Improving the Quality of Solutions in Domain Evolving Environments

Josep Lluís Arcos

IIIA, Artificial Intelligence Research Institute
CSIC, Spanish Council for Scientific Research
Campus UAB, 08193 Bellaterra, Catalonia, Spain
arcos@iiia.csic.es, http://www.iiia.csic.es

Abstract. The development of industrial case-based reasoning systems that have to operate within a continually evolving environment, is a challenging problem. Industrial applications require of robust and competent systems. When the problem domain is evolving, the solutions provided by the system can easily become wrong. In this paper we present an algorithm for dealing with real-world domains where case solutions are evolving along the time. Specifically, the algorithm deals with what we call the *innovation problem*: the continuous improvements on the components that are part of case solutions. We will show how the use of the proposed algorithm improves significantly the quality of solutions in a deployed engineering design system.

1 Introduction

The development of industrial case-based reasoning systems that have to operate within a continually evolving environment, is a challenging problem. Industrial applications require of robust and competent systems. When the problem domain is evolving, the solutions provided by the system can easily become wrong.

In this context, *sustained* case-based reasoning systems [2] are strongly needed. The goal of sustained case-based reasoning systems is to learn continuously from the experience of solving problems. As is argued in [1], a sustained case-based reasoning system requires of an underlying deep knowledge model of the domain, as well as a problem solving mechanism that has to make use of it.

Two main evolving directions can be identified in problem domains. The first one is the change on the type of problems: new types of problems may become important and previously important problems may become irrelevant. The second one is the change on the type of solutions: the same type of problems that were previously solved using a specific domain solution may require a new domain solution to be solved.

Examples of case-based reasoning tasks that have to deal with evolving solutions are design and configuration tasks. Real-world design systems have to incorporate functionalities for dealing with the use of new design components or the improvement of previously existing components. We call this design problem the *innovation problem*.

P. Funk and P.A. González Calero (Eds.): ECCBR 2004, LNAI 3155, pp. 464–475, 2004.

A strategy for solving the innovation problem is to incorporate maintenance processes into the case-based reasoning applications [11, 8, 10]. The most usual techniques used for catching up the changes in the domain environment are case maintenance techniques for reorganizing the case base. Deletion polices are examples of strategies used for dealing with cases with obsolete solutions.

The success of any CBR system is not only motivated by the case base quality. The success of any CBR system depends also on all its knowledge containers [9] and, specially, on the similarity or retrieval knowledge and on the adaptation knowledge. A used policy in design or configuration problems is the improvement of the adaptation knowledge by incorporating rules for component substitution: when a new component is introduced in the solution designs substituting a previous existing component, a new rule is incorporated in the adaptation knowledge.

Nevertheless, not all solution changes can be reduced to the incorporation of new components. One type of the changes is the improvement of a pre-existing component. The main problem with these improvements is that the improvements are usually based on problem specific requirements. That is, there are only experimental evidences of the improvement and not a general theory. Then, a system that has to deal with this incomplete model, could use a case-based reasoning approach. This is the motivation of our proposal, dealing with problem-based improvements for improving the quality of other problem solutions.

In this paper we will present a proposal for dealing with problem domains with evolving solutions concentrating the efforts on the retrieval and the reuse steps. Our goal is to develop an *innovative aware* CBR algorithm, i.e. a CBR algorithm where the retrieval and reuse phases are not degraded by the periodical incorporation of innovations in the problem solutions. We will show how the analysis of solution changes in customary problems – problems that are periodically solved by the system – can be used for improving the quality of solutions in occasional problems – problems that are rarely solved by the system.

We have tested the proposed algorithm in *T-Air*, a deployed engineering design system. *T-Air* is a case-based reasoning application developed for aiding engineers in the design of gas treatment plants [4]. The main problem in designing gas treatment plants is that the diversity of possible problems is as high as the diversity of industrial processes but there are only experimental models for few of them. The knowledge acquired by engineers with their practical experience is essential for solving new problems. The innovation problem in *T-Air* is due to the continuous improvements on the equipment used in the design of gas treatment plants. In this paper we will report the significant improvement of solution quality when using our proposed algorithm.

