and one and a half years. The costs may well have varied between NLG 225,000 and NLG 450,000 when carried out by professionals instead of by students and ourselves.

On the grounds which we put forward in this section, we cannot reject our hypothesis that ESS design can be put into practice effectively, provided that the issues which we indicated as disregarded in chapter 3, are given due consideration.

<table>
<thead>
<tr>
<th>Project</th>
<th>Duration</th>
<th>S</th>
<th>I</th>
<th>D</th>
<th>T</th>
<th>Total</th>
<th>Versions</th>
<th>Costs</th>
<th>ROI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Xpection</td>
<td>1,5</td>
<td>4</td>
<td>4</td>
<td>6</td>
<td>12</td>
<td>26</td>
<td>2</td>
<td>450,000</td>
<td>2,5</td>
</tr>
<tr>
<td>OSS</td>
<td>1,5</td>
<td>3</td>
<td>3</td>
<td>5</td>
<td>10</td>
<td>24</td>
<td>2</td>
<td>350,000</td>
<td>3,5</td>
</tr>
<tr>
<td>IPESS</td>
<td>1,5</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>13</td>
<td>24</td>
<td>2</td>
<td>400,000</td>
<td>?</td>
</tr>
<tr>
<td>PRESSTO</td>
<td>0,67</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>12</td>
<td>1</td>
<td>225,000</td>
<td>2</td>
</tr>
</tbody>
</table>

Table 9.1 Project information.
INDEX

Ackoff, R.L. .................................................. 11, 14
Added value of an ESS .......................................... 47, 148, 152
AI, Artificial Intelligence ...................................... 2
Algorithm .......................................................... 49, 75, 123
Amsz, A. ......................................................... 79
Annual savings .................................................. 154
Application pull .................................................. 37
Area .................................................................. 114, 115
Atlas .................................................................. 124
Bahl, H.C. .......................................................... 28
Benbasat, I. ........................................................ 5
Betz, F. ............................................................... 12, 14
Bloom, M. ........................................................... 66, 73, 80
Bosman, A. .......................................................... 14
Bots, P.W.G. ........................................................ 41, 48
Bounded rationality ............................................... 12, 42, 152
Bredeweg, B. ...................................................... 33-35
Breuker, J. .......................................................... 5, 33-35, 59
Brussaard, B.K. ..................................................... 41
Buchanan, B.G. ..................................................... 86
Burns, R.N. ........................................................... 24
Call .................................................................. 110, 115
Carlson, E.D. ........................................................ 26
Chang, M.G. .......................................................... 27
Check list ............................................................. 144
Chunk ................................................................. 74, 138
Circuit ................................................................. 91
Circuit breaker ..................................................... 83, 90
Collective ............................................................. 68, 69
Confidence level ................................................... 80, 81
Cook, C.R. ............................................................ 3
Coordination task .................................................. 42, 44, 48, 148
	support of a .................................................. 44
Costs of a project .................................................. 154
Creep ................................................................. 63, 68, 69
CSA, Canadian Standards Association ................. 109
Cycle ................................................................. 69
D'Agapieff, A. ...................................................... 5
Data structure ..................................................... 49, 59, 79, 104, 128, 144
Database
	of IPESS .......................................................... 123
	of OSS ............................................................. 98
	of PRESSTO .................................................... 139
	of Xpection ...................................................... 73
Database management system ............................... 49, 57, 60, 149
Datalogical description ........................................ 51
	method for constructing a .................................. 57
Datalogical design ................................................ 51

157
INDEX

method for constructing a ........................................... 57
Datalogical problem ....................................................... 24
   conceptual description in MEDESS .............................. 49, 149
DDA, Deutschen Dampfkessel Ausschuß .......................... 69
De Pater, G. .................................................................. 79
De Romph, P.E. ................................................................. 79
Decision .................................................................... 11
DSS, Decision Support System ........................................ 1
   generator .................................................................. 2, 59, 76
   philosophy behind ...................................................... 2
Dennis, A.R. .................................................................. 24
Depression .................................................................. 96
Design phase .................................................................. 50, 149
Deurwaarder, W.L. .......................................................... 104
Dimension of abstraction ............................................... 55, 57
Discrepancy .................................................................. 11, 16
Dos Santos, B.L. ............................................................... 50
Doukidis, G.I. .................................................................. 1-3
Draijer, J.M. .................................................................. 139
Duty ............................................................................. 90
DZL, De Zoeten Laboratory ............................................. 83
Effectiveness .................................................................. 43, 44
Efficiency ....................................................................... 43, 44, 67
Effort
   spent on designing IPESS ............................................ 128
   spent on designing OSS .............................................. 104
   spent on designing PRESSTO ................................... 144
   spent on designing Xpection ...................................... 80
Elam, J. ......................................................................... 5
Elicitation ..................................................................... 153
Error ............................................................................... 18, 44, 67, 85-87, 106
   lapse .......................................................................... 18, 43, 87, 106
   mistake ....................................................................... 18, 87, 106
   slip ............................................................................. 18, 43, 87, 106
ES, Expert System ............................................................ 2
   shell .......................................................................... 59, 76
ESS design
   in MEDESS ................................................................. 43-45, 148
   methodological contribution on ................................. 25-36, 147
ESS, Expert Support System ........................................... 1, 3, 72, 97, 121, 137, 147
   design methodology .................................................. 6, 23, 147
Experience ...................................................................... 17, 20
Experiential learning ....................................................... 17
Experiment ..................................................................... 66, 80, 86, 87, 105, 111, 114, 129, 130, 153
Expert ............................................................................. 1, 16-20, 42, 134, 147
   objective of an .......................................................... 43, 148
   performance of an ...................................................... 43, 148
Farah, B.N. ..................................................................... 4
Fatigue ........................................................................... 63, 69
# INDEX

