# Cooperative Hybrid Systems[1]

Matthias Gutknecht (gutkn@ifi.unizh.ch)
Rolf Pfeifer* (rolf@arti.vub.ac.be)
Markus Stolze (stolze@ifi.unizh.ch)
AI Lab, Institute for Informatics, University of Zurich,
Winterthurerstr. 190, CH-8057 Zurich, Switzerland
♦currently: AI Lab, Free University of Brussels, Belgium

## Abstract

Recently there has been much criticism in the AI community that knowledge based systems are not situated. We argue that trying to provide for situatedness in a conventional system will lead to the so called model explosion cycle and that for most application environments adaptivity is needed for situated behavior. Cooperative systems are a solution to the model explosion cycle where adaptivity is delegated to the user. A hybrid symbolic/connectionist system offers self tuning capabilities (and therefore adaptivity) but can't cope with the model explosion cycle. Integrating both approaches into a cooperative hybrid system leads to a much more situated behavior than conventional systems can achieve. Our approach is illustrated using a real-life expert system in the domain of technical troubleshooting. Ongoing practical tests indicate that a cooperative hybrid design presents an attractive architecture for knowledge based systems.

## 1. Introduction

In the last few years the terms *situated cognition* and *situatedness* have been brought into the discussion about knowledge based systems (KBS). It is argued that conventional KBS are not situated in several ways. Winograd and Flores [1986] state that these systems (in contrast to humans) lack the ability to cope with "breakdowns", that is situations where the usual "habitual" behavior is interrupted by an impasse. In these situations something which is otherwise "ready-to-hand" - unproblematic, transparent - establishes itself as a problem which has to be dealt with in a novel way (for example the mechanical properties of a jammed screw). Suchman [1987] demonstrates how an expert system for a copier fails to support the users because it does not have access to all the situational factors, and because its models about user intentions and behavior are much too limited. While access to situational factors might be improved by furnishing the system with more sensory data, the limitations of the KBS by its models are more principled: humans use models and plans just to orient themselves in a situation and explain their past behavior; at the most basic level, in its atomic actions, behavior is situational and not model or plan driven. Models of human behavior therefore are always bound to be incomplete and error-prone. Clancey [1989a] argues that "the idea that human-like intelligent behavior could be generated by interpreting stored programs that predescribe the world and ways of behaving must be abandoned, for this view confounds descriptions an observer might make with physical mechanisms inside the agent". He concludes that the designer of a KBS should move from engineering "knowledge structures" in an agent to designing a state-sensory coupling of a system to its environment [Clancey, 1989b].

At first sight these issues may seem fairly theoretical. Nevertheless we were forced to deal with each of them when building the DDT (Device Diagnostic Tool) KBS for technical troubleshooting of a liquid handling robot[2]. A conventional approach doesn't work because it is based on the implicit assumption that the system can be designed and built by the knowledge engineer, who tries to anticipate all the potentially relevant situations. The main problem with this approach is that such anticipation is simply not possible. For example there are costs associated with tests or observations the user (technician) of the KBS has to make. Looking whether the power cord is plugged in correctly is associated with much lower costs than testing the proper connections of an internal circuit board of the robot. But these costs can vary greatly from situation to situation. If the cover of the robot has already been removed by the technician in the process of trying to find the faulty component, the costs of the second test can be very low.

This research was sponsored by Tecan AG, Hombrechtikon, and by the Swift AI Chair of the Free University of Brussels.

This liquid handling robot is a complex, programmable and configurable device that is controlled by a PC. Failures do not only include mechanical, electrical or electronic deficiencies, but as well failures that occur due to a false setup of programs, or use of false liquids or reagents.

