Part I

Automating Consultation
Chapter 2

StatCons-0: Exploring Methodological Design of Support

One of the problems to which new information technology—knowledge technology—might be the solution is the support of humans who lack the knowledge required to achieve a task or a goal. Part I, Automating Consultation, explores the hypothesis that an effective method of support can be based on imitation of the role of a human consultant in human-human advisory dialogues. The method for system development is based on an early version of the KADS methodology. Methods and techniques from software engineering, artificial intelligence and cognitive psychology are used in knowledge acquisition, design and implementation of a system that imitates the role of the human consultant as much as possible.

Two prototype systems have been developed. This chapter reports on the development of StatCons-0, a pilot prototype of an automated statistical consultant. Chapter 4 will report on the development of the final StatCons-1 prototype and provides the final evaluation of this approach to design support for humans with lack of knowledge.


Introduction

This chapter and chapter 4 present the “StatCons case study” in which the KADS methodology was used to develop two prototype expert systems in the field of statistics. In the mid-eighties, building an expert system usually involved an entangled mixture of knowledge acquisition and implementation efforts. An emerging methodology, KADS, based on cognitive psychology and software development, promised a more systematic approach (Wielinga & Breuker, 1984; deGreef & Breuker, 1985; Wielinga & Breuker, 1986). According to early methods for building expert systems based on prototyping, knowledge acquisition and implementation are highly
intertwined. In contrast, early KADS focussed much more on separation of knowledge acquisition and implementation. In analogy with classical software development methods, the software design process is structured according to a Life Cycle Model. The Life Cycle Model views system development as a sequence of stages: analysis of the functional requirements, design of the artifact and, finally, implementation of the system. Deferring implementation allows for a more deliberate choice of architecture. In addition, KADS provided a set of techniques to analyse expert behaviour in a consultancy context. One of the more articulated questions to be addressed by the case study was whether the strict separation between analysis and design would be feasible for knowledge-based systems. Another more detailed question was whether the analysis process (i.e., the knowledge acquisition stage), could be structured in more detail.

In this thesis, the case study is not only presented as an example of the use of KADS in a practical context, but also as a way of exploring the hypothesis that users may profit from an expert system that imitates the human consultant as much as possible. The target for the case study was statistical consultancy as provided by experts at the Psychology Lab of the University of Amsterdam. Students or staff who are conducting an investigation visit the local statistical experts to obtain advise about the statistical analysis of their data. It is a real-life consultation practice in a non-trivial domain and it has a practical advantage: availability of experts.

The task in which clients or users are supported is statistical design. The clients or users lack knowledge to prepare a design that is optimally suited for statistical analysis. The type of support that is given is criticism of the design prepared by the user. Except for the domain (i.e., statistical design), and the level of domain knowledge of the users (i.e., non-experts instead of experts), the type of support provided by the StatCons-0 prototype is similar to Miller’s (1983, 1984), critiquing system. Miller’s system monitors and provides feedback on aenestisists plans. The user may enter a candidate solution, and then the system provides an analysis of pros and cons of this solution and of reasonable alternative solutions.

First an overview will be provided of the KADS methodology of the time and of the issues in the methodology to be investigated in the case study (section 2.1). The next section presents the KADS analysis for the StatCons-0 prototype (section 2.2). The final sections present the prototype (section 2.3), and the conclusion of this first cycle of the StatCons case study (section 2.4, p. 33).

2.1 Early KADS

A major problem in the construction of expert systems is the method for knowledge acquisition. This may be one of the reasons that despite a widespread need there is little advance in building intelligent consultants for statistical problems. There are two distinct methods. The first one is the methodology of rapid prototyping in which knowledge acquisition and implementation are mixed. The knowledge engineer uses interview or textbook data and immediately starts building a system in an implementation formalism. How to make decisions regarding implementation and architecture remains unclear (cf. Hayes-Roth et al., 1983). The second methodology, which we call structured knowledge acquisition, is outlined by Wielinga and Breuker (1984). The StatCons case study aimed to test this early version of the KADS methodology.
A major characteristic of the methodology is a separation between knowledge acquisition and implementation. The task for the knowledge engineer is to bridge the gap between verbal data from experts and the actual implementation of a system. The methodology provides a theoretically founded step in between. It guides the knowledge engineer in the mapping from verbal data onto an intermediate level provided by an interpretation model, which is an implementation-independent description of the domain knowledge on an epistemological (Brachman, 1979; Clancey, 1983), or knowledge level (Newell, 1982). It consists of a typology of basic elements and structuring relationships—e.g., “isa” and “consist-of” relations—for a certain class of problem solving tasks.

The basic elements are objects, knowledge sources, models, and strategies. On the implementation level, knowledge sources can be algorithms or sets of production rules. On the epistemological level a knowledge source is a piece of knowledge that derives new information from existing data. It is equivalent to an elementary subtask, which cannot be decomposed further. A knowledge source that occurs in almost any problem solving task is for instance the classification of objects into categories. Knowledge acquisition consists of repeated cycles of elicitation and analysis of verbal data aimed at refining (and if necessary, rejecting), an interpretation model.

Crucial to the methodology is the use of think-aloud data. They provide the most informative window on expertise in action. However, in knowledge engineering, these data are hardly ever used. They are assumed difficult to interpret (Welbank, 1983), and their use is only recommended as a check on the adequacy of a prototype. However, as psychology of problem solving shows, the analysis of think-aloud data is feasible when an initial model of the task is used as an interpretative framework (Ericsson and Simon, 1984). In KADS a classification of such models is available (Breuker et al., 1987). The knowledge engineer selects one or more interpretation models, describing the expert tasks at a global level. Interpretation models will be explained in more detail below. There are interpretation models for diagnosis, planning, design, etc. The advantage of interpretation models is that the knowledge engineer is equipped with a tool that is much closer to the verbal data than an implementation formalism. There are more practical advantages: repair and refinement of the model does not require throwing away some prototype in an inappropriate formalism.

An interpretation model is a template model for solving problems of a certain type, abstracted from the specific application domain. It provides a categorisation of knowledge and knowledge use that is pertinent to the type of problem solving task, and as such, tells the knowledge engineer what to look for in the verbal data. This can be illustrated with an interpretation model for design tasks (Figure 2.1).

Although this simple example model is not very detailed, it does provide guidance as it suggests the knowledge engineer to try to identify what type of constraints, specifications and solutions there are in the specific domain of application. Interpretation models can be more sophisticated. The point here is that a library of interpretation models, if they are sufficiently general and abstract, and if they still provide a relevant categorisation of types of knowledge and knowledge use, can give guidance and support to the knowledge engineer.

