A Designer's Decision Aiding System: DDAS

Jenny Darzentas, Thomas Spyrou, Eftihia Benaki, John Darzentas

Research Laboratory of Samos,
Dept. of Mathematics,
University of the Aegean

This paper describes the development of a working system for aiding designers of computer systems find appropriate tools and methods to enable them tackle usability problems. The approach taken for the design of the system is based on Soft Systems Methodology (SSM) and fuzzy reasoning in the form of Test Score Semantics and has been extensively described elsewhere. Here the building of the knowledge base and features of interaction with system are explained and illustrated with an example. The importance of the DDAS, lies not just in its usefulness to the designer, who now has access to bodies of knowledge in direct relation to a problem of concern, but in its claim to provide a methodology for decision aid in similar situations where problems exist, and tools to solve them also, but where a short cut or aid is needed to bring the two together

Keywords*: decision aiding systems, design, human computer interaction, expert systems.*

Acknowledgement*: The work reported on in this paper was funded by the ESPRIT Basic Research Action 7040 AMODEUS (Assaying Means of Design Expressions for Users and Systems)*

1. Introduction

The work described in this paper was carried out as part of the Amodeus project, one of the world's largest multidisciplinary HCI consortiums, developing modelling and analytical techniques for Human Computer Interaction.

The motivation for the genesis of the DDAS was as a transfer activity i.e. informing design practitioners of the potential of the Amodeus techniques. The approach that was taken to study the problem situation and design the system has been extensively described elsewhere [5,6,7,8].

The starting point for the architecture and development of the system is the assumption that the design practitioner confronted with a usability problem, - a problem which may be well articulated or only vaguely suspected, - would need some help to assess which techniques are best suited to solve this problem. In effect, by a designer decision aiding system is meant a system whose purpose is to

help a designer by first solving the problem of finding the most appropriate tools, within a specific array, to tackle the situation of concern.

In developing the DDAS some immediate limitations had to be imposed upon this scenario, in order to define the scope of the system.

Firstly and most obviously, it is not possible to offer aid for all problems or classes of problems, but only those that the Amodeus modelling techniques can handle. However, the array of Amodeus techniques covers design from a user oriented, task oriented, system oriented and design rationale points of view, and while one cannot claim that they cover all possible problems, it would probably be true to say that at least at some level of granularity they touch upon all.

Secondly, it is not possible to allow the designer-user to express his problem freely in natural language, because all effort would go to processing the input and trying to match it to the knowledge base of problems handled by Amodeus. Nor would this state of affairs be very desirable. Recent work in active DSS show that the user prefers "active" aid from a system, and wants to be prompted [11]. A user who has a very well formed idea of what his problem is would not mind expressing it , (though he may wonder if the machine is interpreting it as he wants), but a user who just "has a feeling" would have difficulties making that intuitive response to a situation comprehensible to the system.

Thirdly, the DDAS is designed at present to do no more than assess the user's need for technique(s) and make a recommendation and present this to him with some justification which will show the reasoning the system used to arrive at its conclusion. The system does not store knowledge about how a technique is used, or what skills are needed to use it. Thus it could be entirely possible that a recommendation is made which requires a background in software engineering which the user has not been assessed for. The reason for separating out the "what it does" from the "how it does it" was to remain faithful to a desire to have the techniques "compete" on similar terms and eliminate as much as possible constraints that would impose a too early limiting of choices upon the user. From a practical point of view as well, if a problem could be aided by a technique which requires, for instance, skilled personnel, then it is not unlikely that the design practitioner will take steps to acquire the services of said personnel, if at all possible.

The ways for dealing with the above three limitations were, firstly, to build a knowledge base containing descriptions of the aspects of the overall design space which are relevant to the modelling techniques; secondly, to present these descriptions to the user and ask him to select those which most closely resemble his own problem of concern; and lastly, have DDAS complemented by a system, or even documentation, such as the Executive Summaries [2], or a combination of multimedia documents describing a technique, showing it in action, etc. that would

cover supplementary information such as what it is, how it does it, who can do it, what is needed, how to interpret the results, etc. The approach to eliciting the knowledge and designing the overall system was based on SSM [3,4] and has been presented elsewhere [7].

The next section presents details of the implementation of the system, in terms of architecture, and technical details; building and organising the knowledge base, and features of the interaction. In section 3 an example session is given to illustrate the interaction, and section 4 presents conclusions and discussion.

2. Implementation

2.1 Architecture and technical details

The system architecture is represented in Figure 1. This shows the three main components of the DDAS: the knowledge base; the reasoning module; and the communication module.

