Expert Systems: True support for the process of decision making

Edgar A. Whitley
Information System Department
London School of Economics
and Political Science
Houghton Street
London WC2A 2AE
United Kingdom

Abstract
Conventional expert systems design places undue emphasis on obtaining the **answer** in a particular problem situation whereas in practice may problem owners are more concerned with the **process** by which this answer was arrived at. By concentrating on the process of solving a particular problem, it is possible to take into consideration situational factors that are not explicitly accounted for in the knowledge base or find alternatives when the system comes up with 'undesirable' answers. The expert system, therefore effectively becomes a tool for the problem owner to use rather than a machine which generates solutions.
The paper describes this new conceptual approach to using expert systems to assist in the decision making process, showing why it is needed and how it differs from conventional expert systems design. It also describes an expert system development tool that has been created to support this process and briefly discusses a number of examples that have been developed within this framework.

Keywords: Expert systems, decision making, speech acts, explanation.

Introduction

The expert systems that are currently being developed are designed to provide their users with answers in specific problem situations. The beliefs underlying their development together with the tools that support and implement them focus almost entirely on providing the correct answer in the most efficient way. Thus considerable effort is expended on ensuring that knowledge acquisition obtains sufficient, correct knowledge for this task, the expert system development tool or shell used is validated to ensure that it will make proper use of the knowledge base and the user interface is designed to present the answer to the users in an understandable way.

This paper will argue, however, that this emphasis on **the answer** is misplaced. There are many problem situations where what is required is not the answer per se, but rather **the process** by which this answer was arrived at. By focusing on the process of arriving at the answer a far more flexible approach can be developed, one which allows the users to make use of the expert system as a tool to support them in coping with problem situations rather than as a machine that provides an answer for them. In doing so, a wider range of problem situations can be tackled and the users are more likely to accept and sustain the use of the expert system in such situations.

This shift in focus from answer to process cannot be supported with existing tools. What is required is a new approach to developing expert systems (Whitley 1990c) which provides a

theoretical basis to the design of expert system development tools. This paper begins by describing the sort of occasions which require an examination of the process of decision making. It then outlines a theoretical understanding of these occasions and then describes a practical tool that has been developed using this theory.

Occasions where the process becomes important

The need to examine the process by which an answer was arrived at becomes important when a problem situation contains situational factors that are not explicitly accounted for in the knowledge base. It is also important in those occasions where the answer arrived at is not the one 'wanted' in the problem case. Both of these instances will be described in this section.

Situational factors in social circumstances

Suchman (1987) emphasises the importance of particular features in her discussion of **situated actions**. "Purposeful actions", she argues, "are inevitably *situated actions*" which are "simply actions taken in the context of particular, concrete circumstances". Although it is often assumed that we follow plans which state the precise steps needed to achieve a particular goal, in actuality many different actions can be taken to achieve the same goal; the choice of which action depends critically on the particular circumstances that the action is taking place in and *cannot* be predicted in precise terms in advance. For example, the actions involved in actually posting a letter will depend on whether you meet someone who is going to the post office, whether you pass an internal mail basket or if you actually go to the post box yourself. "It is only when we are pressed to account for the rationality of our actions, given the biases of European culture, that we invoke the guidance of a plan". Plans made in advance of an action are necessarily vague because these situational factors *cannot* be determined in advance, whilst those plans described in retrospect "filter out precisely the particularity of detail that characterizes situated actions, in favour of those aspects of the actions that can be seen to accord with the plan".

This problem is further heightened by the observation that no person approaching a new problem situation does so 'from cold'; they are always 'thrown' in the situation. Every new situation is a continuation from a previous one (Winograd and Flores 1986). The experiences of the individuals concerned will make certain features of the problem stand out while others seem less important. Indeed, true expertise can be said to arise when an individual's experiences with a certain type of situation causes the most appropriate features to stand out (Dreyfus and Dreyfus 1986). Even those who are not at the stage of expertise will have their own perspective on the problem domain. At the simplest level, the last topic of conversation or thought will influence the perceptions formed: Compare the case of a group of car mechanics who have been discussing the noise caused by a faulty exhaust with those who have been reading a newspaper when a car with an unusual fault is presented to them. The recent experiences of the first group is likely to focus their initial attention on the noise of the engine rather than on any other factor.

When an expert system application is developed its knowledge base will contain those features that were deemed significant by the development team and the experts they consulted with. Many of the more rigorous approaches to knowledge acquisition make a special effort to try and ensure that the knowledge base includes all the significant features in the problem domain.

In some situations, especially ones based around social situations such as many business environments, there is a likelihood that this consensus about what is important in the domain will not exist. Features of a business problem that are important to someone with an accounting background may differ significantly from those considered important to someone with a background in marketing and product development. Thus the possibility arises that the knowledge base would be developed taking into consideration the views of only one such group. If the resulting system was then used by members of the other group features of the domain that had not been explicitly accounted for in the knowledge base may seem more significant. If these other viewpoints are not acknowledged then the acceptability of the system will be severely limited.

The different actors involved in developing and using expert systems can also have important implications on how certain things are 'measured' and 'labelled' in the domain. Some measurements, such as size and temperature, and labels, such as the use of the term gold for chemists, have been normalised to the extent that standards now exist for them. In other cases, however, no such standards are found. How, for example, should business performance be measured? What exactly is meant by 'responsible action' in a legal situation? There are probably as many answers as there are responses given. If the measurement and labelling used by the expert system developers differs significantly from that assumed by the users of the resulting system then the knowledge base may be useless. For many domains this problem may be quite significant and it is important to realise that even in those situations where consensus on how to measure or label something does exist this is often only the result of "an often difficult social process of negotiating agreement and arriving at a common view" (Stamper 1988).