This paper is organized as follows: In section 2 we present the innovation problem and propose the innovation aware CBR algorithm. In section 3 we exemplify the incorporation of the proposed algorithm in a deployed application an report the results of the experiments performed. The paper ends with a description of the current status of the work and the planned future work.

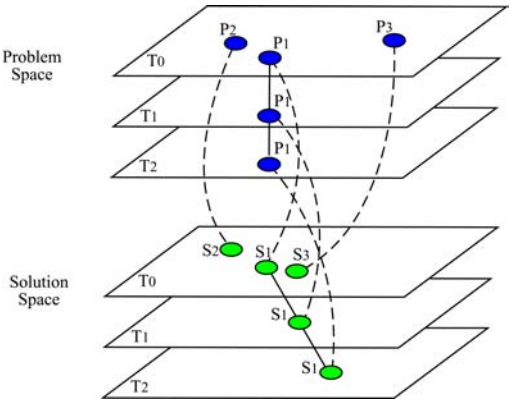

Fig. 1. An Extension of the classical problem and solution CBR spaces for dealing with time factor.

2 The Innovation Aware Problem

The goal of a sustained CBR system is to not degrade the quality of the solutions generated taking into account that the domain environment may evolve. A first naive strategy for solving this problem can be to only use recently solved cases (for instance, including the case date into the similarity measure). Nevertheless, as we will see below this naif approach is not enough.

In a given case base, we can classify the cases into two categories: *customary problems* and *occasional problems*. Customary problems are problems that are periodically solved by the system. Occasional problems are problems that are rarely solved by the system. In customary problems, the strategy of only focusing on recent solutions can be appropriate. The main problem rises with occasional problems: when a new occasional target problem has to be solved, usually only old solutions can be found in the case base. Then, a CBR inference system that is just reusing these old solutions may generate solutions with a low quality, i.e. solutions that may cause the distrust in the system.

Let us illustrate the evolving environment problem using the scheme of the Figure 1. There is a customary problem P_1 that was solved at times T_0, T_1, T_2. Each time the problem was solved, a small innovation was applied to the solution. Whenever a new P_1 target problem has to be solved, it is clear that we can take, as a basis for the reuse, the most recent solution stored in the case-base.

Now let us assume that there is an occasional problem P_2 that was solved at T_0 (see Figure 1), when a new P_2 target problem arises at T_2, we have two alternative cases to consider: S_2 solved at T_0 or S_1 solved at T_2. Using as a criterion the problem similarity, S_2 is the solution designed for the closest problem. Nevertheless, because problems P_1 and P_2 are very similar and P_2 has been solved recently, it can be more feasible to consider S_1 as a candidate for reuse (i.e using the case date when assessing similarity). Taking the last alternative, we are imposing a more powerful adaptation mechanism.

Finally, in the figure, we have another occasional problem P_3 that was solved at T_0. P_3 is a problem far from the other problems P_1 and P_2 but that has a solution S_3 close to the other solutions. When a new P_3 target problem comes to the system, P_1 and P_2 will not be retrieved. Thus, the closest solution is S_3. Nevertheless, taking into account that S_3 was designed at T_0, the solution we can reuse from S_3 will possibly not include recent innovations. Then, taking into account similar solutions more recently i.e. solved (solution S_1 that has been solved at T_2) we can improve the quality of the solution by tuning the solution taking into account the innovations introduced in S_1.

From these simple examples, it is clear that the innovation problem has to be dealt in the CBR inference procedure. Below we will present a variant of the classical CBR inference cycle [3] that incorporates an additional retrieval and reuse steps on the space of solutions.

An alternative approach to deal with the innovation problem could be by implementing a learning module able to incorporate new knowledge into the adaptation model. This approach has the difficulty of managing with the usual incomplete model of the domain and has not been addressed in this paper.

Moreover, it is important to remark that whether we apply deletion policies in the case base, the performance for occasional problems could be affected because we can loose the knowledge about innovation changes. For instance, in our previous example from Figure 1, if we only keep the solution S_1 at time T_2 – deleting the solutions at times T_0 and T_1 – the relationship between the solutions S_3 and S_1 could be not established.

