Chapter 8

Summary and Conclusion

The method presented in the previous chapter is a solution to the original problem statement. We have a method of support: 'Aiding', and explicit psychological requirements and constraints for this type of knowledge enhancement. To design an Aiding function, we have a software engineering method that helps to meet these requirements and constraints.

This chapter summarizes the steps towards this result, provides a concise overview of the result, and reports on follow-up research.


Often humans want to perform tasks, but lack the knowledge to optimally perform the work. Recently, cooperative problem solving (Terveen, 1995), is gaining more and more recognition as an alternative to technology-driven approaches. In cooperative problem solving, user and system supplement and complement each others knowledge and capacities.

This thesis studies cooperative problem solving, describes a number of prototypes, and reports conclusions regarding computer support based on cooperative problem solving. Throughout this thesis, two aspects were distinguished. The first is the question which type of support would be beneficial for users lacking domain knowledge. The second is the question which method can be followed to design and implement such support in a particular application domain.

Part I: Automating Consultation

Part I explores the principle that design of computer support can be based on imitation of human-human support, that is, human consultancy. For software-design aspects, part I was based on KADS, a model-based software-engineering approach. In the StatCons case-study two prototypes of an automated statistical consultant were developed.

Statistical design is a complex task. Breuker (1994), distinguishes three types of design tasks: parametric design in which only the values of parameters need to be determined. Configuration design in which a relation or structure has to be determined, and 'real design' in which both the components and a structure need to be determined. A statistical design can seldom be described by simple parameters. For
allocation of units to different conditions complex structures have to be taken into account. Because in a statistical design new components may be introduced, it belongs to the last, most difficult category. In the discussion of design tasks the following terminology is employed. A design problem comprises: Requirements, Constraints, Design Solution, and Flaws. Flaws are inconsistencies or conflicts among the first three.

**StatCons-0** Chapter 2 designs a certain task distribution between system and user, in imitation of dialogues from statistical consultation practice. The system behaves as a critic of the statistical design of the user. This task distribution comprises three steps. First the user enters Requirements and Design Solution. Second, the system identifies Flaws, and third, the system reports and explains the Flaws to the user.

During the design of the system, detailed analyses were made, models were drafted and re-drafted, and choices were made regarding the knowledge representation of the objects in the statistical domain. These were the basis for the implementation of the StatCons-0 prototype. In the dialogue with the user, the system takes the initiative by asking a question, and the user provides an answer. Users have no difficulties regarding the protocol of such a dialogue. The difficulty, beside understanding certain questions, lies in the rigidity of the dialogue. It is not possible to repair mistakes or change things. The rigidity implies that the user is forced in a straitjacket that may not correspond with the goals of the user.

Furthermore, the comment or criticism by the system should be more than a response to flaws, but should also provide suggestions for repairing flaws by changing the entire design problem, and these modifications should be supported by the system. In the representations used in the prototype one can edit, but consistency cannot be ensured. The representations used are perhaps not the most suited to support simple, consistent modifications of the design problem.

Furthermore, the human-computer dialogue seems rather long; the representations chosen are perhaps not the most concise ones.

**Well-Defined Models** Chapter 3 provides a basis for knowledge representation of objects in the statistical domain. If we use well-defined models of data sets, we have a solution for simple, consistent changes of the entire design problem. Well-defined models also help to classify and order the various user views of the design problem which may be used in different tasks. Models can be ordered on specificity and compression. Together these provide a ‘two-dimensional’ classification of models at different levels of abstraction and conciseness. The meaning of more compressed models can be explained using less compressed models.

**StatCons-1** Chapter 4 presents a second study of the task and the task distribution in cooperative statistical design. A new propose-revise model for design is the basis for a different kind of task distribution. The application task is viewed as an iterative process. Design is not straightforward, and along the way, at any time, changes and adjustments can be made to the design problem, without having to start all over again. Using well-defined models of chapter 3, the domain can be modelled such that revisions and modifications can be a normal part of the design process. In such framework, user or system can exercise each function (i.e., propose, evaluate, modify) at any time during the design process.
A new prototype, StatCons-1, is suited for a highly interactive, collaborative design process. Using its graphical user-interface, the user can enter a specification and have the system generate parts of the solution, or enter his own solution, and have the system act as a critic. This prototype is very flexible, the user can adjust the design problem at any time, but now another difficulty appears. The initiative is in the hands of the user, and the user often is not sure what to do next. The system supports interactive task execution, but offers not enough support for the user lacking knowledge. Especially initiative in the task execution seems important, and in StatCons the initiative is fully in the hands of the user. Therefore, there seems to be lack of guidance for the user.