Situational factors and the social nature of knowledge therefore suggest that in many cases the answer provided by the system may not be appropriate for the particular problem at hand. In particular, it may result from what is perceived to be an inappropriate emphasis on certain parts of the domain. In other cases, situational factors such as the backgrounds of the actors involved may lead to different interpretations of the knowledge base being formed between the users and developers of the application. In both these cases it is only by examining the process underlying the decision made that an understanding of the actions of the expert system can be formed.

Occasions when the 'wrong' answer is found

The second occasion which requires the examination of the process of arriving at an answer is when the answer provided by the expert system is not the one that is wanted.

Consider the case of a legal expert system being used by a group of lawyers to examine a particular case which they are involved in (Capper and Susskind (1988) present such a system for the British Latent Damage Act). In some cases the expert system may determine that, given the facts of the case as they have been presented, the ruling goes against that wanted by the lawyers. In this instance the one thing that can be guaranteed is that the lawyers will not simply accept the decision of expert system. Instead they will examine the process by which this answer was arrived to see which factors were the most significant for their case and which could, perhaps, be presented in such a way that an alternative ruling is reached.

This form of behaviour has also been often noted in managers who, when presenting a new business project on a spreadsheet will use rates of return, estimates of market interest rates etc.

which are more favourable to their own project than any other. What they are doing, once again, is emphasising those parts of the process of arriving at the answer which are most conducive to their own ends while playing down the effects of less useful factors. It is to be expected, therefore, that the users of an expert system would expect similar functionality. However, expert systems that are designed with the sole aim of providing the answer in a given problem situation are not suited to this task.

The role of communication in the decision making process

The importance of the **act of communication** rather than the details of what is communicated has been raised by a number of researchers (Lyytinen 1987, Winograd and Flores 1986). An example of this is the COED system described by Kaplan and Harandi (1989). The system they describe is designed to capture "the complex and dynamic decisions that go into the design of any software" since every part of the design process "must be understood in terms of how earlier parts of the design have evolved". The software supports the conversations that take place between the designers of the software and uses the conversations to record the decisions made in the design process. For example, one designer might decide to use a simple linked list to store a few data items and the communication of this decision through conversation with other programmers will be recorded by the system. When, some time later, the software is maintained, the original reasons for the particular choice as well as the code used to implement it will still be available in the system.

Winograd (1988) describes another software tool based on the notion of communication acts, The Coordinator. This is also structured around the act of communication rather than the details of what is communicated. One of the acts that is supported by the system is the issuing of a request to another person. The nature of the act (making a request) means that one of the following responses is expected: a promise to fulfil the request, a counter offer to modify the request or a rejection of the request. The software is designed to expect one of these further actions and acts accordingly. The theory of human communication used for these applications is also used in this paper and is outlined below.

Saying and doing

Many of the utterances made in the course of a conversation do not relate to true or false statements about the world, rather, as Austin (1962) points out, in many cases the uttering of words may actually be performing a **speech act**. For example, when, in certain circumstances, the words "I bet you ..." are uttered they are *not* describing what has been done, they are not true or false statements about the world, rather by saying the words the bet has been made. Similarly when requesting a particular action uttering "I request ...", in certain circumstances actually performs the request. The "certain circumstances" that affect the bet or request need to be specified in more detail and Austin hoped to discover what these "circumstances" were by "looking at and classifying types of case in which something *goes wrong*" (p. 14). He labels those things that go wrong as **infelicities** and describes three general principles which need to apply if infelicities are not to occur:

To illustrate this point, for the utterance "I bet ..." to be felicitous (appropriate or non-defective), there must be conventional procedures and circumstances that hold for making bets and the particular case must conform to these circumstances. Thus bets can only be placed before the outcome of the event being bet upon and formal bets can only be made in a Bookmakers. Some acts may only be 'appropriate' when performed by certain people, for

1. There must be some conventional procedure which has a conventional effect, in certain circumstances and that the particular persons and circumstances involved in the speech act are appropriate for the procedure being invoked.
2. That the procedure must be executed by all participants correctly and completely.
3. That the procedure is designed for certain intentions, thoughts or feelings and that those involved must have the appropriate intentions, thoughts or feelings.

Figure 1 - Austin's three principles to prevent infelicity in speech acts

example, a formal bet can only be made with the employees of the Bookmaker who are authorised to take bets, and not, for example, with the cleaners. It is also necessary for all the participants to completely and correctly finish the procedure, thus even if the words "I bet ..." are uttered, the bet will not actually be made until there is some response to it - either in the form of a betting slip, or perhaps the verbal response "OK. The bet is on". Whilst the first two conditions can affect the entire act, whether the act is valid or void, the third condition affects the intentions behind it. If I say "I bet" knowing full well that I do not intend to keep the bet if I lose, then the bet is still valid, but the person who bet against me would have justified grievances.

Speech acts and the decision making process

In the course of assisting the users in solving a problem, an expert system may **request** that certain information about the domain be provided, it may **question** the users about the domain and it may make **assertions** about the domain. These three things are speech acts and form the basis of the process of decision making. For the purposes of using expert systems to support decision making it is necessary to examine the intentions behind these acts: the felicity conditions of the speech acts undertaken by an expert system.

This description of felicity conditions (Searle uses a related notion of a speech act being non-defective) for these three acts is due to Searle (1969, ch. 3).

The conditions necessary for a request (for a future act of the hearer) not to be defective include the ability of the hearer to perform the requested act. Also the speaker must believe that the hearer can perform the requested act. It must also not be obvious to both the speaker and the hearer that the act would be performed by the hearer in the normal course of events if the request was not made. If the request is to be 'sincere', the speaker must actually *want* the hearer to perform the act and it is essential that the request must count as an attempt to make the hearer perform the act.