Before describing our innovation aware algorithm for dealing with evolving solutions, we will introduce some basic notation: a case c_i is defined as a tripled $c_i = (p_i, s_i, t)$ where p_i is the problem description, s_i is the solution, and t is the date when c_i was solved. Moreover, we assume that there exists a similarity measure $Sim_p(p_i, p_j)$ between problem descriptions for retrieving and ranking the cases more similar to a new target problem. We say that two cases c_i, c_j are *innovation variants* when their problems p_i, p_j are equivalent ($Sim_p(p_i, p_j) = 1$) and their solutions s_i, s_j are different. Finally, we also assume that exists a similarity measure $Sim_s(s_i, s_j)$ between case solutions.

2.1 The Innovation Aware Algorithm

We propose to incorporate an additional retrieval and adaptation step into the usual CBR inference cycle: after retrieving and reusing the most similar cases for generating the solution of a new target problem – when the reused cases are not recent – we propose to refine the solution generated by looking for recentness paths on the solution space. Our CBR inference algorithm (see Figure 2) is then divided into four phases:

1. Retrieving similar cases using Sim_p: given a new problem p, the first phase uses the Sim_p similarity measure on problem descriptions for retrieving and ranking similar cases (noted C). This phase models the usual retrieval step. Because we are not taking into account the solution date, all the cases with an equivalent problem description will be grouped with the same similarity.

```
    Procedure IN-AW( p , t )
(1)     C = P-Retrieval(p)
(2)     ⟨ s , t' ⟩ = Reuse( p , C)
        if (t - t' < δ) then
          return s
        else
(3)       S = S-Retrieval(s, t')
(4)       s' = Inn-Reuse(s, S)
          return s'
        end if
```

Fig. 2. The innovation aware algorithm. A first a retrieval step on the problem space; then a reuse step on solutions; next a retrieval step on the solution space; and finally a reuse step on the innovation path.

2. First solution proposal: a first solution s for problem p is constructed by reusing solutions of previously retrieved cases C. We say that s is a solution for time context t', where t' is calculated from the dates of cases C. When the most similar cases are recent (we use a fixed δ parameter), the solution S is proposed as final solution and next phases are skipped. Otherwise,
3. Retrieving similar solutions using Sim_s: the goal of the third phase is to retrieve the customary problems S' with a similar solution to s (using Sim_s) solved near t' (using a fixed threshold γ). Only problems with additional solutions more recent than t' are considered–i.e. cases that gather solutions with innovation variants.
4. Applying innovation variants: the goal of the last phase is to apply innovation variants to s. This phase requires domain specific policies for identifying the relevant solution changes.

The threshold parameters δ and γ have to be determined for each domain problem. In the current implementation of the `IN-AW` algorithm we have manually tuned their values for the use in their *T-Air* application.

3 The Application in an Industrial System

We incorporated the innovation aware CBR algorithm in *T-Air*, a case-based reasoning application developed for aiding engineers in the design of gas treatment plants[4]. The gas treatment is required in many and diverse industrial processes such as the control of the atmospheric pollution due to corrosive residual gases which contain vapours, mists, and dusts of industrial origin. Examples of gas treatments are the absorption of gases and vapours such as SO_2, CLH, or CL_2; the absorption of NO_x with recovering of HNO_3; the absorption of drops and fogs such as PO_4H_3 or $ClNH_4$; dust removal in metallic oxides; and elimination of odours from organic origin.

The main problem in designing gas treatment plants is that the diversity of possible problems is as high as the diversity of industrial processes but there are

