Computer Support by Knowledge Enhancement, Constraints and Methodology

de Greef, H.P.

Citation for published version (APA):

General rights
It is not permitted to download or to forward/distribute the text or part of it without the consent of the author(s) and/or copyright holder(s), other than for strictly personal, individual use, unless the work is under an open content license (like Creative Commons).

Disclaimer/Complaints regulations
If you believe that digital publication of certain material infringes any of your rights or (privacy) interests, please let the Library know, stating your reasons. In case of a legitimate complaint, the Library will make the material inaccessible and/or remove it from the website. Please Ask the Library: https://uba.uva.nl/en/contact, or a letter to: Library of the University of Amsterdam, Secretariat, Singel 425, 1012 WP Amsterdam, The Netherlands. You will be contacted as soon as possible.

UvA-DARE is a service provided by the library of the University of Amsterdam (http://dare.uva.nl)
Chapter 4

StatCons-1: Highly Flexible Support

This chapter continues the exploration of the hypothesis that users with lack of knowledge can be supported by automating the human consultant. It presents the design, implementation and evaluation of the StatCons-1 prototype, a system with an advanced graphical user interface that allows highly interactive cooperative problem solving.

This chapter first presents a general KADS model of iterative design in which modifications of Design Solution, Constraints and Requirements are a normal part of the design process. This empty model can then be filled-in, using well-defined models from the previous chapter. Well-defined models help to make the interrelationships among the objects in the statistical design task explicit and they provide a solution for consistent modifications of Design Solution, Constraints and Requirements.

The StatCons-1 prototype is then being reviewed from the point of view of the user and from the point of view of KADS.


Introduction

Chapter 2 presented a first study of computer support based on imitation of the human consultant. The KADS approach for model-based software design was used to guide the knowledge acquisition, design and implementation of the StatCons-0 prototype. In the knowledge acquisition, task analysis, analysis of domain concepts and analysis of expertise in action were used to produce a model for the prospective software system. The prototyping exercise showed that a separation between knowledge acquisition, and subsequent design and implementation of the software system, is feasible. The statistical domain however, is complex and the support provided by the StatCons-0 prototype — criticism or feedback on the user’s statistical design — leaves a few things
to be desired. These were formulated positively as new requirements for the support system (page 31). Taken together, these new requirements point to a more iterative and interactive process of statistical design also known as cooperative problem solving (Fisher, 1990, 1993), or collaborative problem solving (Terveen, 1995).

The system in chapter 2 is based on a procedural model of the consultation task. This model is simple to build, but it is too rigid to support an iterative design process in which design solution and requirements can be modified at any time. The difficulty in building such a model is that requirements and design solutions in this domain are objects with a complex structure. The problem is how these can be represented, in such a manner that consistent modifications are simple.

This chapter presents a new prototype, aimed to overcome the difficulties revealed by the first prototype. In the StatCons-0 development, KADS was used to draft a model of the consultation task. In the StatCons-1 development again a KADS analysis is used, but now aiming at an iterative model of the design task rather than the consultation task.

The StatCons-1 system is based on a new KADS interpretation model for iterative design and on the well-defined-models concept (WDMs) of chapter 3. Both are essential. They are complementary.

WDMs. In statistical design, there is a perhaps unique advantage: all objects in the domain are models for data sets. Chapter 3 explained that if we use well-defined models of data sets to represent objects in the domain, interrelationships among models and effects of modifications on the various models can be made completely transparent. An important result at this level is that model and data set can be consistently modified by using corresponding operations. Well-defined models can thus support modifications in an iterative design process.

If we look at various languages for WDMs and their suitability for representation of objects in the statistical domain, something remarkable appears. The languages provide a means to formulate models of the data set and to consistently modify these expressions, but they fail to capture an important aspect of models as used in statistical design and analysis tasks. The same formula or expression (e.g., a table with counts), can play different roles in the task. It may stand for prior knowledge or for willful selection (bias) by the investigator or it may represent a carefully construed statistical design, or it can be one hypothesis among the set of alternative models, or it can be the final best model that is the conclusion of the statistical analysis. From the language used to express the model alone (e.g., the table, the expression, the formula), one cannot tell which role it plays.

Roles or Meta Classes. Especially this aspect is addressed by KADS: objects in the domain have different roles in the problem solving process. These roles — meta classes, as they were called at the time — are provided by the KADS interpretation model. Using the interpretation model for design, the three major roles are: Requirements, Constraints and Design Solution. Restricting ourselves to WDMs, we can decide which models and languages shall be used in each of the three roles, and the relations among these can be made explicit. We thus need both the interpretation model and the well-defined models. They are complementary.

Iterative Design. The new interpretation model for design resembles the one used in chapter 2, but it makes no assumptions about the course of the design process. According to the model, the design task succeeds if in the end there is a triple (Requirements, Design Solution, Constraints), such that the Design Solution is derivable.
from the Requirements and does not violate the Constraints. Modifications are considered a normal part of the design process, but the interpretation model does not tell how the objects can be represented or modified.

Section 4.1 will first present the new interpretation model for iterative design. Then in Section 4.2 we will start looking for objects and representations to be used for the Requirements, Constraints and Design Solution, just like in chapter 2, but now we know that we have to look for well-defined models.

Chapter 3 also provided a classification scheme to sort available languages for well-models. Using the two-dimensional classification scheme, we can make a selection of languages. Having these languages, we can investigate the relations among the objects. This then provides the basis for a complete system for iterative statistical design (the final KADS expertise model).

Section 4.3 presents the prototype support system. An advanced user interface is designed to allow user and system to pool their knowledge and skill in the cooperative execution of the design task. After an evaluation and comparison of the StatCons prototypes, section 4.4 discusses imitations of human-human support as an approach to the design of computer support for users with lack of task knowledge.

4.1 A Propose-Revise Model for Design Tasks.

In design tasks in general, three kinds of objects are important: Requirements, Constraints and Design Solution. In a general terminological framework provided by Wielinga et al. (1995a), design starts with an analysis of informal ‘needs and desires’ and constraints, yielding formal requirements (Requirements), and formal constraints (Constraints). From then on, a synthesis process derives a candidate design solution from the formalized requirements. This design solution is tested against the formalized constraints. If the candidate solution does not violate any of the constraints, the design task succeeds. Successful completion of a design task thus results in a set of these three objects such that the Design Solution is derivable from the Requirements and does not violate the Constraints. This general definition of design tasks is attractive because it only states when the design task succeeds and does not make any statement about the process by which this result is achieved.

![Diagram of proposed revise model](image)

Figure 4.1: The result of a successfully completed design task.

Iterative models for design tasks are common in A.I (Chandrasekaran, 1990; Fis-
cher et al., 1993; Motta et al., 1994; Schreiber et al., 1994). These models are all variants of the ‘propose and revise’ model developed by Marcus & McDermott (1989). ‘Propose’ takes the Requirements to produce a candidate Design Solution. Testing compares the candidate solution with the Constraints. If, in testing, it appears that the candidate solution violates a constraint, one may try to adapt or revise the candidate solution, towards a better one.

Propose-and-revise may be performed in sequence: ‘complete model and then revise’ (Motta et al., 1994), or in a main loop in a step-wise manner: generate one step, and test the partial candidate solution. If constraints are violated, the last step may be revised: ‘extend-design-then-revise’ (Motta et al., 1994; Schreiber et al., 1994). Although these models are iterative, most of these are consistent with the simple model for the design process introduced in chapter 2, p. 12. The Requirements and Constraints are input to the design process and do not change during the design process.