The act of asking a question is similar to that of requesting an act, except that in asking a question the speaker wants to know about the 'truth' of a proposition (or propositional function), as opposed to making the hearer perform an act. The question would be defective if the speaker 'knew the answer' and it must not be obvious to both parties that the hearer would provide the information about the proposition at that time without being asked. Obviously there are some questions, examination type questions, where the speaker does 'know the answer', but in these cases the speaker still wants to know if the hearer can provide the answer and it is not expected that the hearer will provide the information without the question being asked. If the question is to be sincere the speaker must want the information asked

about and the asking of the question must count as an attempt to elicit this information from the hearer.

The final speech act that needs to be examined in relation to the functionality of expert systems is the act of stating or asserting a particular proposition. For an assertion not to be defective, the speaker must have evidence or reasons for the truth of the proposition and it must not be obvious to both the speaker and the hearer that the hearer knows the proposition (or that the hearer does not need to be reminded of the proposition). A sincere statement requires that the speaker 'believes' the proposition and the assertion must count as an undertaking that the proposition represents an actual state of affairs.

Examining a decision through the use of speech acts

> If he does not make such and such a change, I'm going to ask him why. And if he can explain to me something that's logical and based on sound data, then I'll go ahead and do what he wants to do. (Benner 1984, p. 138).

This quotation describes how a nurse attempts to understand the process underlying a request made by a doctor. Although taken from a field (nursing) that has few apparent similarities with expert systems it offers a useful insight into the domain independent nature of how the use of speech acts can help describe the decision making process.

The nurse is similar to the users of an expert system in that she has an unarticulated assumed background about the problem domain which does not seem to correspond to that conveyed by the 'expert' in the field as it is exhibited in the request made. When this occurs she is prepared to accept the advice of the expert only if the expert is able to explain "something that's logical and based on sound data", i.e. something that allows her to understand why the expert's opinion is to be accepted, something indicating why the felicity conditions for requests have, in fact, been satisfied. In the case of the nurse this will be done by the doctor explaining how the answer was arrived at or by the doctor offering a different perspective on the problem. If this explanation is not satisfactory, however, the nurse will not voluntarily accept the doctor's request.

Designing the support

The PESYS system

The support for the process of decision making described in this paper was added to an expert system shell developed by the author (Whitley 1990a, 1990b). PESYS (Pascal Expert SYstem Shell) is designed to run on inexpensive, standard hardware (an IBM Personal Computer with keyboard and text screen as a minimum) and incorporates many useful features that allow the developers and users of the system to maximise the potential of this technology in decision making situations.

The system offers the developers of applications full control over the inference engine used with a particular knowledge base. It is possible to select the inference method (or combination of inference methods) that are to be used as well as whether a log is kept of the actions performed and the accuracy to be used in numerical comparisons. The shell also includes an implicit design method which encourages the proper, structured design of the knowledge base and includes the use of "sub-knowledge bases" which can be used to structure the application.

The implementation of the inference engine is very efficient, a factor in part due to the choice of programming language (PASCAL), and allows the users of the system to rapidly arrive at solutions to their problems. Their interface with the system is designed to be clear and consistent in use offering access to all the necessary functions of the system without unnecessary inconvenience. A comprehensive What-if facility is also available to allow the users to see the possible effects of changing certain data values. However, despite providing considerable functionality, the system itself is very compact (the executable file is less than 150K in size) and this leaves most of the computer's memory available for the knowledge base and working memory.

Many expert system shells have a series of goals or hypothesis associated with the rules in the knowledge base. The inference engine of the shell then tries to use the rules to arrive at one of these goals or hypothesis. However, by keeping the goals separate from the rules that they refer to, there is a strong possibility of redundancy arising, particular if a large scale application is developed. Consider the case where a new (goal) rule is added to the knowledge base. The then-clauses of this rule refer to a goal yet it is possible that the developers of the system will forget to add the associated new goal to the (separate) list of goals. If this happens the extra rule will never be used, or if it is used, will never be recognised as a goal. Similarly if a goal rule is deleted without removing the associated goal redundancy will occur and the system will try and arrive at that goal although its associated rule never exists.

PESYS overcomes this problem by associating the goal status of a then-clause directly with that clause. Any then-clauses which refer to goals are marked with the keyword **inform** and an associated **inform level** of 1. This marks the clause as being something that the users should be informed about and the level (1) tells the system that the clause is a goal. This approach avoids all the problems of redundancy that arise when the goals are kept separate from the rules they are associated with. If a new rule is added with a then-clause that is a goal, this is marked directly in the rule. Similarly if this rule is deleted from the knowledge base, the associated goal is also deleted and it is therefore impossible for the list of rule to be different from the list of goals.

The use of inform statements and inform levels is used in other ways in a PESYS knowledge base as well. During an interaction with the system, the users may find it useful to be told certain pieces of information. These are normally 'intermediate' results, possibly informing the users about the current state of the problem solving process, and whilst they are not goals they are still useful. These are implemented by attaching the keyword inform to these statements and giving them a level of 2. The keyword inform tells the expert system that the users should be informed about a particular piece of information and the level (2) tells it that this is simply a useful piece of information and once the users have seen it the system should continue to arrive at the final conclusion which is a statement marked with an inform level of 1.

The clauses of the rules in PESYS are written in English in order to aid their readability and convey their purpose in the knowledge base. PESYS also allows knowledge bases to execute certain commands in a restricted 'programming language' that deals mainly with input and output. The use of variables, both numeric and string, is also supported and comparisons and expressions can be evaluated.

Speech acts in PESYS

In order to show how tools to support this process are implemented in PESYS, the act of making an assertion, which arises when an expert system states a goal (or in the case of PESYS an inform statement), will be considered. Those features that relate specifically to requests and questions will be described later.

