Performance Issues in ADAPtER a Combined CBR-MBR Diagnostic Architecture

Luigi Portinale and Pietro Torasso
Dipartimento di Informatica - Universita' di Torino
C.so Svizzera 185 - 10149 Torino (ITALY)
e-mail: {portinal,torasso}@di.unito.it

Abstract

The present work describes some aspects of the diagnostic problem solving architecture of ADAPtER, a multi-modal reasoning system combining Case-Based Reasoning (CBR) and Model-Based Reasoning (MBR). In particular, some issues concerning the performance of such a combined architecture are discussed, with particular attention to the problem of maintaining under control the growth of the case memory. In fact, an over-sized case memory is the main responsible for the arising of the *utility problem* in ADAPtER. We identified such a responsibility through a set of experiments concerning the average behavior of the system with respect to a given domain. As a consequence, we propose two learning strategies regarding the addition and replacement of cases in memory. Experimental results are quite encouraging and suggest that the adoption of such strategies can greatly mitigate the over-sizing of the case memory.

Introduction

The idea of using multiple representations for problem solving has attracted a significant amount of attention and a number of systems have been developed which are able to solve complex problems, mainly in the area of diagnosis, by exploiting some form of integration and co-ordination among different representations (see (David, Krivine, & Simmons 1993) for a survey). Approaches based on multiple representations for problem solving require the solutions to not easy problems such as the selection of specific representations which can be actually useful for the task at hand, the way different representations are used (and when) by a problem solver, in order to gain the maximum of efficacy and efficiency for a particular task.

Taking into account the above issues, we investigated the potential of the integration of Case-Based Reasoning (CBR) and Model-Based Reasoning (MBR) with respect to diagnostic problem solving. In domains where a precise domain theory is available and analytical methods exist for solving the problem, the advantage of using CBR (possibly in conjunction with other methods) could seem less obvious with respect to domains where the domain theory is very partial and weak. However, CBR can still provide advantages when the computation of a solution from scratch is very complex; this is often the case when pure model-based approaches are used, so this kind of integration has been studied for tasks like design (Goel 1989), planning (Veloso 1994) and diagnosis (Koton 1989).

In the following, we will discuss some aspects concerning **ADAPtER**, a diagnostic system based on a multi-modal reasoning paradigm that we recently developed. In particular, we will address some issues concerning the efficiency of an integrated architecture combining CBR and MBR for diagnostic problem solving, by pointing out how the arising of some form of the utility problem makes necessary the definition of appropriate learning strategies.

The ADAPtER System

The name **ADAPtER** means **A**bductive **D**iagnosis through **A**daptation of **P**ast **E**pisodes for **Re**-use and indicates a diagnostic architecture combining Model-Based Reasoning and Case-Based Reasoning (Portinale & Torasso 1995). Differently from other similar systems like CASEY (Koton 1989), the architecture of ADAPtER aims at integrating aspects concerning case management and abductive reasoning in a uniform and flexible framework based on a well-founded specification of the notion of diagnosis; it involves the set of components shown in figure 1 (links represent data flow).

The high-level behavior of ADAPtER can be described by the following pseudo-code:

```
ADAPtER(new-case,Case-Memory,Causal-Model):

IF NOT RETRIEVE(new-case, Case-Memory,
                          retrieved-solution)
   THEN BEGIN
        MBR(new-case, Causal-Model, mbr-solution);
        return(mbr-solution)
        END
```