Like any methodology, KADS contains specifications of activities, tools and techniques that support the system development process. Early KADS uses the Life Cycle Model (Barthéley et al., 1987), as a prescriptive model for system design. In contrast with rapid-prototyping, system development is viewed as a sequence of analysis,
design and implementation stages. In the analysis stage a model of the problem solver is constructed by abstraction from verbal data. The model of the problem solver is input for the subsequent design stage in which the architecture for the future system is specified. Because in the domain of statistics the term “analysis” is already in use, we will consistently use the term “knowledge acquisition” instead of “analysis”.

Among tools provided by KADS are a number of “modelling languages” for expressing domain knowledge and models for problem solving. Central to the methodology is a library of interpretation models, which provide initial frameworks for constructing the model of the problem solver. In addition, KADS provides descriptions of techniques for data collection (elicitation) and data analysis. Some of these techniques are supported by tools in the SHELLEY system (Anjewierden, Wielemaker & Toussaint, 1992).

For the knowledge acquisition stage, KADS prescribes three types of analysis activities:

1. **Task Analysis.** The first is an analysis of functions, the environment and the users of the expertise to arrive at a definition of the operational characteristics of the prospective system. The aim of this task analysis is to define the role of the expertise. A knowledge-based system contains two types of tasks: problem-solving tasks representing the expertise and communication tasks. These communication tasks are by no means trivial; they form the interface between the operational environment and the expertise.

2. **Analysis of Domain Concepts.** The second activity is an analysis of the static domain knowledge, starting with the collection of a lexicon, leading to a set of concepts structured in KLONEtype concept hierarchies (Brachman and Schmolze, 1985).

3. **Analysis of Expertise in Action.** The third engineering activity is the analysis of expertise in action, i.e. the way in which problems are solved. This starts with a
detailed analysis of the reasoning process: selection of one or more interpretation models that appear to represent the structure of the problem solving process. By matching the verbal data from interviews and, in particular, think-aloud protocols, this initial model gets refined and modified into a detailed structure of knowledge objects, knowledge sources and strategies; much in the same way as Bennet’s (1979, 1985) conceptual structures. The final conceptual structure of expert reasoning represents the basic architecture of the prospective system. In the conceptual structure the domain concepts and the actions performed on them become integrated.

This concludes a brief overview of the KADS method for system development. It was used to design a prototype of a statistical consultancy system. The system imitates support as provided by the human statistical consultant. Sections below describe the knowledge acquisition, the design, the implementation and the evaluation of the prototype.

2.2 Knowledge Acquisition for StatCons-0

This section presents the knowledge acquisition for the StatCons-0 prototype. Subsections present the initial task analysis, the analysis of the domain concepts and the analysis of expertise in action. The verbal data to analyze and model the role of system and user were provided by protocols of real and role-playing consultation sessions. The data used for the task analysis of the statistical investigation task were obtained from statistical textbooks.

2.2.1 Task Analysis

The initial task analysis aims at identification of objects, agents and functions in the expert task, as to determine the role or function of the system in the users’ task. The role of the future system should fit well with the user, or, in a wider context, with the work processes of the organization. Various fields such as ergonomics, human-computer interaction and management all use techniques that follow the same pattern: The task or the work is decomposed into parts, that are allocated to the user or to the system, or, in management, to different workers. If necessary, task parts can be decomposed recursively, and in general the decomposition has a tree-structure. Task analysis is used in a similar manner in knowledge acquisition to define the role of the system.

One task may need information produced by another task. The interdependency between the two can be characterized as an information object that “flows” from one to the other. Hierarchical Data flow Diagrams (HDFD’s) can provide the tree-like decomposition of tasks, and, within each diagram and subdiagram the information dependencies among subtasks. Therefore (H)DFD’s provide a suitable language for (hierarchical) task analysis. Below, first the initial task decomposition is presented. Then we turn to the role of the human statistical consultant in the present situation. Finally the role of the future StatCons system is described.

The meaning of Data Flow Diagrams  As soon as the first diagram is shown, the reader may ask for the exact meaning of the elements and relations in the diagram.
In retrospect, there are two alternative interpretations that can be used for the same diagram. The procedural interpretation and the criterion interpretation.

The *procedural interpretation* regards a task with inputs and outputs as a procedure. First, at time $t_1$, the inputs have to be available. Then the procedure can be performed, and then, at $t_2$, the outputs become available. The direction of the arrows in the diagram has a temporal meaning and it provides control information regarding the task execution.

The procedural interpretation provides a rather specific model of the task execution. It is already close to a program in a procedural language, which is the context in which the software engineering diagramming practice developed. In classical software engineering, the data flow diagram is often interpreted in a procedural fashion, but at the same time it is recognized that except for simple cases, the data flow diagrams can be lacking in precision and detail, or give rise to ambiguities regarding the exact timing and control (Wallace *et al.*, 1987). When this occurs, or, as a standard element of a software design method, a separate analysis can be made—a behavioural model—of timing and conditions. For this one can use modelling languages such as state-transition diagrams, Petri nets, or specification languages such as proposed in Chapter 6.

*In conclusion*, in simple cases the data flow diagram (*i.e.*, the functional model), can be used to express control information regarding the task execution. Otherwise a separate model (*i.e.*, the behavioural model), may be used to give this information in all detail.

There is, however, a more fundamental problem, that will surface at the end of this chapter. Even in domains and tasks where it seems possible to perform subtasks according to a simple control structure (*e.g.*, a fixed sequence), this is not what usually happens in real task executions.

If one wants to impose a strict procedure to future task executions as a straightjacket, as a Procrustian bed, then all kinds of difficulties may arise, up to total paralysis. Similar criticism was given on office procedure systems by Gasser (1986), and Bannon & Schmidt (1989).

That the outputs, throughputs and even the inputs of a task may change during the execution, and even afterwards, is something that every researcher and designer knows. One starts with a problem formulation, one begins working on a solution, one changes the problem formulation somewhat, and finally one can only find a partial solution. For the final report one changes the problem formulation into one that fits with the partial solution, now presented as a complete solution. The only place where the sequence implied by a data flow diagram materializes, is in the sequence of presentation in the final report.

One may say that the ordering in the data flow diagram represents the simplest execution trace of all possible traces that lead to same final result. That quality makes it an efficient explanation of the end result. The actual execution trace in the task execution will seldomly be so straightforward.

*In conclusion*, we can use data flow diagrams, if necessary supplemented with, for example, state-transition diagrams, to represent a domain task as a procedure, but then we may be working with too much of a simplification of real task execution.