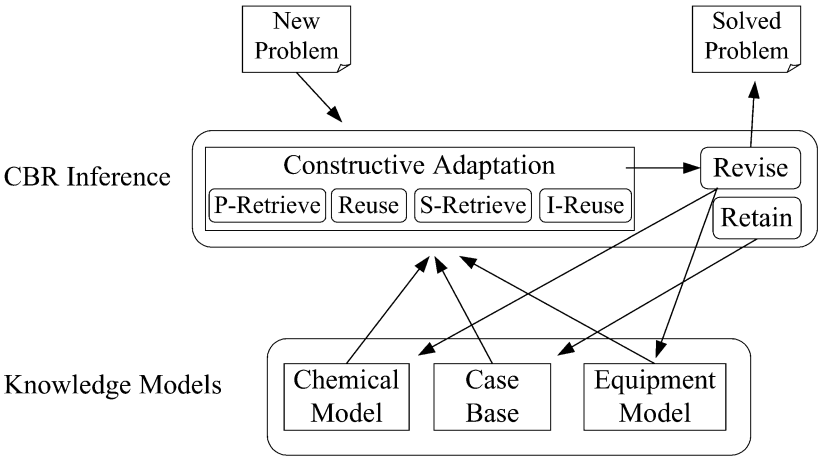

Fig. 3. *T-Air* functional Architecture.

only experimental models for few of them. The knowledge acquired by engineers with their practical experience is the main tool used for solving new problems.

3.1 General Architecture

The *T-Air* architecture is based on three knowledge models and three reasoning modules (see Figure 3). The first knowledge model is the *Chemical Model* and is the basis for determining the similarity between a gas composition in a new problem and those treated previously and stored in the case base. Moreover, since we are mixing a polluted gas with a washing liquid, we also have modeled the properties of the washing liquids, the compatibility among gases and liquids, and some chemical reactions involved.

The second knowledge model is the *Equipment Model*. The equipment model describes the components used in gas treatment plants (gas-washing units, pumps, fans, tanks, and accessories). Each equipment has involved a collection of working parameters. The relationships among these working parameters are modeled by means of a collection of equations, a collection of constraints about their maximum and minimum values, and a collection of safety conditions expressed as heuristics.

Finally, the last knowledge model holds the *Case Base*. *T-Air* uses a highly structured representation of cases. A case is represented as a complex structure embodying four different kinds of knowledge: the *Input Knowledge*, a *Chemical Case-Model*, the solution *Flow Sheet*, and *Annotations*.

The *Input Knowledge* embodies data about the customer such as the industrial sector it belongs or the industrial process that originates the polluting gas; data about the working conditions of the installation such as temperature or gas flow; data about the composition and concentration of input polluted gas; and data about the desired concentration in the output emission. The *Chemical*

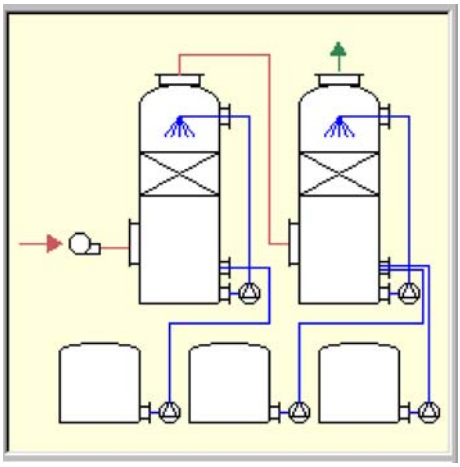

Fig. 4. An example of a simple flow sheet generated by *T-Air* with two scrubbers. each of them with a centrifugal pump that sucks the washing liquids from three tanks to the scrubbers, and one fan at the beginning of the process.

Case-Model embodies a graph structure, generated by *T-Air* using the chemical model, modeling the main characteristics of the input gas. The chemical case-model extends the input knowledge and fixes the thresholds for the working conditions, the main issues to be analyzed, and the kind of gas treatment required. The *Flow Sheet* component describes the solution designed for cleaning a polluted gas. As shown graphically in Figure 4, the flow sheet specifies a collection of required equipment (mainly scrubbers, pumps, tanks, and fans), the design and working parameters for each equipment, and the topology of the installation (the gas circuit and the liquid circuits). The flow sheet is also represented as a graph structure. Finally, *Annotations* are meta-information that, when an external factor influences the solution, describe the reasons for a given solution decision – examples of s annotations are the design decisions forced by the user requirements such as the washing liquid, over-dimensionated parameters because of security reasons, or spatial requirements.