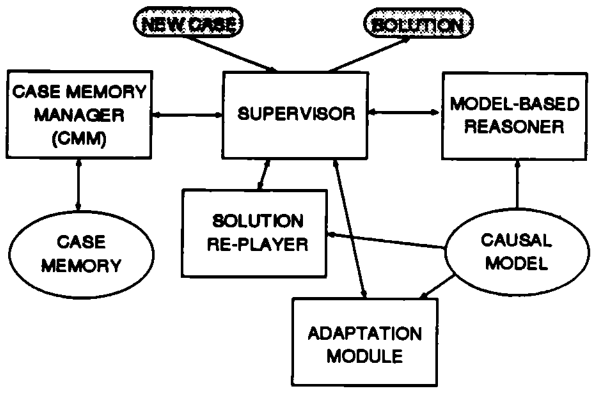

Figure 1: ADAPtER Architecture

```
ELSE
 IF OK-SOLUTION(new-case, retrieved-solution,
                Causal-Model,replayed-solution)
  THEN return replayed-solution
  ELSE
   IF ADAPTATION(new-case, replayed-solution,
               Causal-Model, adapted-solution)
      THEN return adapted-solution
      ELSE
       BEGIN
       MBR(new-case, Causal-Model, mbr-solution);
       return(mbr-solution)
       END
```

When presented with a new case, the SUPERVISOR first invokes the CASE MEMORY MANAGER (CMM) in order to retrieve the most promising cases from the CASE MEMORY (**RETRIEVE**). Such a step evaluates the degree of match between the current case to be solved and the retrieved ones, using a heuristic function which estimates the adaptation effort rather than just the similarity between the current and the retrieved cases. Although ADAPtER has currently the possibility of using two alternative case memory organizations, namely an E-MOP organization or an *associative flat memory* (Kolodner 1993), in the following we will discuss a version of the system using the latter memory structure. In particular, using an associative flat memory, an efficient kind of adaptation-guided retrieval called *Pivoting-Based Retrieval* (PBR) can be used to implement **RETRIEVE** (Portinale, Torasso, & Magro 1997). The PBR algorithm is based on the computation of suitable bounds on the adaptation cost of each solution of a case; such bounds are then used to restrict the search for the best case to be retrieved. In the following we will discuss a version of the system using this kind of strategy. If **RETRIEVE** fails to find a promising case, the control is switched directly to the MODEL-BASED REASONER (**MBR**), otherwise a set of cases with solutions having minimal estimate of adaptation effort is returned and one solution is selected for further elaboration. The SOLUTION RE-PLAYER is then invoked by the SUPERVISOR to replay the retrieved solution (**OK-SOLUTION**). The retrieved solution is used together with the contextual data of the case under examination (potentially different from the contextual data of the retrieved case) and the CAUSAL MODEL (which represent the domain knowledge) to recompute all the possible consequences. The **OK_SOLUTION** step succeeds if some criteria (namely *consistency* and *covering*) between the solution's predicted observable parameters and the current set of observations are met (Portinale & Torasso 1995). If a failure occurs, the replayed solution is passed on to the ADAPTATION MODULE for the **ADAPTATION** step; this step adapts the retrieved solution to be a solution of the current case, by using the same domain knowledge (that is the Causal Model) used by the MODEL-BASED REASONER. The goal is to remove possible inconsistencies in the replayed solutions and to build missing explanations (covering) for some manifestations of the case, in order to obtain a formally correct solution. If adaptation fails, the control switches to **MBR** for solving the new problem from scratch.

A basic feature of ADAPtER concerns the fact that model-based diagnostic problem solving relies on a formal logical theory of diagnosis (Console & Torasso 1991) and that the case-based component also relies on such a characterization in at least two ways:

- the case-based and the model-based components share the same "logical" representation of domain knowledge;
- the adaptation module and the model-based reasoner exploits the same kind of inference steps to perform their tasks. (see (Portinale & Torasso 1996)).

This tight integration allows the system to work in a very uniform framework, while taking advantage of the possibility of avoiding a complete reasoning from scratch, in case relevant past episodes could be re-used.

