Modelling interaction between a KBS and its users

John Kingston

AIAI-TR-141

January 1994

This article was printed in the first edition of the newsletter of the BCS SGES Methodologies Interest Group in the spring of 1992

Artificial Intelligence Applications Institute
University of Edinburgh
80 South Bridge
Edinburgh EH1 1HN
United Kingdom

1 Introduction

Knowledge based systems (KBS) have been a commercially viable technology for over a decade now. As a result of their growing use, users and managers have demanded that KBS be verifiable, maintainable and repeatable. This has led to the development of a number of systematic methods which formalise and direct the knowledge engineering process. A survey of current methods can be found in [2].

The most widely known methodology in the UK & Europe is the KADS methodology. KADS views knowledge engineering as a modelling process [4]; a series of models are developed to represent different aspects of the knowledge engineering process. One of the models suggested by KADS is the *model of cooperation*. This model is used to help in identifying potential KBS applications. The model of cooperation models the processes involved in a real-world problem solving task, identifying the subtasks involved and the inputs and outputs of each subtask. Once this process model has been drawn up. it is used to identify subtasks which could be carried out by a KBS, or by a KBS and user working together. For example, if the real world problem-solving task was the preparation of a meal, then the model of cooperation would closely resemble Figure 1.

When the model of cooperation has identified potential KBS applications (i.e. those tasks which are identified as "system roles" in the model of cooperation), each potential application should be subjected to a feasibility study to determine if it is indeed appropriate to introduce a KBS to fulfil that task.

Once a KBS application has been chosen, KADS recommends further models for analysis and design of the KBS, with the model of cooperation providing details of inputs to and outputs from the KBS. However, the experience of many KBS developers has been that the respective roles of the system and user need to be analysed at a more detailed level; not only **between** problem-solving tasks, but also **within** a single problem-solving task. [1]

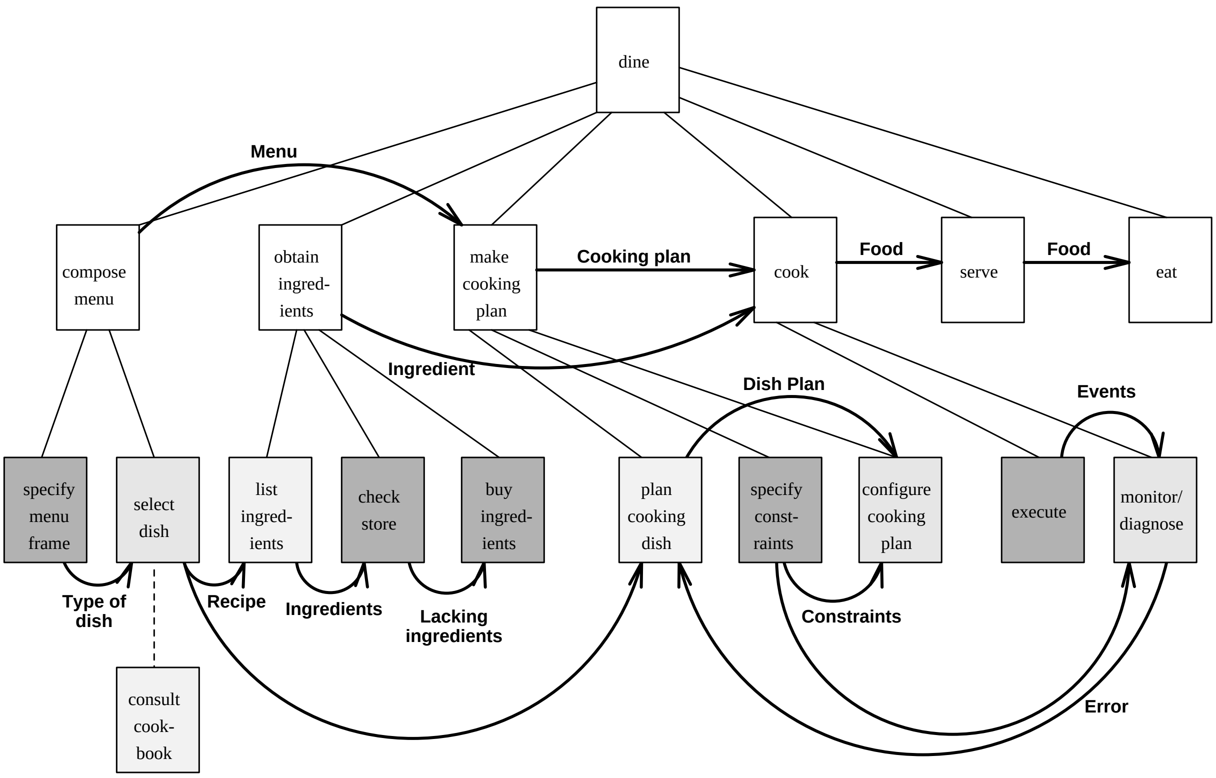
What is needed is a variant of the model of cooperation which analyses the inputs and outputs within a single problem-solving task. The purpose of this article is to describe such a model, and to demonstrate its usefulness. This model is termed the *model of interaction.*

