Extending Case-Based Reasoning with Traces

Amélie Cordier, Bruno Mascret, Alain Mille
Université de Lyon, CNRS,
Université Lyon 1, LIRIS, UMR5205, F-69622, France
{firstname.lastname}@liris.cnrs.fr

March 30, 2009

Abstract

One topical challenge in knowledge engineering is to build systems that can ease the sharing and re-using of experience between a large community of users. This requires a certain plasticity of systems and associated interfaces. Yet, the lack of flexibility is a limitation of current case-based reasoning (CBR) systems. They are often developed in order to solve one particular problem and are not designed to be changed by users depending on the use they want to make of them. In this paper we propose to use interaction traces as a knowledge source for CBR systems and we show how it allows us to drive back the current limits of CBR.

1 Introduction

The fundamental principle of CBR relies on a memory of past experiences (problem solving episodes) instead of a set of rules to solve a new problem. This approach has major well known advantages: CBR systems do not build solutions from scratch, instead they use previous cases. Hence they do not need a full domain theory to reason. Therefore they start working with a small knowledge base, they learn and improve with each problem solving experience, etc. For all these reasons, CBR is considered as a powerful reasoning paradigm and easy to set up.

However, despite these advantages, CBR suffers from re-engineering problems and making a CBR system evolve is always difficult. For example, cases are instant snapshots setting a past experience in a given structure that is not designed to evolve. If one wants to make an experience's representation evolve, for example to take more context into account, one is often limited by the constraints inherent to the case's structure. This lack of flexibility of the knowledge representation is with no doubt a CBR limitation.

We believe it is possible to dramatically extend the possibilities of CBR by giving more malleability to the system, and we are convinced that a trace-based approach would allow us to do so. Indeed, interaction traces used as

continuous records of activities still have "contextualising" problems solving experiences because of their own nature.

We will show in this paper how a trace-based approach (introduced in section 3) could strongly enhance the possibilities of CBR. Section 4 explains how to merge interaction traces and CBR paradigms to take a step towards a general CBR architecture based on traces. We illustrate the benefits of this approach by proposing a theoretical example in which a limitation of the CBR principle is overcome thanks to reasoning traces. Section 5 gives an overview of some of the challenges created by such an approach and discusses several research issues. Section 6 studies the possible applications of our approach in several domains covering different problems with different goals. Finally, a discussion on this contribution concludes the paper in section 7.

2 Case-based reasoning issues

The achievement of CBR is partly due to the fact that it constitutes a solution, at least partially to the knowledge acquisition issue in knowledge based systems. Indeed, CBR systems acquire new knowledge by accumulating cases representing problem solving episodes, and memorise them in a predefined way so that they can be re-used (reused) to solve future problems. Cases thus constitute the main CBR knowledge container.

A case is usually made of a problem part and a solution part, each one being composed of a set of descriptors often defined in a dedicated ontology. A new problem is called a *target case*. Cases are compared on the basis of their problem part. The description of the problem part of a target case is compared to the problem part of each *source case* available in the case base. Differences observed between problem descriptors are then exploited by adaptation operators that are responsible for the adaptation of the solution part's descriptors. During this phase, the solution of a source case is adapted in a candidate solution for the target case.

Similarity knowledge, used to compare problem parts, is also adaptation knowledge [Cordier et al.2007]. When a candidate solution does not work in the context of the target case, it is possible to revise the case (resorting to external knowledge) and, consequently, to revise the system knowledge. Cases, revised or not, are always stored in the case base once they have been solved.

CBR is a good knowledge based system, quite successful in its applications [Leake, Kinley, and Wilson1996a], but its engineering remains a difficult problem. For example, maintenance operations remain tedious and require important knowledge re-engineering efforts. Classical responses to improve CBR engineering are in fact the same than those applied in classical design [Bergmann et al.1998]. Hence, CBR is not really a knowledge base-system that "learns how to solve problems by solving problems" because experience is not able to evolve dynamically. Here are some of the causes of this limitation:

- the **"too-well-defined-so-restricting"** case model: a case has to be fully

described, most of the time according to a static and rigid structure. This limitation is a variant of the classical "frame problem". It really cuts back on CBR's domain of competence.