Performance Issues

Even if in principle the kind of co-operation among the modules of ADAPtER described in the previous section appears to be fruitful, the real advantage of the architecture has to be verified in practice. In fact, there is no absolute guarantee that the combined system is more efficient than the MBR alone. The main reasons for that lie in the following problems:

1. from the theoretical point of view, adapting a retrieved solution has the same complexity than solving the new problem from scratch; indeed, we

have proved that both the above problems are NP-complete, even if adaptation is considered from the conservative point of view (i.e. by trying to keep as much as possible of the old solution) or if the current case to be solved and the retrieved one are very similar (Portinale & Torasso 1996);

2. as most of the CBR systems, ADAPtER suffers from the utility problem, essentially because adding new cases in memory can greatly increase the overhead due to the retrieval and evaluation of cases (van Someren, Surma, & Torasso 1997).

For these reasons, we have performed a number of experiments trying to evaluate the real advantages of combining CBR and MBR for diagnostic purposes.

To enable meaningful comparison among experiments, we have developed a *simulator* automatically generating a case, by taking into consideration a causal model (the domain knowledge base) and some domain-dependent parameters. In particular we identified the following input parameters for the simulator:

p_1 the probability of a causal relation to be enabled;

p_2 the maximum number of initial causes with abnormal values (i.e. faults) to be included in the case;

p_3 the probability that a non-predicted observable parameter has to be included in the case description with a normal value;

p_4 the probability that an observable parameter that has been predicted by the simulator has to excluded from the case description.

These parameters allows us to generate cases with different structural features; indeed, for some experiments we need to distinguish between two different classes of cases: cases to be stored in memory (training sets) and cases representing typical problems to be solved by the system (test sets). In particular, we generated training sets having cases which were all different and (almost) "complete". This means that all relevant observable parameters were been actually observed (included into the case); this has been obtained by setting parameter p_3 to a value very close to 1 and p_4 to 0. Test cases has been generated with smaller values of p_3 and with $p_4 > 0$. Parameter p_1 has been set in all experiments to a value close to 1, meaning that causal relations that are not certain are assumed as a kind of default. Parameter p_2 has been varied depending from the experiments. There are two main issues that has been addressed by the experiments

1. to determine the *average best architecture* for a given domain and a given distribution of problems;

2. to determine the *best architecture* for each single case submitted to the system.

These points will be discussed in the following subsections.

Average Costs and Utility

Concerning the problem of verifying the behavior of ADAPtER with respect to a given domain and distribution of problems, we have performed a detailed set of experiments; in the present work we will present some of them by taking into consideration two particular domains: a mechanical domain represented by a causal model of *car engine faults* and a medical domain represented by a causal model of the *leprosis disease*. The results of such experiments are summarized in figure 2. Average costs (in terms of CPU time) of ADAPtER are