Featheringham, T.R. ................................................. 14
Feigenbaum, E.A. .................................................. 2
Form ................................................................. 50
Framework to describe an ESS design methodology .......... 6, 23, 147
Function automayed .................................................. 48
primary ............................................................... 44
secondary ............................................................ 44
support ............................................................... 48
Gallupe, R.B. ......................................................... 24
Generator section of a circuit .............................. 91, 93
Gold standard ....................................................... 86
Gorry, G.A. .......................................................... 1
Goul, M. ............................................................... 3
Hardware ......................................................... 49, 59, 154
Harmon, P. .......................................................... 30
Hawkins, C.J.B. ...................................................... 5
Hazony, Y. ............................................................. 3
Heslinga, G. .......................................................... 18
Hetharia, R.E.R. ..................................................... 124, 129
Holsapple, C.W. ..................................................... 50
Hunt, R.G. ............................................................ 28
Hypothesis .......................................................... 41, 147, 154
IFE, Initial Factory Evaluation ................................. 109
Ijpeelaar, D.P.N.M. .................................................. 24
In't Veld, J. ............................................................ 44
Individual ........................................................... 47
Infological description ........................................... 51
method for constructing a ........................................ 55
Infological design .................................................. 51
method for constructing a ........................................ 57
Infological problem ............................................... 24
conceptual description in MEDEESS .......................... 48, 149
Information Paradigm .............................................. 41, 42, 133, 148
Information System development life cycle .................. 6
Input information ................................................... 48
Inquiring system .................................................... 14
Inspection list ....................................................... 69, 73
Inspection regime .................................................. 70, 74
Inspection trip ....................................................... 110
Inter-organizational system ..................................... 45
Interview ............................................................ 59, 74, 138, 139
Interviewing technique ........................................... 59
Introspection ......................................................... 59, 74, 138, 153
IPRESS ............................................................... 121-130
IS, Information System ........................................... 41
IT, Information Technology ...................................... 4, 20, 147
Ivanov, K. ............................................................. 37
Janse, J.W. ............................................................ 79
INDEX

Jansen, E. ........................................ 86
Jonker, D.W. ..................................... 144
KADS ............................................. 33-36
KBS, Knowledge Based System ................. 1, 2, 147
  shell ........................................... 2
Keen, P.G.W. .................................... 28
KEMA .............................................. 8, 63, 83, 109, 131, 149
Kerschberg, L. .................................. 32
Kien, T.W. ........................................ 128
Kleersteeg, P.C. ................................ 123, 128
Knowledge ....................................... 17
Koelewijn, J.J.A. ................................ 144
Kolh, D.A. ........................................ 17
Learning effect ................................... 80, 105
Learning mode ................................... 17
Learning style ................................... 17
  of an expert ................................... 43
Level of aggregation ......................... 52, 55
Limited cognitive ability .................. 12, 20, 153
List .............................................. 50
Luconi, F.L. ..................................... 3
Lüthi, H.J. ....................................... 2
Macro-perspective ............................... 23
  in MEDESS .................................... 45, 54, 148
Malone, T.W. .................................... 3
Management Science .......................... 12
Maus, R. .......................................... 30
MEDESS .......................................... 8, 41, 147
Menu ............................................. 50
Meso-perspective ................................. 23
  in MEDESS .................................... 44, 54, 148
Method .......................................... 6
Methodology ..................................... 6
Meyberg, S.H. ................................... 79
Micro-computer ................................ 59, 76, 100, 124
Micro-perspective ............................... 23
  in MEDESS .................................... 42, 43, 54, 148
MIS, Management Information System ........ 1
Mitroff, I.I. ..................................... 12-14
Model ............................................. 12
  conceptual .................................... 13-15
  descriptive .................................. 14
  descriptive conceptual ..................... 14, 15
  descriptive empirical ....................... 14, 15
  empirical ..................................... 13-15
  prescriptive .................................. 14
  prescriptive conceptual ..................... 14, 15
  prescriptive empirical ....................... 14, 15
Model management system .................. 49, 57, 60, 149
INDEX