However, if the cover has to be removed first to make the observation, costs will be higher. Furthermore the costs of the observation greatly depend on the skills of the user. Therefore they cannot be assigned globally once and for all.
In a traditional system this problem might be approached by developing a user model and a spatial model of the device. But even these detailed models do not allow the anticipation of situations like the following: the user does not have the appropriate tools available at the moment (e.g. he does not have a screw driver), he cannot remove a screw because it is jammed (and thus he cannot remove the cover), he cannot execute a basically easy test (e.g. the Knowledge Engineer has not anticipated that to test, if "contacts are ok", many factors may have to be considered, for example corrosions, oil on the contact points, the solder points of the contacts aren't ok, etc.). It is important to note that the additional non-anticipated complications caused by a specific situation are not irrelevant On the contrary, they can be crucial to the whole task. For example, if the user cannot determine whether the contacts are ok, an alternative and potentially much more costly solution path may have to be followed. Generally speaking, however detailed the models, there will always be additional *relevant* factors which should be included: a phenomenon which we called the *model explosion cycle.*

Moreover, there will always be a tradeoff between the benefits of a more detailed model and the computational and interactional costs of applying the model to a particular situation. In other words, even if the model is more detailed - it includes potentially very many situations - much effort will have to be expended in actually applying the model: the user will have to be asked many questions which might not make much sense to him in this situation, and there will be high computational costs associated with maintaining the model internally. Sometimes things get even worse when the user requires additional expertise to answer these questions. In summary, every knowledge base in a sufficiently complex real world domain will always by necessity be incomplete.

But in most application environments it is not sufficient to escape the model explosion cycle in considering all the relevant factors because relevancy itself is changing due to changes in the environment. For example in the domain of technical troubleshooting the following changes may occur: the devices to be diagnosed and repaired are redesigned; there are different configurations of the devices; the operators of these devices show different behavior (e.g. some are more careful than others), which leads to different types of failures; the users of the expert system have different skills, etc. Thus, these KBS have to be adapted continually to maintain a situated behavior. By *adaptation* we understand the proper coupling of a system to its environment including the user (this matches Clancey's [1989b] request). If a system is able to adapt itself, it is called adaptive, that is, it has an inherent *adaptivity.*

Hence, situatedness of a KBS means, that it has access to all the relevant information of a situation and is able to react appropriately in novel situations due to experience with similar situations. Learning and performance cannot be separated in a situated system.

## 2. Our own approach

Our approach to build a situated KBS resulted in a cooperative hybrid system. We will briefly introduce the notion of cooperative systems on the one hand and hybrid systems on the other and then motivate our approach. In *cooperative* systems several actors contribute to a problem solution. Each of this actors is capable of sophisticated problem solving on its own, but the solution to the complete problem cannot be achieved without cooperation. All the tasks to be performed to solve a specific problem are allocated to the individual agents in a cooperative system. These agents are either computer systems or humans, that is users. In this paper the focus will be on joint human-computer systems.

Most cooperative systems will be *hybrid* in the sense that they contain components of different types. The term hybrid has been used for systems that include analog and digital components, for KBS that include rules, frames, and constraints, or for systems including a traditional KBS and a neural network part. We will focus on the last. More concretely, a system will be described in which the neural network learns some tasks from the user, which require situation specific knowledge. In other words, in this system the neural network picks up parts of the task that were originally entirely delegated to the user. In the past few years many hybrid systems have been proposed. A number of them are briefly reviewed in Gutknecht and Pfeifer [1990].

In the introduction we mentioned that KBS, in order to be situated, should include all the relevant situational knowledge without running into the model explosion cycle and provide for adaptivity. To escape the model explosion cycle the system must be cooperative. Only the user of the system has access to the relevant information since he is present in the real-world problem solving situation [Winograd and Flores, 1986; Suchman, 1987]. To take our example of the jammed screw, it is unnecessary to have a complete model of the fact that potentially all screws might be jammed and what should be done if this is the case. This is only considered if it actually arises. For this purpose the user is needed. Although much of the knowledge needed is related to common sense - which is assumed to be present in all users - specific skills are also required. If the user has dealt with jammed screws in other situations he will be better able to cope with it. So, there is a high dependence of the joint human-computer system on the level of experience of the user. Less skilled users should be able to benefit from the expertise of more skilled ones, but as we showed, this cannot be achieved by standard knowledge acquisition techniques only .