When an inform statement is asserted the expert system may cause the users of the system to want to examine the process underlying it. This can occur when the system asserts a result that is not expected or alternatively when the system fails to make an assertion that is expected. In the PESYS system assertions are made whenever a rule fires and one of the then-clauses is marked with an inform level of 1 (goal) or above (useful information). The asserted clause is displayed on the screen and the system waits for the users to press any key. By pressing the **SPACE** bar the users can indicate that they want to examine the process underlying the assertion.

The theory of speech acts describes general conditions that are assumed (at least on the part of the receiver of the act) to apply when an assertion is made. As was discussed above, for a non-defective assertion the speaker is expected to have evidence or reasons for the truth of the proposition being asserted. It must not be obvious to both the speaker and the hearer that the hearer knows the proposition (or does not need reminding of the proposition). Also, a sincere assertion is made if the speaker 'believes' the proposition and the assertion must count as an undertaking that the proposition represents an actual state of affairs. The notion of commitment in the communication suggests that the speaker is expected to be "willing and able to articulate why" the assertion is believed.

When the expert system makes an assertion about which the users are uncertain, there is an implied commitment on the part of the speaker to explain why the assertion is believed. The nature of computers means that this commitment to explain cannot be generated automatically. The users of the system, however, can make use of tools that allow them to determine the reasons why the assertion is believed.

```
RULE IDENTIFY-10
IF
animal is mammal
animal is carnivore
animal has tawny colour
animal has black stripes
THEN
animal is tiger
```

Figure 2 - Winston's rule for identifying a tiger (Winston 1984, p. 282)

In an expert system, the only evidence for the assertion being made is that a particular rule fired. The expert system only 'believes' in the assertion to the extent that its working memory contains facts (entered by the user - i.e. indirectly believed) which were used to fire that rule. Thus using Winston's animal recognition knowledge base (1984) the only reason for asserting

that "the animal is a tiger" is because the rule Identify-10 fired, see Figure 2. This rule only fired because of the facts entered by the user which were added to the working memory.

In this case, the 'belief' in the animal being a tiger follows directly from the facts that are known to be true and the rule that was fired. The rules in the knowledge base can therefore be seen as the means by which the experts and knowledge engineers convey their belief in certain assertions. It is possible, however, that the eventual rules used may misrepresent the intentions of the expert or fail to take into consideration certain situational factors if there have been problems in the knowledge acquisition or implementation stage of the development process. Such cases of 'unreasonable' rules leading to unjustifiable assertions would be observed and ignored by the users of the system who can then use the system to follow other reasoning paths (see below).

Implementation of the support

An assertion is made when a rule with an inform marked then-clause is fired. The rule can only fire if all the if-clauses of the rule are known to be true (i.e. are found to be true in the working memory of the system). It is therefore advisable to display the entire rule for the users of the system indicating the if-clauses that are known to be true. In general, however, some if-clauses may be known to be false and others may be unknown and all three types should be displayed. In PESYS a √ is used to signify those clauses that are known to be true, X is used to signify clauses that are known to be false and ? is used for clauses whose truth is unknown. Commands, which are "true" by default, are marked with - to show their special status. A typical rule is shown in Figure 3.

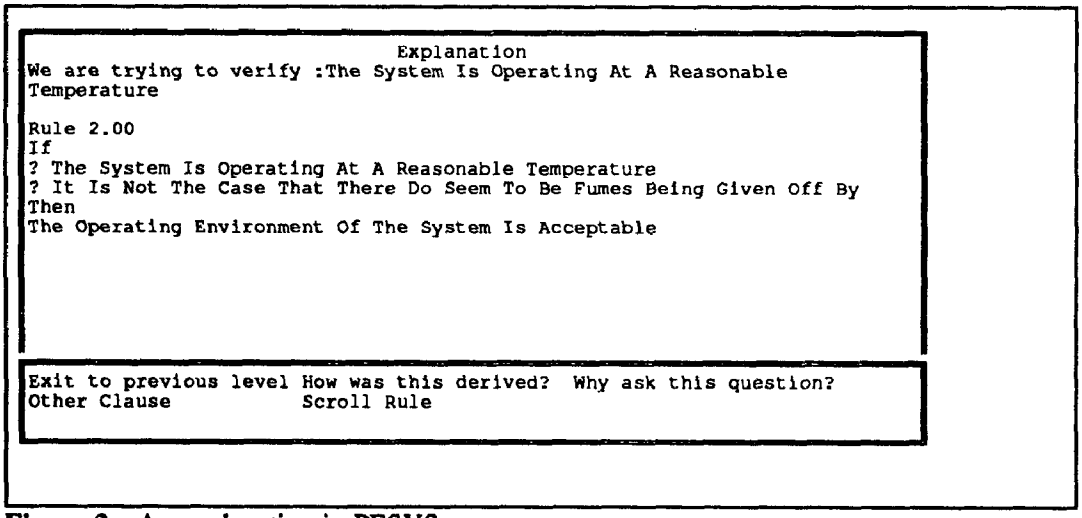

Figure 3 - An explanation in PESYS

When commands such as comparisons and variable manipulations are used it is useful to have the actual values displayed on the screen rather than simply stating that the commands were executed and the comparisons evaluated. Thus rather than just specifying that Number_of_components > 20 was true, the actual value of the number of components should also be shown. When commands such as Enter are used, it is possible to specify a large number of variables that are to be requested from the users and their actual values may be longer than one line. In such a case, the system only displays one screen line of values but allows the users, through the scroll rule option, to highlight the particular line and scroll through all the possible values. The scroll rule option is also used when the if-clauses and

then-clauses in the rule cannot all be shown on the screen at the same time and an example of such a rule is shown in Figure 4, with the same rule in a scrolled form shown in Figure 5.

```
                                Explanation
We are trying to verify :The Expressions Have Been Evaluated

Rule-4
If
- Let Result = 0 ----> 0
- Let Coefficient = .27 ----> 0
- Let Error_range = 4.45 ----> 4.45
- Form Descrip.frm
- Form Moreinfo.frm
1.00 √ The Setting Of The Device Are Reasonable
2.00 √ The Operating Environment Of The System Is Acceptable
- Enter Parameter_1 Parameter_2 Parameter_3 Parameter_4 ----> 23.00
? The Expressions Have Been Evaluated
Then
More to follow ...

Exit to previous level How was this derived?  Why ask this question?
Other Clause          Scroll Rule
```