With the synthesis of a statistical design, a different kind of iteration is sometimes necessary, because Constraints are identified along the way (i.e., constraints are not only input but also output of synthesis). Even the Requirements may be cut-down along the way (like in Schreiber et al., 1994), because the initial specification appears to be too ambitious, because any of its solutions violates the real-world constraints or exceeds the available resources.

That constraints are identified during the design process and that even the specification of the requirements may change during the design process (see Smithers, 1992), are probably characteristics of any non-trivial design task (for example the design of software). Therefore, the final Requirements and the final set of Constraints may be regarded as an output of the design process (Figure 4.2).

For the statistical design task a ‘propose-revise’ model was used, with two important refinements. The first is the notion of a central solution, that is, a single, ‘paradigmatic’ solution that is optimal in certain aspects. To exemplify this, a more familiar design task can be used as a metaphor: ‘road design’. The specification of requirements for a road design task is, for example, to connect town A with town B. The single central solution is a straight strip of tar between the two towns, or,
more abstractly described, a straight line segment between points A and B. In statistical design the abstract solution is a more complicated mathematical object, but the reason that there exists a single central solution is the same. The problem is well-understood and there is a formalised theory in which a clear optimum can be identified. This optimum is the central solution. It is perhaps not an unimportant class of design tasks in which a formalised theory identifies an optimal solution.

Although the central solution is in a sense optimal, it is not always used because of real world constraints, or because of resource constraints. It simply may be impossible or too expensive. In the road design example, there may be a mountain between A and B and the central solution needs to be adapted.

One must first have a candidate solution before it can be tested against the real-world (e.g., trying it on the map, or, if that is not available, surveyors can explore the area between A and B). That is, the real-world constraint is identified by generating a candidate solution that violates it. Without the candidate solution, one would not know where to look for constraints that are a threat to the feasibility of the candidate solution. New Constraints may necessitate modifications of the Design Solution or even of the Requirements.

![Diagram](image)

Figure 4.3: Decomposition of the synthesis task.

The second refinement of the model for statistical design is the implementation of revisions as modification operations that transform an entire pair (Requirements, Design Solution), into a new one. Not many operations are needed, due to the neat mathematical nature of the domain.

Some of these operations preserve the 'derivable' relation between Requirements and Design Solution. In this manner, the 'derivable' relation can be defined as those pairs that belong to the central solution, and by those pairs that can be obtained by the derivability-preserving operations.

In this new model, transformations of the entire design problem are a normal part of the design process. Figure 4.3 presents the decomposition of this synthesis task.

Modifications of Design Solution or Requirements are difficult. Compare this to the
architect, who, for some reason, has to move a wall at the ground floor. This will have consequences in many other parts of the building and in the end this may perhaps have an effect on the Requirements that can be satisfied. The difficulty lies in the web of interrelationships among different parts of the Design Solution and Requirements. A change at one place may have far-reaching consequences.

The above has presented characteristics of the statistical design task that may apply to design tasks in other domains. Although this represents conclusions and insights obtained during the knowledge acquisition, it is presented at the start as an advance organiser. The general model shows the ingredients necessary for an interactive design process in which modifications of the design solution or even the requirements are a normal part, but it does not tell us how the complex objects can be represented nor does it tell us how the 'derivable' relation can be defined.

4.2 Well-defined Models in Iterative Statistical Design

The well-defined models (WDMs) of chapter 3 can be used retrospectively to explain the StatCons-1 design. This appears straightforward with languages for the Design Solution and section 4.2.3 is modernized accordingly. With languages for Requirements, this appears more difficult.

Chapter 2 presented a 'finger exercise' in knowledge acquisition, targeting statistical design. From that chapter, we can still use the results of the hierarchical task analysis (i.e., the functional analysis of the domain task), especially the decomposition of the objects. These functional objects have been decomposed to a small-enough level such that it becomes possible to select representations. A similar decomposition of functional objects is shown at the top in Figure 4.4. Except for a few minor extensions and improvements, it was already provided in the StatCons-0 functional decomposition in Figure 2.7 (p. 20).

This chapter uses well-defined models (WDMs) from chapter 3. The WDMs concept helps in two ways. It helps in the implementation of modifications and it helps in surveying existing languages.

Modifications WDMs provide a basis for the implementation of consistent modifications of functional objects. WDMs provide for a separate and independent and more formal domain analysis. The domain analysis is illustrated in the two layers at the bottom of Figure 4.4. WDMs of data sets and operations on WDMs provide a system in itself to represent the domain, independent of any functional analysis or problem solving process. Using WDMs, corresponding operations provide for consistent modifications.

Therefore, all functional objects must be represented as WDMs. This will give the functional object a clear and unambiguous meaning and it makes the operations on WDMs available for the functional object. For the representation of a functional object, a choice can be made from model languages for WDMs. In this manner we can select or design a coherent set of different views or representations suited to users and to the task to be performed in the domain. Using these, the leafnodes of the hierarchy
of functional objects in figure 4.4 become defined at last, and this propagates upwards in the hierarchy, perhaps improving the original functional decomposition, and leading to a better understanding of the interrelationships of the functional objects.

Summarizing figure 4.4 from the bottom, first WDMs are defined with respect to a central data set concept. That is, a central data set concept is selected, and various model languages can be defined as homomorphisms from data set concept to language. Second, functional objects are defined with respect to WDMs. Modifications of functional objects can then be performed by the consistency-preserving operations of WDMs.

---

Survey  The WDMs concept clarifies interrelations among various languages. It can thus serve as a tool for analysis in a survey of languages for WDMs. Sections below survey languages for WDMs and construct a mapping from functional objects to WDMs. A broad survey is performed, collecting many different languages that potentially qualify for parts of the Requirements, Design Solution or Constraints. Compared to chapter 2, we are much better equipped to oversee the jungle of available languages for models of data sets. It has become much easier to see whether lan-
guages are potential candidates for our use—are they about the data set, are they well-defined? WDMs thus provide for a more systematic approach in surveying available languages. Different languages for the same aspect can be related and classified as homomorphisms or by using the ‘specificity x compression’ diagram explained in section 3.4.

Below we first present an analysis of a few model languages as WDMs, without regard of their use. Then we will discuss the representations of Requirements, Design Solution and Constraints.

Sources. In the inventory of potential languages various statistics textbooks, software systems and human experts were consulted. From the textbooks especially Hays (1974), needs mentioning since it presents the function schema as tool for conceptualizing at a global level.

Important sources for languages for the structure of statistical models and structure of statistical designs were (manuals of) software systems for statistical analysis such as SPSS, GLIM and GENSTAT (Nelder, 1974). For statistical software, various languages for the structure of the statistical design and of statistical models have been developed. Wilkinson and Rogers, in their classical paper (1973), presented design and model formulae. These extend the idea of using operators like the cross product to describe or to construct a design structure or alternative models for statistical analysis. GENSTAT needs special mentioning as it provides a factorization of the statistical design in sampling structure, treatment structure and assignment.

In the context of the generalised linear model (Nelder & Wedderburn, 1972), McCullagh & Nelder (1983), explain the use of model formulae and operations on these theoretically. An implementation in software is provided by the GLIM system, and, in an experimental mode, by the system GLIMSE (Wolstenholme & Nelder 1986).

Central Data Set Concept This chapter uses the data set concept that chapter 3 finished with. The elementary datum \( \{ \text{unit}, \text{variable}, \text{value} \} \) is extended with time (time points), and with operationalization (treatment, measurement, or sampling/selection). With the extended data set we have a basic representation of the data collection process and the statistical design in the data set itself (section 3.5). This data set concept can be used as the most specific (and the least compressed) language for the plan, design and scheduling of the data collection.