Now, how can we provide less skilled users with access to the expertise of more skilled ones without running into the model explosion cycle again? Our solution is to have the system pick up the control strategies of experienced users, since control - choosing the optimal thing to do next - is at the very heart of every kind of expertise. To do this, a list of hypotheses to pursue and tests that might be performed are presented on the screen from which the - preferably experienced - user can choose. These selections arc recorded and used for training a neural network. This way the network eventually picks up the strategy of the user. We use neural networks for convenience, because of their power to build correlations. This inclusion of a neural network leads to a hybrid system.

Another possibility would be to simply ask the user why he is preferring a particular alternative to another. This would represent an efficient knowledge acquisition procedure which is, in fact, applied in quite a number of projects that we are aware of. However, this approach can lead to the problem of the user getting annoyed by too many questions, and the difficulty of making implicit knowledge explicit. As is known from the knowledge acquisition literature, such explanations are prone to be "post hoc rationalizations" [Nisbett&Wilson, 77]. By simply learning the choice of the user both of these problems can be avoided. In our approach the recorded selections may show that the user chooses different alternatives in the same situation. In this case, we can look for the hidden piece of information the user is apparently implicitly using. The knowledge engineer may then decide whether the discriminating knowledge should be introduced in the knowledge base or not. This provides us with an incremental knowledge acquisition procedure: we are using information about cases which have actually occurred, but we are not misled by potentially irrelevant cases and by rationalizations.

In adapting to the (adaptive) behavior of skilled users the system also implicitely tunes itself to changes in the environment. If due to some consistent error, (e.g. the cover screws being jammed frequently), there is a design change, (e.g. a different way of fixing the cover), the errors due to jammed screws will no longer occur, the users will change their behavior and the system will adapt to the changed design without direct intervention of a knowledge engineer.

## 3. System description

An overview of the system architecture of DDT (Device Diagnostic Tool) is shown in figure 1. We will describe it in two parts. In the first part each *component* of the architecture will be introduced and in the second part the *tasks* relating the components to each other will be explained.

### 3.1 Components

In figure 1 there are two types of components: *humans* (head-symbols) and *artifacts* (boxes). The human components of the system are the user, the expert and the knowledge engineer. The *user* is the person who selects the best failure hypothesis from the interface, chooses and executes tests that discriminate between competing (failure) hypotheses and updates the list of current findings (see below). He is also the component most sensitive to the environment. The *expert* is the person responsible for the adaptation of the knowledge base (i.e. maintenance and extension). He sometimes assumes the role of the user. In this role he "produces" high quality cases for the case base (see below) to teach the hybrid system. The *knowledge engineer* is the person who designs the system. This includes the assignment of tasks to humans and artifacts. The knowledge engineer is in fact both, part of the system itself (where maintenance is concerned), and outside of it (as its designer)[Clancey, 1989a&b]. Her responsibilities include the knowledge acquisition up to the point where the system is maintainable by the expert.