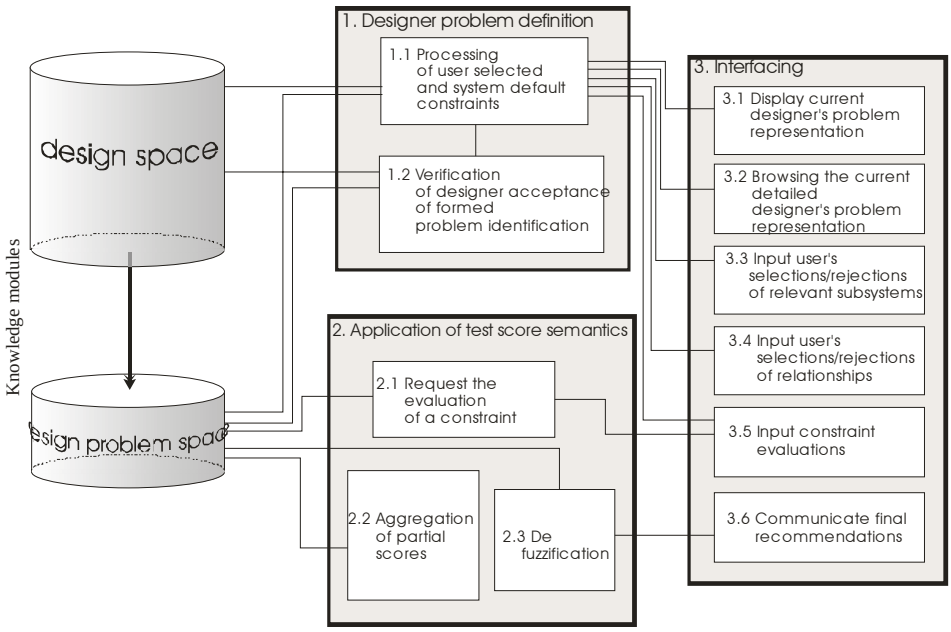

Figure 1. DDAS architecture

Implementation was carried out using CLIPS, an expert system environment developed by NASA and HARDY, a hypertext based diagram editor for Xwindows and windows 3.1. developed by AIAI of the University of Edinburgh.[9,10]

2.2 Knowledge base

The design of the content of the knowledge base, in terms of its organisation and its manipulation is the result of a methodology that goes through four phases:

1. the extraction of statements that describe the potential of the modelling techniques
2. the extraction of subproblems that the above statements (1) refer to
3. the specification of the relationships of the above subproblems (2) with the modelling techniques
4. the specification of relationships between and among the above subproblems (2)

2.2.1 Extraction of statements that define the modelling techniques

The modelling techniques were examined in turn and defining statements about each were taken from the relevant literature [1,2]. The statements were descriptions of what a technique does in relation to system usability design, as opposed to how to use an approach.

The statements that were extracted were shown to the modellers to check for interpretation and consistency in order to arrive at a final set of working statements. Examples of “what” statements are given below:

> CTA (Cognitive Task Analysis) identifies aspects of design that place heavy demands upon the user’s cognitive resources (memory, attention span, etc.)
>
> FSM (Formal Systems Methods) provides a framework for representing and understanding the compatibility between functional (system) state and perceived state.

2.2.2 extraction of subproblems that the above statements (2.2.1) refer to

Considering that for each statement of the modelling techniques, resulting from the previous phase, there is a purpose that justifies its existence, that purpose is used to extract the design subproblem that lies behind the claims of the modelling technique. That is to say, searching for the answers to the question `What problems does each modelling technique try to solve?’.

The CTA example given above implies that there exist aspects of design which are difficult for users to cope with cognitively. In metamorphosing from claims to subproblems, whole sets of subproblems are revealed. It is possible that more than

one statement of a modelling technique refers to the same subproblem. It is also possible that more than one statement from different modelling techniques refers to the same subproblem. In the DDAS, the subproblems are considered in the sense of Checkland's [3,4] purposeful activity subsystems.

These design subproblems are noted down and then compared and contrasted to see which are common in order to arrive to a sets of subproblems which eliminates redundancy.

Some examples of the subproblems are the following:

> isolate features that the user will find hardest to learn
>
> reason whether some program correctly implements a given specification
>
> predict reasonable user behaviours that the designer did not intend and did not want etc.

2.2.3 specification of the relationships of the above subproblems (2.2.2) with the modelling techniques.

Each one of the subproblems is related with one or more modelling techniques. Only one type of relationship is considered here, that which specifies how well the modelling technique satisfies the particular subproblem.

For example, a subproblem A may be well satisfied by the CTA modelling technique, somewhat less satisfied by the FSM modelling technique and not satisfied at all by other modelling techniques. This means that the particular subproblem has this `degree of satisfaction' relationship with two of the modelling techniques.

The knowledge about this relationship between the techniques and the subproblems were elicited from the modellers, who were presented with the sets of subproblems and were asked to give a degree of satisfaction of their modelling technique to each one of the subproblems within them. The modellers were given the opportunity to use either a given scale of quantifiers (a lot ... a little) or to define their own scale of quantifiers, in order to give the degree of how well their modelling technique satisfies each one of the subproblems. In the second case, when the modellers defined their own scale of quantifiers, they had to explicitly state what the scale meant. The modellers were invited to make comments on the subproblems, and especially useful were comments regarding the presentation of the subproblems e.g. as disjointed statements, placed in a hierarchy, etc. Examples were taken from case studies to illustrate the subproblem descriptions or modelling technique claims

In the spirit of Checkland, the methodology is viewed as a learning process rather than a requirements/specification process. The modellers were presented with "problems" that were often little more than paraphrases of statements about

modelling techniques. The modellers, in rating the subproblems, were asked to ignore these strong associations and to try to assess each subproblem statement as though it were free of connotations. The modellers commented upon the fact that this exercise made them aware of the scope of their techniques as well as how they might be viewed by designers.