Modelling
  descriptive ........................................... 15
  prescriptive ......................................... 15
Monhemius, W. ........................................ 11
Moore, J.H. ........................................... 27
Morrissey, W. ......................................... 30
Nieuwenhuis, M.A. .................................... 86
Nijssen, A ............................................ 113, 114
Noordzij, E ........................................... 79
Novice .................................................. 16, 19, 134
Object
  instance ............................................... 15
  type ................................................... 15, 49
Objective
  of an expert ......................................... 43, 148
  of an inter-organizational system .................. 45
  of an organization ................................... 44, 148
Operations Research .................................. 12
Organization .......................................... 44, 46
  interest of an ....................................... 44
  objective of an ..................................... 44, 148
  performance of an ................................... 44, 148
OSS .................................................... 97-104
Output information .................................... 48
Paradigm
  Bounded rationality .................................. 42, 152
  Information Paradigm ................................ 41
Peeters, A.M.A. ....................................... 123, 128
Performance .......................................... 1, 20, 147
  of an expert ......................................... 43, 148
  of an inter-organizational system .................. 45
  of an organization ................................... 44, 148
Personal computing .................................... 2
Pfeifer, R ............................................. 2
Phasing ................................................ 24
  direct acceptance of requirements .................. 24
  in MEDESS ........................................... 50, 149
  incremental .......................................... 24
  iterative ............................................ 24, 51
  linear ................................................ 24
Pondy, L.R. .......................................... 12-14
PRESSTO ............................................. 137-145
Primary function ..................................... 44
Primary objective ..................................... 46
Problem ................................................ 11
  datalogical .......................................... 24
  degree of novelty to problem solver ................ 16
  ill-structured ........................................ 1, 2, 16
  infological ......................................... 24
<table>
<thead>
<tr>
<th>INDEX</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>owner</td>
<td>11</td>
</tr>
<tr>
<td>semi-structured</td>
<td>16</td>
</tr>
<tr>
<td>solver</td>
<td>11</td>
</tr>
<tr>
<td>structured</td>
<td>2, 16</td>
</tr>
<tr>
<td>structuredness</td>
<td>16</td>
</tr>
<tr>
<td>systological</td>
<td>24</td>
</tr>
<tr>
<td>technological</td>
<td>24</td>
</tr>
<tr>
<td>Problem class</td>
<td>16, 46</td>
</tr>
<tr>
<td>Problem solving</td>
<td>11-20</td>
</tr>
<tr>
<td>behaviour</td>
<td>48</td>
</tr>
<tr>
<td>coordination</td>
<td>19, 20</td>
</tr>
<tr>
<td>experience</td>
<td>16</td>
</tr>
<tr>
<td>knowledge-based behaviour</td>
<td>19, 133</td>
</tr>
<tr>
<td>monitoring</td>
<td>19, 44, 68, 69, 148</td>
</tr>
<tr>
<td>rule-based behaviour</td>
<td>19, 133</td>
</tr>
<tr>
<td>skill-based behaviour</td>
<td>19, 133</td>
</tr>
<tr>
<td>Program</td>
<td>49</td>
</tr>
<tr>
<td>Project</td>
<td></td>
</tr>
<tr>
<td>costs of</td>
<td>154</td>
</tr>
<tr>
<td>organization of</td>
<td>60, 149</td>
</tr>
<tr>
<td>period of</td>
<td>154</td>
</tr>
<tr>
<td>progression of</td>
<td>60, 149</td>
</tr>
<tr>
<td>Prototyping</td>
<td>24</td>
</tr>
<tr>
<td>Rasmussen, J.</td>
<td>17</td>
</tr>
<tr>
<td>Reactance coil section of a circuit</td>