- the **"active-but-frozen-knowledge"** paradox: the knowledge of a CBR system is assumed to be consistent, or invariant (solved cases, domain knowledge, similarity and adaptation knowledge). In such a situation, how is it possible to make the knowledge of the system evolve, while staying consistent and without being too demanding for the knowledge engineers?

- the simplistic **"good-for-me-now-so-always-good-for-all"** assumption: in non-trivial application, it is frequent that the question that one wants to solve changes radically. However, CBR systems are usually designed to solve one typical problem. How is it possible to create a system able to solve several problems, even if they have not been anticipated by the system designer?

Hence, the debate is open: how to design a CBR system able to take into account an unknown context, permanently evolving, and that doesn't need a costly re-engineering? As it is hopeless to foresee all the possible uses of a system, how to make systems able to adapt themselves to their users? In order to overcome these issues, is it possible to associate learning and problem solving in a revised reasoning cycle?

Our proposal is to go further than specifically collected and coded problem solving episodes, and to consider the interaction trace of an activity as a container in which various problem solving situations can be found. We propose an extended variant, in terms of knowledge representation of CBR that we call traced experience based reasoning (TBR). In TBR, the reasoning is still based on cases but cases are dynamically "built" from the trace for the reasoning purpose, they are no longer stored in a case base but are present in the "context" of the activity trace. The TBR cycle relies on the notion of trace of interaction, which is a record of an activity represented at an appropriate abstraction level. Traces are handled by a trace management system (see section 3).

During the **elaboration step**, a request for experience reusing is triggered. This request corresponds to an *explained case signature*, i.e. a pattern characterising the problem to be solved. This pattern is similar to the target case in CBR. Elaboration is guided by the system knowledge. It may be necessary to transform the trace such as it is described with the available vocabulary in the ontology used to express the request. The notion of trace transformation is very important (section 3). The request describes the problem part and the explained case signature indicates the expected solution descriptors. Consequently it also gives the constraints needed for the mining of similar patterns. In CBR, the elaboration step is often limited to a form filling process. In TBR, this step is the main one and it provides knowledge that will efficiently guide the reasoning. Additional knowledge can be added by the user depending on

the precise context of the request (specific case signature or specific similarity measure for example).

On the basis of the request (target case), several episodes found in the trace are considered as being similar to the signature elaborated by the request. The similarity is computed according to a similarity measure (taking into account adaptation knowledge) associated to the case signature and possibly specialised during the elaboration step. The **retrieval process** can be interrupted at anytime in order to define the request and to refine the similarity measure. This brings about performing a new elaboration and learning from failures of the retrieval process.

When the user (or the system) decides to use a previous episode, this episode is adapted exactly as cases are adapted in CBR. **Adaptation** rules are associated to each explained case signature. If the adaptation fails, then adaptation knowledge can be corrected during the **revise step**. It is often necessary to make a loop on the elaboration step in order to enrich the request context. The revised case is then proposed for the current activity and is "naturally" stored in the ongoing traces.

There is no specific **learning step** since learning is at the heart of the cycle. However, effective learning (i.e. memorisation of new knowledge built during the problem solving episode) is made concrete during the elaboration and during the revision step.

3 Traces

The concept of "trace" refers to a record of "something" that has occurred in the past. A trace is a "footprint": what remains of a phenomenon after it has ended. In computer sciences, traces are everywhere (log files, navigation history, versionning, etc.) and have been studied for several purposes [Diekert and Rozenberg1995]. Recently, several researchers have contributed to the elaboration of a *trace theory* [Settouti et al.2009]. According to this theory, a trace is a set of temporally and spatially situated elements that are inscribed in the environment during an activity. One should observe that a trace is inscribed intentionally or not.

The trace theory and its related work provide all the material needed to exploit digital traces: vocabulary, methods but also tools to manipulate them. According to this approach, traces are handled by the Trace-Based Management System (TBMS) which provides methods for collecting, transforming, storing, manipulating and visualising traces. These methods address a major challenge: how to build traces (by collecting a relevant set of objects) that will be reusable for a given purpose? The challenge is only partly addressed because the methods require models, and models have to be defined according to the target application. In order to tackle this issue, the trace theory defines the concept of `M-trace` (modelled trace) thus enlightening the fact that a trace and its model are indivisible. `M-traces` can be processed by transformations tools to filter, merge or reformulate traces.