The inference in *T-Air* is performed by three modules (see Figure 3): the *Constructive Adaptation* module, the *Revision* module, and the *Retention* module. The main inference is performed by the constructive adaptation module and will be described in the next subsection.

Because the chemical knowledge required for covering all the potential problems is extremely huge, the approach followed was to only model the knowledge related with the initial case base. This design decision had two main consequences: i) the system required a *Revision* module where the engineers are able to correct the system solution, and ii) the system required a mechanism for gradually built up the chemical knowledge when it is required. Moreover, the engineer have a complete control over the plant parameters (the system only

prevents safety violations). The approach we have adopted is close to work of capturing expert design knowledge through "concept mapping" [6]: when the engineer is analyzing a problem involving a gas not previously covered, the engineer uses the navigation interactive capabilities of *T-Air* for determining the possible characteristics of the gas and improves the chemical knowledge of the system by means of completing the information.

An important issue during the development of the system was to provide useful navigation tools for inspecting the chemical knowledge and the case base. This requirement is crucial when the engineer is trying to design the solution for a problem that is not covered by the knowledge of the system.

The retention module is the responsible of storing all the problems solved and manages the indexing structure. Cases are stored in an external database and accessed by SQL queries.

3.2 Inference Process

The *T-Air* inference process has been implemented using *constructive adaptation* [7], a generative technique for reuse in CBR systems. A solution in *T-Air* is constructed by combining and adapting several previously solved designs. We use the input knowledge and chemical knowledge (stored in the chemical case-models) as the basis for determining the similarity between a new problem and those treated previously and stored in the case base. The Chemical Model and the Equipment Model are used in the adaptation stage for assessing the working parameters of each equipment of the flow-sheet. The design of a solution in *T-Air* is organized in four task levels:

a) selecting the class of chemical process to be realized;
b) selecting the major equipments to be used – and their inter-connections;
c) assessing the values for the parameters of each equipment; and
d) adding auxiliary equipment.

Task levels a) and b) are mainly related with retrieval mechanisms. Tasks levels c) and d) are mainly related with adaptation mechanisms.

The innovation problem in *T-Air* is due to the continuous improvements on the equipment used in the design of gas treatment plants. Specifically, the innovation in the scrubbers (the gas washing elements) that are the core elements in a gas treatment plant. It is not usual to incorporate new models of scrubbers. The usual procedure is to apply innovations to the current models. For instance, an innovation on the scrubber cover can decrease the pressure drop– i.e. increase the washing efficiency. Moreover, because there are many washing parameters estimated experimentally, the behavior and the in-site measurements of the deployed gas treatment plants is also a source of knowledge continuously incorporated into future designs. In *T-Air*, the innovation problem decreases the quality of proposed solutions. Because the *T-Air* system has a revision step, the first consequence is that engineers have to spend more efforts in verifying and correcting the proposed solutions. Moreover, each time the engineer has to correct a lot of equipment parameters, the trust in the system decreases.

When designing gas treatment plants two classes of problems can be identified: customary designs and occasional designs. Customary designs – for instance the gas treatment inside wastewater treatment plants – are good examples for tracking the innovations introduced in scrubbers. Solutions for occasional designs have to be tuned with customary designs. Otherwise, the quality of a solution for an occasional design may become very low.

The innovation aware algorithm is only relevant for *T-Air* tasks levels b) and c): the selection of the major equipments to be used and the assessment of the values for the parameters of each equipment.

3.3 Maintenance Policies

As it was argued in section 3.1, due to the impossibility of covering all the chemical knowledge, we only modeled the knowledge required by the cases present in the case base. Thus, the first maintenance component was designed for refining the similarity knowledge. The maintenance of the chemical similarity hierarchy can be performed graphically by adding new leaves or refining intermediate nodes. Moreover, each node in the hierarchy has associated a set of *perspectives* that locally determine the set of 'important' input data. Perspectives [5] is a mechanism developed to describe declarative biases for case retrieval in structured and complex representations of cases. This mechanism is also very powerful for assessing similarities among case solutions where, as in design tasks, solutions are also represented as complex and structured representations. *T-Air* performs automatic subsumption tests for restricting the incorporation of a new perspective that can conflict with the pre-existing ones.