Figure 4 - An explanation based on a long rule

```
                                Explanation
We are trying to verify :The Expressions Have Been Evaluated

- Let Error_range = 4.45 ----> 4.45
- Form Descrip.frm
- Form Moreinfo.frm
1.00 √ The Setting Of The Device Are Reasonable
2.00 √ The Operating Environment Of The System Is Acceptable
_1 Parameter_2 Parameter_3 Parameter_4 ----> 23.00 45.00 56.00 67.00
? The Expressions Have Been Evaluated
Then
Form Result1.frm
Form Result2.frm
Form Result3.frm
There Should Be No Problems Operating The Device

Exit to previous level How was this derived?  Why ask this question?
Other Clause          Scroll Rule
```

Figure 5 - The same rule after scrolling

Understanding the justification for the clauses

In some cases, simply viewing the rule that was fired will be sufficient for the users to accept the process behind the assertion made. This is particularly likely to be the case when the users have forgotten or failed to consider a possible link between various conditions and the assertion made.

In other cases, however, simply displaying the rule that fired will not be sufficient. They may accept that if certain conditions are true then a certain assertion can be made in good faith but they disagree that the conditions for making the assertion have in fact been met. The users should, in these circumstances, be able to see the process underlying the assertion of the if-clauses in the particular rule to see how they have come to be known to the system, i.e. the justification for these if-clauses needs to be made apparent.

Selecting the if-clauses

Since most rules will have more than one if-clause it is necessary to select the particular if-clause that is being examined so that the justification or rationale for why that clause is known to be true can be examined. It is possible to examine the rationale behind all the known if-clauses, but each one needs to be examined separately. On machines that have high resolution screens and mouse type pointing devices, the most appropriate method of selecting a clause to be examined would be to move a pointer to the clause and pressing a button on the mouse. On a standard IBM PC, without a mouse type input device, a more feasible method would be to number the known if-clauses (i.e. those clauses that are not commands and whose truth is true or false but not unknown). To examine a particular clause, the users simply need to enter the number of that clause. Rules with only one known if-clause will cause that clause to be automatically selected and only valid choices will be accepted by the system. The clause chosen to be examined can be determined from its number.

Once the clause has been identified, the means by which it was added to the working memory needs to be found. The inference engine in PESYS only allows clauses to be added to the working memory in two ways. They can be arrived at as a result of a rule firing or they can be entered by the users in response to a question. It would be possible to store the rule used to arrive at a clause with the clause in working memory, however this method is rather wasteful of memory, particularly since many of the clauses may not need to be examined by the users.

An alternative method would involve examining all the rules that contain the clause in its then-clauses and seeing which of these rules fired. These rules could then be used to provide the explanation. At first sight this seems to be a computationally intensive task, however it should be borne in mind that only those rules that have fired need be examined (since the clause can only be arrived at when a rule fires) and it is possible to mark those rules that have fired with a boolean flag. Moreover, in most applications the number of rules fired will only be a proportion of the total rules found in the knowledge base. If no fired rules could be found which contain the clause in their then-clauses then it follows that the clause *must* have been entered by the users.

Multiple rules that have fired

Some knowledge bases may have more than one rule which fires adding the particular clause to the working memory and in these cases the users are asked to select the rule which they wish to follow the reasoning through. Once the rule used has been selected, it can be displayed to the user in the same way as before.

In some cases, displaying the rule that asserted this fact may be sufficient for the users to accept the original assertion, in which case they can return to the previous level and continue with the inference process. In other cases, however, they may still want to examine the if-clauses of this rule further in which case the entire process is repeated again recursively.

The situational factors surrounding the use of the system cannot be determined in advance by the designers of the expert system or the developers of the application and it is therefore impossible to plan for them arising. There should therefore be no limits on the amount of browsing in the knowledge base that the users can undertake and, indeed, they should be

encouraged to perform this task until they are completely satisfied that the felicity conditions for the assertion have been satisfied.

In practice there is a slight memory limitation in the PESYS system as it stores the previous screen image as each rule explanation is displayed. This is only likely to cause a problem if a single line of reasoning is examined to a considerable depth. Differences in interpretation are likely to reveal themselves at a far earlier stage than this however and so problems are unlikely to arise in most applications.

Why-not justifications

The description that has been presented so far has concentrated on allowing the users of the system to examine how the felicity conditions associated with making an assertion are satisfied. The need to examine the felicity conditions of an assertion may also arise when a particular assertion is not made. The realisation that something 'ought' to have been asserted can only come about when the system performs a particular act; the users of the system can only realise that an assertion ought to have been made when some other assertion is made instead (or when an unexpected question is asked or an unexpected request is made - see below). The users may therefore want to know why a particular answer was not arrived at or why certain situational factors were not included.

Felicity conditions can again be used to help determine why a particular assertion has not been made. For a rule to fire and make an assertion its if-clauses need to be known to be true. One possibility, therefore, is that the responses made by the users have caused some of the if-clauses of the particular rule to be entered as false when they would need to be entered as true for the rule to fire and the assertion to be made.