2.2.4 specification of relationships among the above subproblems (2.2.2)

Apart from the degree-of-satisfaction type of relationship that each subproblem may have with one or more modelling techniques, the subproblems may also be related with one or more other subproblems. These relationships among the subproblems exist regardless of the modelling techniques. Some of these relationship types are generality/specificity, low possible concurrency, high possible concurrency, and are defined below.

Type generality (gen): if A and B are subproblems, members of the relationship 'A (gen) B', this means that subproblems B is a more specific subproblem than subproblem A and subproblem A is more general than B. In other words, B is a subproblem of subproblem A. This type of relationship includes the type specificity (spec) as well. Each time there is A (gen) B, there is also B (spec) A. Such relationships derive from questions asked such as:

- move towards the specific (How can you tell it's a ...?, Can you give examples of ...?)
- move towards the general (What have ... got in common?, What are ... examples of?, What distinguishes ... from..?)
- move orthogonal to the axis (What alternative examples of ... are there to ...?)

It should be noted that for the following types of relationships A is considered to be one subproblem or a conjunction of subproblems (A1 [and A2 [and ...Ai]]) and B to be one subproblem or a conjunction of subproblems (B1 [and B2 [and ... Bj]]).

Type low possible concurrency (lpc): if A and B are subproblems and also members of the relationship `A (lpc) B', then the possibility that both A and B are parts of the user's problem is very low.

Type high possible concurrency (hpc): if A and B are subproblems and also members of the relationship `A (hpc) B', then the possibility that both A and B are parts of the user's problem is very high.

2.2.5 Results of the methodology

Phase 2.2.3 provides a set of subproblems related to the modelling techniques. Phase 2.2.4 takes this set of subproblems and specifies the relationships among the

subsystems-problems that follow the above definitions and thus converts the set of subproblems to a network of related purposeful activity subsystems.

2.3 Interacting with the system

A main assumption of the DDAS is that the designer-user will express his problem according to the subproblems of this network that have their source in the modelling techniques. The following sections discuss the presentation of the subproblems to the user; how the user can be guided to express/identify his problem and what facilities are available for this; and finally how the system outputs recommendations.

The interaction with the user is based upon two types of presentation elements: the graphic display of the subproblems, and the commands that manipulate the interaction.

The subproblems are displayed in the form of labelled shapes and are laid out in a series of screens browsable by the user. Two types of shapes are used, one to represent the fact that there exist more specific problem descriptions in the knowledge base, while the other shape represents the most specific expression of a subproblem contained in the knowledge base. In the current version, the former are shown as rhombii, the latter as circles. Shapes may be linked by arcs which denote different types of relationships existing between subproblems, for example, green arcs represent “high possible concurrency” and blue arcs, “low possible concurrency”.

Commands are displayed as buttons on a toolbar which is permanently on screen. These commands aid the user to choose amongst the available facilities of the system, for example the facility of moving to diagrams/screens that correspond to different levels of analysis is performed by double arrow buttons.

2.3.1 Guiding the user to identify his problem

During the interaction the subproblems are displayed to the user, in order for him to search for and identify the subproblems descriptions that he considers as most relevant to his problem. The objective is for him to make a selection of these relevant subproblems which is a way of expressing his situation of concern. Whilst selecting (by left-clicking on the subproblems), the user can also specify the degree of relevance of the subproblems to his problem, and should he change his mind, he can unselect any subproblems he has already chosen. Each time he clicks on a subproblem, its colour changes. Each colour shows the degree of importance of the specific subproblems to the user The set of used colours are white, turquoise, yellow, magenta and red, signifying least to maximum importance respectively. This is also the sequence of the colours which appear when left clicking. After red,

(most relevant) comes white again and the user can go through this cycle as many times as he wants.

In order to guide the user through the network of subproblems, these are presented to him at various levels of detailed description. This is done by presenting them in several screens according to the subproblems' degree of generality. It is possible for the user to go backwards and forwards between screens (by using the buttons << >>).

Another feature of the system is that the user can ask for comments from the system about the set of the subproblems he has chosen (`Comments on Choices`) This facility is available any time during the interaction when selections are made. The comments that the system is able to give are based on the relationships of the subproblems that exist in the knowledge base. For instance, the user who has chosen both of the subproblems that are parts of a "low-possible-concurrency" relationship, is warned that these subproblems are not usually concurrent. The subproblems that are mentioned in the warning messages are highlighted with a black outline in order to find them more easily. The system is flexible in the sense that it allows the user to ignore the warning messages. Should the user want to follow the advice given, he may decide how he wants to solve the implications, by either selecting and unselecting accordingly. Once all the warning messages that the system has to show according to the relationships have been displayed, the system reverts to the normal interaction state where the user can choose/unchoose subproblems or choose one of the other available facilities.

A further facility available at any time is that of providing a formatted text description of the set of subproblems chosen (`Current State`). The relationships that exist in the knowledge base form the basis for the text description of the chosen subproblems. Once the text description of the set of chosen subproblems has been presented, the system reverts to state where the user can choose/unchoose subproblems or choose one of the other available facilities.