The *criterion interpretation* regards a task with inputs and outputs as a function or relation in the mathematical sense: there is no temporal ordering implied. Data
Flow diagrams are then connected function schemas (Figure 2.2), and the hierarchy can be based on function composition or conjunction and disjunction operators. Under this interpretation, the same diagram provides much less information. It defines only the names of domains (information objects) and names of functions among these. It thus provides a model of the task execution of rather low specificity; it contains no temporal or control information whatsoever. Therefore, it is rather an agenda for further knowledge acquisition than a solution. In this criterion interpretation the functions are no longer processes that can achieve inputs from outputs, but reduce to criteria for a situation \((e.g., (x, y, x) = (x, g(x), h(x, g(x))))\), that must be achieved in the end of a successful task execution.

![Data Flow Diagram](image.png)  
![Diagram of Functions](image.png)

Figure 2.2: Data flow diagram as set of function schemas.

In the StatCons-0 knowledge acquisition the data flow diagrams of the hierarchical task analysis were drawn with the procedural interpretation in mind (the input of a task precedes the output of a task; the activity of a producer must precede the activity of a consumer). These diagrams will still be valid in Chapter 4, but with the criterion interpretation.

Task Analysis of the Statistical Investigation Task

StatCons is conceived as a system that can support a user performing a statistical investigation. A statistical investigation, in general, is a method for obtaining new information. For example, a survey to find out about the average household budget in the Netherlands, or a laboratory experiment to compare two user interface designs, as presented in Chapter 7. The case study focused on statistical investigations using so-called designed experiments.

As a first task decomposition, Statistical Investigation can be decomposed into three subtasks: Conceptualize, Data Collection and Statistical Analysis. The decomposition principle being used is simply this: if a task contains a clearly identifiable subtask (with a clearly identifiable output), the task may be decomposed in that subtask and "the rest". This can be made stronger when "the rest" can be decomposed in "the rest before" and "the rest after" the subtask. In Statistical Investigation there is a clearly demarcated subtask, namely Data Collection, with the Data Set as a clearly identifiable output. Data collection is a physical activity. For example, interviewers go door to door with a questionnaire, or a lab assistant takes measurements of blood pressure and body weight and hands out the pills with the drug or the placebo. The widespread use of computers for Statistical Analysis has brought about conventions for the formal encoding of the Data Set as a data file.

Conceptualize is simply defined as all the work before Data Collection. It comprises setting the objectives or requirements of the investigation, the selection or design of
measurement procedures, the overall planning and the statistical design of the data collection. We call this output Conceptualization and we regard it as an input to Data Collection.

Likewise, Statistical Analysis is defined as all the work after Data Collection. Statistical Analysis uses the Conceptualization and the Data Set as input and results in New Information, as a summary of the Data Set. Figure 2.3 represents the result of this task analysis as a data flow diagram. This diagram shows which types of information—"information objects"—need to be exchanged between subtasks. At the beginning of the analysis these information objects are, except for the Data Set, not well defined, but they can easily be related to examples.

Example of the domain task. Figure 2.4 shows an example from a statistics textbook. It is not untypical for investigations that take place in the Psychology Lab, but it is less complex.

The example in figure 2.4 is annotated to show how it can be related to the task model—the data flow diagram—in Figure 2.3. The initial text represents the Conceptualization and the cross tabulation with observed values of the dependent variable represents the Data set. The cross tabulation shows the structure of the statistical design: there are 6 observations in each cell. This structure is already defined in the initial text, that is, the structure of the Data Set is already part of the Conceptualization. The New Information is not visible in Figure 2.4. The student is supposed to complete the exercise—to finish the investigation and to extract the new information from the data set—and to do so the student has to compute the analysis of variance statistics and interpret these.

In conclusion. The first decomposition of the statistical investigation task in Figure 2.3 is rather strong, almost beyond dispute, because it is based on particular characteristics of the domain. The fact that Data Collection is such a clearly identifiable subtask, with the Data Set as a well-defined formalized output, makes this task decomposition almost obvious. The nature of the work material, data sets, dictates the clear-cut partition in activities that have to be carried out before data collection and activities that can (only) be carried out after. A task analysis like the one presented above can be used as a basis to describe the roles of the user and the future system.

Social Perspective: Critiquing the Conceptualization

Consultation scenarios. StatCons is envisioned as a system that could partially replace the role of the statistical expert in the Psychology Lab. Students or staff who
Conceptualization

In the construction of a projective test, 40 more or less ambiguous pictures of two or more human figures were used. In each picture, the sex of at least one of the figures was only vaguely suggested. In a study of the influence of the introduction of extra cues in the pictures, one set of 40 was retouched so that the vague figure looked slightly more like a woman, in another set each was retouched to make the figure look slightly more like a man. A third set of the original pictures was used as a control (variable CUE). These pictures were administered to a group of 18 male college students and an independent group of 18 female college students (variable SEX). Six members of each group saw the pictures with female cues, six the pictures with male cues, and six the original pictures. Each subject was scored according to the number of pictures in which (s)he interpreted the distinct figure as a female (variable FINT). The results follow:

<table>
<thead>
<tr>
<th></th>
<th>FEMALE CUES</th>
<th>MALE CUES</th>
<th>NO CUES</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>FEMALE SUBJECTS</strong></td>
<td>29 36 28</td>
<td>14 5 10</td>
<td>22 25 23</td>
</tr>
<tr>
<td><strong>MALE SUBJECTS</strong></td>
<td>35 33 38</td>
<td>8 7 16</td>
<td>20 30 32</td>
</tr>
<tr>
<td><strong>FEMALE</strong></td>
<td>25 35 26</td>
<td>3 5 4</td>
<td>18 7 8</td>
</tr>
<tr>
<td><strong>MALE</strong></td>
<td>31 32 34</td>
<td>8 9 6</td>
<td>15 11 10</td>
</tr>
</tbody>
</table>


Complete the analysis of variance.

Figure 2.4: An example taken from a statistics textbook (Hays, 1974). The example problem is decomposed in parts corresponding to the information objects in Figure 2.3.
Comment by an expert:

"... forget about the single observation (female subjects, no cue). Your variable cue has only two possible values, really, you have only data for male and female cues. You have two cells that can be compared in their effect on FINT. Because female subjects only get female cues and male subjects only male cues, you cannot distinguish between an effect of SEX and an effect of CUE. You may recode the data. You can recode the information contained in SEX and CUE. You create a new variable, say “GROUP”, and all subjects with sex=female and cue=female are coded as group=1. All male subjects with sex=male and cue=male are coded as group=2. Then you can do one-way ANOVA with GROUP as independent variable and FINT as dependent variable. This will tell you whether there is an effect on FINT, but you will not be able to distinguish an effect of sex, an effect of cue, or a complicated joint effect of sex and cue."