<td>91, 95</td>
</tr>
<tr>
<td>Reality</td>
<td>11</td>
</tr>
<tr>
<td>notion of desired</td>
<td>11, 15</td>
</tr>
<tr>
<td>perception of</td>
<td>11, 15</td>
</tr>
<tr>
<td>Reason, J.</td>
<td>19</td>
</tr>
<tr>
<td>Relational algebra</td>
<td>49, 73, 98</td>
</tr>
<tr>
<td>Relational model</td>
<td>49</td>
</tr>
<tr>
<td>Reluctance</td>
<td>95, 96, 114, 117, 129</td>
</tr>
<tr>
<td>Rennings, F.W.J.</td>
<td>79</td>
</tr>
<tr>
<td>Research approach</td>
<td>6</td>
</tr>
<tr>
<td>Research question</td>
<td>6</td>
</tr>
<tr>
<td>ROI, Return On Investment</td>
<td>60, 153</td>
</tr>
<tr>
<td>Rouse, W.B.</td>
<td>18, 43</td>
</tr>
<tr>
<td>RS, Real System</td>
<td>41</td>
</tr>
<tr>
<td>Rule</td>
<td>74, 75, 139</td>
</tr>
<tr>
<td>Rule-based level</td>
<td>68, 70, 93, 95, 96, 99, 115, 116, 138, 148</td>
</tr>
<tr>
<td>support on the</td>
<td>43</td>
</tr>
<tr>
<td>S-trip</td>
<td>115, 117</td>
</tr>
<tr>
<td>Sagasti, F.</td>
<td>12-14</td>
</tr>
<tr>
<td>Sage, A.P.</td>
<td>12, 43</td>
</tr>
<tr>
<td>Scanner</td>
<td>124</td>
</tr>
<tr>
<td>Schreiber, G.</td>
<td>33-35</td>
</tr>
<tr>
<td>Scott Morton, M.S.</td>
<td>1, 3, 28</td>
</tr>
<tr>
<td>Secondary function</td>
<td>44</td>
</tr>
<tr>
<td>Secondary objective</td>
<td>46</td>
</tr>
<tr>
<td>Index item</td>
<td>Pages</td>
</tr>
<tr>
<td>---------------------------------------------------------------------------</td>
<td>---------------------</td>
</tr>
<tr>
<td>Self report</td>
<td>59, 99, 138, 153</td>
</tr>
<tr>
<td>Seligmann, P.S.</td>
<td>23, 60</td>
</tr>
<tr>
<td>Shane, B.</td>
<td>3</td>
</tr>
<tr>
<td>Shortliffe, E.H.</td>
<td>86</td>
</tr>
<tr>
<td>Simon, H.A.</td>
<td>1, 11, 12, 17</td>
</tr>
<tr>
<td>SKE</td>
<td>35, 36</td>
</tr>
<tr>
<td>Skill-based level</td>
<td>69, 70, 93, 95, 96, 99, 116, 117, 119, 137, 138, 148 support on the</td>
</tr>
<tr>
<td>Software</td>
<td>49, 59</td>
</tr>
<tr>
<td>Sol, H.G.</td>
<td>4-6, 14, 23, 24, 36, 37, 60, 76, 80, 100, 138</td>
</tr>
<tr>
<td>Sommerville, I.</td>
<td>59</td>
</tr>
<tr>
<td>Source code</td>
<td>59, 79, 104, 128, 144</td>
</tr>
<tr>
<td>Sprague, R.H.</td>
<td>25, 49</td>
</tr>
<tr>
<td>SQL, Structured Query Language</td>
<td>49</td>
</tr>
<tr>
<td>Statement</td>
<td>79, 104, 144</td>
</tr>
<tr>
<td>Stoomwezen</td>
<td>68</td>
</tr>
<tr>
<td>Structure chart</td>
<td>79, 104, 128, 144</td>
</tr>
<tr>
<td>Structured English</td>
<td>49, 99, 123</td>
</tr>
<tr>
<td>Sub-task</td>
<td>41, 42, 48, 148</td>
</tr>
<tr>
<td>support of a</td>
<td>43</td>
</tr>
<tr>
<td>Subramaniam, M.R.</td>
<td>4</td>
</tr>
<tr>
<td>Supply of a circuit</td>
<td>91, 93</td>
</tr>
<tr>
<td>Support structure</td>
<td></td>
</tr>
<tr>
<td>of IPESS</td>
<td>121</td>
</tr>
<tr>
<td>of OSS</td>
<td>97</td>
</tr>
<tr>
<td>of PRESSTO</td>
<td>137</td>
</tr>
<tr>
<td>of Xpection</td>
<td>72</td>
</tr>
<tr>
<td>Systelogical description</td>
<td>51</td>
</tr>
<tr>
<td>method for constructing a</td>
<td>52</td>
</tr>
<tr>
<td>Systelogical design</td>
<td>51</td>
</tr>
<tr>
<td>method for constructing a</td>
<td>53</td>
</tr>
<tr>
<td>Systelogical problem</td>
<td>24</td>
</tr>
<tr>
<td>conceptual description in MEDESS</td>
<td>46, 47, 148</td>
</tr>
<tr>
<td>T-test</td>
<td>80, 105</td>
</tr>
<tr>
<td>Tas, P.A.</td>
<td>41</td>
</tr>
<tr>
<td>Task</td>
<td>42, 47, 148, 149</td>
</tr>
<tr>
<td>decomposition of a</td>
<td>55</td>
</tr>
<tr>
<td>dynamic aspects of a</td>
<td>48</td>
</tr>
<tr>
<td>static aspects of a</td>
<td>48</td>
</tr>
<tr>
<td>Task coordination structure</td>
<td>48, 149, 152</td>
</tr>
<tr>
<td>Task interface</td>
<td>50, 100, 149</td>
</tr>
<tr>
<td>Task precedence structure</td>
<td>48, 149</td>
</tr>
<tr>
<td>Task support structure</td>
<td>48, 149</td>
</tr>
<tr>
<td>Technical implementation</td>
<td>49</td>
</tr>
<tr>
<td>Technique</td>
<td>6</td>
</tr>
<tr>
<td>Technological description</td>
<td>51</td>
</tr>
<tr>
<td>method for constructing a</td>
<td>59</td>
</tr>
<tr>
<td>Technological design</td>
<td>51</td>
</tr>
<tr>
<td>method for constructing a</td>
<td>59</td>
</tr>
</tbody>
</table>