Should the user want an illustration of a particular problem description, he can obtain examples of use of the he by using the `example` button (or by shift-left-clicking on the subproblems in question).This feature can be useful in helping the user decide about how close (if at all) the specific subproblem description is to his own particular problem.

Finally, the user is also able to see instructions regarding the use of the system (`Help` button).

2.3.2 Output of the interaction: Recommending

When the designer-user feels that the subproblems he has chosen describe his problem situation adequately, he can request a recommendation from the system.

The DDAS recommends which the modelling technique(s) are suitable for his problem. This facility is available whenever no other facility is active.

In the current version of DDAS, the recommendation is a formatted text which recommends the user the most appropriate technique(s). The reasoning behind this recommendation, which is based upon fuzzy logic, is also given in the formatted text, in order to give the user the justification of the rationale behind the recommendation. The compensation oriented score operator from test score semantics [12] is used to compute the recommendation. For its computation the degrees that specify how important each chosen most specific subproblems is to the user (3.2.) and the degrees of how well the modelling techniques satisfy each chosen subproblems are used (2.3).

3. Example

To illustrate some of the system's capabilities, an example detailing a designer's specific problem and how he can handle it using DDAS follows. In this example, a designer's concern is that interface users are often confused by the outcome of clicking a button X. e.g. there can be two different results of clicking the same button X in two different contexts respectively.

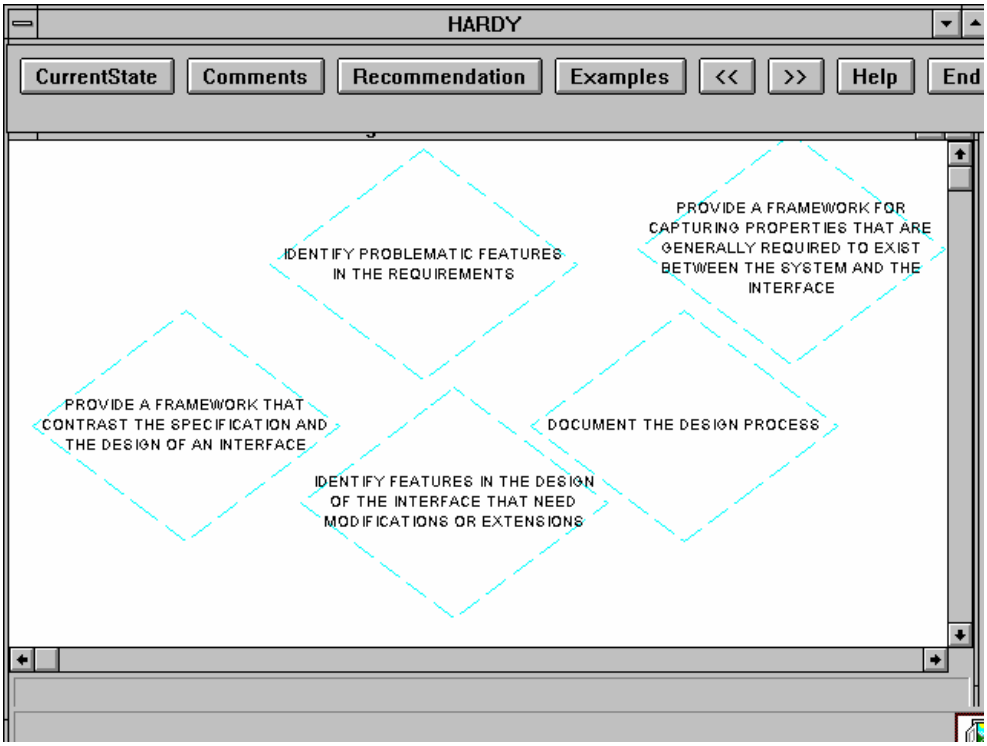

Figure 2. *A snapshot of the most general subproblems diagram*

The designer wants to resolve this problem. For the sake of the example, it is assumed that he wants the solution to enable the users of the interface to distinguish clearly what are the corresponding effects on the system when a button is pressed. It is also assumed that the designer wants to check that this problem of the design of the interface does not start from a confusion in the requirements.

The designer is firstly presented with a diagram which uses rhombii to represent the most general subproblem descriptions, such as that given in Fig 2. The designer searches through the diagram for labels which come closest to expressing his problem. In this case, he chooses the rhombii with the following labels and assigns to them a degree of relevance:

- IDENTIFY FEATURES IN THE DESIGN OF THE INTERFACE THAT NEED MODIFICATIONS OR EXTENSIONS (red)
- IDENTIFY PROBLEMATIC FEATURES IN THE REQUIREMENTS (yellow)
- PROVIDE A FRAMEWORK FOR CAPTURING PROPERTIES THAT ARE GENERALLY REQUIRED TO EXIST BETWEEN THE SYSTEM AND THE INTERFACE (magenta)