Traces are sets of objects collected from a primary trace (a raw source of data) in which one could retrieve "episodes", i.e., records of previous situations by using manipulation and transformation methods. The TBMS is a general tool dedicated to the management of traces; it can be used for various purposes. Here, we consider using the features offered by the TBMS to increase the CBR possibilities. We advocate that introducing the concept of interaction trace in CBR is "not so complicated" a solution to better integrate context to CBR systems and thus to facilitate the reuse of experience.

4 Extending CBR with traces: an out of the box example

This section describes the advantages of our approach first by giving a theoretical example in a fictive application domain: *FLATLAND*. This example is based upon the novel of Edwin A.Abbott [Abbott1884]. It has been used firstly in Cordier's IAKA approach [Cordier2008] and followed up by Mascret [Mascret2008].

In this domain, problems consist in ordered shapes and solutions consist in shapes. Shapes have two properties: number of edges and colour. There is no available rule allowing the immediate computation of the solution given the knowledge on the problem.

In the IAKA[1] approach, Cordier describes a decomposable and differential process of adaptation shown on figure 1: each (similarity) difference ($\mathtt{r}_i$) between source (`srce`) and target (`tgt`) problems is interpreted with a specific adaptation operator ($\mathtt{r}_i, \mathcal{A}_{\mathtt{r}_i}$) to produce step by step a candidate solution ($\widetilde{\mathtt{Sol}}(\mathtt{pb}_1)$ or $\widetilde{\mathtt{Sol}}(\mathtt{tgt})$) from the preceding solution (resp. $\mathtt{Sol}(\mathtt{srce})$ or $\widetilde{\mathtt{Sol}}(\mathtt{pb}_1)$). In other words the IAKA approach links directly and explicitly similarity and adaptation knowledge in the CBR process.

The domain behaviour described on figure 1 is "having a child". In *FLATLAND*, a child (solution) is computed by following two rules: the solution has one more edge than the first parent shape, and has the colour of the second. We remind that the CBR system doesn't know these rules and this behaviour. Thus we have a CBR system able to represent the domain knowledge "having a child" trough similarity and adaptation knowledge.

Let's now assume that another behaviour exists in *FLATLAND* but has not been yet implemented (or seen) by the CBR designer. This behaviour may be "fighting" for example. Rules associated with "fighting" differ from "having a child": the first shape wins the fight but looses an edge. Figure 2 shows the adaptation path to be followed to produce the candidate solution. The common answer to this kind of problem in CBR is to perform a new complete cycle of knowledge engineering including at least new domain knowledge. Depending on the models used, there are several and not exclusive possibilities: new problem definitions, new similarity knowledge and/or measures, new