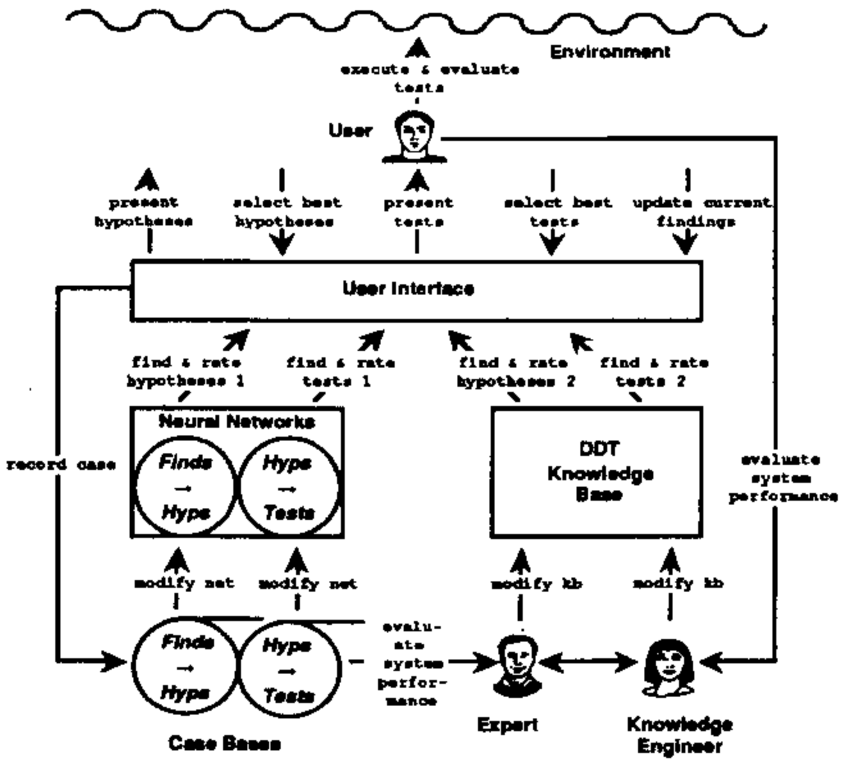

Figure 1: Architecture and embedding of the cooperative hybrid system. Boxes represent components, arrows tasks relating the components to each other. (See text for further details.)

The artifacts in the system are the knowledge base, the neural networks, the case bases, and the user interface. The *knowledge base (KB)* stores the knowledge in a structured form and makes it available to the inference mechanisms and - for maintenance purposes - to the expert and the knowledge engineer. The diagnostic knowledge is divided into *failures, observations,* and relations between them. The *neural networks (NN)* are three layer feedforward nets that learn the focusing interactions from skilled users and are able to give hints to less skilled ones due to this knowledge. There are two NNs in the system. One NN is trained on the users' selection of hypotheses, given the set of current findings, the other is trained on the users' selections of tests (or new observations), given the set of hypotheses which are currently in his focus. These selections which reflect the users' focusing actions are recorded in the *case bases.*

The *user interface* is supporting the user in solving his tasks by displaying relevant information. This information is extracted from the KB and the NNs. But the user interface is also responsible for recording the user interactions (selections of hypotheses/observations/tests and updates of findings) in the appropriate case base. The user interface consists of windows and buttons to manipulate the information shown in the windows. One window is the observation window, where general symptoms or tests relevant to the current situation are displayed. In another window, the hypothesis window, failure hypotheses are displayed. In front of each hypothesis there is an evaluation part with the KB rating for this hypothesis and the NN rating.

## 3.2 Tasks

In figure 1 the basic tasks are shown as arrows. The system has to perform two main types of tasks, namely consultation and modification tasks. Three task loops can be identified: a consultation loop and two modification loops. In the following, the loop of consultation tasks is described as it is performed after the addition of some new findings to the set of current findings. The loop starts, when the user enters a set of initial findings. First the *find&rate-hypotheses* task will be performed. The result of this task is a set of rated hypotheses related to the set of current findings. The task can be performed by two problem solving methods. Either the relations between findings and hypotheses in the KB are evaluated, or the NN which maps findings to hypotheses is consulted. In both cases the result is a rated set of hypotheses. (In contrast to the suggestions of the NN, which depend on previous cases and are therefore not complete, the KB always finds all valid hypotheses due to its observation-failure-relations.) Then the *present-hypotheses* task filters and groups the set of rated hypotheses before presenting them to the user (filtering means either showing the highest rated KB proposals or those NN suggestions, which are valid and have a high rating). Knowledge about groups of hypotheses is used for this task. The *select-best-hypotheses* task is performed by the user. He selects on the display the hypotheses he wants to focus upon, or marks the irrelevant hypotheses. This may involve changes of the hypotheses display (e.g. switching between KB and NN hypotheses). After the user has chosen a set of interesting hypotheses, this set is used by *the find&rate-tests* task to find a set of rated tests. Again, there are two problem solving methods for this task. Either the relations between hypotheses and tests in the KB are evaluated, or the NN which maps hypotheses to tests is consulted (*cf.find-and-rate-hypotheses).* As for the hypotheses, a *present-tests* task groups the set of rated tests, before they are presented to the user (cf. *present-hypotheses).* Knowledge about groups of tests is used for this task. Then the user performs the *select-best-actions* task by selecting appropriate repairs or tests to discriminate between the interesting hypotheses. He executes the selected repairs or tests and evaluates the results *(execute&evaluate-actions* task). Afterwards he updates the set of findings on the interface accordingly *(update-current-findings* task). The consultation ends when all symptoms have been removed.