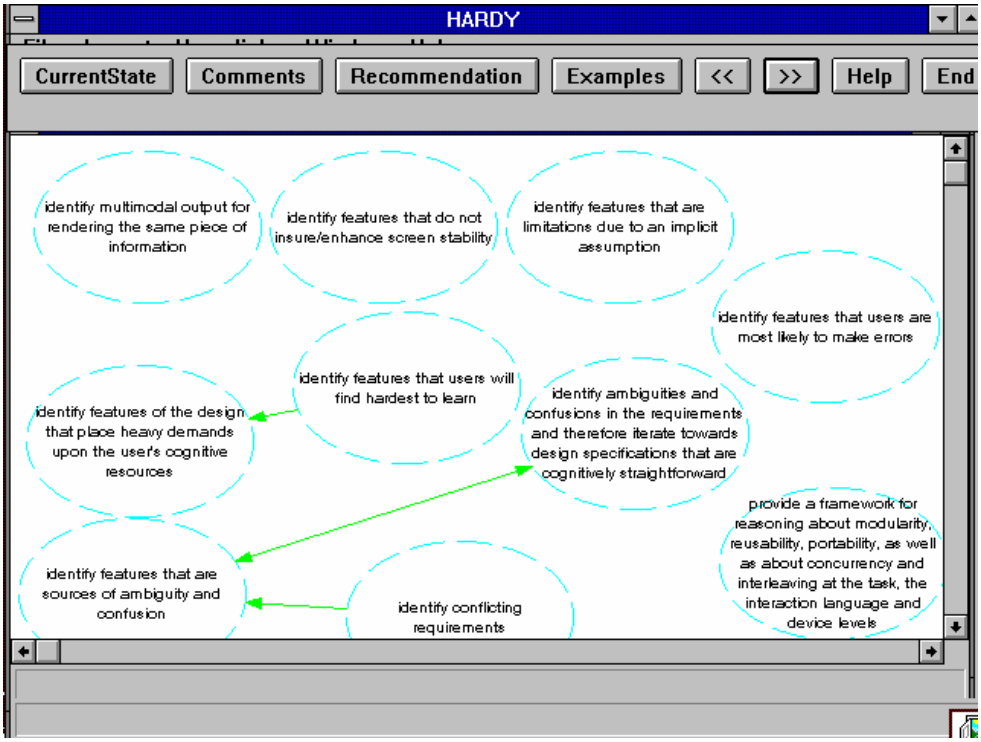

Figure 3. *A snapshot of the most specific subproblems diagram*

The designer presses the button with the label «>>« in order to move to the next diagram with the more specific subproblem descriptions. He is then presented with a diagram which uses circles and arcs to represent the possible subproblem descriptions and the relationships between them, such as that given in Fig 3. The designer searches through the network diagrams for labels which come closest to expressing his problem. In this case, he chooses the circles with the following labels and assigns to them a degree of relevance (colour).

- identify features that are sources of ambiguity and confusion (red)
- identify ambiguities and confusions in the requirements and therefore iterate towards design specifications that are cognitively straightforward (yellow)
- provide a framework for representing and understanding the compatibility between functional (system) state and perceived state (conformance) (magenta)
- provide a framework for representing and understanding the trade-off between what the representation in itself will support and what must be supported by the system (affordance) (turquoise)

- provide a framework for representing and understanding the property of predictability: supporting the system tasks by providing enough information to indicate to user what effect his new actions will have (magenta)

Before going on to choose some more subproblem descriptions from the DDAS diagram, the designer would like to have a commentary from the system about his choices. He clicks on the «COMMENTS ON CHOICES» grey button. This advice is given in a message window as shown in Fig. 4.