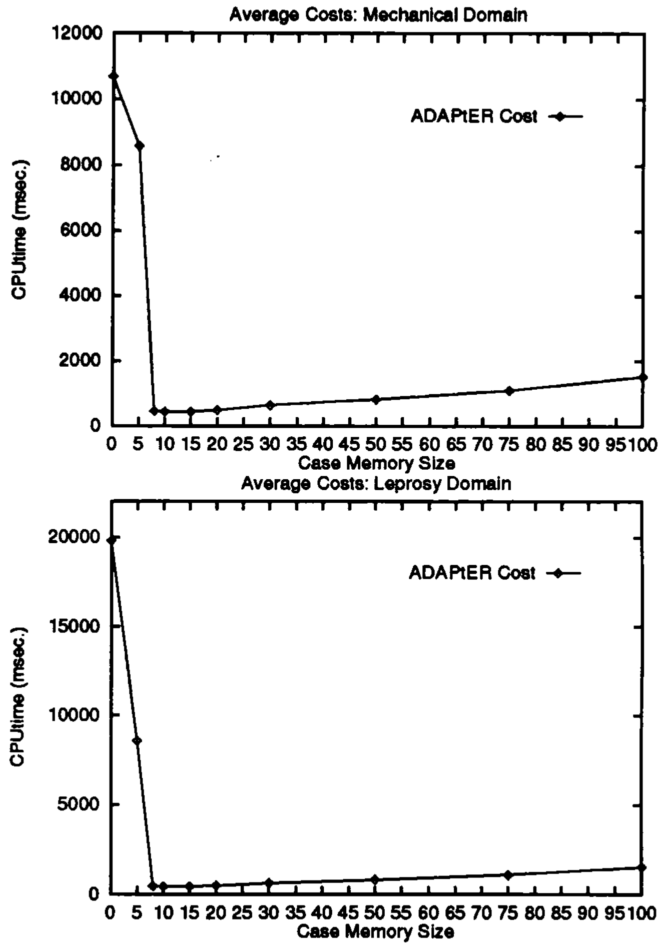

Figure 2: Average Costs for Mechanical and Leprosy Domains

plotted with respect to case memories of different sizes.

The case memory was filled with training sets of cases of different sizes and the resulting system was run on a test set of 100 problems. Such experiments show the arising of the utility problem: the average cost of ADAPtER, after having reached a minimum, tends to linearly increase with growing case memory sizes. An in-depth analisys shows that this effect is essentially (but not completely) due to the increased retrieval time when the number of cases in memory also increase and points out the importance of having good monitoring strategies on the case memory. Retrieval is the main responsible for the utility problem, however also adaptation seems to play a role. Indeed, we also measured some estimate (based on counting) of the probability of success of the single steps of ADAPtER. In particular, we measured an estimate of the probability of success of the **OK-SOLUTION** and **ADAPTATION** steps[1]. Results with respect to the case memory size parameter are plotted in figure 3 for the two considered domains. We noticed in both domains a slight increase in the probability of success of **OK-SOLUTION** with growing memory sizes (that however, does not fully compensate the increase in the cost of **RETRIEVE**). On the other hand, a peculiar behavior can be identified, in the mechanical domain, for the probability of success of **ADAPTATION**; it shows a decreasing patterns with growing memory sizes, meaning that it is possible that, even if more cases are in memory, less useful cases can be retrieved. This seems to suggest that in some situations, when more cases are available, the probability of getting an estimated promising case that is actually not so good can increase.

In any case, such results show that a parameter like the number of cases in memory must be seriously taken into account in order to tune the architecture and to test the utility of its components. For this reason we studied the possibility of defining *learning and forgetting strategies* in order to decide whether to add, replace or delete a case from the case memory.

Learning Strategies

The definition of suitable learning strategies controlling the growth of the case memory size are of primary importance in a combined architecture like ADAPtER, but also in any general CBR systems (see (Smyth & Keane 1995)). A deep investigation of the problem is still under examination, but we have some preliminary results concerning two strategies applied to the mechanical domain.

The first strategy we tested is a strategy of *pure addition* of cases into memory. We start with an empty case memory and we add a case into memory if and

[1]In the current implementation, the **RETRIEVE** step has a probability of success equal to 1 almost every time and it is not interesting to consider it.