[1] InterActive Knowledge Acquisition

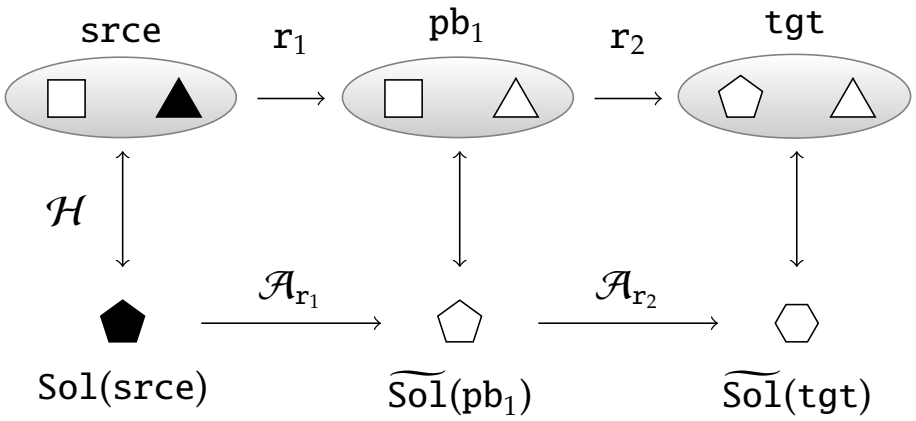

Figure 1: An adaptation path in IAKA. The $\mathcal{H}$ relation (vertical) represents the link between problem parts (below) and solution parts (above). The first step takes into account the similarity difference $\mathtt{r}_1$ (second shape's color is different) and proposes an intermediate candidate solution $\widetilde{\mathtt{Sol}}(\mathtt{pb}_1)$ by adapting the source case solution $\mathtt{Sol}(\mathtt{srce})$ thanks to the adaptation function $\mathcal{A}_{\mathtt{r}_1}$ (change the colour of the solution). The pair $(\mathtt{r}_1, \mathcal{A}_{\mathtt{r}_1})$ is called an adaptation operator. The next step uses the same principle: $\mathtt{r}_2$ (one more edge on first shape) implies $\mathcal{A}_{\mathtt{r}_2}$ (one more edge on the solution).

adaptation knowledge. These operations are costly, often long and need the involvement of both designers and domain experts. The final user has to wait until improvements are published and then hope not to encounter another unsupported behaviour.

However the existing knowledge is not so bad. It is only partial because it doesn't know the "fighting" behaviour. The final user is stuck and has no choice but to wait. But he may know *WHY* the CBR process failed. Thus the question is: "Why not offer him the possibility to explain this knowledge to the system?". We are going to show that it depends only on a constraint release or wide interpretation of it.

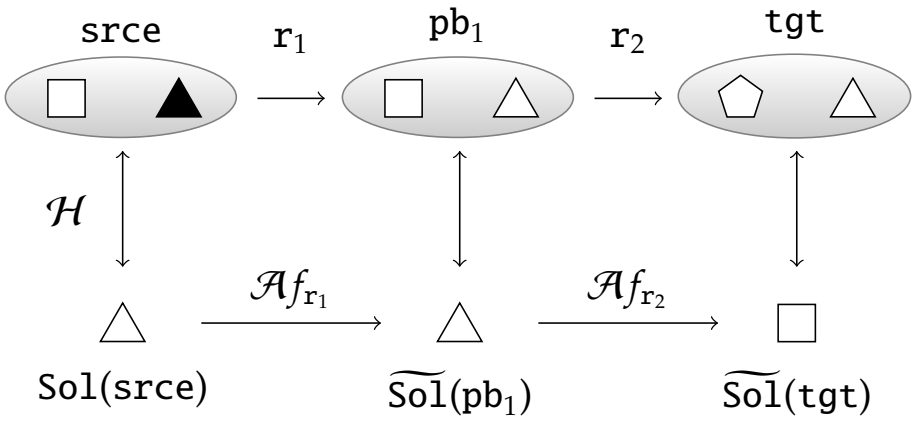

Figure 2: An adaptation path for the "fighting" behaviour. Source and target problems are the same as in the previous adaptation path (figure 1) but solution parts differ from the "having a child" behavior. Adaptation operator $(r_1, \mathcal{A}f_{r_1})$ means "a difference of colour between second shapes doesn't change the solution colour" and $(r_2, \mathcal{A}f_{r_2})$ is the same that $(r_2, \mathcal{A}_{r_2})$.

CBR main assumption is that "similar problems have similar solutions". "Having a child" problems are different from "Fighting" problems but their descriptions are the same for the moment. What would happen if we assumed that a problem may have different solutions according to the context?

We have to allow our CBR system to propose several solutions and their corresponding context. Then the system requires to ask the user which context is running. A candidate solution can be build thanks to this information. Moreover the only difference between our two examples is the new adaptation operator $(r_1, \mathcal{A}f_{r_1})$. It can be discovered by adapting the $(r_1, \mathcal{A}_{r_1})$ to the first shape. In the case of trivial "adaptations of adaptation methods", a quite autonomous user can produce the solution himself without waiting the designer improvements. Moreover he can then share this solution by giving his new adaptation operator to other users and to designers. This is a feature closed to

Leake's approach used in DIAL [Leake, Kinley, and Wilson1996b].