The second possibility is a direct consequence of the way in which the inference engine is implemented and used since it is possible that the required rule has not fired simply because the truth of some of its if-clauses are not known; they have not been considered yet. If the users are to examine the design rationale of the system, to follow the felicity conditions of the assertions, then these cases must also be considered.

Expected assertions are not made

When the system fails to present an expected assertion the users must be able to examine how this came about. Since the expected assertion has not been made, the rule that didn't fire is not immediately available to the system and, more particularly, the system has no way of determining which assertion was expected.

From the use of inform levels in PESYS knowledge bases two paths can be followed to select the rule that would have fired to make the expected assertion. The users can choose to examine a particular goal to see the reasoning that would need to be followed for that goal to be arrived at. Alternatively the users may choose to start the examination from a particular inform 2 statement, particularly if the assertion was an intermediate statement. A list of rules which contain inform 1 then-clauses (goals) and a list of rules containing inform 2 statements are created when the knowledge base is loaded and they are used to allow the users to select the reasoning path they want to follow and the choices available are shown in Figure 6. Depending on the choice made, the appropriate list of clauses is shown and the users are asked to select the reasoning that is to be examined.

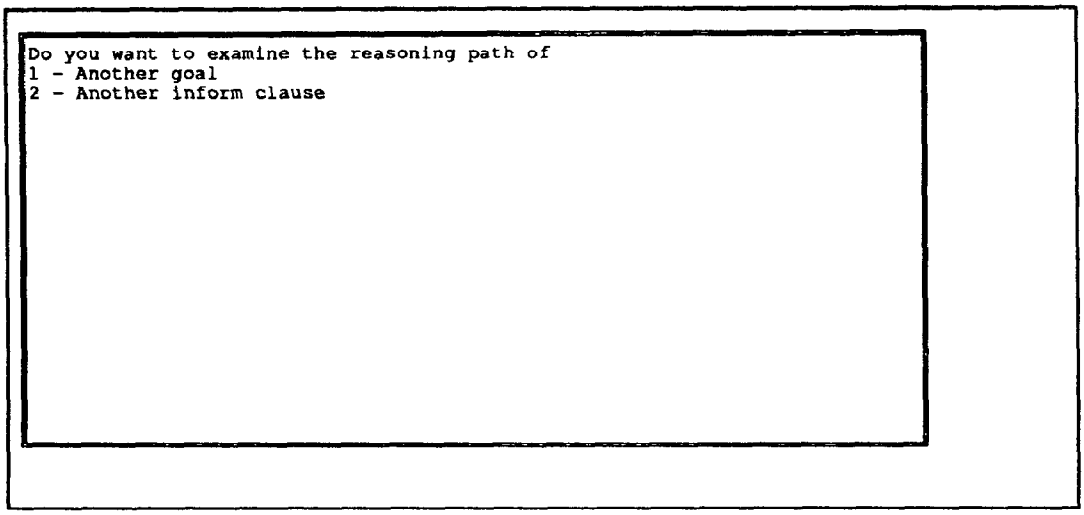

Figure 6 - The option to examine other goals or other clauses

The appropriate rule is then displayed in the same way as for assertions that have been made. Once the rule is displayed, a second problem arises, one which is particularly important for those occasions where a rule has not fired because the necessary information has simply not been obtained. Previously all the if-clauses were known and the means by which they were known could be examined. In the case of an assertion that has not been made, however, not all the if-clauses are known, or they are known to be false. Assertions that are known to be false can be tackled in the same way as those that are known to be true, but those that are unknown need extra support to deal with them. Whereas previously the system allowed the users to examine the reasoning through rules that fired, clauses that are unknown can only be examined through rules that have not fired. In PESYS this is performed by considering all the rules that could be used to assert the requested fact, not just the ones that have fired. Another consequence of this is that the choice of clauses to be examined must now include all the possible clauses rather than just those that are known to be true.

Rousset and Safar (1987) describe a system that provides negative and positive explanations in an expert system. They give no indications, however, as to why such explanations are required. The explanations generated are not particularly easy to understand either, as is shown in Figure 7.

Questions and requests

In many respects the way that the design rationale of the system is made available when questions and requests cause the users to examine the decision making process is very similar to that for assertions. Questions and requests do, however, differ in that they are made in order that a particular rule will fire rather than being a consequence of a rule having fired. As in the case for assertions however, the sincerity for the acts is simply a consequence of how the knowledge base was written and how the inference engine uses the knowledge base.

When a question or request is being performed, therefore, it is being performed to make a particular rule fire. This, again, is the extent of the sincerity behind asking the question or making the request. If the rule that is being examined is then displayed for the users it will provide information as to which clauses will be arrived at (and possibly asserted) if the question is answered appropriately or if the response to the request satisfies a certain condition.

```
Why-not bidding = diamonds?
        - possible bidding = diamonds
                - possible bidding = the longest
                - two long suits hand
                        - biddable spades
                        - biddable diamonds
                        length1 <> length2
                the longest = diamonds
        but
                bidding = the biddable major suit
                - preferring major suit
        We cannot then conclude on bidding = possible bidding
```

Figure 7 - A negative explanation from the system described by Rousset and Safar (1987)

In some cases, however, simply being told that a question is being asked to arrive at certain then-clauses is insufficient. In these cases the system must provide further information relating the clauses of the rule to other rules in the knowledge base. Since the inference engine is simply manipulating the knowledge base that was written by the knowledge base designers, the only reason for asking a question or making a request is that it will allow a rule to fire which adds then-clauses to the working memory of the system, which allows another rule to fire ... until a rule fires which adds a goal (inform 1 statement) to the working memory. This is the rationale behind performing the particular act when the inference engine is doing a backward chain and forms the basis for the explanation provided. When forward chaining is taking place no further explanation can be provided *because* no other rationale exists for the act to be performed.