Conclusions Part I

At this point we do have an advanced and flexible system, but we do not have adequate support. We have insufficiently addressed the user, his work situation and his mental workload.

For the functioning of the user, computer support may have benefits, but certainly it has costs and there is a complicated trade-off (cf., Rouse, 1988). Design of support based on imitation of human-human interaction ignores the increased overhead in human-computer interaction. The extra overhead originates at all levels of interaction: First there is a lack of communication bandwidth in the interaction using screen, mouse and keyboard. Second, there is a mismatch between the languages used by the system and by the user, and among users there can be large differences. The third is a lack of mutual adaptation, the user is supposed to adapt to the system, and this provides the user with additional work. More generally, compared to human-human support, computer support brings the user more additional difficulties and additional mental workload, at various levels, from pressing the right key to transforming problem representations to articulating goals. Imitation of human-human practice cannot be a real basis for design. Too many new elements are introduced, and each of these contributes to the load on the user. To design effective support, care must be taken that the outcome of the trade-offs regarding the user's work situation is positive.

This is difficult. Although there are many architectures, guidelines and standards, there is not much research into the conditions in which a user lacking knowledge might be able to profit from computer support. We have a real problem here. On the one hand there are technological means and design choices. On the other hand there are users who want to perform tasks, and who have certain characteristics, but little is known regarding performance as a function of design choices and user characteristics. The experience with the prototypes, and a user-psychological interpretation of the results, suggest that initiative and level of knowledge of the user are important factors in the outcome of the trade-off. Lack of knowledge is not only detrimental to performance in tasks, it is also detrimental to the efficacy of support.

Workbenches The line of development in the prototypes is paradigmatic for many projects. One starts with a simple question-answer interface that takes the initiative and thus provides strong guidance. The next step in the development is an extremely flexible ‘workbench’ with graphical editors. The user then has the initiative to work on design documents and the system provides no steering or guidance whatsoever. This
happened in the KADS project with Shelley en KEW (Anjewierden, Wielemaker & Toussaint, 1992; Terpstra et al., 1993).

These workbenches have two characteristics which make them hard to use for users lacking knowledge. First the user is supposed to recognize the objects and operations in the domain, on screen. Second, the user himself must decide what to do next, as he has the initiative. Both tasks are especially difficult for a user lacking knowledge. A workbench is therefore more suited to experts than to novices.

Fischer's system for kitchen design (see chapter 1), is a-typical, as there will be no reader that is not an expert in kitchens. All users will be able to recognize the objects and be aware of the operations that can be performed. This is not the case with the typical workbench. It usually is not that obvious, what the objects on the screen mean, let alone that it is 'intuitive' what is supposed to happen next.

The explanation for this line of development is the availability of new technology for graphical user interfaces, allowing such workbenches to be built for the first time, with the hope that the new graphical user-interface technology would solve most if not all problem in human-computer interaction. In the world of educational systems this technology led to open, explorative learning environments. Although even there it is recognized that the user needs a certain amount of guidance or support, there is no acceptable solution yet in terms of a user interface that provides adequate support, but is not too rigid. One question is for example the level of specificity of the guidance: is it only a global structuring of the task or perhaps also a detailed instruction with specific domain knowledge?

Lack of Knowledge The problem with the user lacking knowledge is that the efficacy of computer support is impeded by his lack of knowledge. This user has goals, but does not know how to elaborate these into action specifications (Norman, 1986). That is why he needs guidance, but wrong guidance or guidance that is too rigid may conflict with the goals of the user. The question is whether it is possible to find a form of support that provides sufficient support, without becoming too rigid. To make this puzzle more difficult, the user lacking knowledge is less reachable for information on the screen. Less is recognized, less is understood, there is less overview and less orientation. This greatly reduces the possibilities for indirect control by providing information or special design of the environment.