Modification tasks are performed in two loops: an automatic loop which tunes the system to the user and the environment, and a manual one that is performed by the knowledge engineer and the expert who adapt the KB based on their evaluation of the system performance. In the automatic loop the *record-case* task of the interface keeps track of the user's selections of tests or hypotheses and records them in two case bases. During the *modify-network* task the case bases are used to train the appropriate NN with the backpropagation learning algorithm [Rumelhart *el al.,* 1986]. Because this task requires a lot of time and system resources, it will typically be performed overnight when the KBS is not being used. The manual loop starts with the evaluation of observations about unsuccessful consultations by the expert and the knowledge engineer *(evaluate-system-performance* task). These observations can either be made directly by users of the system or indirectly by analyzing the cases recorded in the case bases. Based on this analysis they modify the KB *{modify-knowledge-base* task).

## 3.3 A sample session: adaptation and knowledge acquisition

We will show how the self tuning mechanism of the system leads to an adaptive behavior, and we will point out how specific characteristics of the system performance can be evaluated and used for knowledge acquisition.

The case presented here is a reproduction of a session of one of our experts with the system. This session was recorded in the case bases together with about four others of the same day. The case base with the data from the hypotheses selection task has been used afterwards to train the NN, which learns to focus on hypotheses. After 200 training passes through this case base, we replayed one of the five sessions to assess the quality of the now self tuned system. The results of the first five steps of this session are shown in table 1. In the leftmost column the step number together with the newly added finding is shown. In the next three columns, the hypotheses focused on by the expert and the corresponding suggestions of the NN and the KB are shown. The NN and the KB proposals appear with their associated ratings. For the NN this figure is the scaled activation level of the output node representing this proposal. Only proposals with a rating above 80 are shown. For the KB this rating is computed from the number of findings related to this hypothesis and an apriori assessment of its frequency which is also stored in the KB. Only the proposals with the highest ratings are shown.

| New findings | Hypotheses focused on by the expert | Net proposals with rating > 80 | Best KB proposals |
|---|---|---|---|
| 1. ERROR PLUNGER OVERLOAD | DISPENSE SPEED TOO FAST<br>TIP CLOGGED | DISPENSE SPEED TOO FAST N95<br>TIP CLOGGED N91<br>PIPETTNG TUBING CLOGGED N98<br>DILUTER OLD EPROM VERSION N96<br>TIP CLOGGED N91<br>T-BLOCK CLOGGED N83<br>VALVE CLOGGED N83<br>DRIVE PLUGGED IN REVERSE N83<br>QUAG ADJUSTMENT INCORRECT N82<br>SYRINGE VOLUME TOO SMALL N82 | TIP CLOGGED W12 |
| 2. SOFTWARE/PROGRAM MANIPULATED | DISPENSE SPEED TOO FAST<br>SYRINGE VOLUME TOO SMALL<br>DILUTER EPROM OLD VERSION<br>TIP CLOGGED<br>TUBING CLOGGED/BENT | SYRINGE VOLUME TOO SMALL N99<br>DILUTER EPROM OLD VERSION N94<br>TIP CLOGGED N96<br>TUBING CLOGGED/BENT N91 | TIP CLOGGED W12 |
| 3. FAILURE PERSISTANT | DISPENSE SPEED TOO FAST<br>TIP CLOGGED | DISPENSE SPEED TOO FAST N96<br>TIP CLOGGED N88 | DISPENSE SPEED TOO FAST W12<br>TIP CLOGGED W12 |
| 4. SYSTEM LIQUID HIGH VISCOSITY | DUST ON DECODER WHEEL | DUST ON DECODER WHEEL N85 | DISPENSE SPEED TOO FAST W14 |
| 5. SYRINGE EXCHANGED | DILUTER DUST ON DECODER WHEEL<br>DISPENSE SPEED TOO FAST<br>SETUP PORT INCORRECT | DUST ON DECODER WHEEL N99<br>DISPENSE SPEED TOO FAST N81<br>SETUP PORT INCORRECT N84 | DISPENSE SPEED TOO FAST W14 |