This possibility provides new advantages compared to a classical CBR system. However, the system is now dependent of the user's choice and has not learnt how to resolve context changes. The user has to explain it each time a problem has different solutions. We will show now that traces can be a good answer to this interactive context learning. Figure 3 represents a `M-trace` part corresponding to an activity in *FLAT LAND*. Episodes (cases) have been discovered by using trace transformations and similarity analysis. This `M-trace` contains not only the cases but other observations (circles, diamonds and stars). Our assumption is that context knowledge may be found by the user by interacting with this `M-trace`. In our example, the star stands for a war declaration. The exploitation of this observed item has not been taken into account yet, but the user may indicate to the system this information is important to automatically solve the context ambivalence. He only needs to explain it once and this new knowledge can be learnt immediately by the reasoning system.

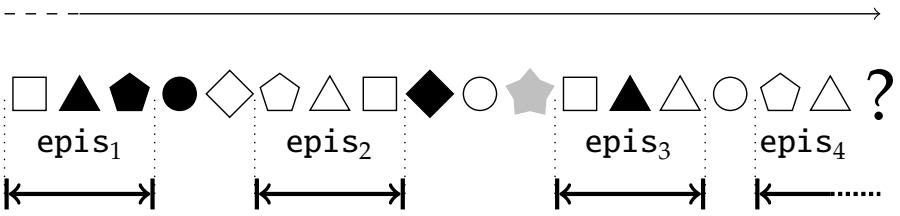

Figure 3: A trace with a change of context: the grey star indicates a context mark which can be used to resolve ambiguity in the adaptation process for computing episode 4 solution.

Another possible way is to see the last adaptation of episode. As a trace is temporally ordered, temporal logic [Allen1984] can be used to identify the context. In our example episode 3 has been adapted using the "fighting" behaviour. As episode 4 follows, the user may be able to specify a temporal context knowledge rule, like *if preceding episode adaptation contains $\mathcal{A}f_{r_1}$ then use "fighting" adaptation operators*.

This simple example shows what kind of improvements could be made using traces as case supports. The *too-well-defined-so-restricting* limitation can be avoided by defining a new context knowledge. The trace provides a concrete support to the user and gives him the possibility to directly explain to the system what it needs to resolve ambiguities and how to do so. This acquisition of new knowledge has immediate repercussions that get the reasoning process to evolve lively. The *active-but-frozen-knowledge* is not set anymore and catches easily the user expertise.

A lot of contextfull domains may benefit from this interactive approach of CBR. The main problem in this area is the multiplicity of point of views corresponding to multiple users. TBR get over the *good-for-me-now-so-always--good-for-all* assumption and can be used to fit with heterogeneous behaviour. It

offers the possibility to CBR to solve less structured problems and to be applied to changing domains. We detail more these aspects in the following sections.

5 Research issues

In order to build a functional CBR system based on traces, several issues have to be tackled. These issues are briefly described below and they constitute our road map for future work.

From cases to traces: retrieval and adaptation: this issue is raised by the fact that, in such an approach, cases are neither stored anymore, nor organised in a given structure. They are only implicitly present in the trace. Hence, before being able to perform any CBR, one needs to build "cases" from the trace. This underlines both the importance of the elaboration in TBR's retrieval step and the great advantage that allows a user to see directly the result of his request. The reflexivity of the tracing process gives him the possibility to identify immediately an unsuitable similarity measure. He may decide to stop the process here. In this case, the TBR systems should let him choose to refine the measure, try or adopt another one, or he may continue the cycle because the problem seems to have a better resolution further on with other kind of knowledges (adaptation or context ones). The automation of adaptation has raised a more complicated issue: we know how to adapt in context, but specific context and adaptation knowledges must be stored somewhere, particularly when they are shared by several users. Moreover, the different kinds of knowledge are really linked in TBR and we have to let them exchange naturally.