Solutions can be searched in hybrid mixtures: Question-answer dialogues with a possibility of escape to a graphical editor, workbenches with wizards, and so on. However, one may have to search a long time. Search can be more directed, and some insight into the shape of the search space may be gained by a better analysis of the Requirements and Constraints stemming from the psychology of the human task executor lacking knowledge.

Part II: Cognitive Support

The previous part provides a solution for representing statistical knowledge, but it does not solve the problem in which manner the user lacking knowledge can be supported adequately. The task execution by the user is critical for the efficacy of the overall system, and the trade-offs associated with the user’s role determine the effect of the support based on cooperative problem solving. Existing human factors methods and techniques are not geared to the analysis and design of the joint execution of
expert tasks. What seems to be needed is a cognitive task analysis (Woods & Roth, 1988).

Many existing system design methods, including KADS, provide methods and techniques for the modelling of the system, but little means for the analysis of the task distribution and the role of the user. To make such user analysis model-based, there is a need for models of collaborative work processes. In part II such methods are investigated and the implications of putting the user in the center are made explicit and operational in a method for the design of a support method called ‘Aiding’.

Analysis of Cooperation Chapter 5 presents a first version of a method for model-based system design. An overview is provided of the concept of cooperative problem solving, and of means for the analysis and design of cooperative systems. Hierarchical task analysis, task distribution and user analysis are the main ingredients.

Task distribution can be represented by partitioning data-flow diagrams, and the partitioned diagram immediately indicates which communication tasks are involved in a particular distribution. This well-known modelling technique is supplemented with initiative markers, showing whether user or system will have the initiative for communicating the data, since the locus of control appeared to be so important. These can be used for a better, more detailed analysis, compared to standard techniques for analysis and allocation of tasks.

The design process has to be organized differently, to take information about the user into account, and to accept that there is so much uncertainty regarding the effects of computer support. The uncertainty is partly due to lack of knowledge about the effects of various forms of computer support, but is partly inevitable. We will never be in the position to predict with absolute certainty whether users will be able to profit from a specific application in particular domain. Generation of a candidate system design and evaluation with respect to usability are essential elements in the design process. Nevertheless, further investigation and development of a practical theory will help to reduce uncertainty and help to find designs users can profit from. This thesis is a step into the right direction, a contribution to practical theory.

The models for the task distribution describe the roles of system and user, and are based on hierarchical task analysis. They are therefore rather coarse grained. One can define functional decompositions and communication tasks, but one cannot describe the dynamics of the joint process. It is attempted to solve this by analysing the more abstract task of managing collaborative work. It is possible to use the same diagram techniques for abstract tasks such as negotiating and other cooperative phenomena, but this does not provide an increase of the expressive power of the diagram language. More grip on the description of the process is needed.

Models of Cooperative Work Chapter 6 surveys languages, models and techniques for the description of interactive, collaborative work processes, at different levels of specificity. These can be used for protocols in the application domain, and for protocols at the more abstract ‘managerial’ level. This provides the means for modelling roles or task distributions in complex cooperative work processes, but it does not yet tell which model, which role or task distribution, constitutes effective computer support.
Cognitive Support: Aiding  Chapter 7 uses the means from the previous chapters to design a task distribution that is not guided by imitation of human support, but that is based on a simple idea. If the problem of the user is ‘lack of knowledge’, why not intervene by offering the knowledge that is lacking. From a trade-off perspective, the benefits of such Aiding are in the enlargement of the users knowledge, and the costs are in the additional work and workload in utilizing the support.

The detailed requirements for such an Aiding function were derived from psychological and human-computer interaction literature. For the design of support that meets such requirements, a complete method is provided. In this method for system design the knowledge acquisition is very much controlled by constraints and requirements stemming from the user. In the method, existing systems, prototypes and even simulators can be used for a realistic simulation of the user’s work environment, and for knowledge acquisition from experts.

For a realistic approach, we start with real users. The design method comprises, among other things, an assessment of the needs for support. Chapter 7 starts with users from a locally used program for statistical analysis. We first use modern user-interface technology to bring the user interface up-to-date. Then, in this environment, we design and evaluate computer support for users lacking knowledge of the statistical analysis method. Experiment shows users with a not too large lack of knowledge can profit from the support.