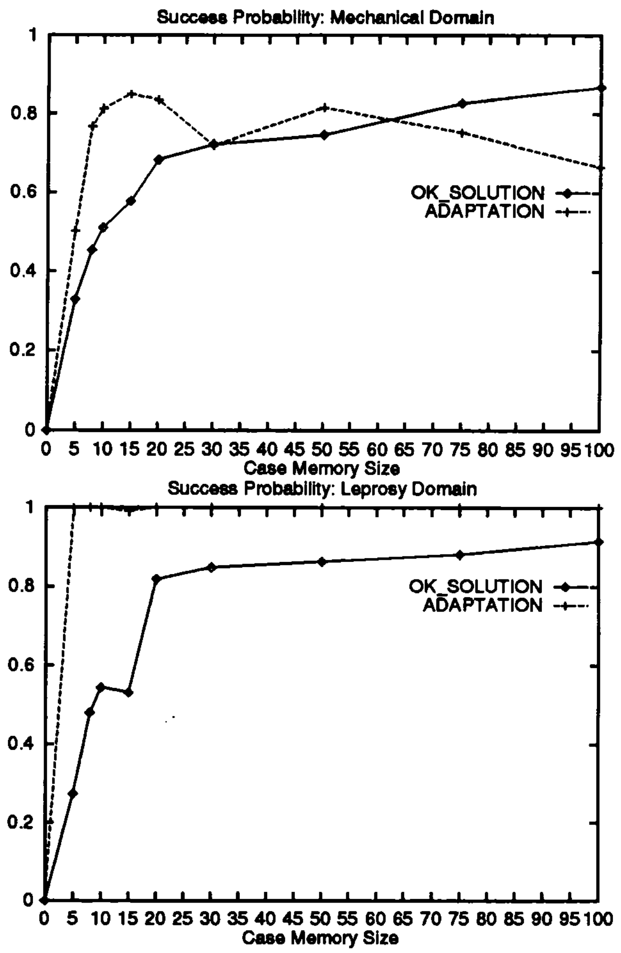

Figure 3: Success Probabilities for Mechanical and Leprosy Domains

only if the case has to be solved by the MBR component (i.e. if either no case is retrieved or adaptation fails). This strategy, called **ADD_ONLY**, implements the principle that the only cases worth to be stored into the case memory are the ones solved by MBR, since they represent the gaps to be filled in the competence of the CBR component.

A second strategy we tested concerns the possibility of adding a new case by replacing an old one. This strategy, called **REPLACE**, is activated when an input case I_j has to be solved by the MBR component, that is the CBR component was unable to retrieve and successfully adapt a solution of a retrieved case C_i to be a solution for I_j. Let us denote with:

- MBR_TIME(I_j) the cost (in terms of cputime) of solving I_j by MBR;
- $h(C_i, I_j)$ the heuristic function estimating the cost of

adapting the stored case C_i into the input case I_j[2]; it is worth noting that $h(C_i, I_j) \leq h(C_k, I_j)$ for any C_k stored in the case memory since C_i was chosen as the most promising case for adaptation;

- $h(I_j, C_i)$ the estimated cost of adapting the solution of the input case I_j to be a solution for the case C_i.

The strategy **REPLACE** decides to replace the stored case C_i with the input case I_j into the case memory (i.e. to add I_j and to delete C_i) if and only if

- adaptation using C_i has failed;
- $h(I_j, C_i) < \sigma$ where σ is a suitable threshold.
- $\text{MBR\_TIME}(I_j) > \text{MBR\_TIME}(C_i)$;

Notice that the main reason for the strategy to work is that the function h is not symmetric; indeed, if C_i has been retrieved using I_j as input, then $h(C_i, I_j)$ was not a good estimate because a failure in the adaptation process occurred. However, the lack of precision for $h(C_i, I_j)$ does not mean that the same problem affects $h(I_j, C_i)$; in particular we can reasonably assume that if the $h(I_j, C_i)$ is low, there is no necessity of maintaining C_i because there is evidence than case C_i can be adapted at inexpensive cost to I_j. Moreover, the replacement of C_i by means of I_j should provide larger saving in computation time, because of the principle that only the most expensive cases should be retained into memory.

In order to evaluate the relative merits of the different strategies, we have submitted a test set of 5000 cases to be solved by ADAPtER with **ADD_ONLY** and **REPLACE** strategy respectively. We evaluated the average cost of solving a case as a function of the number of cases submitted so far. Figure 4 shows the two corresponding plots[3]; in particular, the point $(n, C(n))$ of each plot in the graph represents the average cost $C(n)$ after the examination of n cases, where $n = 100 * k$ $(k = 1 \ldots 50)$ (i.e. values are plotted every 100 cases). We can still notice the arising of the utility problem, since both strategies may produce case memories with increasing sizes as long as new test cases are examined and no suitable adaptable cases can be found (i.e. MBR has to be invoked). However the **REPLACE** strategy seems to effectively deal with the utility problem (the average cost increase very slowly), since it is able to limit the growth of the case memory,

[2]In fact, the case C_i may have more than one solution and $h(C_i, I_j)$ is considered to be the minimum cost among all the solutions of C_i.