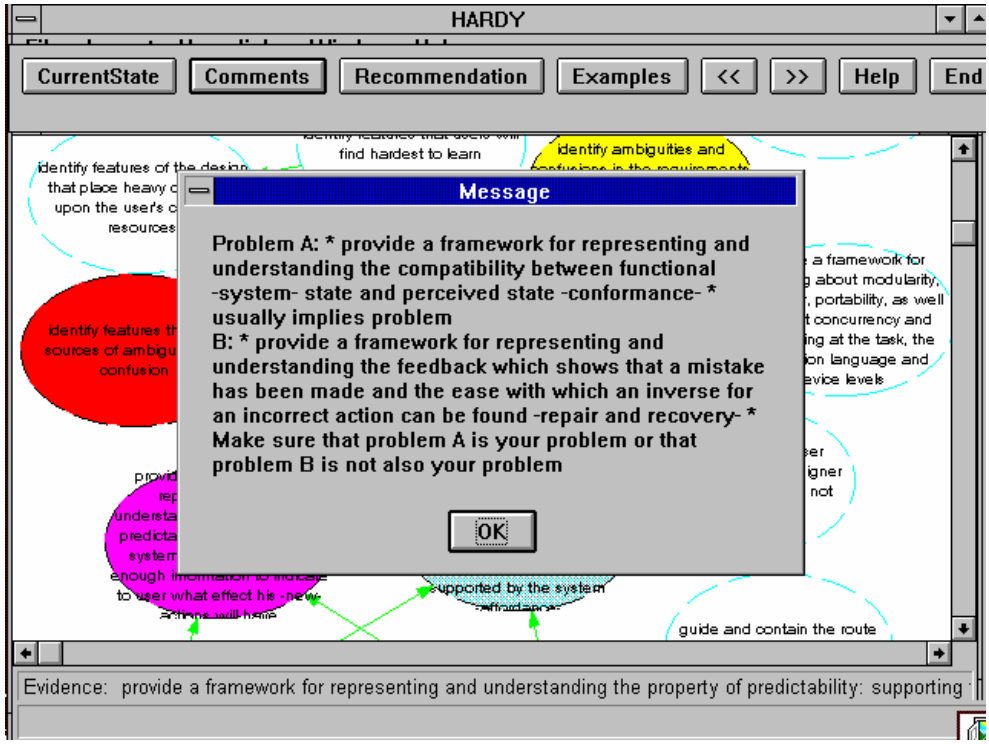

Figure 4. *Comments on Choices message window*

In this particular case the displayed message comments that according to the system, the subproblem description «*provide a framework for representing and understanding the compatibility between functional (system) state and perceived state (conformance)*» usually implies the one with the label «*provide a framework for representing and understanding the feedback which shows that a mistake has been made and the ease with which an inverse for an incorrect action can be found (repair and recovery)*» and therefore the second could also be chosen.

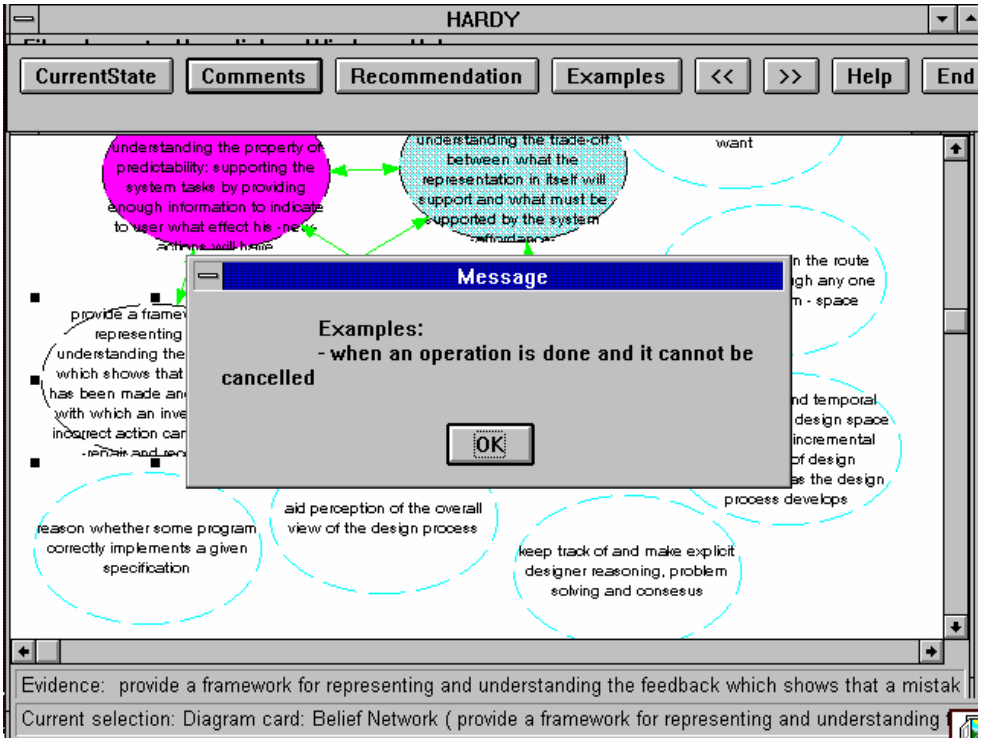

Figure 5. *Message window with the available example(s) for the chosen specific subproblem*

The designer can shift-left-click on a subproblem in order to see the available examples (if any) of the specific subproblem. The examples help him understand some characteristic situations that the subproblem should be chosen. The examples are given in a message window as shown in Fig. 5.

Each time the designer wants to see a text description of the chosen subproblem he clicks on the «CURRENT STATE» grey button. A window appears with formatted text which consists of sentences that contain either one selected subproblems description or two selected subproblems that are related with a type of relationship expressed in words. In this example, a part of the text description that the designer sees is shown on the window in Figure 6.

In this way, the system, utilising its knowledge of the design space, and subproblems associated with it, prompts the user and aids him to consider subproblem descriptions which may be relevant to his problem of concern and which he has not chosen. The user considers the system's advice and is free to reject it should he not think it relevant.