**Table 1:** The first five steps of a session after the NN has been trained on several sessions. The leftmost column shows the number of the step together with the new finding. The next columns show the hypotheses the expert is focusing on, the suggestions of the NN together with their rating, and the top rated suggestions of the KB with their rating.

In *the first step* the NN suggests a focus which is too broad. Beside the two hypotheses the expert has focused on, the NN suggests seven more hypotheses. This can be either due to a poor generalization on the part of the NN or to inconsistencies in the case base stemming from inconsistent behavior on the part of the users. Inconsistent behavior can have two reasons: either the users were selecting different hypotheses/tests in the same situation due to their individual preferences and experiences or some relevant piece of discriminating information about these situations is missing in the system. In the first case, the cause of the individual preferences and experiences of different users could be found in peculiarities of their local environment (e.g. in some countries failures of a device due to insufficient maintenance are more frequent than in others) which would suggest that alternatively tuned system components (KB and/or NN) should be used. The second case is interesting, because it points out deficiencies of the KB itself. The KB must be modified to include some new hypotheses or observations (incremental knowledge acquisition). In our case the expert made a trial session with the system where he used the same initial finding, but chose a very broad focus. This session had been recorded into the case base too and was the reason that the system had been trained with this broad focus. The KB proposes only one of the two hypotheses of the expert. The other got a lower rating than the proposed one and is therefore not shown.

In the *second step* the expert includes three more hypotheses into his focus. The NN suggests the same hypotheses as the expert except one ("dispense speed too fast", which in fact had been proposed, but with a rating below 80). It is interesting to note that the expert's hypotheses had already been proposed by the NN in step 1 with a significantly higher activity level than the rest. This can be explained as follows: when a hypothesis is part of the focus over several selection steps, it will be trained several times together with the initial findings. This means that strong connections will be built up between this finding and the hypothesis (reflecting the high correlation between them). When this finding is present in a consultation using the trained NN, the strongly correlated hypotheses will get a lot of activation from it leading to a high rating. The top rated suggestion of the KB was also part of the expert's focus.

In the *third step* the expert made a test which discriminated between the hypotheses in his focus. The NN and the KB both make a correct suggestion.

In the *fourth step* the observation that the liquid had a high viscosity made the expert eliminate both hypotheses remaining in his focus and switch to a totally new hypothesis. This focusing switch had also been learnt correctly by the NN whereas the top rated suggestion of the KB was not useful in this case.

In the *fifth step* the expert again broadened his focus and even reconsidered a hypothesis he had focused on before. The NN suggested all three hypotheses correctly. As before, the highest rated NN suggestion was considered the most likely one by the expert. The KB suggestion was part of the expert's focus, too.

## 4. Discussion and further work

Initially we stated, that if we want to build a situated KBS, it should meet the following requirements: it should include relevant situational factors without running into the model explosion cycle, and it should be adaptive to its environment We argued, that traditional KBS technology cannot appropriately fulfill these requirements, while the proposed approach based on cooperativity and tunability can.

*Cooperativity:* Cooperativity was shown to be a way to cope with the problem of the model explosion cycle. In our system cooperativity has been realized through a task distribution, which was developed in close collaboration with a domain expert, who has worked with us on the development of the system for more than a year. The NN part was only introduced recently (see below). It seems that the experts feel comfortable with this cooperative approach. One of our experts is a strong supporter of this technology in the company and sees a great potential for further applications. A preliminary evaluation was so successful, that the system has been introduced this spring into a productive environment. But extensive and systematic evaluations of experiences in various productive environments will have to be conducted to make a final assessment.