<table>
<thead>
<tr>
<th>FEMALE CUES</th>
<th>MALE CUES</th>
<th>NO CUES</th>
</tr>
</thead>
<tbody>
<tr>
<td>FEMALE</td>
<td>29</td>
<td>17</td>
</tr>
<tr>
<td>SUBJECTS</td>
<td>25</td>
<td>18</td>
</tr>
<tr>
<td>MALE</td>
<td>13</td>
<td>5</td>
</tr>
<tr>
<td>SUBJECTS</td>
<td>4</td>
<td>9</td>
</tr>
</tbody>
</table>

Figure 2.5: An example of a buggy design, with comment of an expert.

Example of the consultation task. For an example of the critiquing role, we can use the example problem presented above in Figure 2.4. The example has a perfect design, and if the investigator in the scenario would visit the expert, the expert would say: "well, what a neat design, you can use analysis of variance, of course". In real-life consultations, the design is usually not so perfect. For a really faulty example, we can change the example problem by replacing the cross table with the table shown in Figure 2.5. This Figure also shows the expert’s critique of the design and his advice to make repairs in the design and to adjust the requirements. The example illustrates the importance of the table structure in the experts’ assessment of the users’ design. The example also illustrates that flaws in the design may have an effect on the statistical analysis.

To model this role of the consultant, Figure 2.6 adds a new task to the initial task analysis in Figure 2.3. The Debug & Advise task identifies flaws and takes these into account in producing the Advice for statistical analysis. In Figure 2.6, the roles of client and consultant are characterized by partitioning the diagram in tasks for the client and tasks for the consultant.

An analysis of the roles of user and system like the one presented above helps to outline the boundary of the system. Using the initial task analysis, the role of the system can be characterized as the set of tasks —subtasks or new tasks related to the subtasks— allocated to the system. These are to be analyzed in more detail to further develop the system, but the boundary between user and system defined in Figure 2.6 is still rather vague.
This social perspective sets the agenda for further analysis. Regarding the future system, Figure 2.6 begs two questions. The first is the question how a user's Conceptualization can be represented, or, more specifically, what kind of language or notation one may use to encode and enter a conceptualization to be judged by the system. The second is the question how a Conceptualization can be judged, how flaws can be identified and what kind of remedies can be prescribed.

Besides these, there is a question at the level of the knowledge acquisition process. We have made a step in task analysis, we are in a slightly better position to state what the future system should do, but the question is how the knowledge acquisition process should proceed. Should we continue the hierarchical task analysis, or should we consider "Task Analysis" as finished, and move on to "Analysis of Domain Concepts"? Both were tried, and, as will be explained in the next section, the bottom-up "Analysis of Domain Concepts" failed. Progress in defining the Conceptualization could only be made by a continued, more detailed analysis of the task Conceptualize and its output.

Further Analysis of the Conceptualize Task

The hierarchical data flow diagram at the top of Figure 2.7 presents the final result of the analysis of the domain task Statistical Investigation, with emphasis on the Conceptualize task. The first level of decomposition in the analysis of the domain task was presented before as a data flow diagram in Figure 2.3.

The second decomposition, level (2) in Figure 2.7, is based on the KADS interpretation model for design tasks (Figure 2.1). The Conceptualize task at the previous level is identified as containing a design task. The interpretation model for design tasks then provides the decomposition of Conceptualize in two subtasks and of the Conceptualization object in two parts: Requirements and Design Solution. This decomposition is simple, due to the availability of a template model that could be selected after identifying Conceptualize as a design task. The use of generic models as templates appears to be a worthwhile extension to hierarchical task analysis.

The third decomposition, level (3) in Figure 2.7 reaches the bottom. We can stop here, because we have identified languages or notations that may be used to encode the four objects that compose the user's Conceptualization. For the "Research Plan" a ready made pictorial language provided by O'Keefe (1981) is used. Figure 2.8 shows selected notations being used to encode different parts of the Requirements and Design Solution.

The decomposition of objects (e.g., Requirements decomposes in Variables and...
Figure 2.7: The complete task analysis of the domain task Conceptualize.
Requirements/Variables: (set of elements with attributes)

SEX  {male, female}
CUE  {male, female}
FINT  N

Requirements/Research Question: (graph or schema over the Variables)

Design Solution/Operationalizations: (more attributes of Variables)

SEX  sampling
CUE  treatment
FINT  measurement

Design Solution/Research Plan: (O'Keefe diagram)

Figure 2.8: Notations for the Information Objects
Resesarch Question), suggests a decomposition of tasks that produce them (e.g., Specify(Requirements) decomposes in Specify(Variables) and Specify(Research Question). This brings the task decomposition also at the third level (See Figure 2.7). The pictorial languages also provide an intermediate step towards a set of KLONE concepts as will be shown in the next section.

2.2.2 Analysis of Domain Concepts

KADS prescribed the collection of a domain lexicon—a list of domain specific terms—and structuring these in KLONE concept hierarchies (Brachman, 1979; Brachman and Schmölze 1985). KLONE is a semantic network or frame language that can be used to define concepts or frames in inheritance hierarchies. The domain lexicon was collected from textbooks, from interviews and from protocols of consultation sessions. This provided a list of more than 500 terms. Structuring these, or at least an important part of these more than 500 terms in various is-a and consist-of hierarchies was not very successful due to the large amount and different views on their roles by different statisticians.

It is impossible to collect and integrate “all” domain concepts, but perhaps we do not need “all” domain concepts. Perhaps a large share of the domain terms are what Clancey (1983) calls ‘support knowledge’, i.e. the terms are part of underlying theories and principles in the domain. This support knowledge is not used and functionally accessed in actual problem solving. If we look in protocols of consultation sessions, only a subset of domain concepts seem to be used, and, moreover, their meaning varies less. In fact, inter-expert discrepancies do not disappear completely, but different experts, using different concepts, seem to reach, generally speaking, the same results. We assume therefore that we do not have to collect and integrate “all” domain concepts, only those that are relevant in actual problem solving.