163
## INDEX

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Technological problem</td>
<td>24</td>
</tr>
<tr>
<td>concept description in MEDESS</td>
<td>49, 149</td>
</tr>
<tr>
<td>Technology push</td>
<td>37</td>
</tr>
<tr>
<td>Ter Bekke, J.H.</td>
<td>55</td>
</tr>
<tr>
<td>Tijmensen, A.G.H.</td>
<td>123, 128</td>
</tr>
<tr>
<td>Timesharing computer system</td>
<td>1</td>
</tr>
<tr>
<td>Todd, P</td>
<td>5</td>
</tr>
<tr>
<td>Tonge, F.</td>
<td>3</td>
</tr>
<tr>
<td>TPS, Transaction Processing System</td>
<td>1</td>
</tr>
<tr>
<td>Transformer section of a circuit</td>
<td>91, 93</td>
</tr>
<tr>
<td>TRV control of a circuit</td>
<td>91, 95</td>
</tr>
<tr>
<td>TRV, Transient Recovery Voltage</td>
<td>95</td>
</tr>
<tr>
<td>Turban, E.</td>
<td>3, 25</td>
</tr>
<tr>
<td>Turbine</td>
<td>135</td>
</tr>
<tr>
<td>Turbo Pascal</td>
<td>76, 100, 124, 140</td>
</tr>
<tr>
<td>Understanding phase</td>
<td>50, 149</td>
</tr>
<tr>
<td>Usage factor</td>
<td>66, 68</td>
</tr>
<tr>
<td>User interface</td>
<td>49, 149</td>
</tr>
<tr>
<td>Van der Ende, A.</td>
<td>49, 57</td>
</tr>
<tr>
<td>Van Heijnigen, R.H.C.</td>
<td>79</td>
</tr>
<tr>
<td>Van Poelgeest, R.</td>
<td>128</td>
</tr>
<tr>
<td>Van Rikxoort, R.P.</td>
<td>75</td>
</tr>
<tr>
<td>Van Walsum, T.</td>
<td>79</td>
</tr>
<tr>
<td>Van Weeldeken, J.A.</td>
<td>76, 80, 100, 138</td>
</tr>
<tr>
<td>VDI, Verein Deutscher Ingenieure</td>
<td>136</td>
</tr>
<tr>
<td>Versloot, J.</td>
<td>128</td>
</tr>
<tr>
<td>Vibration analysis</td>
<td>132</td>
</tr>
<tr>
<td>Friend, J.</td>
<td>128</td>
</tr>
<tr>
<td>Waern, Y.</td>
<td>16, 20</td>
</tr>
<tr>
<td>Waterman, D.A.</td>
<td>2, 29, 63</td>
</tr>
<tr>
<td>Watkins, P.R.</td>
<td>3, 25</td>
</tr>
<tr>
<td>Way of control</td>
<td>24</td>
</tr>
<tr>
<td>in MEDESS</td>
<td>60, 149</td>
</tr>
<tr>
<td>Way of experiencing</td>
<td>17</td>
</tr>
<tr>
<td>Way of modelling</td>
<td>24</td>
</tr>
<tr>
<td>in MEDESS</td>
<td>46, 148</td>
</tr>
<tr>
<td>Way of organizing</td>
<td>60</td>
</tr>
<tr>
<td>Way of thinking</td>
<td>23</td>
</tr>
<tr>
<td>behind MEDESS</td>
<td>41, 148</td>
</tr>
<tr>
<td>Way of transforming experience into knowledge</td>
<td>17</td>
</tr>
<tr>
<td>Way of working</td>
<td>24</td>
</tr>
<tr>
<td>in MEDESS</td>
<td>50, 149</td>
</tr>
<tr>
<td>Weghorst, P.M.G.</td>
<td>75</td>
</tr>
<tr>
<td>Weitzel, J.R.</td>
<td>32</td>
</tr>
<tr>
<td>Welke, R.J.</td>
<td>24, 50</td>
</tr>
<tr>
<td>Wielinga, B.</td>
<td>5, 31-35, 58</td>
</tr>
<tr>
<td>Wijers, G.M.</td>
<td>23, 60</td>
</tr>
<tr>
<td>Williams, A.C.</td>
<td>3</td>
</tr>
<tr>
<td>Window</td>
<td>50</td>
</tr>
</tbody>
</table>
Wirth, N. ................................................................. 49
Woods, D.D. ......................................................... 5, 16, 37
Workstation ......................................................... 59
Xpection ............................................................... 72-81
Zeidner, L. .......................................................... 3
Ziya Aktas, A. ......................................................... 49
Zwijnenberg, G.J. ................................................... 98, 104
SAMENVATTING