*Tunability* : The need for having an incrementally tunable system is given by a changing environment which includes the skill levels of different users. Our incrementally trained NN's can track the changes in the environment and represent a means for making skills of experienced experts available to less skilled ones. They function as a kind of associative case-base of previous user-system interactions. This setting allows the system to adjust itself to a certain extent to its environment Since this was the most recent addition, there are no extensive tests available yet However, preliminary evaluations by one of the experts have been very positive. He clearly stated that the selection tasks present real problems for less skilled users and that having useful hints on an appropriate subset of hypotheses or tests for focusing would be of great advantage. He was also enthusiastic about the idea of tuning systems to different countries (the company distributes these robots world-wide) and to different user groups (e.g. to sales personnel, technicians, or operators). Of course one has to be careful to train the NN with experienced users only.

Independent statistical tests (with simulated cases) were performed to assess the capabilities of the NN. They clearly show the NN's capabilities to generalize to cases they were not trained on: The NN which learns the hypotheses selection task had been trained overnight on a case base of 250 cases[1]. After 277 epochs of training with the standard backpropagation algorithm the NN could reproduce 239 (95%) cases of the training set correctly. Tested on a set of 250 new cases it was able to solve 160 (64%). This is a good result, considering the fact that in our application the interface filters out all NN suggestions that are not part of the KB suggestions (i.e. what is shown on the interface is the intersection of the NN suggestions with all KB suggestions). In other words: if the user is solving a problem that has occurred previously, he can focus properly with the help of the NN in 95% of the cases; if it is a new problem, he will be better off (compared to a system which doesn't tune itself) in 64% of all the cases. But even in the remaining 36% of the cases he won't pursue any wrong hypotheses, he will just focus on hypotheses which will lead to a less optimal solution (but nevertheless a correct one).

The initial experiments have been encouraging. Thus, it seems sensible to do further work in this direction. One task will be to analyze, how the optimal balance between design knowledge (acquired via traditional knowledge acquisition methods) and situation specific knowledge (acquired via adaptive mechanisms) can be found. Moreover, it has to be investigated what the impact of this point of view is on knowledge engineering. We suppose, for example, that it may not be desirable to develop enormous libraries of problem solving methods but that more time and effort should be spent on studying experts and users in how they interact with their social and physical environments, and how the introduction of knowledge systems changes their work. These insights will in turn "feed back" into the design of better knowledge systems as discussed in this paper.

[1] The net was a three layer net with 167 input nodes (findings), 80 hidden nodes and 142 output nodes (hypotheses). We used a learning rate of 0.5 and a momentum value of 0.9. The weights were changed after each pattern presentation.

## Acknowledgements

We would like to thank B. Aiken, D. Allemang, T. Bratschi, and T. Wehrle for their helpful comments.

## References

[Clancey, 1989a] W.J. Clancey. The knowledge level reinterpreted: Modeling how systems interact. *Machine Learning,* 4:285-291, 1989. Kluwer Academic Publishers, Boston, 1989.

[Clancey, 1989b] W.J. Clancey. The frame of reference problem in cognitive modeling. In *Proceedings of the Annual Conference of the Cognitive Science Society,* pages 107-114. Lawrence Erlbaum, Hillsdale, New Jersey, 1986.

[Gutknecht and Pfeifer, 1990J. M. Gutknecht and R. Pfeifer. An approach to integrating expert systems with connectionist networks. *AI Communications,* 3(3): 116-127, September 1990.

[Nisbett and Wilson, 1977] R.E. Nisbett and T.D. Wilson. Telling more than we can know: Verbal report on mental processes. *Psychological Review,* 84:231-259, 1977.

[Rumelhart *et ai$_t$* 1986] D.E. Rumelhart, J.L. McClelland, and the PDP Research Group. *Parallel Distributed Processing,* Vol. 1. MIT Press, Cambridge, Massachusetts, 1986.

[Suchman, 1987]. L.A. Suchman. *Plans and situated actions.* Cambridge University Press, Cambridge, 1987.

[Winograd and Flores, 1986]. T. Winograd and F. Flores. *Understanding Computers and Cognition.* Addison Wesley, Reading, Massachusetts, 1986.