The task analysis in the previous section provides a notion of relevance, especially the decomposition of the Conceptualization object. Figure 2.8 shows a set of notations/pictorial languages for the different parts of the Conceptualization. This decomposition and the pictorial languages have finally guided the “Analysis of Domain Concepts”. In this analysis a KLONE data model was drafted for the elements and relations described by the pictorial languages/notations. Figure 2.9 shows part of the domain concepts for StatCons-0 in a KLONE-based language, restricted to those that would be needed to encode the Conceptualization as exemplified by Figure 2.8.

The set of KLONE concepts may be regarded as a detailed scheme for a formalized encoding of a Conceptualization. Each concept comprises a number of roles. Each role has a name and a definition of the number of fillers (i.e., a cardinality range \((x, y)\)), and of the type of filler. This type refers to another concept or to a predefined type like Number. Concepts may be organized in an ‘isa’-hierarchy, meaning that concepts lower in the hierarchy inherit the roles from concepts higher in the hierarchy (Brachman and Schmolze, 1985). The figure shows only the StatCons-0 concepts related to the Conceptualization object. With these concepts, as a formal notation for the information objects, the hierarchical task decomposition reaches a solid bottom.

The KLONE data model complements the hierarchical task analysis by adding detail and by providing a formalization for the information objects involved in the task. The concepts provide a precise definition of what the information objects are.
Such a model in KLONE can be regarded as a data model—it defines data structures and provides a formal coding scheme—or as an "ontology". As Gruber (1992), states: it constitutes a view on what kinds of objects and relations are relevant in the domain and establishes a vocabulary of terms (names of sets, relations, functions), that shall be used to represent these.

In conclusion. With the above, the Analysis of Domain Concepts is finished. Together with the Task Analysis started in the previous section, it provides us with a fairly comprehensive model for the Conceptualize part of Statistical Investigation. The model includes a decomposition of the information object Conceptualization, and thereby it provides much more detail to the system boundary in Figure 2.6. The information objects, at the most detailed level, are described as domain concepts. That is, a data model is provided that can be used to encode a Conceptualization.

It should be emphasized that the data modelling is driven by the hierarchical task analysis. The analysis of domain concepts is not a separate activity that can be performed independent of hierarchical task analysis. The hierarchical task analysis steers selection of domain concepts that are relevant, that is, domain concepts that can be used for the information objects. The bottom-up approach starting with the collection of "all" domain specific terms and trying to structure these into a single framework has failed, but the top-down approach starting from information objects was more successful and resulted in a coherent framework for the domain knowledge.

Conceptualize is a task outside the system boundary according to Figure 2.6, but the model is useful as it provides much more detail and precision on what the boundary comprises.
2.2.3 Analysis of Expertise in Action

The third and final stage in knowledge acquisition is the Analysis of Expertise in Action. It is an analysis of how the expert solves problems, and it results in a model of the problem solver (an expertise model), that can serve as input for the design and implementation of software. Ideally, the third and final analysis activity is only a matter of selecting and refining an interpretation model into a model of expertise for the problem solving task at hand. Unfortunately, there is no readily available model for a task like Debug & Advice. One might think of a diagnosis task, but there does not seem to be any malfunction or complaint which would normally be involved in a diagnosis task.

To resolve this, it was decided to use the same hierarchical task analysis technique as was used for the analysis of the domain task Conceptualize. The Debug & Advice can be decomposed in three functions:

1. a dialogue with the user that yields the user's Conceptualization. The analysis of the domain task Conceptualize can be used here.

2. identification of flaws in the user's Conceptualization and selection of a statistical analysis method.

3. providing the user with advice.

A second decision, to get on with it, was to overlay a procedural interpretation over the functional analyses. We read the functional models —the data flow diagrams— as single sequential processes. There is thus a strict separation of functions: each function is processed individually at one time, and there is a fixed sequence. In this manner, we have a model for the task execution process that is as simple as possible.

Figure 2.10 presents a more detailed analysis of the task of the statistical consultant. If there are no flaws, the Advice needs to be no more than a recipe for the statistical analysis, it can remain simple, just the name of the analysis method or analysis program, perhaps extended to a complete sequence of commands for using a statistical analysis program. For the user this is direct instruction how to act in the statistical analysis. Generating this is not a difficult task. Textbooks offer decision tables (Siegel, 1956; Fienberg, 1977), there is a decision tree for selecting statistical analysis methods available (Andrews et al., 1981) (See also: Van den Berg, 1992).

If there are flaws, then the system must point out what is wrong and how these flaws may affect the user's goals. Generate Advice then becomes a complex task.

Both Debug and Select Analysis Method can be based on Clancey's (1985), model of heuristic classification. In this model, there is an enumerated set of "solutions" (e.g., labels of analysis methods or labels of flaws). Solutions are selected using heuristic associations, that is, rules with on the left-hand side conditions in the "problem data" (i.e., the Conceptualization), and on the right-hand side the name of a solution. Making the heuristic association may be preceded by a data abstraction step, in which problem data entered by the user are transformed into problem data at the level of the heuristic association rules. So, the conditions need not be at the level of detail as used in the dialogue with the user. The heuristic association may be followed by a
refinement step, in which rules are used to determine the values of parameters of the selected solution.

The data abstraction from the Requirements is simple: computing the number of independent variables and the number of dependent variables. The data abstraction from the Design Solution is much more complicated: simple properties need to be extracted from the complexly structured Research Plan. These, and properties of the Requirements are used in the heuristic association with flaw labels. Generate Advice presents the selected analysis method and/or flaws as text to the user.

In conclusion. At this point the knowledge acquisition stage finishes. In the description above a fair overview is provided of the scope and difficulties of the knowledge acquisition in the first cycle of the StatCons case study. Although the knowledge acquisition was in no sense complete, it was taken as the starting point for a finger exercise.

2.3 The StatCons-0 Prototype

The previous section provided an overview of the knowledge acquisition for the StatCons-0 prototype, which was documented as a technical report in the KADS project (DeGreef, 1984). The document, that is, the expertise model summarized in previous section, was the basis for design and implementation of a prototype. This section shows the trace of the dialogue with the system on the example scenario and finishes with a critical evaluation of the prototype.

The prototype was designed following the Expertise Model presented in Figure 2.10. There is a one to one mapping from tasks in Figure 2.10 to modules of the prototype. The various modules were coordinated by a sequential control structure and if-then-else constructs.

A session with StatCons-0 has the following structure (see also Figure 2.10):

1. A question-answer dialogue about the user’s Conceptualization. The structure of the dialogue follows the structure of the hierarchical task analysis of the Conceptualize task. So we can decompose the dialogue in the part about the Requirements, and the part about the Design Solution

2. The system detects flaws.
3. If there are no fatal flaws the system selects a statistical analysis method.

4. The system presents the user with feedback on the Design Solution. The user will get a text explaining flaws, and, if possible, repairs.