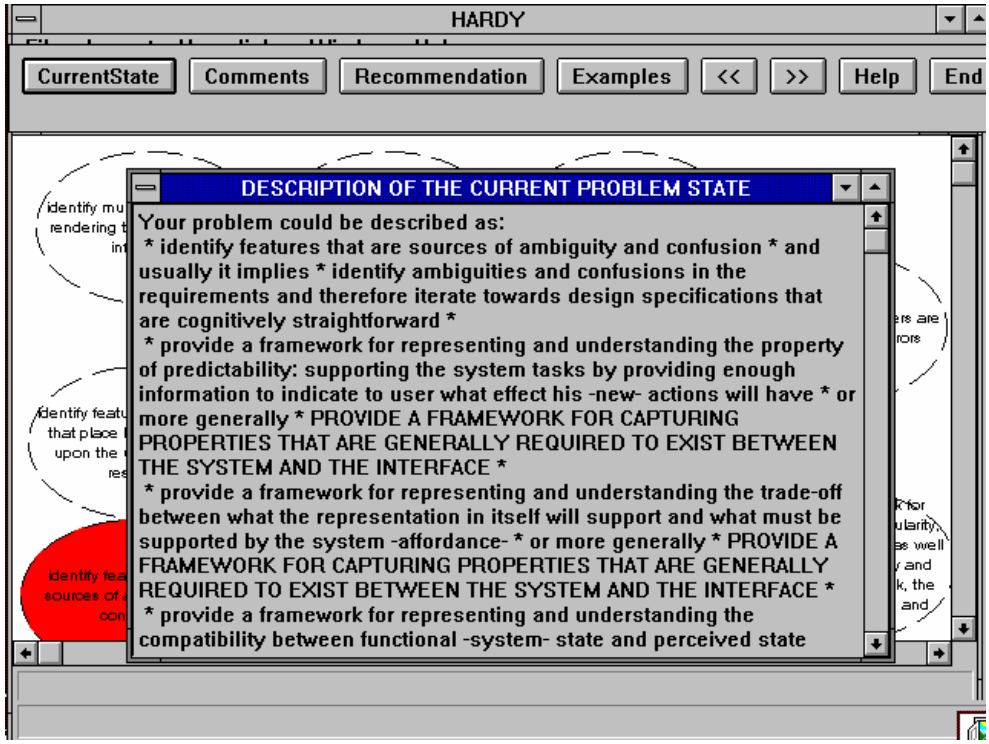

Figure 6. *Current state message window*

Otherwise, the system highlights the subproblems mentioned with a black outline (Figure 7) to help the user find the subproblems that the message refers to.

The user continues in this way, making selections, reading the comments on current choices and reselecting until he is satisfies with what the current selection represents. During this cycle he can get at any time a text description of the current state.

When the designer is satisfied that he has a final set of chosen subproblem descriptions (i.e. he doesn't want to choose any more subproblem descriptions by clicking on them and that he doesn't want to change his belief about the importance he gave to the selected subproblem descriptions, by changing their colour), he then clicks on the «Recommendation» button to get a recommendation about the most appropriate modelling technique(s) for his problem. A window appears with the recommendation. The computation representing the reasoning behind this result is also displayed in the same window for traceablity. This can be transformed to formatted text, in order to give the user the opportunity to understand and justify the system's reasoning (Figure 8).