The second maintenance policy implemented in the *T-Air* system was the deletion of problems with obsolete solutions. Because we had all the designs performed by TECNIUM from 1989, some of the solutions provided are now obsolete. The system provides automatic tools for identifying divergent solutions for the same problems and options for manually eliminate cases (useful for problems with non-standard solutions). Moreover, the user can mark a new solution as a 'non-standard solution' and then, the solved problem will not be considered as a case.

The third maintenance policy in *T-Air* was incorporated in the adaptation knowledge. One of the components of the adaptation knowledge is the list of obsolete components. The engineers manually introduce the new components and their corresponding obsoletes. Unfortunately, only auxiliary equipment become obsolete.

Finally, the innovation aware algorithm was introduced for dealing with the improvements of gas-washing units. In the next section we will describe the experiments performed for assessing the use of the innovation aware algorithm in *T-Air*.

3.4 Experimentation

The goal of the experimentation was to compare the quality of solutions provided by *T-Air* using the innovation aware algorithm regarding the correct solutions

Table 1. Summay of experimentation results.

	Main Equipments		Parameter Ass.	
	Inn.	err.	Inn.	err.
Standard Algorithm	0.0 %	25.0 %	0.0 %	83.3 %
IN-AW Algorithm	21.6 %	8.3 %	75.0 %	18.3 %

and the solutions proposed without the innovation aware algorithm. We say that the quality of a solution is preserved when the solution proposed by the innovation aware algorithm prevents the manually revision of the solution.

For testing the performance of the innovation aware algorithm we experimented with a case base of 900 design solutions. These 900 cases are the performed installations since 1997 (the last seven years). Then, the worst situation for the *T-Air* system is to propose a new solution for an occasional problem based on a solution of 7 years old.

From the case base we selected as testing problems 60 occasional designs – 20 from 2001, 20 from 2002, and 20 from 2003. All the testing problems had as similar cases designs with at least 4 years old. An important remark is that we tested each problem using only the cases older than the problem – i.e problems from 2001 were tested only with cases from 1997 to 2001.

We analyzed the quality of the solutions at two different task levels: the selection of the main plant components and structure (task level b) and the selection of equipment working parameters (task level c). Errors performed at task level b) are critical because the system is proposing an inappropriate type of solution. Errors performed at task level c) are less critical for the chemical process involved, but they increase the analysis cost because the design has to be revised by a more qualified engineer.

Performance at 'Main Equipments' Task Level. The `IN-AW` algorithm proposed a different design solution for 13 problems (representing the 21.6 %). Two of the proposed solutions were not justified by the case base content. After the experimentation, this situation was easily corrected by refining the equipment model. Moreover, there were three problems with better solutions that were not identified (the error percentage representing both situations is 8.3 %). When *T-Air* is not able to identify the best solution for a given problem, the inference algorithm behaves like in the standard CBR algorithm. The improvement of quality of solutions at this level was not high but this is coherent with the fact that the design improvements occur mainly in the equipment parameters.

Performance at 'Parameter Assessment' Task Level. The important difference between the `IN-AW` algorithm and the standard CBR algorithm arisen in this task level. `IN-AW` proposed different parameter values in 45 of the test problems (representing the 75 %). All the proposed parameter values were correct improvements – comparing with the standard CBR algorithm. The error of 18.3 % arose when comparing the differences with the real solutions. The difference is motivated by the conservative policy we chose when reusing

innovation variants: only those innovation variants with a high confidence degree are applied. The consequence is that in those cases the solution proposed by the `IN-AW` algorithm represents an intermediate point between the solution provided by the standard CBR algorithm and the engineer solution – i.e. the quality of the solution was improved but the optimal was not reached.

4 Conclusions

We presented a CBR algorithm for dealing with problem domains with evolving solutions concentrating the efforts on the retrieval and the reuse steps. The algorithm has been developed for design and configuration tasks, where a solution has a complex and structured representation and is usually constructed with the contribution of different cases. The *innovative aware* CBR algorithm manages the periodically incorporation of innovations in the problem solutions avoiding the decreasing of the system performance. The solution provided incorporates an additional retrieval and adaptation step into the usual CBR inference cycle for capturing the innovations performed in customary problems and applying them to the solutions of occasional problems.