2 The model of interaction

The model of interaction can be used in conjunction with any methodology (or in isolation), although it is derived from, and fits particularly well with, the KADS methodology. The procedure used to develop the model of interaction is a simple

[1]The KADS-II project is extending the analysis of cooperation to be more wide-ranging, but the focus is still on between-task analysis.

three-step procedure which is very similar to procedures used for the development of the model of cooperation.



Key:
Light grey: *system* roles
Dark grey: *user* roles
Medium grey: *system & user* roles

Figure 1: Model of cooperation for the preparation of a meal (from [1])

2.1 Developing the model of interaction

- Step 1 is to draw up an ordered list of the different subtasks which a KBS must perform in order to fulfil its problem-solving task. If KADS is being used, this list is provided by the *task structure*, which is developed as part of the analysis of acquired knowledge. An example of a task structure is given below; it is drawn from a KBS which helps civil engineers to check the design of a building against British standards:

task assessing-building-against-British-standards

goal check that a building design conforms to British standards

task structure

assessing-building-against-British-standards(results of checks)

obtain(numerical description of building)

transform(numerical description → model of the building)

select(a check to perform)

obtain(any further information required for that check)

match(model of building + standards relevant to the chosen check → result of check)

report(results of check)

- Step 2 is to identify the dependencies between subtasks i.e. the inputs and outputs of each subtask. Any I/O with external databases, files, the user, or other sources of information should also be noted.

- Step 3 is to decide which subtasks will be performed by the KBS, which by the user, and which by the KBS and user together. This final model can be represented in a diagram, as shown in Figure 2.

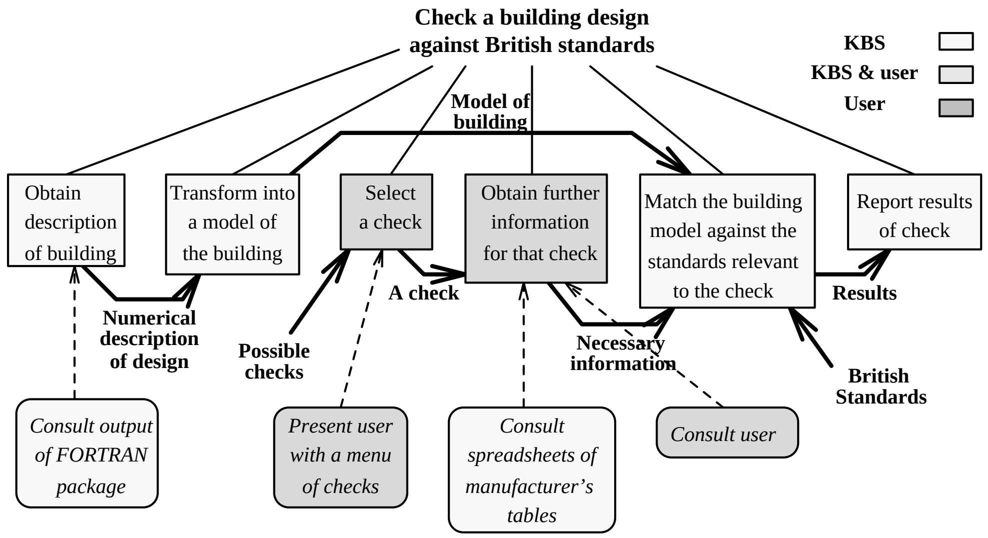


Figure 2: Model of interaction for the KBS which checks the design of a building against British standards