[3]Notice that the cost values of figure 4 are quite different from those reported on figure 2 because different kind of hardware platform have been used.

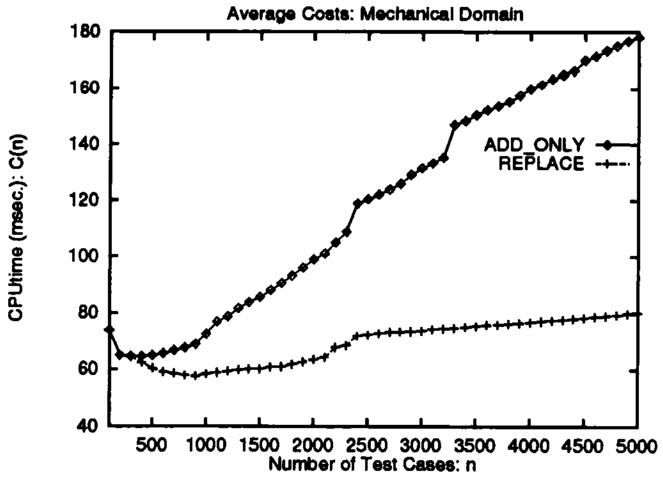

Figure 4: Average Costs Mechanical Domain: MBR and Learning Strategies

by keeping stored just a few significant cases. This can be noticed in figure 5 that shows that the number of cases stored in memory after 5000 test cases is limited to 43 cases when using **REPLACE** against 141 cases when using **ADD_ONLY**. This is a surprisingly

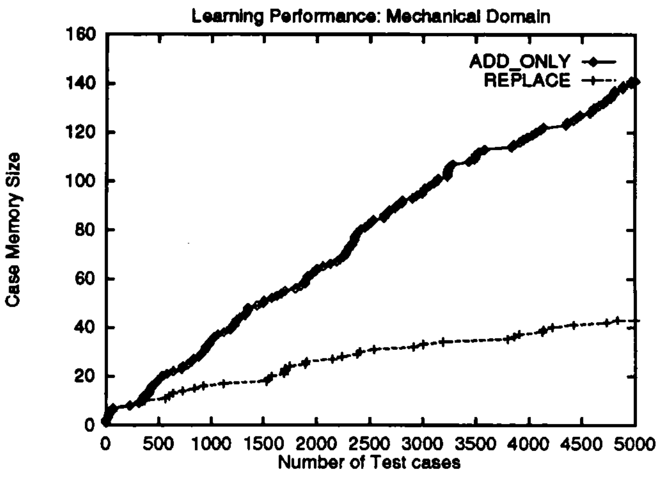

Figure 5: Learning Performance: Mechanical Domain

small number when we consider that we started from an empty case memory and the test set is composed of cases automatically generated with parameter $p_2 = 5$, that is up to 5 different faults can be simultaneously present in each test case.

These results are even more interesting from the computational point of view if we compare them with the average cost of solving the above 5000 cases by using the MBR module alone. In fact, average cost analysis shows that the average cost of pure MBR is quite high; in particular the sample mean of the MBR cost after 5000 cases results to be equal to 855.41 msec. with a maximum error of ±158.14 msec. in a confidence interval at 0.05 confidence level (i.e. 95% of precision).