With the Aiding interface there is a solution for the problem formulated in the first chapter, and, partly, for the general problem noted at the close of part I: the dilemma between guidance and flexibility. With the Aiding function we have an interface that provides a strong guidance, but which is added as a non-compulsory instruction-action window to the base system. In this manner, it is more a helping hand for the intelligent user than a rigid harness in action. And it is shown that users lacking knowledge, and even users lacking capacity (Neerincx & De Greef, 1998), can profit from this type of support.

8.1 Main Results

This section lists the main results of this thesis.

A) Cognitive engineering of support  The method for design of computer support based on cooperative problem solving has special characteristics.

- The design method is a complete operational procedure that tells exactly which steps to take to design support that meets these requirements.
- The design method is integrated with software engineering methods and techniques.
- The design method applies user analysis at different points for different purposes:
  - to ensure the base system for ‘experts-in-ideal-conditions’ is flawless.
- for detailed analysis of user requirements, using the base system (cognitive task analysis, user performance analysis).
- for knowledge acquisition, using the base system.
- for the final evaluation, sometimes called 'summative' evaluation, using base system + support function.

The individual elements of which the design method is composed are not new and can be found here and there in other approaches, but the method as a whole, compared with other approaches, is comprehensive. The method provides a concrete and operational version of the global cognitive engineering approach advocated by Woods & Roth (1988) that is simple and complete. Except that the design method can be used to design an application, it can also be used to investigate human performance models and the effects of support in realistic task settings. An example of such a follow-up investigation is given in the next section.

B) Models for Cooperative Work Chapter 6 provides modelling approaches for multi-agent protocols. These have a role in the design method for support based on cooperative problem solving, but these specification methods are much more general. The meeting protocol in chapter 6 is an example of a more general application.

C) Well-Defined Models The concept of well-defined model in chapter 3 is a fundamental contribution to knowledge representation and interoperability of software systems. Well-defined models are clear and unambiguous representations. The concept of well-defined models provides for consistent modifications and it provides a better insight into the design space for representations. Well-defined models are therefore a good basis for cooperative problem solving of a highly-interactive, iterative form.

It is known for quite a while that homomorphisms can be used to define abstractions of properties (e.g., Time — Timepoint) such that certain operations are preserved (e.g., Coombs et al., 1970). Chapter 3 shows that different types of homomorphism can be used to define abstractions of complex structure in a data set.

Another important advantage of the 'well-defined models concept' is the explicit separation between the choice of aspects that need to be considered in the meta data (these are included in the central data set concept), and the choice of the representations (model languages) to be used in task execution by users or software modules. This disentanglement facilitates the design of representations and the interoperability of systems using different representations.

Well-defined models do not solve everything. The StatCons-1 prototype in chapter 4 is based on well-defined models and on a 'propose-revise' model for iterative design. The value of KADS models is that they suggest meta classes or roles such as: Requirements, Design Solution, and Constraints, that can be used to add meaning to a well-defined model, as to indicate its role in the execution of a certain task. To use this added meaning correctly, these roles or meta classes must be taken into account in models of problem-solving tasks. KADS models do so explicitly.

Methodological Knowledge. In the research areas of statistical experts systems and meta-data, one of the main problems is the representation of methodological knowledge (Adèr, 1995). Adèr proposes a new notation, F-notation, for methodological
knowledge. It is based on a number of primitives, including time. F-notation, as yet, is an empty formalism; no interpretation rules are given and there is no direct semantic relation with data sets. F-notation can, potentially, be used for anything, but the disadvantage is that there is no grounding. Some examples of F-diagrams are about a data set, and these can be interpreted as well-defined models, but this relationship is not made explicit.

This thesis claims that the well-defined models concept can be used to establish a direct semantic relation with data sets and operations on data set. Representations of methodological knowledge can be devised as well-models of a central data set, and the well-defined models concept provides a basic solution for questions about representation and adaptability.

Chapter 3 shows that well-defined models allow for the use of different notations simultaneously, and provide adaptability as a fundamental characteristic, even for heterogeneous sets of models. Based on the concept of well-definedness, heterogeneous model languages can be related, compared and sometimes ordered. In the well-defined models concept, the central data set concept acts as the Esperanto through which all languages may be related. Additions to the data set concept allow the introduction of new topics in models of this central data set. Therefore, the well-defined models concept is suited to integration of a broad spectrum of representations and notations, including those people are actually using. Well-defined models seem to possess many more desirable characteristics, and open up a whole universe for further study.