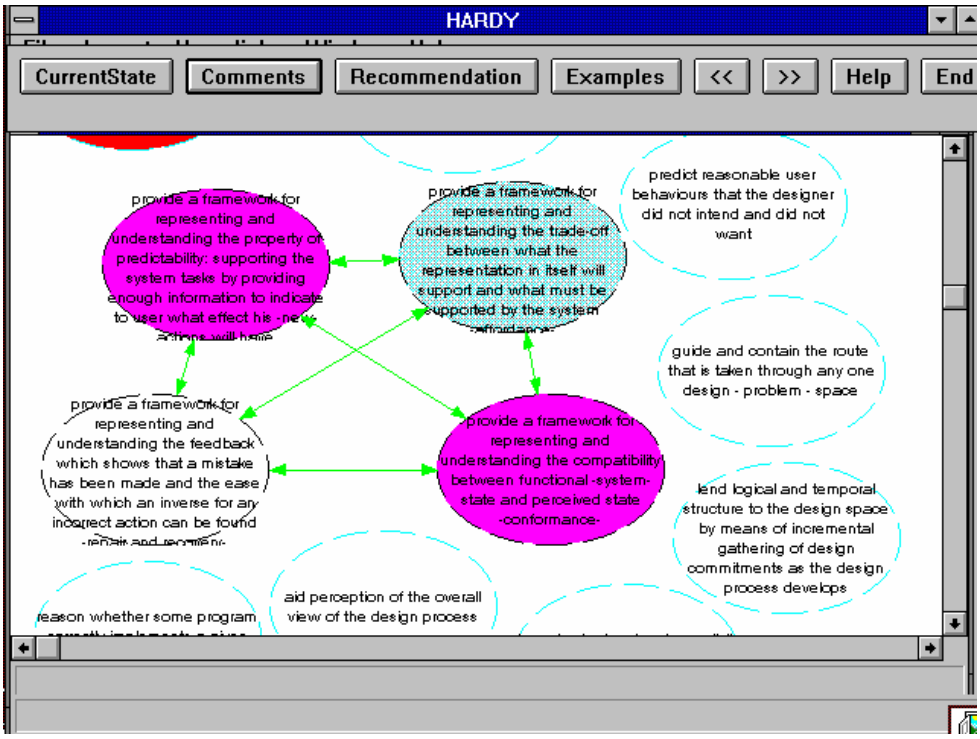

Figure 7. *Selected subproblems*

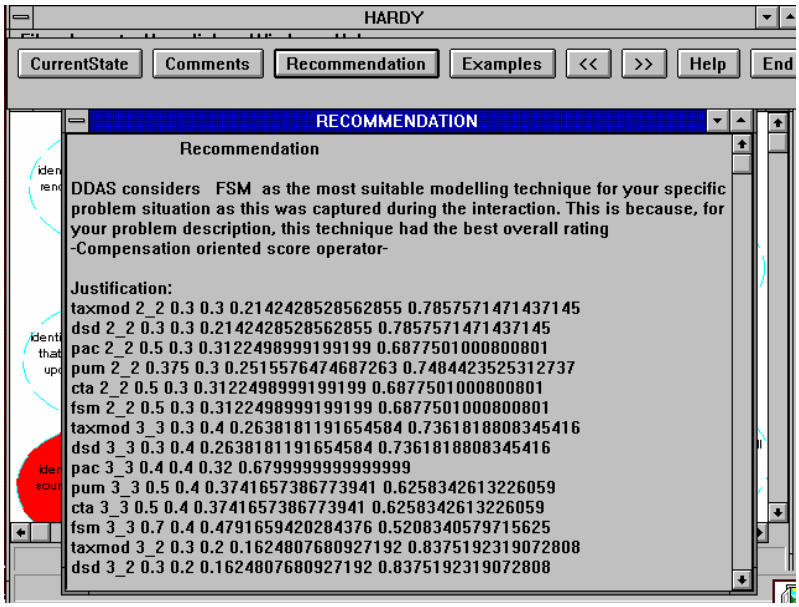

Figure 8. Recommendation message window

4. Conclusions and Discussion

Making a generally applicable catalogue of design problems, and/or even making a taxonomy of such problems, is a difficult task to undertake and it is doubtful, in view of the rapidly changing nature of technology, and users' responses to it, whether such a task could be achieved satisfactorily. The DDAS concentrates on those design space subproblems which can be aided by the modelling techniques and upon classifying these into a network according to the relationships that exist among them.

This approach does not compare and contrast the modelling techniques. Instead, the methodology looks at what design problems these techniques are capable of dealing with and works by comparing and contrasting the problems, not the techniques themselves. In this way a network of subproblems is generated. The network and the relationships within it relate to the degree with which a modelling technique can deal with a subproblem, as well as to the relationships between the subproblems themselves. When a user selects a sample set which most closely resemble his own situation of concern, the system reasons by means of the test score semantics using a number of operators, to provide a recommendation as to which modelling technique(s) are the most appropriate for the user's particular problem, and backs this up with a justification of its rationale.

The importance of the DDAS, lies not just in its usefulness to the designer, who now has access to bodies of knowledge in direct relation to a problem of concern, but in its claim to provide a methodology for decision aid in similar situations where problems exist, and tools to solve them also, but where a short cut or aid is needed to bring the two together.

5. References

[1] Amodeus, ESPRIT Basic Research Action 3066, 1989-1992, AMODEUS (Assimilating Models of Designers Users and Systems) and ESPRIT Basic Research Action 7040 AMODEUS II 1992-1995 (Assaying Means of Design Expressions for Users and Systems) Documentation available by anonymous ftp (ftp.mrc-apu.cam.ac.uk) or by www (http://www.mrc-apu.cam.ac.uk/amodeus/qref.html).

[2] Buckingham Shum S., Jørgensen A.H., Hammond N. and Aboulafia A.(Eds), Amodeus-2 HCI Modelling and Design approaches: Executive Summaries and Worked Examples , *Amodeus Project Document: TA/WP16*., 1994

[3] Checkland P.B. Systems Thinking, Systems Practice, Wiley, New York, 1981.

[4] Checkland P.B., Scholes J. Soft Systems Methodology in action, Wiley, New York, 1990.

[5] Darzentas J., Darzentas J.S., Spyrou T. Defining the Design "Decision Space": Rich Pictures and Relevant Subsystems , Amodeus Project Document: TA/WP 21, 1994.

[6] Darzentas J., Darzentas J.S., Spyrou T. Fuzzy Reasoning and Systems Thinking in a Decision Aid for Designers in Proceedings of Second European Conference on Intelligent Techniques and Soft Computing, Aachen, pp1609-1614, 1994

[7] Darzentas J., Darzentas J.S., Spyrou T. Designing a Designers' Decision Aiding System (DDAS): a Designers' Decision Aiding System, Journal of Decision Systems Hermes (in press)

[8] Darzentas J., Darzentas J.S., Spyrou T. An Architecture for Designer Decision Aiding in Brannback, M. and.Leino, T (Eds) DSS-Galore! Ebo Academy Press, Ebo Ser. A. 427, pp115-132, 1995.

[9] Giarratano J. and Riley G. Expert Systems: Principles and Programming. PWS Publishing, Boston, MA., 2nd. edition, 1994.

[10] NASA Johnson Space Center, Houston, TX, "Clips Programmer's Guide, Version 6.0, JSC-25012", June 1993.

[11] Raghav Rao H., Sridhar R., Narain S. An active intelligent decision support system - Architecture and Simulation. *Decision Support Systems*, 12, pp. 79-91, 1994

[12] Zadeh L.A. Knowledge Representation in Fuzzy Logic, *IEEE Transactions on Knowledge and Data Engng* 1 n^{o} 1, pp. 89-100, 1989.