As said before, the extra properties of the elementary datum need not be included in real data sets. They serve a theoretical purpose, to provide a data set that can serve as a common basis to consider and investigate different (well-defined) models as candidate representations.

### 4.2.1 Selected WDMs

This section presents a few WDMs in the manner of chapter 3, as well-defined models of structure in data sets, without regard for their functional use or role (meta class) in the statistical design task.

Figure 4.5 shows several model languages for structure in the data, defined as homomorphomic functions from a central data set concept. These different model languages are about the same aspect of the data set, but provide different levels of specificity and compression.
Table structures can be considered equivalent to table formulae, provided that the latter are taken in conjunction with the schema (the valuesets of variables are needed to form the cross product). If the actual structure is a cross product or is close to the cross product, table formulae provides a large amount of compression. Because of the equivalence between the two, the compression is lossless.

Tables with counts, under conditions of equal number of units in each cell, are equivalent to ‘table formula with counts’. The actual formula in the ‘table formula with counts’ shows whether this symmetry exists (one number in the set versus more than one number). The formula then provides a high degree of lossless compression of (part of) the data set.

For each of these model languages operations can be defined that correspond to the set of basic operations defined for the data set. Below table structures are selected to give examples of these corresponding operations.
Operations on Table Structures  For the data set concept one can define a set of operations that take a data set and return a modified data set (chapter 3). Figure 4.6 illustrates corresponding operators on the 'table structure language', that is, operators that take a table structure and return a table structure.

Crossing The crossing operator is the cross product of sets. It adds a variable to a table structure.

<table>
<thead>
<tr>
<th>SEX</th>
<th>CUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>male</td>
<td>male</td>
</tr>
<tr>
<td>male</td>
<td>female</td>
</tr>
<tr>
<td>male</td>
<td>no-cue</td>
</tr>
<tr>
<td>female</td>
<td>male</td>
</tr>
<tr>
<td>female</td>
<td>female</td>
</tr>
<tr>
<td>female</td>
<td>no-cue</td>
</tr>
</tbody>
</table>

Collaps or Projection The projection or collaps operator removes a variable from a table structure and it can be used to undo a crossing.

<table>
<thead>
<tr>
<th>SEX</th>
<th>CUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>male</td>
<td>male</td>
</tr>
<tr>
<td>male</td>
<td>female</td>
</tr>
<tr>
<td>male</td>
<td>no-cue</td>
</tr>
<tr>
<td>female</td>
<td>male</td>
</tr>
<tr>
<td>female</td>
<td>female</td>
</tr>
<tr>
<td>female</td>
<td>no-cue</td>
</tr>
</tbody>
</table>

Delete Cell or Restriction This operator removes cells from the design.

<table>
<thead>
<tr>
<th>SEX</th>
<th>CUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>male</td>
<td>male</td>
</tr>
<tr>
<td>male</td>
<td>female</td>
</tr>
<tr>
<td>male</td>
<td>no-cue</td>
</tr>
<tr>
<td>female</td>
<td>male</td>
</tr>
<tr>
<td>female</td>
<td>female</td>
</tr>
<tr>
<td>female</td>
<td>no-cue</td>
</tr>
</tbody>
</table>

Figure 4.6: Examples of the three basic operators for table structures.

The three operators are sufficient to transform any table structure into any table structure. Especially in the context of statistical design, there is use for special operators to construct special structures such as nesting and latin square. These special structures can be reduced to compositions of the three basic operators.
Nesting  The nesting operator creates a table structure which is not a crossing. Whereas a crossing $A \times B$ uniquely identifies a table structure, a nesting $A/B$ can be met by various table structures. But by introducing balance as an additional criterion, and by using a standard order, nesting can be regarded as an operator. For example:

$$
\begin{array}{ccc}
\text{SCHOOL} & / & \text{TEACHER} \\
\text{school1} & \text{teacher1} \\
\text{school2} & \text{teacher2} \\
\text{school3} & \text{teacher3} \\
\text{school4} & \text{teacher4} \\
\text{school5} & \text{teacher5} \\
\text{school6} & \text{teacher6}
\end{array}
= 
\begin{array}{ccc}
\text{SCHOOL} & \text{TEACHER} \\
\text{school1} & \text{teacher1} \\
\text{school2} & \text{teacher2} \\
\text{school3} & \text{teacher3} \\
\text{school4} & \text{teacher4} \\
\text{school5} & \text{teacher5} \\
\text{school6} & \text{teacher6}
\end{array}
$$

Latin Square  The latin square operator can be applied to three design variables or factors provided that all factors have the same length. To provide an example of a latin square for three factors $A,B,C$, all of length 3:

$$
\begin{array}{ccc}
A & B & C \\
\hline
a1 & a2 & a3 \\
b1 & c1 & c2 & c3 \\
b2 & c2 & c3 & c1 \\
b3 & c3 & c1 & c2
\end{array}
$$

Special structures such as nesting and latin square can be composed as a sequence using only the three basic operations, but the sequence may become rather long.

4.2.2 Requirements

One of the most obvious candidate languages for formulating Requirements is the function schema, but it is not obvious in what way a function schema may be regarded as a well-defined model of the data set, or how the data set concept should be enlarged. What has been used instead in the StatCons-1 design, and in most statistical software, is the use of abstractions from statistical models. For example, a function schema may be regarded as an abstraction from an ANOVA model or from a Regression model. As will be put forward in the section below, this is not unreasonable, since Requirements at the most detailed level use the language of a statistical model.

If pressed for a definition of a function schema in relation to the central data set concept, I would suggest a function that enumerates the set of all possible function schemas. For a data set with two variables, $A$ and $B$, this would be the set $\{A \rightarrow B, B \rightarrow A\}$. If the data set contained information about the design, for example that variable $A$ is a treatment variable, the function would have to produce the set $\{A \rightarrow B\}$. With a larger number of variables this set of function schemas may become extremely large. Therefore, in practice, the user has to say which ones are considered interesting.

The Requirements can be discussed at a low level of specificity. An investigator can state his requirements at a minimal level of specificity by providing only the schema of the data set. This list of variables implicitly states: "These are the variables the prime investigator is interested in." In audio recordings of dialogues between statistical
consultant and researcher, we can see signs of increasing specificity. In a consultation dialogue regarding the example problem from chapter 2 (p. 17), the following two fragments are typical. The first fragment occurs in the beginning of the dialogue:

investigator: "I want to study the effect of SEX and CUE on FINT".
consultant: "Do you also want to look at the joint effect?"
investigator: "Yes".

Then a rather long dialogue about the Design Solution follows. Near the end of the dialogue the following fragment takes place:

consultant: "Then you can use ANOVA and look at the effects of SEX en CUE separate, together, and at the interaction effect".

This example shows two things: the increase of specificity, and, at some point, the introduction of a statistical model as the most specific formulation of the Requirements.

Languages for the formulation of requirements at different levels of specificity are not hard to find. Figure 4.7 shows samples of languages/models for requirements (expressing the requirements of the same example problem). The language expressions can be ordered on conciseness and specificity. From top to bottom the specificity increases. From left to right the conciseness diminishes, but the languages/models are equivalent.

The first three levels. The three levels at the top are to be used without regard of the statistical analysis methods. The least specific one at the top is the schema for the data set, listing the variables. The second level of specificity is a function schema, outlining the domain and the codomain of a function, by listing the independent and the dependent variables. When there are two or more independent or dependent variables, the domain or codomain is not fully defined. The first two levels seem to capture the first line of the dialogue.