The explanations for backward chaining can easily be generated from the backward chaining system. Whenever the backward chaining algorithm recursively considers a new rule, the current clause and the rule that it is found in are stored in a special stack. When the explanation of the request or question is being created, the necessary information can be found on this stack. This process can be repeated until the rule that will be fired to add a goal clause to the working memory is found and the stack is therefore empty.

As well as providing assistance with the questions and requests that are asked on some occasions it is beneficial to examine questions and requests that could have been made. The system may ask the users about topic A when they are expecting to be asked about topic B instead. In this case a why-not type explanation is required. However, the number of possible questions or requests made by the system is normally large. PESYS, therefore uses the previously described other clauses option instead and allows the users to examine other goals or other inform 2 clauses. This particular choice is made because the questions or requests are only considered to be significant in relation to the inform statements that they are used to arrive at.

How this method differs from more conventional approaches

The method of explanation described above differs from more conventional approaches in a number of respects which will now be summarised. One of the arguments that has been

presented in this paper has been that the users of the expert system may form a different understanding of the system or may emphasise different situational factors to those intended by the designers of a particular application. The users of the system may decide that this difference of interpretation requires them to examine the decision making process.

The system described above makes use of speech act theory and considers the three acts that are performed by the expert system, namely making an assertion, asking a question and making a request. It was argued that for each speech act, the users of the system expect certain felicity conditions to be met and that their problems arise when these conditions are not met. The explanation facility must therefore attempt to show the users why the conditions were in fact met so that the decision making process can be explained.

In addition to the speech acts that have been performed, problems may also arise when an expected speech act is not performed. The system therefore allows the users to examine reasoning paths that could have been followed in addition to those that were followed. Again this feature is not normally present in conventional explanation facilities.

The limited communicative resources of computer based expert systems mean that it is not possible for the expert system itself to be aware of most cases of the problems that have arisen and it is therefore necessary for the users to initiate and control the use of the explanation facility.

Rule identifiers

In the previous discussion it was implicitly assumed that each inform clause was only found in a single rule and that each known if-clause in a rule was only arrived at from one other rule. In practice, however, this is not the case and the system needs to be able to distinguish between the same clauses that were or could be arrived at using different rules.

In PESYS this is performed by using the rule identifier to allow the users to select which rule they want to examine. For example, when examining how an if-clause was arrived at there may be two rules which have (or could have) fired, adding the clause to the working memory of the system. The rule identifiers of the two rules are then presented to the users and they select the one which they want to use.

The rule identifiers that have been assumed so far are simple titles such as *rule-one* or *identify-10*. When such a list is presented to the users they will have little indication as to which rule they want to examine. They will either require a printed version of the knowledge base which they can use or they will have to examine all the rules individually. This is not an ideal situation.

Moreover, when a rule is presented to the users to enable them to examine the rationale of the system, it has been assumed that the users are able to determine the intended effect of the rule simply by examining the if-clauses and then-clauses of the rule. To some extent the use of natural language clauses in the rules simplifies this matter, but it is likely that the intended effect of many rules will not be readily conveyed simply by the clauses in that rule. A description of the intended effect of the rule, which plays no part in how the rule fires, would be sufficient to convey the intended effect.

Using rule identifiers

PESYS seeks to overcome these two problems by using the rule identifier as a description of the intended effect of the rule. The rule identifier is already displayed with the rule itself in the explanation screens and plays no part in the actual inference process yet is normally sufficient to convey the intended effect of the rule to the users, see Figure 8. The limit to the length of the rule description is 255 characters, which is the size limit for strings in the Turbo Pascal compiler used.

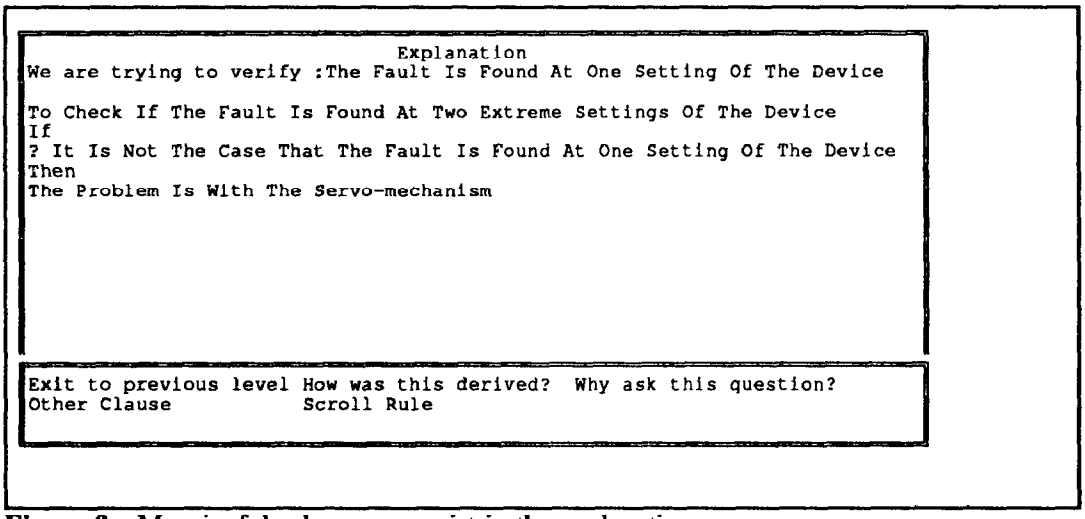

Figure 8 - Meaningful rule names assist in the explanation

The only obvious disadvantage associated with replacing the rule identifier with a free text description of the intended effect of the rule arises when the knowledge base is being debugged since there is no longer a direct way to identify particular rules. However, most editors now support sophisticated search facilities which enable particular rules to be identified using their descriptions. Moreover, if required the description of the intended effect of the rule can include the rule number as well, see Figure 9.