The User Interface

Below the dialogue is presented that a user would have with the system if he would use the system to obtain advice about the example scenario that was presented in Figure 2.4. First the user answers questions about the Requirements, then about the Design Solution and then the system presents feedback.

Requirements/Research Question. The dialogue starts with a few questions about the type of schema needed to formulate the research question.

**SC:** First you will have to answer a number of questions about the conceptual model of your investigation.

**SC:** What kind of research question do you want to solve ?

SC: 1- a question about one variable
SC: 2- a question about a set of variables
SC: type <number> <dot>
CLIENT:: 2.

**SC:** What kind of problem do you want to solve ?

SC: 1- a relations problem
SC: 2- a dimension problem
SC: 3- a classification problem
SC: type <number> <dot>
CLIENT:: 1.

**SC:** What do you want to know about the relations ?

SC: 1- calculation of strength of relation
SC: 2- testing a hypothesis
SC: type <number> <dot>
CLIENT:: 2.

Requirements/Variables. Then the names of the variable are asked:

**SC:** Which variables did you use in your research ?

SC: type <name1> <return>
SC: <name2> <return>
SC: .. etc ..
SC: type <number> <dot>
CLIENT:: sex, cue, fint.

Requirements/Research Question. Then it is tried to give each variable a place in the schema for the Research Question:

SC: What can you say about the variable fint?
SC: (1) fint influences the value of some other variable
Design Solution/Operationalization. With the above, the dialogue about the Requirements is finished. The second part of the dialogue is about the Design Solution. It starts with the operationalizations of the variables:

SC: How is fint operationalized?  
SC: 1- measured  
SC: 2- selected  
SC: 3- manipulated  
SC: 4- not_operationalized  
SC: type <number> <dot>  
CLIENT:: 1.

SC: How is cue operationalized?  
SC: 1- measured  
SC: 2- selected  
SC: 3- manipulated  
SC: 4- not_operationalized  
SC: type <number> <dot>  
CLIENT:: 3.

SC: How is sex operationalized?

...
Design Solution/Research Plan/Sampling Structure. Having established that there is one sampling variable, the dialogue now turns to the sampling structure:

**SC:** How many units (for example: subjects) do you have in the sample sex=male?  
**CLIENT:** 18.

**SC:** Do you have an equal number of units in the other samples as well?  
**CLIENT:** yes.

Design Solution/Research Plan/Treatment Structure. Knowing that there is one treatment variable, the dialogue now turns to the treatment structure:

**SC:** What are the possible values of cue?  
**SC:** value1 value2 ....  
**CLIENT:** male, female, no-cue.

Design Solution/Research Plan/Assignment. The dialogue now addresses the assignment of units to treatment conditions. At this point the dialogue follows the O'Keefe diagram. An attempt is made to find one path from the sample (at the left of the diagram) to the final measurement (at the right of the diagram):

**SC:** Let us have a closer look at the sample sex=male.

**SC:** How are the units (subjects) assigned to the experimental conditions (treatments)?

**SC:** (1) Each unit (subject) is assigned to more than one experimental condition (treatment).

**SC:** (2) All units are random assigned to one of the experimental conditions.

**SC:** (3) Each unit is somehow split in two or more subunits, and these subunits are random assigned to the experimental conditions.

**SC:** (4) First, the sample is split into homogeneous groups of units. This is done by a measurement(s) on what is(are) called the block variable(s).

**SC:** Second, the units in each group are random assigned to one of the experimental conditions.

**SC:** (5) The assignment is done in a non-standard way.
CLIENT: 2.

SC: Did you apply exactly the same technique as with sex=male
SC: to the other sample(s) as well. That means the units in the
SC: other samples are assigned to experimental conditions
SC: (treatments) in the same manner?
CLIENT: yes.

Design Solution/Research Plan/Measurements. The structure of the Research
Plan regarding the assignment of units in sampling conditions to treatment conditions
is now established. The next step is to locate the position(s) of the measurements.

SC: Which variable(s) are measured after the assignment
SC: of units (subjects) to the experimental conditions (treatments)?
CLIENT: \text{fint}.

SC: Which variable(s) are measured both before and after the
SC: treatments?
SC: If none answer: none.
CLIENT: none.

Design Solution/Operationalization. The details of the operationalization of the
measured variable are filled-in.

SC: On what kind of scale is \text{fint} measured?
SC: 1: interval scale
SC: 2: ordinal scale
SC: 3: nominal scale
SC: 4: dichotomous scale
SC: 5: help
SC: type <number> <dot>
CLIENT: 1.

SC: What is the minimum value of \text{fint}?
CLIENT: 0.

SC: What is the maximum value of \text{fint}?
CLIENT: 30.

The Advice to the User. With the above, the dialogue to obtain the user's Conceptualization is finished. The final part of the dialogue comprises the feedback (flaws and Repairs) to the user and the advice about the Statistical Analysis Method.

***************
NO ERRORS detected in RESEARCH PLAN
***************

***************
ADVICE: analysis of variance with
sex and cue as independent variables and
fint as dependent variable.
***************
**The Advice to the User in case of flaws.** The example above shows no flaws, but the reader may think of the buggy example in Figure 2.5. The dialogue above about the Design Solution changes somewhat, and the feedback as the prototype would provide is shown below.

```
********************
ERRORS detected in RESEARCH PLAN
Unequal number of units per cell.
Empty cells.
- less power in the statistical analysis
- independent variables are correlated: distinguishing effects of different independent variables becomes difficult.
Repair: Fill all cells with equal number of units.
********************
ADVICE:
Analysis of Variance (ANOVA) with sex and cue as independent variable and fint as dependent variable.
```

The feedback to the user is perhaps not detailed enough. Very often the client or user will already have collected the data set. In case of flaws in the design of the data set the feedback provided by the prototype tells the user how to repair the design and which statistical analysis would be possible after the repair, but, if the data set according to the flawed design has already been collected, this type of feedback does not tell the user how to make the best of the available data. This will be discussed in the next section.

**Evaluation of the StatCons-0 Prototype**

A first conclusion is that KADS has helped to design and implement a working prototype. The user interface of the prototype is determined by the technology of its time: the dumb ASCII terminal. No formal evaluation was held, but a few persons have tried to use the system and in general the system appears to work. The user interface is based on menu-questions and requires hardly any instruction or learning. The persons that tried the system had knowledge about statistics and experimental design, but were no experts. Quite regularly a question required some explanation, but then users can complete the dialogue. The system could easily have been extended with canned help texts to provide such explanations.