At the third level the domain of the example function is made more precise. The table formula to the left defines it in a succinct manner as the cross product of (the valuesets of) the independent variables, whereas the table structure to the right expresses the same model more explicitly. Table formulae and table structures are equivalent. The latter can be used as an explanation of the first. This third level seems to capture the second and third line of the consultation dialogue above (i.e., the joint effect of the two independent variables).

The fourth level. The fourth level of specificity is very close, or identical, to the specification of the statistical analysis. Statistical analysis always works with a number of different models for the data set. The maximal model is the largest, most extensive model. It has a number of components, and each component has a number of parameters. The last line of the consultation dialogue above enumerates the components of the maximal model in the standard ANOVA solution for statistical analysis. This maximal model comprises four components: an overall mean, two components for the separate effects of the variables, and one component for the joint interaction effects of the variables. The language of Model Formulae can be used to show the set of components of the maximal model (Figure 4.7, to the left, level 4). Each component represents a set of parameters with values yet unknown, which are to be estimated in the statistical analysis. The set of table structures to the right provides an alternative language, showing more explicitly what the model formula to the left means.
Figure 4.7: Model Languages for Requirements.

The maximal model can be simplified by leaving out a component, that is, by setting the parameters of that component to zero. In this manner the maximal model can be simplified into models that are smaller, that is, have less parameters. Parameters of one component may be functionally dependent on other parameters in other components. This gives rise to the partial order among components. In simplifying a model, the partial ordering of components must be respected. In simplification one may only remove a component that is not smaller than (i.e., not depended on) any of the other components that are present. Figure 4.8 shows another language: the Hasse diagram, (Dougherty & Giardina, 1988), that makes this aspect of the detailed requirements more explicit.

The alternative models form a set of models comprising the maximal model and all models that can be arrived at by simplification of the maximal model. In statistical
Figure 4.8: Model Languages for Detailed Requirements.

In statistical analysis, comparisons are made between these alternative models on how well they fit the data set. If, in the statistical analysis, a smaller model fits the data equally well as a larger model, the smaller model is preferred. Simple hypothesis testing is an example. The null-hypothesis is the smaller model, with the parameter set to zero, and the alternative hypothesis is the larger model. The simplest model that describes the data well gets the title of best model. The maximal model determines the search space of alternative models. The maximal model thus provides a specification of a set of the alternative models (i.e., hypotheses), to be compared, during the statistical analysis, on how well they summarize the data set.

The fifth level. Figure 4.7 shows a fifth level of increased specificity. It represents the rare case when Requirements are about specific values for parameters.

The Requirements are dynamic. Not only can they be refined by making them more specific, they can also be modified at all levels, using corresponding operations. The design process may start with a simple version of the research question, but during the design process, variables may be added or deleted, and the initial requirements may become a network in which auxiliary variables and background knowledge have a place.

The Requirements comprise a representation of the research question (the function/relation/model in the data set the researcher is interested in), but may also contain elements from the design introduced in the design process. Both Requirements and Design Solution are much interwoven with theory of statistical analysis (cf., Bailey, 1981). We notice the entanglement of statistical design and requirements already in the most abstract formulation of the requirements, the schema of the data set. For example, the original example problem (p. 17) contains a variable:

\[ \text{cue} \in \{\text{male, female, nocue}\} \]

This variable can be modified or replaced by a new variable:

\[ \text{cue2} \in \{\text{male, female}\} \]

There are more parts of the Requirements that have a design-like character or that are very similar to parts of the Design Solution.
4.2.3 Design Solution.

In the StatCons-0 model in chapter 2 the O'Keefe diagram was used to represent the Design Solution (p. 21). With some benevolence, this diagram language can be interpreted as a well-defined model of the data set with time and operation.

Figure 4.9 shows how the O'Keefe diagram can be projected over the time dimension to obtain the design structure and how it can be projected over the design variables to obtain the process structure. For design solutions in which each unit has the same process structure, the conjunction of design structure, process structure and design schema are equivalent to the O'Keefe diagram.

The process structure and the 'sequences of variables language' were discussed in section 3.5.3 as well. The process structure limits itself to with the temporal ordering of data operations and associated variables. Other model languages of the process structure were presented in (section 3.5.3). The design structure deals with on the specific values for sampling and treatment variables and the numbers of units for each combination of values.

Sampling Structure, Treatment Structure and Assignment. Using the system of operators on table structures, the construction of a complex design can be modularized in the manner of GENSTAT. The design structure of the design can be factorized into a sampling structure, a treatment structure and the assignment. As structures, the design structure, sampling structure and treatment structure are similar and can employ the same languages.

The sampling structure is the design structure for the variables in the sampling design, that is, the variables with sampling/selection as operationalization. The treatment structure is the design structure for the treatment variables, those with treatment as operationalization. The assignment of sampling structure to treatment structure may be regarded as an operator such as crossing, latin square, and others, which yields the overall design structure.

In the example scenario, the sampling structure and the treatment structure each have only one variable. The cross product operator is used to construct the (overall) design structure from the sampling structure and the treatment structure. The operator is used to assign units in different cells in the sampling structure to different cells in the treatment structure.

In the example scenario, the sampling structure and the treatment structure each have only one variable, but in larger designs factorizing the design can help to achieve compression. The distinction between sampling and treatment structure has especially utility for designs with a complex error structure. As for example in split-plot designs where different treatment variables are assigned to different levels in a hierarchical sampling structure.

Solving Design Problems

A design problem is a triplet (Requirements, Design Solution, Constraints).

Constraints There are three types of constraints.

The first type are real-world constraints that lead to structural zeros in the design structure. For example, there is a variable marital status: \( marstat \in \text{MARSTAT} = \ldots \)
Figure 4.9: Design Solution: various user views as well-defined models (tree of homomorphisms) of a central data set concept (at the bottom).
{unmar.mar} and a variable age: \( \text{age} \in \text{AGE} = \{\text{young, old}\} \) and the investigator wants to look at the cross product. It appears that there are hardly individuals in \( \text{mar.stat} = \text{mar} \) and \( \text{age} = \text{young} \). In a table with counts using maximum counts these constraints can be expressed by setting parameters to zero.

The second type are resource constraints. These are constraints on the numbers of units that can be used. They can be expressed as maximum number of units in a table with counts.

The third type of constraint is the 'Data Set already collected' constraint. If the data set is already collected according to a faulty design, there are still options to modify and improve the design and data set (by corresponding operations). Options are limited, however. The data set and the design can only be made smaller, by deleting units or by deleting variables. So the available data set serves as a maximum. If the design of the data set is expressed as a table with counts, then this table is taken as the table with maximum number of units.

**Central Solution** There is a core of solutions presented in textbooks, here referred to as the central solution, for which it is clear which Design Solution goes with which Requirements.

The design structure is constructed as a cross product \( X_1 \times \ldots \times X_n \), or by special operators such as nesting or latin square. For these structures maximal models can be derived that can be properly analysed by statistical methods.

In the central solution, the design structure has the same count of units in each cell. This is desirable for the statistical analysis. Under these conditions, as said before (figure 4.5), the table formula with counts is equivalent to the table with counts. Therefore, as long as one stays within the framework of the central solution, the design task can work with highly compressed representations. One can first determine a solution at the level of a table formula, and then decide about the number of units per cell (i.e., finishing with a table formula with counts), as a highly compressed, but lossless (see Chapter 3) version of the structure in the data.

**Interrelations** Typically, the global Requirements (levels 1-3 in figure 4.7) can be represented as a function schema of the form:

\[
X_1, \ldots, X_n \rightarrow Y
\]

The domain of the function is a cross product \( X_1 \times \ldots \times X_n \) minus real-world constraints. What is left after taking such constraints into account is a local Universe that can be represented as a table structure.