2.2 Benefits of the model of interaction

What are the benefits of the model of interaction? At the very least, it serves as a useful *aide memoire* for the designer of the I/O components of the KBS, both for the inputs & outputs of the KBS as a whole, and for any I/O which occurs within the KBS. The model of interaction is at its most useful when it is used to identify that one or more subtasks should be carried out by the user of the KBS; if this is so, the design of the KBS may be radically affected. An excellent example of this comes from a project carried out by AIAI for an insurance company. The task of the KBS was to identify errors on forms; the task structure is shown below.

task identify-errors-on-forms

goal check each field on a form against its predicted value to identify errors made when filling in the form

control-terms

fields = set of all fields on the form

task structure

identify-errors-on-forms(classified-errors)
decompose(form → fields)
do for each field ∈ fields
  specify(expected value)
  read(actual value)
  match(actual value + expected value → mismatches)
  classify(errors → classified-errors)

It was decided that the forms would not actually be input into the KBS; instead, the KBS would advise the user on fields to check, and the user would perform the actual checks. In the terminology of the model of interaction, the *match* subtask would be carried out by the user, as shown in Figure 3.

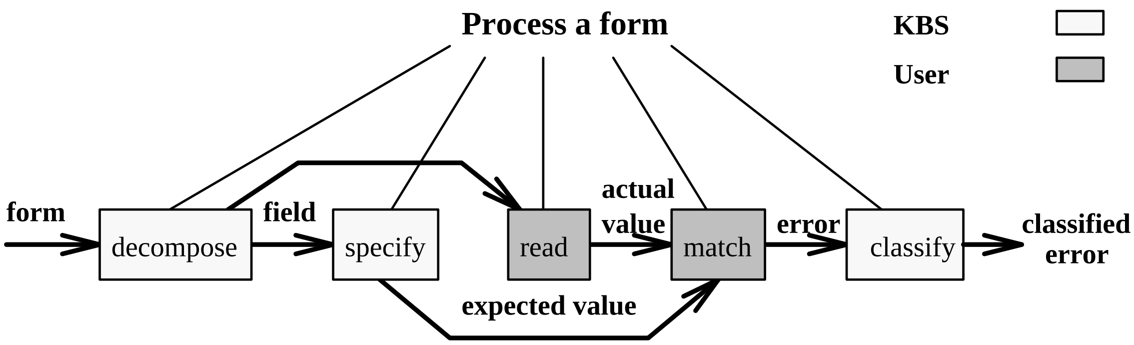


Figure 3: Model of interaction for the form processing KBS

The effects of assigning the matching task to the user were immense. KBS designers often use production rules to implement matching tasks; but, with the matching task being performed by the user, it was decided that the forms processing KBS could be developed entirely using object-oriented programming.

If KADS is being used, the model of interaction should be used as an input to the functional decomposition stage of KBS design. The model of interaction is most useful in a KADS-based project when a KBS is designed to handle different modes of interaction based on the same inference structure. For example, a KBS may allow users either to volunteer information or to be prompted for the same information. Many users of KADS associate an inference structure with a single task structure, and by implication with a single mode of interaction. The model of interaction explicitly represents the fact that one inference structure can be open to multiple modes of interaction.

The model of interaction is also useful in helping the knowledge engineer identify omissions from the task structure.

3 Worked example

3.1 Task structure

A model of interaction was developed as part of a recent AIAI project. The project was to assist a small manufacturing company to build a KBS for diagnosing faults in plastic moulding machinery [3]. The KADS analysis was based on an inference structure for systematic diagnosis; the task structure was as follows:

task *diagnose-contamination-problem*

goal diagnose the cause of contamination in plastic mouldings

control-terms

Possible tests → the set of all possible tests

Possible faults → the set of all possible faults

Expected results of tests → the set of expected results on each Possible test for each Possible fault

Related tests → the set of all tests which can be performed at the same time as the chosen test

Hypotheses → the set of all suspected faults

task structure

diagnose-contamination-problem (symptom + Possible faults + Possible tests + Expected results of tests → conclusion)

obtain (symptom)

identify (symptom + Possible faults → Hypotheses)

if (only one hypothesis in Hypotheses) then *report* (conclusion)

select-1:

for each test in Possible tests **do**

compute (time required for test → cost of test)

compute (likelihood of associated hypothesis/hypotheses → information value of test)

compute (cost of test + information value of test → utility value of test)
select (test with highest utility value → recommended-test)
obtain (user's choice of test → chosen test)
update (chosen test → log file of tests performed)
update-2 (chosen test → state of the machine)
report (changes to be made)
perform (chosen test → result of chosen test)
for each related test in Related tests **do**
inform user of related test
obtain (user's choice of test → chosen test)
perform (chosen test → result of chosen test)
for each hypothesis in Hypotheses **do**
compare (result of chosen test(s) with expected result for that hypothesis → hypothesis confirmed/ruled out/still suspected)
update (Hypotheses + hypothesis confirmed/ruled out → revised Hypotheses)