Meta Data. Statistical offices collect and disseminate data. The common European statistical office EUROSTAT has promoted research into meta data and meta-data management. The Modelling Meta Data project (Darius et al., 1993) and Froeschl (1997) all explicitly consider operations on data sets and view data sets as three-dimensional structures in Units x Variables x Timepoints. Almost always, with some benevolence, their meta data may be interpreted as well-defined models, but the relationship between data and models is not made the focus of attention. Chapter 3 did and came up with well-defined models with respect to a central data set concept. The data set concept in Chapter 3 can be extended, and the formulation of requirements and characteristics of well-defined models is kept at a general level. Therefore this thesis shows general principles on which such meta data management can be based. The work on prototypes by Boucneau (in Darius et al., 1993) and Froeschl (1997) can be taken as evidence of the soundness of these principles. Adhering to these principles more explicitly and systematically may help to make data swamps more manageable.

8.2 User Support

This section summarizes the current understanding of computer support for users lacking knowledge, taking follow-up research into account.

Even the most naive theory about support has two parts. A performance model, for example: ‘Performance increases with an increase of available knowledge’, and a practical theory, for example: ‘A knowledge-based system application can help to increase the level of available knowledge’. From these follows the prediction that the knowledge-based system application can help to increase the level of performance. In Part I, and in Post (1996), this prediction failed to materialize, the practical theory being the culprit. It became apparent that in the end it is the user who does the
performing, and to better understand and predict what will happen, we need a human performance model and a practical theory relating to human performance.

Chapter 7 showed that even a rather simple model of human performance can help to derive Requirements to be satisfied by the Design of the support function. Chapter 7 provides a method to design support that meets these Requirements. In follow-up research (Neerincx & De Greef, 1998), the human performance model and the effects of support were investigated in more detail. Below a summary is provided of the human performance model and the follow-up research.

**Human Performance Model** Chapter 1 posed a human who has a goal, wants to perform tasks to achieve the goals, but lacks the knowledge to decide about the best action. We can now look at such a situation in more detail. With knowledge, it is certainly not only the presence or quantity that matters, but also its type, the processing associated with that type, and the capacity required for such processing.

With increasing expertise, the type of knowledge and processing changes from declarative to procedural (Anderson, 1990), respectively from knowledge-based, through rule-based to skill-based (Rasmussen, 1983, 1986). Therefore, with increasing expertise, less capacity is required to reach the same level of performance. A task that requires a large effort from the novice can be performed by the expert without much effort. The availability of directly applicable knowledge and the available processing capacity are considered to be the main factors that determine human performance.

![Available Knowledge + Available Capacity = Performance](image)

**Figure 8.1: A human performance model.**

In general, the more knowledge is available that is directly applicable, that is, the more expertise, the better the performance. There is also a positive effect of available capacity on performance. This is indicated by the plus signs in Figure 8.1. To get the complete picture, one needs to look at the joint effect of capacity and knowledge. For a fixed task, performance can be regarded as a response surface of capacity x expertise. Contours of equal performance contain the various combinations of capacity and knowledge that give rise to the same performance level. Limited knowledge can be compensated by a large amount of capacity and vice versa, but the human task executor can never do without a minimum of capacity, even if he has all the knowledge. Also, even an abundant capacity cannot function without a minimum amount of knowledge. Towards these limits (zero capacity or zero knowledge), the return of substitution diminishes. Because of this, equal performance contours are curved as shown in Figure 8.2.

The shape of the curve is is consistent with the empirical evidence (cf. Rasmussen, 1984): knowledge-based processing requires an extremely large capacity, rule-based processing requires much less capacity. Skill-based processing has the lowest capacity.
requirements, but the decrease is less than in the transition from knowledge-based to rule-based processing. Human processing capacity is limited (Miller, 1956; Just & Carpenter, 1992). For a user low on expertise, the capacity required to attain a certain level of performance may exceed the maximum available capacity.

This is the human performance model adopted here.