The design structure in the Design Solution is the sampling plan that prescribes which parts of the Universe will be present in the final data. It can also be represented as a table structure. A first criterion for a solution is that the design solution must be contained in what is left of the universe after constraints have been taken into account.

To give an example, let us recall the table structure used as illustration of a nesting. The SCHOOL/TEACHER structure (see p. 73), can serve as an example of the Universe - Real World Constraints (using the restriction operator from Figure 4.6). An investigator of educational methods may have phantasies about a crossing SCHOOL\( \times \)TEACHER, but what he can sample, that is, the Design Solution, cannot exceed SCHOOL/TEACHER.
A second criterion is that the maximal model, that is, the detailed Requirements (level 4 in figure 4.7), must be contained in the design structure. That is, effects or components present in the maximal model need certain parts of the universe to be present in the data set sampled. If the design structure is a nesting SCHOOL/TEACHER, the largest maximal model that can be analysed is [1,SCHOOL,SCHOOL/TEACHER].

These criteria can be formalised. If we concentrate on the structural aspect of statistical design, then the relevant parts of Requirements, Design Solution and Constraints can all be represented as table structures or table formulae. A containment relation between table structures can be defined as follows:

\[ T_1 \subseteq T_2 \text{ if } T_1 = T_2 \text{ or if } T_1 \text{ can be obtained by applying projection and restriction operators to } T_2 \text{ (see p. 72).} \]

Then the desired relationships between Requirements, Design Solution and Constraints represented as table structures can be summarized as:

\[
\text{Requirements} \subseteq \text{Design Solution} \subseteq \text{Universe - Constraints}
\]

where ‘\(-\)’ is the restriction operator, used to subtract Constraints from a Universe that is the crossing of all domains of variables. Real world constraints are restrictions of this local universe. Detailed Requirements are represented by a maximal model, a set of table structures. The maximal model (i.e., each component of the maximal model), must be contained in the Design Solution. The Design Solution, in turn, must be contained in what can exist according to knowledge about real world constraints. Table 4.1 provides some more examples.

| General rule: Requirements \(\subseteq\) Design Solution \(\subseteq\) Universe - Constraints |
|-------------------------------------|---------------------------------|------------------|
| Examples:                          |
| 1 \([1,A,A/B]\) \(\subseteq\) \(A/B\) \(\subseteq\) \(A/B\) |
| 2 \([1,A,B,AxB]\) \(\subseteq\) \(AxB\) \(\subseteq\) \(AxB\) |
| 3 \([1,A,B,C]\) \(\subseteq\) \text{latinsquare}(A,B,C) \(\subseteq\) \(AxBxC-(a=1,b=2)\) |

Table 4.1: Examples of Solved Design Problems

Error Reduction Strategies To make do with less units, various error reduction strategies have been devised. The statistical analysis becomes less sensitive with less units. To compensate for this, the analysis can be made more sensitive (i.e., the statistical model will have a smaller error component), if one includes auxiliary variables into the statistical design and into statistical analysis. The auxiliary variables must have a known effect or at least very likely have an effect on the dependent variable. If this device is used, the auxiliary variable becomes part of the model used in the analysis, that is, it becomes part of the detailed requirements and it shows up in the model formula. Often it is not considered interesting to study interaction effects with auxiliary variables and such components will then be absent from the model formula expressing the maximal model.

The special design structures such as nesting and latin square, may be used to build auxiliary variables into the design.
Flawed Designs  When a design is flawed, that is, when it is outside the central solution, a small set of operations may be tried to move into the direction of the central solution. These too can be thought of as operations that transform a pair (Requirements, Design Solution) into a new one.

Conclusion  This is the outline of the model for iterative statistical design. As a design task it is atypical, there is hardly any search let alone backtracking.

In the design process the representation of the objects in the domain can often remain at the rather high level of compression provided by table formulae. The generation of the design can be done at this level. Expansion to the table structure or tables with statistics is straightforward and can be carried out afterwards, after reaching an acceptable solution at the highest level of abstraction. For the StatCons-1 prototype different languages at different levels of compression were used.

There are reasons for having this type of redundancy. For example, one cannot readily see in a table whether there is a complete crossing with equal numbers of units per cell. The table formula shows immediately whether these criteria are satisfied. If a table formula needs to be explained, it can be expanded to the table structure or to the table with counts. Languages may differ in degree of difficulty for the user.

To actually modify a (partial) candidate Design Solution, and, if necessary, the Requirements, a few well defined operations suffice, due to the mathematical nature of the domain. Part of these are based on the operations on data sets mentioned above. The set of operators is small and it appears feasible to determine the effect of each operator on the different parts of the pair (Requirements, Design Solution) systematically. The next section shows how this model can be the basis for the design of a system to support a user who lacks knowledge of statistical design.

The above serves as a more extended example of the use of well-defined models in the analysis and design of statistical software. There are a number of aspects of statistical design that are not covered by this concept of the data set and models thereof. Random sampling and random assignment are the most important ones. In the prototype these are dealt with as simple properties of the data set. Well-defined models do not provide for everything, but they certainly helped to untangle the complex structures in statistical design tasks.

4.3 The StatCons-1 prototype

The model for statistical design as an interactive, iterative process provides a solid basis for a system for collaborative statistical design. According to the previous sections, the aim of the design process is a Design Solution that satisfies the Requirements and does not violate the Constraints. The use of operations that modify these objects are a normal part of the design process. In a cooperative or collaborative design process, the aim of the design process is the same, but the knowledge and skills may be distributed over the partners.

If partners can join forces, together they may perform better than alone. What is needed is pooling of the knowledge of user and machine. Viewed from this metaphorical perspective, we only have to design a user interface that allows user or system to create a design that satisfies both the user’s knowledge and the system’s knowledge.
The user interface for the joint collaborative design process is based on a shared object metaphor and a speech-act dialogue model. The objects in the design task, Requirements, Design Solution and Constraints may be regarded as a shared object. Both partners can propose extensions or improvements or enter new Constraints using a simple interaction framework for sharing. Sharing involves communication, but is not the same as communication. The meaning of sharing can be made more precise using the most simplified version of a dialogue model à la Winograd and Flores (1986). It provides a protocol composed of speech-acts: propose, accept, reject. An agent can try to share an object X with another agent by proposing it to the other. Then there are two options: the other agent accepts, meaning the object has become shared, or the other agent rejects, meaning it is not. Thus, after a propose there exist two contexts, one with X and one without X. A reject may involve a reason and a counter proposal, and we may interpret criticism as a reject. This model shows that minimally two contexts are needed in dialogues that negotiate about a shared object.

The StatCons-1 User Interface

The design of the StatCons-1 user-interface explored new user-interface technology: graphical interfaces with direct manipulation of objects on the screen (Norman and Draper, 1986). This technology offers new opportunities, combining the advantages of menus, which require only recognition rather than recall, with the advantages of large sets of commands: flexibility and more direct task-action mapping. Windowing systems allow parallelism, or at least interleaving, or multi-threaded dialogues —again more flexibility. Pictures or diagrams may be used, which, as many believe, may tell more than one thousand words. Principles as WYSIWYG —What you see is what you get— make it easier for the user to understand and control what is happening. WYSIWYG and direct manipulation dictate that the objects in the design task are explicitly shown on the screen of a workstation. These can then be directly manipulated by the user and they can be changed by the system. This would be possible using a number of graphical editors representing the design problem as a shared object, understandable to both the user and to the machine. The WYSIWYG principle can also be applied to show the structures in the statistical design at the abstract symbolic level —design formulae— and —simultaneously— in their expanded tabular form (see figure 4.10). This would allow users unfamiliar with the symbolic notation to actually see what they mean.