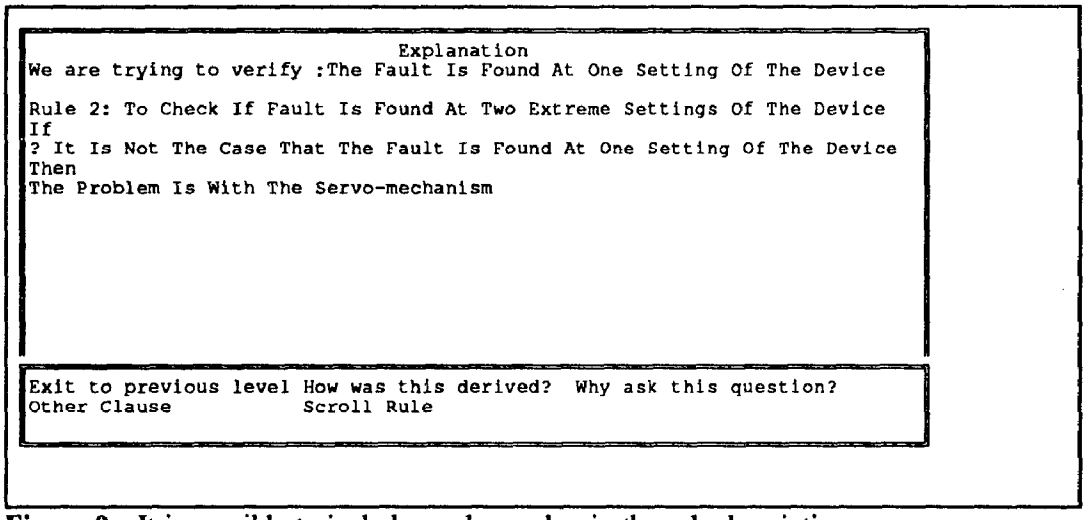

Figure 9 - It is possible to include a rule number in the rule description

Applications developed using PESYS

PESYS has been widely used as a teaching and research tool at the London School of Economics and Political Science and earlier versions of the software were used to implement an expert system to assist in filing Income Tax returns (Whitley *et al.* 1989) and to support the process of effort estimation in preliminary systems design stage of software projects (Levy 1990). It has also been used to develop numerous smaller projects.

Conclusion

This paper has described occasions of expert systems use when the process of arriving at the answer is more important than the actual answer arrived at. Conventional expert systems tools, however, are not designed with this in mind and hence provide little support for those users who wish to examine the decision making process.

An understanding of human communication, based on the theory of Speech Acts, was presented and its applicability to expert systems was shown. This theoretical basis was used to develop an expert system shell, PESYS, which aims to provide support for users in those situations where the process used to make a decision is most important.

References

Austin, J.L. "How to do things with words: The William James lectures delivered at Harvard University in 1955", London: Oxford University Press, 1962.

Benner, Patricia, "From novice to expert: Excellence and power in clinical nursing practice", Reading, MA:, Addison-Wesley, 1984.

Capper, Phillip and Richard Susskind, "Latent damage law: The expert system", London: Butterworths, 1988.

Dreyfus, Hubert L. and Stuart E. Dreyfus with Tom Athanasiou, "Mind over machine: The power of human intuition and expertise in the era of the computer" (updated, paperback edition), New York: The Free Press, 1986.

Kaplan, Simon M. and Medhi T. Harandi, "Expert assistance in conversational design tools", Proceedings of CASE '89, The 3rd Annual Workshop on CASE (July 17-21), BCS/IEEE, London, 1989.

Levy, Zeeva, "Software development effort estimation", Unpublished Ph.D Dissertation, University of London, 1990.

Lyytinen, Kalle, "Two views of information modelling", Information and Management, Volume 12, Number 1, (1987), pages 9-19.

Rousset, Marie-Christine and Brigitte Safar, "Negative and positive explanations in explanations", Applied artificial intelligence, an international journal, Volume 1, Number 1, (1987), pages 25-38.

Searle, John R., "Speech Acts: An essay in the philosophy of language", Cambridge: Cambridge University Press, 1969.

Stamper, Ronald, "Pathologies of AI: Responsible use of artificial intelligence in professional work", AI & Society, Volume 2, Number 1, (January - March 1988), pages 3-16.

Suchman, Lucy A., "Plans and situated actions: The problem of human machine communication", Cambridge: Cambridge University Press, 1987.

Whitley, Edgar A., Ashwajit Singh and Georgios I. Doukidis, "An expert system to assist in filing tax returns: The case of Indian Income Tax" in The proceedings of the fifth international expert systems conference, London, 1989, pages 115-129.

Whitley, Edgar A., "Embedding expert systems in semi-formal domains: Examining the boundaries of the knowledge base", Unpublished Ph.D. Dissertation, University of London, 1990a.

Whitley, Edgar A., "The PESYS User Guide", Working Paper and Software Package, Information Systems Department, London School of Economics and Political Science, 1990b.

Whitley, Edgar A., "Two approaches to developing expert systems", Working Paper, Information Systems Department, London School of Economics and Political Science, 1990c.

Winograd, Terry and Fernando Flores, "Understanding computers and cognition: A new foundation for design", Reading, MA: Addison-Wesley, 1986.

Winograd, Terry, "Where the action is", Byte, Volume 13, Number 13, (December 1986), pages 256A-258.

Winston, Patrick H., "Artificial Intelligence" (second edition), Reading, MA: Addison-Wesley, 1984