In dit proefschrift schenkten wij aandacht aan het verbeteren van de prestatie van een expert door middel van informatietechnologie. Wij hebben een ontwikkeling waargenomen waarin de concepten van een beslissingsondersteunend systeem en een kennisysteem geïntegreerd worden in het concept van een expert ondersteunend systeem (ESS). Vele organisaties zijn geïnteresseerd in het integreren van ESSn in hun informatiesystemen. Echter, men kan zich afvragen of aan de verwachtingen van deze organisaties in de praktijk voldaan wordt. We hebben waargenomen dat een aantal auteurs de veelbelovende mogelijkheden van het ESS concept onderschrijven maar dat zij tevens van mening zijn dat het ontwerpen van ESSn niet op effectieve wijze in de praktijk wordt gebracht. Deze waarnemingen hebben ons geïnspireerd naar te gaan hoe ESS ontwerpmethodieken het ontwerpen van een ESS ondersteunen.

In tien hedendaagse, methodologische bijdragen hebben wij een denkwijze, een modelleringswijze, een werkwijze en een beheerswijze trachten te identificeren. Het blijkt dat probleemopvattingen eerder vanuit het micro-perspectief beschouwd worden dan vanuit het meso- en het macro-perspectief, dat huidige situaties zelden dienen als referentie voor het ontwerpen en evalueren van ESSn, dat het systeem in monopolistisch en infologisch ontwerpprobleem weinig aandacht krijgen in vergelijking met het datalogisch en technologisch probleem, dat slechts enkele methoden en technieken voor het modelleren van ontwerpproblemen aangerekend worden en dat een beheerswijze vrijwel ontbreekt. Op basis van deze waarnemingen formuleren wij de hypothese dat het ontwerpen van ESSn op een effectieve manier in de praktijk gebracht kan worden door het gebruiken van een methodiek die aan de achtergestelde zaken aandacht besteedt. Om deze hypothese te testen hebben wij een denkwijze geformuleerd en gecompleteerd met een modelleringswijze, een werkwijze en een beheerswijze. De hieruit voortkomende methodiek heet "MEDESS", een acroniem van "Methodology for Designing Expert Support Systems".

In de denkwijze achter MEDESS kan een probleemopvatting niet alleen vanuit het micro- maar ook vanuit het meso- en het macro-perspectief beschouwd worden.

SAMENVATTING

Vanuit het meso-perspectief staan de doelstelling van een organisatie en diens prestatie centraal. Het ontwerpen van ESSn kan in een bredere context geplaatst worden waarin de doelstelling en de prestatie van een organisatie heroverwogen worden. Het is mogelijk dat men bestaande probleemklassen, individuen en taken herziet en nieuwe specificeert. Men kan ook de volgorde waarin taken worden uitgevoerd herzien. Pas als deze zaken zijn opgelost, komt het ontwerpen van ESSn aan de orde.

Vanuit het macro-perspectief staan de gemeenschappelijke doelstelling van twee of meer organisaties en hun prestatie centraal. Men kan deze organisaties als één organisatie beschouwen. Het ontwerpen van ESSn kan dan op eenzelfde wijze opgepakt worden als vanuit het meso-perspectief.

In de modelleringswijze van MEDESS onderscheiden wij een systelosdig, infologisch, datalogisch en technologisch ontwerpprobleem.

Het systelosdig probleem kan beschreven worden in termen zoals organisatie, probleemklasse, individu en taak. Deze concepten zijn essentieel voor het beschrijven van een problemsituatie vanuit een organisatorisch perspectief en voor het specificeren van de toegevoegde waarde van een ESS.

In het infologisch probleem bestaat een taak t uit vier elementen: de informatie die nodig is om t uit te voeren, de informatie die resulteert door het uitvoeren van t, het gedrag dat een expert vertoont tijdens het uitvoeren van t en de ondersteunings-structuur van een ESS waarin functies zijn opgenomen die de uitvoering van t kunnen ondersteunen of automatiseren. Een taak kan uitgewerkt worden in een precedentie structuur, een ER model van diens RS en een coördinatie structuur.

In het datalogisch probleem staat centraal hoe informatie gegroepeerd en verwerkt wordt, onafhankelijk van de technologie waarmee dit verwezenlijkt wordt.