**Practical Theory 1** Chapter 7 is based on a human performance model as discussed above, and a simple practical theory: Support by offering knowledge enhances the available knowledge and diminishes the available capacity (practical theory 1). Support can thus lead to an improved performance, provided that the negative effect on available capacity is not too large. It was assumed that the effect of support on capacity would always be negative and should be kept as small as possible, to avoid outweighing the potential benefits.

Offering knowledge tends to have a positive effect on available knowledge, but the size of this effect also depends on the amount of knowledge the user already has available. If the user has hardly any knowledge available, he may not be able to understand the knowledge offered (apart from shortage of processing capacity). If the user is an expert, the knowledge offered may contribute little to the knowledge the user has available. Therefore, the expected improvement is negligible at the extremes of the knowledge or expertise dimension. Figure 8.3 shows the theoretically expected effect on available knowledge given the level of knowledge the user initially has available. The results of experiments in Chapter 7 confirm that for users low on expertise the effects of offering knowledge on performance and learning are negligible. For users with intermediate levels of expertise there are positive effects on learning and performance.

Figure 8.4 summarizes the current view on the working of the support function, taking follow-up research into account. It contains the human performance model discussed above, and external influences on availability of knowledge and capacity. The understanding of the effect of offering knowledge on capacity has evolved. In
Chapter 7 it was assumed to be negative, but the effect on capacity was not investigated. The follow-up study below showed that the effect on capacity can be positive. It is annotated with a question-mark in Figure 8.4.

**Practical Theory 2** Figure 8.4 also contains the effect of distractions. In real life there are uncontrolled distractions that have a negative effect on the performer's capacity. In investigations of the effect of support, use can be made of subsidiary tasks (Brown & Poulton, 1961), as a means of influencing the available capacity. Neerincx & De Greef (1998) used such a secondary task, to assess effects on capacity. This is an elegant and efficient method compared to physiological measurements of mental workload. In experiments it was shown that the Aiding function has a positive effect on the available capacity. Accordingly, the question mark in Figure 8.4 can be replaced by a plus sign. A possible explanation is that the support can help the user in following a different type of task execution, with lower capacity requirements.

In summary, it is possible to design support functions that enable the user to function as if he has better knowledge, leading to better solutions and less errors, and increased capacity, allowing the user to keep up with a higher pace, or to complete the work faster.
In Conclusion. Aiding as a form of knowledge enhancement is a general concept, based on psychological requirements. It is known that human performance models (Rasmussen, 1983, 1986; Kieras & Meyer, 1994) can provide real predictions. Here it is shown that even a simple human performance model can help to design better support. A human performance model provides more detail about the factors that affect performance, it suggests problems the human task executor may have, and it provides an opportunity for theorizing about the assumed working of a support function. Knowledge and capacity are two very important factors, but certainly not the only ones that affect overall performance.

The design method is suited to design Aiding functions and to design other forms of support. The method provides an efficient procedure to design high-quality support that enhances the user's knowledge and that can off-load the user. In the method, user analysis does not only serve to test usability, but to get information about the knowledge and capacity of users during task execution, and to develop support that supplements the user's knowledge and capacity.

Therefore I claim that the design method is a constructive contribution to the software design process, in contrast to the current practice of usability testing, which tends to provide criticism only. The method is efficient and has been elaborated to a single operational procedure that helps to design better applications.

The design method and models for human-computer cooperation also provide a clear and operational basis for further research and development of human performance models and practical theory about the workings of computer support. The design method is an efficient and valid method to investigate performance theories in experiments under realistic yet controlled conditions. Realistic simulation appeared even possible in an investigation of handling of emergency calls (Post, 1996). The assessment of effects on workload using a secondary task is elegant and unobtrusive compared to physiological measurements, and the workings of the support can be studied in more detail. In such a research framework (see also Neerinckx & De Greef, 1998), the design method can help to improve our knowledge and understanding of computer support.

Therefore I also claim that the design method is a constructive contribution to the research methodology of a psychology of human mental work in realistic settings.

The success of the design method is in the integration of the psychology of the human task executor with important methods and techniques from software engineering that can be used to analyze and design cooperative tasks and knowledge. In this manner, theoretical psychological concepts are made operational in the context of software design.