A second conclusion is that the support provided by the prototype is probably less than optimal. The behaviour of the prototype is, apart from the application domain, similar to Miller’s (1983, 1984), critiquing. It is a simple sequential process: First a question-answer dialogue about the problem and solution (i.e., about the Conceptualization comprising the Requirements and the Design Solution), then evaluation
of the user's design solution, presented as text for the user. It may very well be
that critiquing is an effective method of support in the context of Miller's application
(small-size problem and solution, experts that once in a while have a less than optimal
solution), but in the evaluation of the StatCons-0 prototype a more interactive kind
of support seems more desirable.

Reviewing the StatCons-0 dialogue there appear to be two problems, two flaws,
in the StatCons-0 design. The first is not surprising. The rigid procedural inter­
pretation of the Conceptualize task is suited to drive the question-answer dialogue, but
is somewhat too rigid. The user can make mistakes in the question-answer dialogue,
but cannot make any changes along the way. If the user detects an error in the pre­
viously entered data, there is no other way to repair it, but to start the dialogue all
over again. The dialogue about the requirements and design solution can be quite
lengthy and starting all over again can be extremely annoying. The user should there­
fore be able to change something at any time, that is, during the dialogue about the
Conceptualization

A more serious problem reveals itself if we study the feedback provided by the sys­
tem. The textual feedback to the user as provided by the prototype comprises three
elements. First a notice (e.g., Unequal number of units per cell.), then a general state­
ment about the effect of the flaw (e.g., Distinguishing effects of different independent
variables becomes difficult.), and finally a “repair” that states what the user should
have done in his design (e.g., Use equal number of units in each cell.). The advice
to the user is thus to use a re-design without flaws, as illustrated by the example on
page 30. This is adequate in so far the dialogue is about choices, when the Design
Solution is only a plan that can be freely substituted by another. Unfortunately, this
feedback may be inappropriate as the flawless design may not be feasible in the real
world. Sometimes “flaws” are inevitable due to real world constraints. In this case,
the feedback should tell the user how to make the best of the data set according to
the buggy design.

If the user already has collected the data, which is often the case, the advice to
use a flawless design implies that the user has to throw away the available data set
and collect a new one using the new design, or to collect additional data to patch
the available data set. Because data collection is an expensive effort, the user would
want to be advised on how to make the best of of the data set according to the flawed
design. This user needs a repair that tells him what he should do in his current
situation, to make the best of it. If we think this over, we have to conclude that a
more adequate repair would comprise of specific operations that can be applied to
transform the user’s Conceptualization. The fictitious feedback below illustrates this.

ERRORS detected in RESEARCH PLAN

The design structure is a 2 by 3 table of 6 cells.
There are two cells with 6 units.
There is one cell, (sex=female,cue=no-cue), with only one unit.
There are three cells with no units.

The cell with one unit can be regarded as one with no units.
Delete the unit with (sex=female,cue=no-cue).

31
Now there are no units with cue=no-cue.
Cue is now a variable with two values: male, female.
The design structure is a 2 by 2 table of 4 cells.
There are two cells with 6 units.
There are two cells with no units.

You have two cells that can be compared in their effect on fint.
These cells differ on two variables: sex and cue. Therefore, in the analysis, if you find an effect, you cannot attribute it precisely
to an effect of sex, of cue, or both.

ADVICE: Delete the unit with cue=no-cue.
Analysis of Variance (ANOVA) with
either sex or cue as independent variable and
fint as dependent variable.

In conclusion, the StatCons-0 prototype relies too much on standard solutions for statistical design and analysis. To adapt to prevailing real-world constraints, or to make the best of the available data set (i.e., the user’s design), the system should provide the user with context-specific feedback about how to transform the Conceptualization, (i.e., Design solution, Requirements, or both), as to make the best of it. The desired feedback must thus comprise of operations that modify the Conceptualization. This kind of feedback would constitute a major improvement of the quality of the support.

These new requirements were not explicit in the knowledge acquisition document for the prototype, and it can be said that the prototype provides a fair coverage, albeit not complete in all respects, of the expertise model in Figure 2.10. As will be discussed in the next section, the problems in the dialogue may arise from problems in this expertise model and, indirectly, in the model of the social perspective (Figure 2.6).

Implementation of modifications. These new requirements for the system behaviour explicitly or implicitly require that the system is able to modify its internal data structures that represent the Conceptualization. The question is whether we can adapt and improve the prototype. The internal storage of the Conceptualization is according to the KLONE data model, which, in turn is based on pictorial notations like O’Keefe’s for the Research Plan. This organization of storage is not particularly conducive to implementing modifications of the Conceptualization. Of course one can edit the data structures, but such changes may result in an inconsistent Conceptualization. The StatCons-0 data model can be used as a coding scheme, but operations on the encoded object, and how these may be represented, have not been addressed at all. Therefore the prototype cannot easily be adapted as to meet the new requirements.

In conclusion. The prototype is a success in the sense that it meets the requirements of the knowledge acquisition document, but it has drawbacks that reveal requirements that were not made explicit in the knowledge acquisition.
2.4 Conclusions StatCons-0

The case study investigates computer support for users with lack of task knowledge. The aims were twofold. The first was to try out KADS in the design of a support system, with the emphasis on software development, that is, on methods for realizing a software system. The second was to test the hypothesis that by imitating human consultancy we may design better support systems, with emphasis on the question whether the system has a beneficial effect for the user. Assuming human-human consultation has evolved into an optimal way of supporting the client with lack of knowledge, automation of the human consultant may help to arrive at optimal computer support.

The results of the case study are threefold:

- The model for the system. The KADS analysis resulted in a text and diagram description of the system (i.e., section 2.2), here referred to as expertise model. This model of the system is summarized in Table 2.1 below.

- The StatCons-0 prototype. Using the model for the system, a prototype was implemented that can, in some sense, play the role of statistical consultant (section 2.3).

- Evaluation. The evaluation of the prototype concludes that the prototype in many respects is conform the model. The prototype satisfies the model, in some, as yet informal, sense. Analysis of the behaviour of the prototype (i.e., the human-computer interaction), shows the critiquing scheme to be too simplistic and two new explicit requirements were proposed (p. 31).

The subsequent section discusses the utility of the model-based approach to system development. The final section discusses model-based design of support and argues that a particular type of support can be described as a particular kind of model for the system's behaviour. To support this argument, we look back at the model on which the prototype is based. The problematic behaviour of the prototype can be traced back to this model. The origin of system behaviour can even be pointed out in the model of the social perspective. It is put forward that the desired behaviour can be brought about by a system based on a new model.