We incorporated the innovation aware algorithm in *T-Air*, a deployed application for aiding engineers in the design of gas treatment plants. The experiments we performed with occasional problems demonstrated the utility of the algorithm: the quality of the solutions provided by *T-Air* were improved. This effect was clearly significant at the 'Parameter Assessment' task level where the percentage of solutions improved is the 75 %.

When the performance of the system degrades, the psychological effect on the users is the lack of trust. An important contribution of the algorithm is that increased the trust of the users in the *T-Air* application.

In the current implementation of the `IN-AW` algorithm we have manually tuned the values of the recency thresholds δ and γ for their use in the *T-Air* application. As a future work we plan to investigate the use of variable recency thresholds.

Although the use of learning techniques for acquiring adaptation knowledge is a hard task in incomplete domain models, it can be an interesting complement to our proposed algorithm. Future research will continue in this direction.

Acknowledgements

The author thanks Ramom López de Mántaras and Enric Plaza for their helpful and stimulating suggestions in the elaboration of this research. The research reported in this paper has been partially supported by the Spanish Ministry of Science and Technology under the project TIC 2003-07776-C2-02, EU-FEDER funds, and the company TECNIUM. Any possible mistake or error in the description of the working principles of chemical gas treatment is the sole responsibility of the author.

References

1. Agnar Aamodt. A computational model of knowledge-intensive learning and problem solving. In Bob Wielinga et al., editor, *Current Trends in Knowledge Acquisition*. IOS Press, 1990.
2. Agnar Aamodt. Knowledge-intensive case-based reasoning and sustained learning. In Luigia Aiello, editor, *Proceedings of the 9th European Conference on Artificial Intelligence*, pages 1–6. Pitman Publishing, 1990.
3. Agnar Aamodt and Enric Plaza. Case-based reasoning: Foundational issues, methodological variations, and system approaches. *Artificial Intelligence Communications*, 7(1):39–59, 1994. online at http://www.iiia.csic.es/People/enric/AICom_ToC.html
4. Josep Lluís Arcos. T-air: A case-based reasoning system for designing chemical absorption plants. In David W. Aha and Ian Watson, editors, *Case-Based Reasoning Research and Development*, number 2080 in Lecture Notes in Artificial Intelligence, pages 576–588. Springer-Verlag, 2001.
5. Josep Lluís Arcos and Ramon López de Mántaras. Perspectives: a declarative bias mechanism for case retrieval. In David Leake and Enric Plaza, editors, *Case-Based Reasoning. Research and Development*, number 1266 in Lecture Notes in Artificial Intelligence, pages 279–290. Springer-Verlag, 1997.
6. David B. Leake and David C. Wilson. Combining cbr with interactive knowledge acquisition, manipulation and reuse. In *Proceedings of the Third International Conference on Case-Based Reasoning (ICCBR-99)*, pages "218–232", Berlin, 1999. Springer-Verlag.
7. Enric Plaza and Josep Ll. Arcos. Constructive adaptation. In Susan Craw and Alun Preece, editors, *Advances in Case-Based Reasoning*, number 2416 in Lecture Notes in Artificial Intelligence, pages 306–320. Springer-Verlag, 2002.
8. Thomas Reinartz, Ioannis Iglezakis, and Thomas Roth-Berghofer. Review and restore for case-based maintenance. *Computational Intelligence*, 17(2):214–234, 2001.
9. Michael M. Richter. Introduction. In M. Lenz, B. Bartsch-Spörl, H.D. Burkhard, and S. Wess, editors, *CBR Technology: From Foundations to Applications*, pages 1–15. Springer-Verlag, 1998.
10. Barry Smyth and Elizabeth McKenna. Competence models and the maintenance problem. *Computational Intelligence*, 17(2):235–249, 2001.
11. David C. Wilson and David B. Leake. Maintaining case-based reasoners: Dimensions and directions. *Computational Intelligence*, 17(2):196–213, 2001.