In het technologisch probleem staan de technische implementaties van een database management system, een model management system en een gebruikersinterface centraal. De gebruikersinterface bestaat uit verscheidene taakinterfaces die elk de communicatie tussen een expert en een ESS ondersteunen tijdens de uitvoering van een sub-taak of een coördinatie taak.

In de werkwijze van MEDESS hebben wij een fasering gespecificeerd waarin wij een kenfase en een ontwerp fase onderscheiden. In deze fasen kunnen systelosdig, infologische, datalogische en technologische beschrijvingen en ontwerpen op iteratieve wijze geconstrueerd en herzien worden. Wij hebben methoden voorgesteld om het systelosdig en infologisch probleem op te lossen. Voor het datalogisch en technologisch probleem hanteren wij methoden die beschreven staan in de literatuur.
SAMENVATTING

De beheerswijze van MEDESS bewaakt de balans tussen de effectiviteit en de efficiëntie waarmee een ESS ontworpen wordt. Hierin onderscheiden wij de organisatie van een project en de voortgangsbewaking ervan.

Om MEDESS te testen hebben wij het toegepast in vier casusposities waarin KEMA een probleem wenste op te lossen dat betrekking had op de prestatie van een expert.

De eerste casuspositie betreft twee consultants die werken bij de afdeling WMK en inspectierregimes opstellen voor het onderhoud aan conventionele centrales. De hoeveelheid werk die gemoeid is met het verwerken van een aanvraag was aanzienlijk aan het toenemen. De tijd-efficiëntie van de consultants moest verbeterd worden. Wij hebben een ESS ontworpen dat hen kan ondersteunen met het verwerken van twee maal zoveel werk als dat zij momenteel kunnen verwerken.

De tweede casuspositie betreft vier proefleiders die werken bij de afdeling DZL en circuits ontwerpen voor het beproeven van vermogensschakelaars. Om de service van DZL naar cliënten toe te verbeteren moest het aantal fouten dat door de proefleiders gemaakt wordt, verminderd worden. Tevens moest hun tijd-efficiëntie verbeterd worden. Wij hebben een ESS ontworpen dat hen in staat stelt circuits driemaal sneller en vrijwel zonder fouten te ontwerpen.

De derde casuspositie betreft een planner die werkt bij de afdeling LTI en reizen plant waarin in geheel Europa elektrotechnische produkten en hun produktiefaciliteiten gekeurd en geïnspecteerd worden. De klachten van de inspecteurs met betrekking tot de relatief grote hoeveelheid reisuren die zij tijdens een inspectieris maken, moesten verholpen worden. Wij hebben een ESS ontworpen dat de planner helpt om sneller inspectiereizen te plannen die minder reisuren vergen.

De vierde casuspositie betreft de opvolger van het hoofd mechanisch onderhoud van een Nederlandse kerncentrale. Wij hebben ingeschat dat de opvolger, door een gebrek aan specifieke ervaring, moeilijkheden kon krijgen bij het diagnostiseren van de conditie van de turbine. Wij hebben een ESS ontworpen dat hem in staat stelt de conditie van de turbine beter en sneller te diagnostiseren omdat hij de beschikking heeft gekregen over de expertise van zijn voorganger.

Op grond van onze ervaringen kunnen wij de volgende conclusies trekken.

In twee van de vier casusposities bleek het voordelig een probleemsituatie vanuit het macro-perspectief te beschouwen. In de eerste casuspositie hebben wij de afdeling WMK geadviseerd de taak van het achterhalen van de onderhoudshistorie van een onderdeel aan de consultants toe te kennen in plaats van aan de medewerkers van een elektriciteitsbedrijf. In de vierde casuspositie hebben wij de kerncentrale geadviseerd de taken van het meten van trillingen en van het omzetten van trillingen in spectra toe te kennen aan de opvolger in plaats van aan een medewerker van de afdeling MAP.

169
SAMENVATTING

In alle casusposities bleek het interessant om een probleemsituatie vanuit het mesoperspectief te beschouwen. In de eerste casuspositie hebben wij voor de afdeling WMK de mogelijkheid geschapen om door te gaan met het afhandelen van alle aanvragen voor advies zonder daarbij meer consultants in te schakelen. In de tweede casuspositie hebben wij de afdeling DZL geholpen de kwaliteit van de service naar diens cliënten te verbeteren. In de derde casuspositie hebben wij de afdeling LTI ondersteund in het behouden van een aanvaardbaar werkklimaat. Tenslotte hebben wij in de vierde casuspositie de kerncentrale geholpen de conditie van de turbine zo goed mogelijk te blijven bewaken.