The relatively bad news is that the **REPLACE** learning strategy is quite sensitive to the threshold σ that is used to replace a case. Figure 6 shows a plotting of the number of cases that are stored after 5000 tests, depending on the value of the threshold. Notice

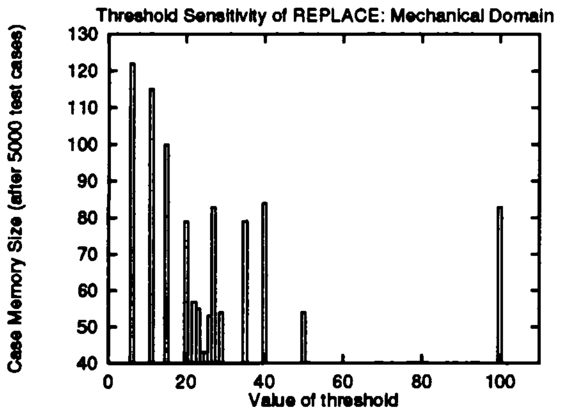

Figure 6: Sensitivity of REPLACE to threshold: Mechanical Domain

that results plotted in figure 5 are relative to the best threshold, however even in the worst case, the number of cases in memory is relatively small and still less than the number of cases obtained using **ADD_ONLY** (that is producing a small number of cases anyway)

Conclusions

In the present work we have presented the analysis of some performance aspects of a multi-modal reasoning architecture for diagnostic problem solving, combining CBR and MBR. We pointed out in particular, the arising of a form of *utility problem* similar to that discussed in (Francis & Ram 1993; Smyth & Keane 1995). A critical role in the arising of such a problem is played by the size of the case memory that must be searched. For this reason we investigated two different learning strategies able to add or replace cases to memory. Preliminary results on a domain concerning car engine faults suggest that these form of learning can be really effective in taking under control the growth of the case memory and allow a significant reduction in average computational cost with respect to a pure MBR approach (about one order of magnitude). Future works will concentrate on the problem of selecting a good start-up case memory (instead of starting with an empty one) and on the definition of *opportunistic strategies* to be adopted by the system in order to decide whether to try to solve the case by CBR or directly by MBR.

Acknowledgments

We would like to thank M. van Someren for many fruitful discussions and P. Tavano for running the experiments. The work has been partially supported by CNR project SCI*SIA under grant 97.02436.CT12.

References

Console, L., and Torasso, P. 1991. A spectrum of logical definitions of model-based diagnosis. *Computational Intelligence* 7(3):133–141.

David, J.; Krivine, J.; and Simmons, R. 1993. *Second Generation Expert Systems.* Springer Verlag.

Francis, A., and Ram, A. 1993. The utility problem in case-based reasoning. Technical Report ER-93-08, Georgia Tech.

Goel, A. 1989. Integration of case-based reasoning and model-based reasoning for adaptive design problem solving. Technical report, (PhD Diss.), Ohio Univ.

Kolodner, J. 1993. *Case-Based Reasoning.* Morgan Kaufmann.

Koton, P. 1989. Using experience in learning and problem solving. Technical report, MIT/LCS/TR-441.

Portinale, L., and Torasso, P. 1995. ADAPtER: an integrated diagnostic system combining case-based and abductive reasoning. In *LNAI 1010.* Springer Verlag. 277–288.

Portinale, L., and Torasso, P. 1996. On the usefulness of re-using diagnostic solutions. In *Proc. 12th European Conf. on AI - ECAI 96*, 137–141.

Portinale, L.; Torasso, P.; and Magro, D. 1997. Selecting most adaptable diagnostic solutions through Pivoting-Based Retrieval. In *LNAI 1266.* Springer Verlag. 393–402.

Smyth, B., and Keane, M. 1995. Remembering to forget. In *Proc. 14th IJCAI*, 377–382.

van Someren, M.; Surma, J.; and Torasso, P. 1997. A utility-based approach to learning in a mixed case-based and model-based reasoning architecture. In *LNAI 1266*, 477–488. Springer Verlag.

Veloso, M. 1994. *Planning and learning by analogical reasoning.* LNAI 886, Springer Verlag.