This task structure is more complicated than the examples presented above; however, with 12 knowledge roles (within parentheses) and 10 inference actions (in **bold font**) as well as a number of interface-related actions (in *italic font*), it is not exceptionally large for a commercial KBS project.

3.2 Model of interaction

The model of interaction for this KBS is shown below. The model was built up in three stages, which are shown in the following three diagrams:

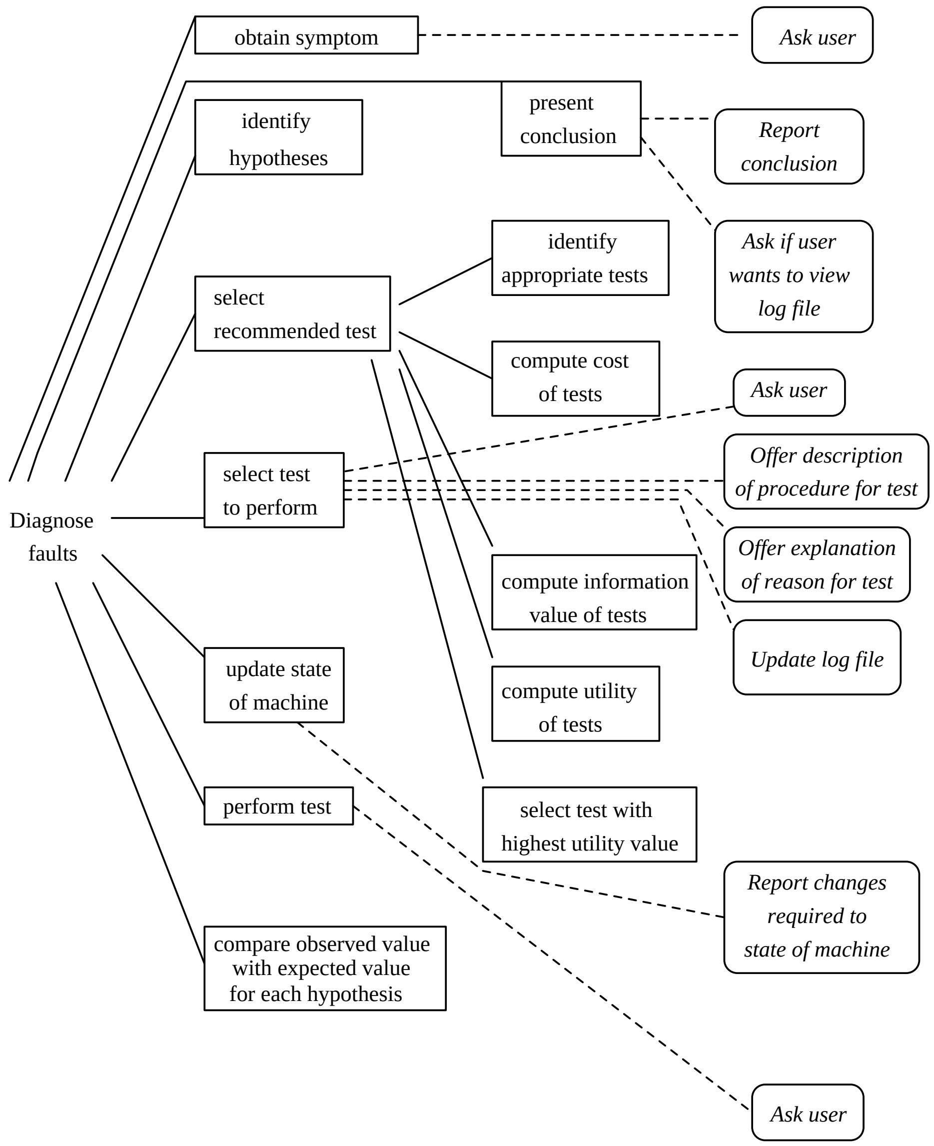


Figure 4: The task structure in graphical form