The StatCons-1 user interface, from the point of view of the user, comprises a number of editors for various parts of the Requirements and Design Solution. The user has to take the initiative and use the editors. Figure 4.10 shows the editors for the variables (the schema for the datat set), the Requirements, and the Design Solution. The Design Solution can be communicated about at two abstraction levels (design formulae or tables with counts). This set of editors comprises a design workbench, somewhat analogous to Fisher’s Kitchen Design system shown in chapter 1 in figure 1.1.

The use of editors and direct manipulation allows for flexibility. At any time the user can change part of the Requirements or Design Solution. On startup the system shows the variables editor and the user first has to enter at least the names of the variables. When finished, the user must click the OK button and then the system shows the Requirements editor and the Design Solution editor. Then it is at the user's
Variables

<table>
<thead>
<tr>
<th>sex</th>
<th>cue</th>
<th>Operationalization</th>
<th>Scale Type</th>
<th>Min</th>
<th>Max</th>
<th>Step</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>measurement</td>
<td>interval</td>
<td>1</td>
<td>30</td>
<td>1</td>
</tr>
</tbody>
</table>

Requirements

- sex
- sex x cue
- cue

Design Solution

- **Sampling Variables**: cue
- **Treatment Variables**: sex
- **Number of Units**: 36
- **Number of Units per Cell**: 6

**Sampling Structure**
- **Typology**: simple
- **Distribution**: balanced
- **Randomized Sampling**: yes

**Treatment Structure**
- **Typology**: simple
- **Assignment**: crossed
- **Distribution**: balanced
- **Randomized**: yes

**Overall Design Structure**

<table>
<thead>
<tr>
<th>sex</th>
<th>cue</th>
</tr>
</thead>
<tbody>
<tr>
<td>male</td>
<td>12</td>
</tr>
<tr>
<td>female</td>
<td>12</td>
</tr>
</tbody>
</table>

**Overall Design Structure**

<table>
<thead>
<tr>
<th>sex</th>
<th>cue</th>
</tr>
</thead>
<tbody>
<tr>
<td>male</td>
<td>6</td>
</tr>
<tr>
<td>female</td>
<td>6</td>
</tr>
<tr>
<td>no-cue</td>
<td>6</td>
</tr>
</tbody>
</table>

*Figure 4.10: The user interface of StatCons-1*
discretion to enter the Requirements or the Design Solution, or both.

The Analyse and Synthesize buttons are necessary for the user to control the system. In retrospect, perhaps better names for these buttons would be Critique and Generate, respectively. If the user wants the system to generate a design, he can click the Synthesize button —after entering the Requirements— and the system will generate a Design Solution. If there are no Requirements yet, the system shows a pop-up window saying that the user must first enter Requirements. The user must first provide the Requirements and then the system can generate a Design Solution and show it in the design editor. The knowledge of the system is not complete and the system needs the user to make sure the system's solution does not violate real world Constraints the system is not aware of. In a sense the user has to critique the system's design. To do so, the user can directly modify parts of the design that violate real world Constraints. After such change, the system in turn may critique the updated version of the solution.

If, in a different scenario, the user wants the system to critique the user's design, he has to enter both Requirements and Design Solution and click the Analyse button. When the system detects bugs, it can open a window and report on the problem, and if there is any feedback on the Requirements, it can be effected and shown in the editors.

These buttons are somewhat clumsy, but it is necessary that the user controls the operation of the system. Even if the system would only be a critiquing system, it would still be necessary that the user can turn critiquing on or off, to avoid critiquing during a major modification of the design which may involve many inconsistent intermediate states and it is undesirable that the system critiques these. This is contrary to the idea of timely system initiated criticism, but we were confronted with the problem that an edit operation is not the same as a speech-act. A propose may take n edit operations, and the Analyze (Critique) button is needed for the user to signal the end of the propose.

I wonder how Fischer's system for Kitchen Design takes care of this. Suppose (see Figure 1.1) that the user wants to swap the stove and the fridge, and to do so, first moves the fridge to the middle of the kitchen. Would the system then show a pop-up menu: "It is better not to put a fridge in the middle of the kitchen as one could trip over the power cord."

**Evaluation of the StatCons-1 Prototype**

Experience with the StatCons-0 prototype in chapter 2 resulted in new requirements for the support system (p. 31). The StatCons-1 prototype provides sufficient functionality to meet these. As explained in the previous section, it does not provide immediate feedback, because that may interfere with edit sequences by the user. Timing of feedback is indeed an important characteristic of computer support, at some periods it is desirable that it is immediate, but during editing it should be turned off.

At its time (1987), the direct manipulation interface was quite advanced. Apple Macintoshes were still novel and most users were only familiar with command-based or menu-based user interfaces. In informal evaluations —asking a few acquaintances to try to use the system— users find the StatCons-1 prototype quite difficult to use. They do not know what to do or where to start, and they need much explanation, instruction and stimulation to get started and to keep going. It appears that the
StatCons-1 prototype is much more difficult to use than its predecessor, the menu-based StatCons-0 prototype described in chapter 2.

The StatCons-1 user interface provides more options and choices for the user than the StatCons-0 question-answer loop. Beside being more complex, the direct manipulation interface gives very little guidance about what to do when. In accordance with Norman and Draper's classical (1986) user-centered system design, the control is fully in the hands of the user: the user must control which task will be performed when. However, the objects on the screen and the use of the various editors — each being different — are apparently not intuitive to non-experts.

In the dialogue with StatCons-0, the system takes the initiative. With StatCons-1 the initiative is in the hands of the user. The user is in control and the user must decide what to next. For users with only a small amount of domain knowledge, it may be difficult to be responsible for planning the execution of the design task. In retrospect, a workbench interface such as StatCons-1 or Fischer's kitchen design is more suited to experts rather than to novices. Similar problems have appeared with workbenches such as Shelley (Anjewierden, Wielemaker & Toussaint, 1992) and KEW (Terpstra et al., 1993).

The next section will provide a more detailed comparison and interpretation of the differences between the prototypes.

4.4 Conclusions of the StatCons Case Study

Chapter 2 and this chapter 4 presented designs of computer support based on imitation of human-human consultation. We have collected verbatim protocols of consultations, performed analyses, drafted diagrams, and designed and implemented two prototypes, using the model-based system development method KADS as a guiding framework. This shows that KADS is indeed helpful in software design, but we must also conclude that the support provided by the prototypes is inadequate. Imitation of human consultation practice is a risky guiding principle.

User Support

The two prototypes have a common basis: a decomposition of collaborative design in three subtasks or functions. At this abstract level the prototypes share these functions with the human consultant.

1. Propose, or enter (Requirements, Design Solution).
2. Evaluate, or identify flaws in (Requirements, Design Solution).
3. Modify, revise or repair (Requirements, Design Solution).

The two prototypes differ in the control structure overlaying the functional decomposition. StatCons-0 is based on a single execution sequence of the three functions. StatCons-0 is thus unsupportive of an interactive design process. StatCons-1 is based on an iterative and incremental process in which each function may become active again and again. StatCons-1 supports interactive, iterative design. The two prototypes also differ in user-interface technology. StatCons-0 has a question-answer interface for the first function. StatCons-1 has an advanced direct-manipulation user-interface.
There is a large difference between, on the one hand, the human consultant, and, on the other hand, the two prototypes. Small, informal evaluations of the prototypes with prospective users showed that neither is a very satisfactory substitute of the human consultant. The human-computer interaction, the computer dialogue, is a problem. Only in a certain abstract sense the prototypes can be said to be a (partial) replacement of the human consultant. There appears to be a human-computer interaction bottle-neck.