KADS.

The main conclusion is that especially the model-based approach of KADS helps the development of a working prototype. In the model-based approach to system development, knowledge acquisition results in a model of the future system. This model is input to design and implementation, resulting in a realization of a system that—if all is executed correctly—satisfies the model.

The knowledge acquisition stage is mainly concerned with drafting, refining and modifying candidate models. The final model of the future system is like the architect's drawings, describing the artifact to be, but not in all detail. The model takes a central role in the communication between knowledge acquisition and implementation. Both members of the implementation team praised the guidance and steering provided by the KADS knowledge acquisition document, i.e., by the model of the future system. Thus separation between knowledge acquisition and implementation appeared feasible.
The result of knowledge acquisition. The models that are the result of the StatCons-0 knowledge acquisition are summarized in Table 2.1.

<table>
<thead>
<tr>
<th>Topic of Analysis</th>
<th>Type of Analysis</th>
<th>Model Language</th>
</tr>
</thead>
<tbody>
<tr>
<td>The domain task</td>
<td>Hierarchical task decomposition</td>
<td>Hierarchical DFD Figures 2.3, 2.7</td>
</tr>
<tr>
<td></td>
<td>Data modelling</td>
<td>KLONE Figure 2.9</td>
</tr>
<tr>
<td>The roles of user and system task</td>
<td>Tasks for the user and tasks for the system</td>
<td>Partitioned DFD Figure 2.6</td>
</tr>
<tr>
<td>The consultancy task</td>
<td>Hierarchical task decomposition</td>
<td>Hierarchical DFD Figure 2.10</td>
</tr>
</tbody>
</table>

As can be seen in the table, the most prevalent activity in knowledge acquisition appeared to have been task analysis. We needed it to discuss the role of the expert c.q. the future system, we also needed task analysis to steer the analysis of the domain concepts. Finally, in the analysis of expertise in action, task analysis was applied to analyse the system’s tasks in more detail.

The task analysis is viewed as modelling and the hierarchical task analysis results in a hierarchical task model. Data flows and information objects play an important role in such a task analysis. Information objects, like tasks, may be decomposed in parts. The hierarchical task model keeps track of the coordinated decomposition of tasks and information objects, that is, inputs and outputs of tasks. Techniques from conventional software engineering, for example, hierarchical data flow diagraming can be used as a task description language.

In early KADS, the functional perspective dominated, that is, the hierarchical task analysis and the use of data flow diagrams. In the analysis of domain concepts, data modelling was no explicit activity, but the KLONE concepts used in the prototype constitute a data model. The set of domain concepts—the data model—provides an important supplement to the hierarchical task analysis. Like tasks, information objects are decomposed in parts, and once parts are small enough, domain concepts or data models can provide a formalized notation for the objects in the domain. Using these, information objects can be described by attributes, components, and relations among components.

The Knowledge Acquisition process. KADS prescribed a sequence of three stages for the knowledge acquisition process: Task Analysis, Analysis of Domain Concepts, and Analysis of Expertise in Action.

We have tried a different ordering: to do only a limited hierarchical task analysis of the domain task as far as needed to define the role of the system. Then it was tried move to the next stage, the “Analysis of Domain Concepts”. However, it turned out too difficult to complete the analysis of domain concepts in a bottom-up approach starting from terms collected in textbooks, consultation protocols and interviews. Therefore, the analysis returned to hierarchical task analysis of the domain task, until the inputs and outputs of subtasks (i.e., functional objects), are small enough.
such that languages or notations for all inputs and outputs, or parts thereof can be identified. Only after that there was enough focus for deciding about a set of domain concepts using KLONE as modelling language. The resulting set of concepts can be regarded as a data model, as a set of data structures to encode the user's Conceptualization. Such a data model is an important supplement to the functional model (i.e., the task decomposition using hierarchical data flow diagrams).

The conclusion is that indeed it is necessary to perform Task Analysis first, as to have sufficient steering and guidance towards domain concepts.

The third and final stage, "Expertise in Action", was spent on a task analysis of the consultancy task, that is, the set of tasks for the system. For want of a process model, we submitted to the simplest possible model: a procedural interpretation of the hierarchical task analysis.

In conclusion. The model-based approach works to produce a working prototype. If executed neatly, the prototype complies with the model. Knowledge acquisition is mainly task analysis, aiming at a model in which the tasks for the system are well-defined. The model may contain a number of sub models, and different sub models may utilize the same model language.

Support and Model-Based System Development

A support system can be characterized by a model. The framework of model-based system development can thus be useful in the design of support, one only has to take care to devise the right model. Below we trace the problems in the behaviour of the prototype back to the model on which it is based. It is not hard to point to the erroneous decision in the knowledge acquisition process. One of the earliest models, the one of the social perspective (fig 2.6), contains the germ for faulty behaviour. This diagram states that the Conceptualization is only an input. However, if the feedback of the system comprises proposals for changes of the Conceptualization, the latter is no longer just an input, but also a result. The model should change, to allow for modification of the Conceptualization.

A new diagram for the social perspective, in Figure 2.11, views Debugging as a separate activity that acts on the Conceptualization. Selection of a Statistical Analysis Method is considered part of Conceptualize, or, if the user has not yet come to that decision, Debug will make the Conceptualization complete, after necessary repairs (i.e., modifications), of flaws in the user's Conceptualization.

These alternative models of the system (Figures 2.6 and 2.11), show that important characteristics of computer support can, in part, be specified in the abstract model of the future system. Different types of support may be characterized by different models. These models are still very global and do not address social behaviour in detail. KADS at the time did not provide means to analyze the system behaviour towards the user.

A New Prototype

StatCons-0 was only the first of two prototypes. With Figure 2.11 only a start is made with an improved design. Much has to change or to be redone entirely to accommodate for changes of the user's Conceptualization. We cannot simply improve
the software of the first prototype. It is better to start all over again and first design a better model for the system.

The next chapter provides the theoretical groundwork we need for designing modifiable representations of the user's Requirements and Design Solution. Chapter 4 then presents the development of StatCons-1, a system intended to overcome the problems revealed by the StatCons-0 prototype.

Acknowledgements. Many thanks are due to the experts who made themselves generously available for interviews and who provided opportunities to record real consultation sessions on audio tape. Next I would like to thank Kees Buisman and Dick van de Vlucht for implementing the StatCons-0 prototype. This research was performed within the KADS project, which was partly funded by the ESPRIT program of the European Commission.