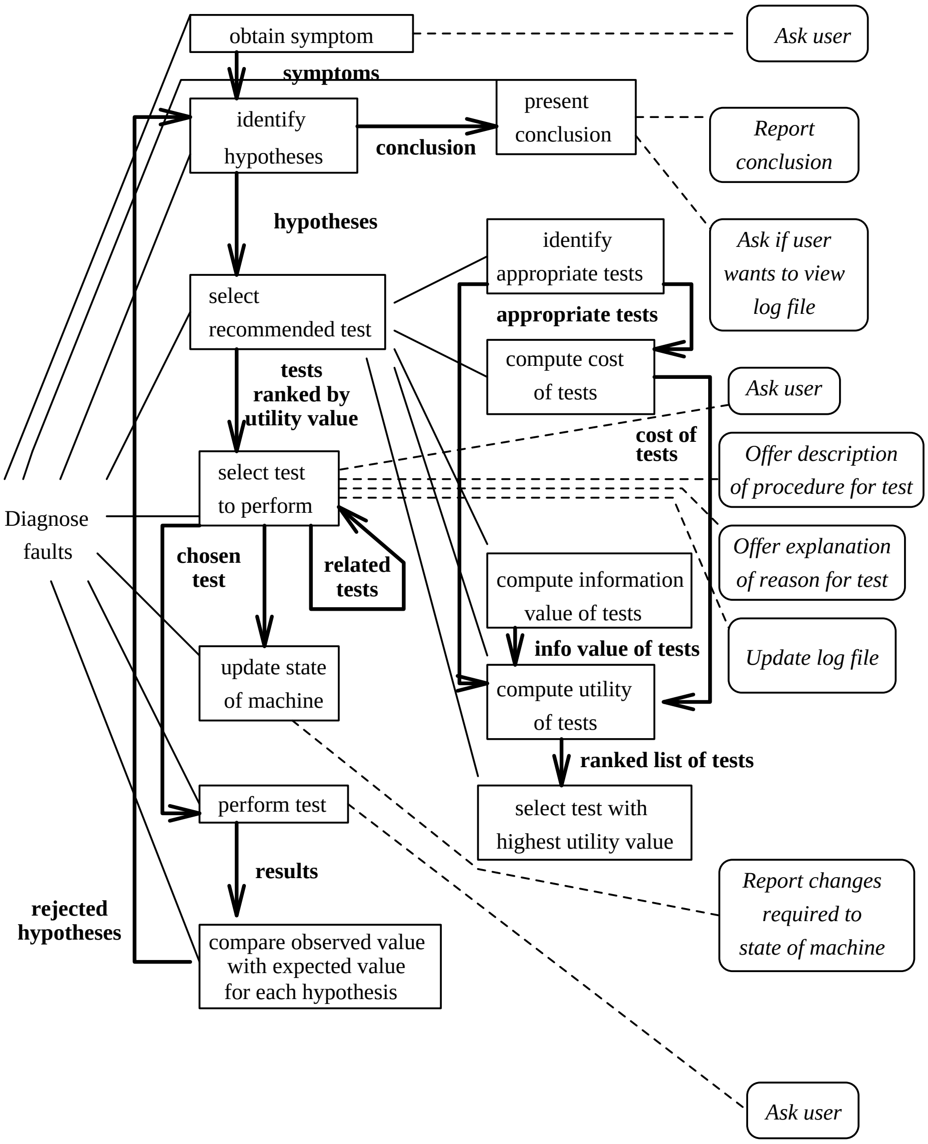


Figure 5: The task structure showing dependencies and user I/O

The main decision made when developing the model of interaction for this system was that the selection of a test to perform would be done by the KBS and user in conjunction, rather than by the KBS alone; in other words, the KBS would recommend a test to perform, but the user would be free to reject the recommendation. This is indicated by the medium grey shading of **select test to perform** on the following diagram:

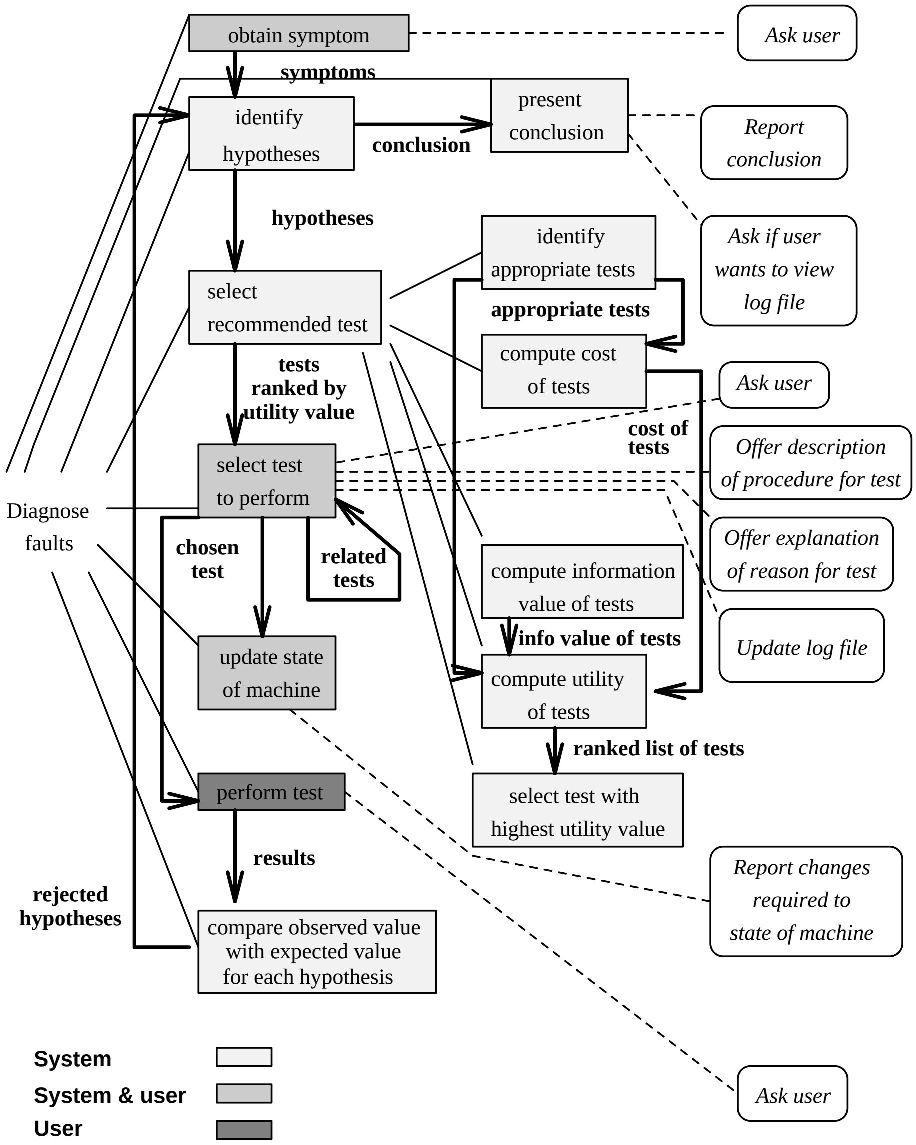


Figure 6: The model of interaction, with tasks assigned to different agents

3.3 Evaluation of the model of interaction on this project

Using a model of interaction was beneficial on this project, because it clarified the need for a decision about who would perform the selection of tests. It also proved useful in identifying all the interface-related actions which needed to be performed; while the task structure specified actions such as "inform user", the model of interaction notes every occasion when the user needs to be consulted. This information is useful in the design phase of KBS development (cf. the "cooperation functions" in the functional decomposition stage of the KADS design phase).