Both systems are difficult to learn or use, but the StatCons-1 prototype with its flexible user interface appears more difficult to learn and use than the StatCons-0 question-answer interface. Ironically, the final prototype which, using new interface technology, would solve all problems with the pilot prototype, turned out to be more difficult to use. We can explain this, in retrospect, if we take the perspective of the user.

The user’s levels of expertise and experience  In the use of the support system, the human needs two types of knowledge: knowledge of the domain, to formulate goals and subgoals in the domain task, and knowledge of the user-interface, of how to act in the dialogue with the system to achieve subgoals. This distinction can be explained using the GOMS model (Card, Moran and Newell, 1983), or Norman’s (1986), approximate theory of action. Human-computer interaction is characterized as a process in which the user has goals, decomposes these into subgoals or tasks. Here domain knowledge is used to decompose goals and select tasks that will help to accomplish goals. Continuing the user process, knowledge of the system is used to transform tasks to action specifications (task-action mapping), which can then be executed. To monitor and control this goal-satisfaction process, outcomes of actions are perceived, interpreted in terms of task outcomes and evaluated in terms of progress towards the goal. Knowledge of the domain is important at the level of goals, evaluating progress and in deciding about subgoals or tasks that will help to accomplish the goal. Knowledge of the system, especially the user interface is important at the level of specifying the actions and perceiving and interpreting the outcomes.

Although domain knowledge and system knowledge are not entirely separate, the distinction appears useful. J. Fisher (1991), articulates this as there being two independent dimensions that are to a large extent independent: expertise refers to domain knowledge (from novice to expert) and experience refers to system knowledge (from naive to experienced). In relation to software for statistical analysis, each of four categories occurs in reality: (novice, naive), (novice, experienced), (expert, naive) and (expert, experienced). Using these concepts we can say that StatCons aims at naive novices, that is, users low on domain expertise and low on system experience.

Initiative Although direct manipulation interfaces with windows, menus and buttons are considered to be easy to learn and use, the direct manipulation interface of StatCons-1 appears to be more difficult to use than the question-answer interface of StatCons-0. The main difference between the two user interfaces appears to be system-initiative versus user-initiative in the planning and execution of the domain task (page 84).

Direct-manipulation interfaces are successful in facilitating task-action mapping by presenting the required system knowledge on the screen, thus relieving the user from the need to possess elaborate system knowledge. Direct manipulation interfaces
can thus help compensate lack of system knowledge. Direct manipulation interfaces offer little help for users that lack domain knowledge. That is, when the user has difficulties at the level of goals and deciding about subgoals or tasks, support in task-action mapping will not help. The StatCons-0 question-answer interface helps by providing (even forcing) a proper sequence of subgoals. The direct manipulation interface provides no help in setting subgoals. It is entirely up to the user.

Users with lack of knowledge to an extent that they do not know what to do, need guidance in the process, that is, system initiative. Direct manipulation interfaces, such as StatCons-1 and Fischer's Kitchen Design system (chapter 1), expect the user to take the initiative, but the gap between system and user can only be bridged if, on the user's side, there is familiarity with the objects in the domain (i.e., stoves and sinks and an outline of walls), and if the user is able to formulate goals and subgoals in the domain. This assumes the user has a considerable level of expertise. System initiative may relieve the user entirely from this responsibility and also help to off-load the user.

Another difference between the prototypes is that the direct manipulation interface provides a richer garden of behavioural syntax to be mastered. In contrast, the question-answer interface does not require much experience from the user to understand the question-answer loop. It is simple and straightforward in its use: just answer each question. Lack of experience (skill) with graphical user interfaces may become less of a difficulty when they have become more common and more standardized, but the factor of user-initiative vs. system initiative, with its implications for the responsibility for planning the task execution, with its implications for guidance or lack thereof, will remain an important factor, especially in case of users with little domain knowledge. Users with considerable lack of expertise need guidance instead of a bewildering toolbox. In this respect, the StatCons-0 user interface is better than the more advanced direct manipulation interface of StatCons-1.

In summary, a cooperative system may take a larger or smaller role in planning the (collaborative) problem solving process. The more the system takes the initiative in the collaborative process, the less the user has to do in planning and in articulating the collaborative execution. For users low on domain expertise and system experience, planning and articulating the task execution is a stumbling block and a large load contribution. Assigning the initiative to the system can help to lower the load on the user. Initiative is therefore an important design parameter in computer support.

With the above we have an understanding why the StatCons-1 workbench user-interface is more difficult. In the above there have been hints at extra tasks involved in using computers. The fact that both prototypes are more difficult to use than the human consultant can to some extent be attributed to the additional work or overhead users are burdened with.

Articulation Work. Theorists of human-human cooperation stress the additional work or overhead inherent in any form of cooperative work: articulation work (Strauss, discussed in Gasser, 1986), or co-ordination work (Malone & Crowstone, 1990). This additional work does not only comprise communication, but also mutual adaptation and synchronization, and whatever else is needed to 'manage interdependencies'. As long as the additional work in cooperative problem solving does not exceed certain limits and as long as it does not exceed the benefits, the human may be able to profit
from engaging in cooperative modes of problem solving. Then the support has added value, that is, a positive effect.

Consultation and Articulation Work. Articulation work is part of human-human and human-computer consultation. In the latter situation, the amount of overhead tends to be much larger, due to the limited bandwidth of human-computer interaction. In human-computer interaction there are numerous aspects that contribute to extra articulation overhead. It is not only that, in the dialogue with the computer, the human has to type or manipulate a pointing device. Chapanis (1975, reprinted in Chapanis 1988), compared performance of subjects in different rooms working together and communicating over audio or teletypes. In the last condition groups need twice as much time to find a solution, even if the members are trained typists. Differences are not ascribed to the typing work, but to the 'communication richness' of the audio channel.

Another source of additional work with computer support is that the user has to reformulate or translate his problem to the abstract language or ontology used by the system. In the StatCons domain, there is a high level of abstraction and the mapping from the user's specific problem to the language of the system is left entirely to the user. Even in less abstract domains such abstractions are necessary; the user must encode problem data in the language of the machine. A medical expert system, for example, may ask "Is there cyanosis?", or "Does the patient look blueish?".

Another important difference is that in human-human support very quick mutual adaptation occurs. For example, in a conversation about a patient, a doctor may ask a nurse whether there was cyanosis, and if the nurse does not answer immediately, the doctor immediately asks a new question: "Did the patient look blueish?". In human-computer dialogue the adaptation is fully in the hands of the user. More generally, all kinds of subtle mechanisms for mutual adaptation that are routine among humans are absent in human-computer dialogue. Therefore, the user must adjust himself much more to the system, compared with the human-human situation.

Imitation of Human Consultation as Guiding Principle. There are a number of aspects in the area of user-interface technology, human factors and computer literacy that contribute to the user-interface difficulties, but even if these can be improved, it seems that in general human-computer consultation involves much more overhead for the user than human-human consultation. In automation in general the tasks of the human change, and in automation of consultation the user is burdened with more articulation work than the client. Task demands that are within limits in human-human contexts, may turn into overloading and stagnation in a human-computer context. The trade-off between potential benefits and increased articulation work may turn negative. This is likely to happen when in the human-human situation cognitive demands are already very much stretched to their limits. Support systems can thus have a harmful influence (cf. Neerinx & deGreef, 1993).