In alle casusposities hebben wij een probleemsituatie vanuit het micro-perspectief beschouwd. In de eerste casuspositie hebben wij de tijd-efficiëntie van de consultants verbeterd. In de tweede casuspositie hebben wij de tijd-efficiëntie van de proefleiders verbeterd en hun foutenpercentage gereduceerd. In de derde casuspositie hebben wij de tijd-efficiëntie van de planner verbeterd en zijn tegenzin om topografische informatie te raadplegen en om gebieden beter op te delen weggenomen. Tenslotte hebben wij in de vierde casuspositie de opvolger de mogelijkheid geboden dezelfde effectiviteit te behalen als zijn voorganger en bepaalde taken op een kosten-efficiëntere manier uit te voeren.

Wij concluderen dat door een probleemsituatie niet alleen vanuit het micro- maar ook vanuit het meso- en het macro-perspectief te beschouwen, wij meer mogelijkheden kunnen identificeren om de prestatie van een expert te verbeteren.

In alle casusposities bleek de conceptuele beschrijving van het systelогisch probleem een compleet en krachtig gereedschap. Probleemhebbers worden gestimuleerd om vanuit een organisatorisch perspectief tegen hun problemen aan te kijken, om belangen te onderscheiden en te vertalen in primaire en secundaire doelstellingen en om de prestaties in het verwezenlijken van doelstellingen vast te stellen. Alleen in de derde casuspositie stond een secundaire doelstelling centraal.

In alle casusposities bleek de conceptuele beschrijving van het infologisch probleem ook een compleet en krachtig gereedschap om een taak te beschrijven vanuit het paradigma van beperkte rationaliteit. Dit paradigma legt de nadruk op de procedurele aspecten van de manier waarop een expert een probleem oplost. In de tweede en derde casuspositie bleek vooral het concept van een coördinatie structuur hiertoe de juiste handvaten te bieden. De visie van Rasmussen om onderscheid te maken tussen een skill-based, rule-based en knowledge-based niveau van probleemoplossend gedrag, bleek gemakkelijk in de praktijk te brengen en ondersteunde ons bij het vaststellen van het type functie waarmee een taak het beste ondersteund kan worden.

170
De fasering in MEDESS, bestaande uit een kenfase en een ontwerpfase, bleek met name op te gaan voor het systeologisch en het infologisch probleem. Voor wat betreft het datalogisch en technologisch probleem hebben wij ons voornamelijk gericht op het ontwerpen van oplossingen.

De methode voor het construeren van een systeologische beschrijving is volledig gevolgd. De methode voor het construeren van een systeologisch ontwerp is slechts gedeeltelijk gevolgd. Wij hebben ons hoofdzakelijk gericht op het herzien van prestaties van een organisatie en/of een taak. Met name in de eerste en de tweede casuspositie waren wij in staat de toegevoegde waarde van een ESS aan te tonen. Dit is essentieel. Immers, een ESS dient een bijdrage te leveren aan het behartigen van de belangen van een organisatie en van de individuen die deze te werk stelt. In de derde casuspositie was dit een stuk moeilijker. Hierin hadden wij eerder te maken met een secundaire dan met een primaire doelstelling. Persoonlijke belangen zijn erg subjectief en zo ook persoonlijke doelstellingen. Het meten hiervan kan tot sterk speculatieve en omstreden resultaten leiden. Voor wat betreft de primaire doelstellingen van een organisatie, concluderen wij dat door het construeren van een systeologische beschrijving en ontwerp, men in staat is uitgangspunten te definiëren om een ESS te ontwerpen en de toegevoegde waarde ervan aan te tonen.

Voor wat betreft de methoden die wij voorgesteld hebben om het infologisch probleem aan te pakken, kunnen wij zeggen dat de methode voor het construeren van een beschrijving volledig is gevolgd. De methode voor het construeren van een ontwerp is slechts gedeeltelijk gevolgd. Wij hebben ons hoofdzakelijk beziggehouden met het ontwerpen van ondersteuningsstructuren. Wij concluderen dat de voorgestelde methoden compleet zijn en ons ondersteunen bij het identificeren van mogelijkheden om de beperkte cognitieve mogelijkheden van een expert uit te breiden.

Als elicitatietechnieken hebben wij introspection en self report toegepast. De eerste techniek is toegepast in situaties waarin een expert zonder veel aarzeling aan kan geven hoe hij een taak uitvoert. In andere gevallen hebben wij self report toegepast.

In een beheerswijze onderscheiden wij de organisatie en de voortgangsbewaking van een project. De looptijd van de projecten varieerde tussen acht maanden en anderhalf jaar. Wij hebben het aantal manmaanden besteed aan elk ontwerpprobleem aangegeven. De kosten van de projecten zouden bij professionele tarieven tussen NLG 225.000 en NLG 450.000 gelegen hebben. Op basis van deze bedragen zouden de terugverdienperiodes van de ontworpen ESSn tussen twee en drie en een half jaar gelegen hebben.

Op grond van onze ervaringen kunnen wij de geformuleerde hypothese niet verwerpen.

171
CURRICULUM VITAE