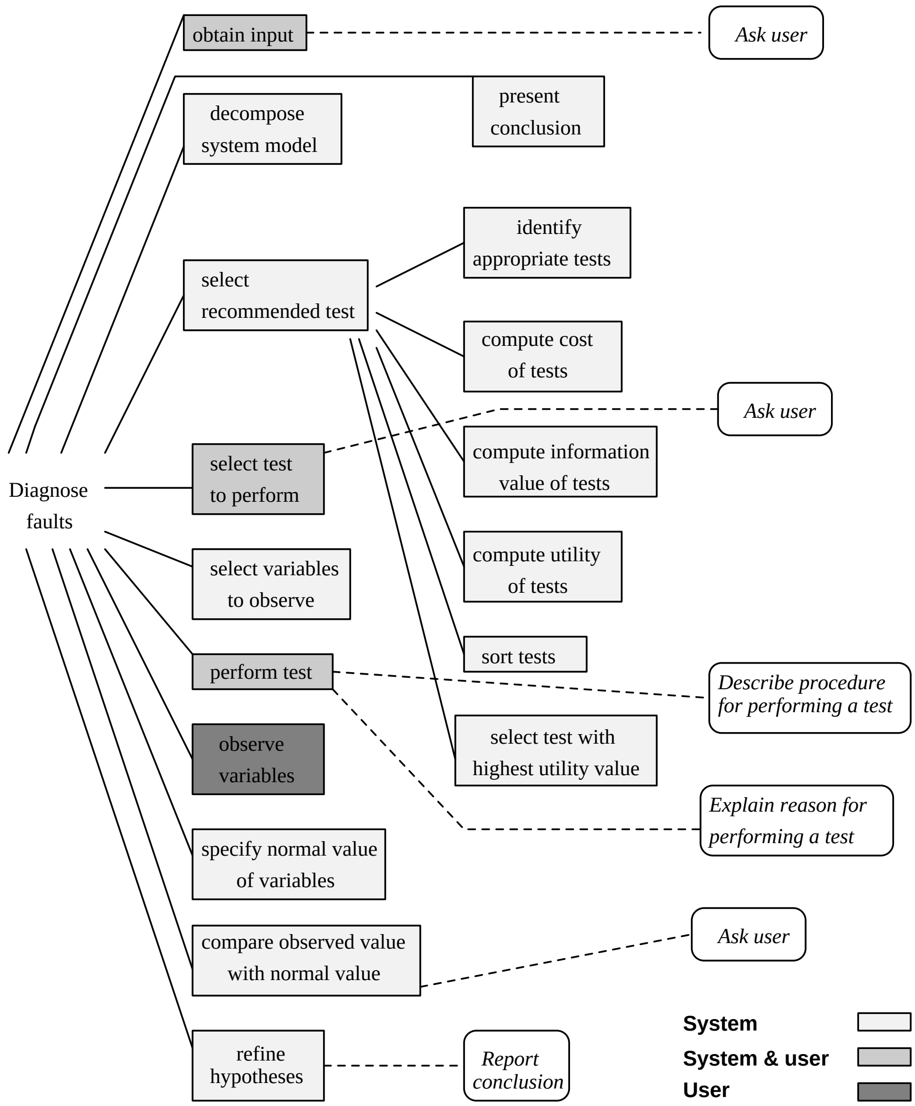


Figure 7: Simplified model of interaction for diagnosing faults in machinery

However, it is obvious that the model of interaction for this project contains too much information to be easily comprehensible by anyone except the developer of the model. This reduces one of the major advantages of a methodological approach to KBS development - that of producing good documentation of design decisions made during the development of a KBS. As a result, it is proposed that the model of interaction is simplified before it is delivered to anyone outside the project. The suggested simplification is to remove the arrows representing dependencies from the final model; while these dependencies provide useful information to the KBS developer, few external readers are likely to be interested in such a low level of detail, especially since it is represented in other models in KADS, or similar methods. The resulting simplified model for this project can be seen in Figure 7.

4 Conclusion

The model of interaction provides a method for modelling the respective roles of a KBS and its users within a single problem solving task, which has proved useful in identifying important decisions on more than one occasion. It is derived from the model of cooperation proposed by the KADS methodology, although it can be used in conjunction with any methodology, or in isolation. It provides a useful *aide memoire* for KBS designers, and its use may identify factors which have major implications for the design of the KBS. The model can become complicated, and so it is recommended that dependency information is removed from the model before presenting it to external readers.

References

[1] H.P. de Greef and J. Breuker. Analysing system-user cooperation in KADS. *Knowledge Acquisition*, 4(1):89–108, March 1992.

[2] R. Inder and I.M. Filby. A Survey of Knowledge Engineering Methods and Supporting Tools. In *KBS Methodologies Workshop*. BCS Specialist Group on Expert Systems, December 1992.

[3] J. Kingston. KBS Methodology as a framework for Co-operative Working. In *Research and Development in Expert Systems IX*. British Computer Society, Cambridge University Press, 16-17 Dec 1992. Also available from AIAI as AIAI-TR-130.

[4] B.J. Wielinga, A.Th. Schreiber, and J.A. Breuker. KADS: A Modelling Approach to Knowledge Engineering. *Knowledge Acquisition*, 4(1), March 1992. This article is also available from the University of Amsterdam as KADS-II/T1.1/PP/UvA/008/1.0.