In Conclusion, the strategy of imitating human-human support takes no account of the increased articulation workload in the transition from client to user, and it may therefore be misleading as a guiding principle for the design of computer support. That is, the imitation strategy entails the risk that no users can be found for the system. To take the user's work situation into account we adopt a new guiding principle for the design of computer support: 'maximise the benefits for the user and
minimise the workload of utilizing the support'. It is as if in the invention of the flying machine we abolish the idea of flapping artificial wings and turn to the essentials: high lifting force and low weight.

Placing the user in the center implies a shift in the overall perspective. User and system may still be engaged in a joint execution process pooling their knowledge and skill, but not as equal partners. The user now has a special status, and the role of the system is to support and enhance the problem-solving by the human, rather than aiming to control or be involved as much as possible (cf. Woods & Roth, 1988). Such computer support we will call cognitive support, because the aim is to enhance and amplify human cognition rather than automating as much as possible. The scope of methods to achieve this aim is larger than knowledge-based systems technology alone. There are many gradations; from critiquing based on a complete expert model to more subtle forms of support — which might provide substantial improvement of performance of the human — such as a check-list or forms or notepads as external memory aids.

At this point in the StatCons case study, there is hardly any theory available and little systematic investigation of the effects on users of different methods of support. From the new guiding principle we can only conclude that the choice, design or invention of a method of support may best be driven by an analysis of the needs and problems of the user's of the base system.

Implications. Following these new insights, computer support designed as integrated add-on to existing software may have a larger probability of a positive trade-off. Being integrated helps to avoid dialogue, that is, overhead. Having an existing system as a basis provides a context in which there are real potential users and benefits. In statistics there are many widely used systems for statistical analysis. The critiquing function of StatCons could be integrated into such a context, which I henceforward call 'base system'. Then the user would have less work to provide the data needed by the support function. Most of this information is available from the base system. Specifically for the StatCons Case, part of the user's Conceptualization is already present from the dialogue between user and statistical analysis system. Another part, the structure of the design, can be computed from the Data Set. This greatly reduces the amount of dialogue that remains needed to provide support. From the user's point of view, there is little or no additional work in utilizing the support.

The fact remains in this integrated StatCons-2 fantasy that we do not know whether the critiquing function would provide any benefits for the user. If we really put the user in the center, and strive to optimize the trade-off for the user, we must also analyse the problems and needs of the users of the base system, as not only to minimize the costs, but also maximise the benefits of using the support. Other forms of support outside critiquing may also come into consideration.

KADS

Given its point of departure, the StatCons-1 prototype is a reasonable system: it contains an expert model for the design task and can be used to generate or to critique a statistical design. The use of a systems or knowledge engineering approach such as KADS has helped to achieve this prototype.

The general model for iterative design in Section 4.1 is a contribution to the KADS collection of generic models. Computational models for design tasks tend to be
limited to parameter space: figuring out the values of a parameters like, for example, the diameter of the drum and the strength of the cable in elevator design. The model for iterative statistical design in Section 4.2 in this chapter is an instance of the general model and it addresses the construction of a structure.

It has to be said that the KADS approach has gradually been extended. Early KADS in chapter 2 was too much limited to functional analysis. The modelling of information structure was underdeveloped. In contrast with diagnosis or assessment tasks, statistical design shows complex structures and interrelationships. KLONE is not specially suited to represent such structures in a manner that makes the interrelationships explicit.

In COMMONKADS there is now a Conceptual Modelling Language (CML), but this offers no ready-made solution. It will always be necessary to analyze the models and operations in the domain. In the StatCons case, in statistical design, the key to the domain is the insight that all domain objects are models of the data set. One cannot expect a representation language such as CML to solve this. The current view on KADS also states that modelling a design task in a new domain requires theory construction. For the dependencies and interrelationships between different models in the design task is no standard method, no effective clerical procedure nor any ‘algebra of discovery’, available.

With well-defined models we have found a solution to consistent representation and modification. The contributions of the functional task analysis and of the domain analysis to the final model of iterative statistical design were illustrated in Figure 4.4. The layer at the top shows the hierarchy of functional objects. This aspect of the functional analysis from chapter 2 is, with a few modifications, still present in the final model.

KADS interpretation models provide templates for decomposing different types of tasks such as design tasks. In the decomposition tree of functional objects, in Figure 4.4, one can find the Conceptualization object being decomposed in the three objects of design tasks: Requirements, Constraints, Design Solution. In the functional analysis, these ‘functional objects’ have no definition, but it is easy to find many examples in many different languages or representations. In fact one is easily overwhelmed. The functional analysis nor KADS interpretation models nor the modelling language itself provides any clue for a systematic approach to resolve the chaos. In the final model, the functional objects are defined as well-defined models.

The well-defined models approach can be transferred to other design tasks. For example architectural design, in which the central object is a building. One can define different models and use various languages. By using well-defined models, it becomes simple to consistently modify and update an entire building project. To ensure that all models are well-defined, a set of operations on buildings has to be devised, and for each model the corresponding operations have to be defined. In statistics the central object is the data set and the number of operations is not too large. The number of building operations that are needed may be larger.

In Conclusion, early KADS was too much fixated on functional models, with neglect of domain analysis. The latter was limited to simple hierarchies of attributes, without recognizing that the fillers of the attributes in this domain should be designed as well-defined models of the central data set.

When it is the case that objects have an internal structure and that there are complex interrelationships among these, then a strict separation of task analysis and domain
analysis is not feasible. An initial task analysis is useful and necessary to zoom-in on the domain and to collect views on domain objects. After that, when the domain analysis has provided a better understanding of the objects and operations on objects, the functional task analysis can be rebuilt from the bottom up. COMMONKADS now has a much richer analysis method for domain knowledge, but still no definite method for complexly structured objects. Well-defined models provide a solution in the statistical domain, but scaling this up to domains with a very large number of operations is not trivial.

Model-Based Design of Support

In the model-based approach there is room for improvement. Two aspects seem important. The first is that the method for system development should help to develop the right system. The KADS approach helps to develop a system that is technically sound, but this is not necessarily a system that fits well with the users. As put forward in the above, the design of task support must put the user in the center and it is desirable to improve or extend the method for system development as to design a better-functioning human-computer system. This may not always be possible, depending on the domain and the prospective users, and the method should help to assess lack of feasibility at an early stage.

The second aspect is that the repertoire of modelling techniques needs to be extended. As yet, not all aspects relevant to the human-computer interaction have been covered. What is missing is a modelling technique to describe the method of support, that is, the process of joint task execution. A term like critiquing is too vague. Both prototypes implement a form of critiquing, but they are very different in their behaviour. In the development of the prototypes, their behaviour was only described loosely and informally. It is desirable to have a modelling technique to specify the behaviour of user and system in the joint execution process with greater precision (chapter 5 and especially chapter 6 will provide for this). In a more extended model-based approach, such specification may serve as input to user-interface design.

In conclusion, KADS is focussed on knowledge and expertise. the modelling approach appears helpful to actually design a system, but not necessarily an effective system to support a user with lack of knowledge. For the design of support the modelling approach can be extended with various forms of user analysis, and models for collaborative execution that address the behavioral aspects of the interaction.

Acknowledgements. First of all I would like to thank all statistical consultants and theorists who made themselves available for interviews, for role-playing consultancy sessions, and for discussions about prospects and limitations of statistical expert systems. Many thanks are also due to Guus Schreiber and Jan Wielemaker who helped by designing and implementation the StatCons-1 prototype. This research was performed within the KADS project, which was partly funded by the ESPRIT program of the European Commission.