"Horizontal" and "vertical" context extensions: this process has to be more defined to overcome the *too-well-defined-so-restricting* problem issue. When a problem cannot be solved with a retrieved episode, this episode has to be extended (for example by seeing what happened "before", "during", or by using other temporal Allen's relations). But context extension must not be limited to horizontal (temporal) analysis. Comparison and adaptation of episodes is made possible through transformations of traces. In order to be compared, traces sometimes must be generalised and thus produce several levels of abstraction. This vertical organisation corresponds to a trace provenance history[Leake and Dial2008]: navigation between transformation levels (a top-down approach) could help the user to formulate this context by showing him what happened to the `M-trace` above. Then he could tell the system which transformation is wrong (an observed item could have been merged with another one or simply be skipped by the transformation). This approach is now bottom-up.

Towards a CBR assistant: the user's skills and involvement play an important part in this process. We have to think how to support him in these operations, and to help him to determine when context must be extended. TBMS's role should not include this kind of feature to preserve him to become dependant of the application domain. That's why we prefer to focus on the designing of TBR assistants. One great challenge would be to achieve the spec-

ification of generic interfaces between TBMS, assistants and viewers. Common task for assistants should include replay of retrieved episodes or solutions, tools for identifying what is relevant or not, appropriation of the system by taking into account the variety of the users and their modalities of interaction. Assistants would support too the sharing of traces between users. They should be considered as operators able to drive the TBMS in a generic way.

6 Related work and applications

A certain number of researches concern computer traces for software behaviour diagnosis [Diekert and Rozenberg1995] or for human behaviour diagnosis [Penelope and Fisher1994]. As far as we know, if some of these researches focus on sequence mining in traces [M.Gaber, A.Zaslavsky, and Krishnaswamy2005], none of them use traces in order to reuse previous experiences by adaptating their context. Studying the notion of "point of views" [Karacapilidis, Trousse, and Papadias1997], and the concept of "conversational CBR " [Aha, Breslow, and Munoz-Avila2001], has given importance to the elaboration step of the reasoning. This step relies on the system knowledge to help users to describe their problems. However, CBR systems do not use traces in order to assist users in this complex task, though it could be useful for example to easily extend the context of study.

TBR could be efficiently used in various types of applications: problem solving tools, assistives technologies, intelligent tutoring systems, etc. Application domains are also numerous and we believe that promising results could be obtained web-based applications and in social network tools such as described in [Briggs and Smyth2008].

7 Discussion

In this paper, we have proposed a new way of performing CBR which facilitates experience reuse in context and is more user-centered. This approach uses interaction traces. It is the result of a synergy between several researchers of our team[2]. Each one is working on different topics (CBR, traces, knowledge acquisition and knowledge management, assistive technologies, ergonomics) but all are focusing on one single question: *how to reason and to learn from experience?* This approach relies on a robust trace theory and on a set of tools allowing us to handle traces and to use them as CBR knowledge container. Traces enable us to overcome several limitations of traditional CBR systems described above.

The **"too-well-defined-so-restricting"** issue. Since cases are dynamically elaborated from the trace, they are always considered "in their context". Thus, it allows us to overcome the limitations of rigid case structures that do not allow a reliable representation of experience. Dynamic elaboration of cases is possible thanks to transformed traces and explained case signatures. The TBMS

[2]SILEX team (Supporting Interactions and Learning by EXperience), http://liris.cnrs.fr/silex

provides users reformulated traces. Each user can transform traces using his own knowledge for the problem he must to solve. With this approach, each user can have his own point of view on a problem.

The **"active-but-frozen-knowledge"** issue. Our interactive approach of CBR provides an opportunistic revision of similarity and adaptation knowledge. A failure of retrieval or of the adaptation is often related to a failure in the available knowledge. An interactive mining of the trace then allows to elaborate in collaboration with the user the necessary knowledge. The mining functionalities are provided by the TBMS.

The **"good-for-me-now-so-always-good-for-all"** issue. An experienced user might want to freely reuse his experience in the system. In order to do so, he must be able to design new explained case signatures. In order to ease his task, the system has to provide him customisable trace mining mechanisms using a set of constraints on the problem [M.Gaber, A.Zaslavsky, and Krishnaswamy2005]. Hence, explained case signatures (and their associated similarity measures) are co-constructed by the user and the system.

The combination of CBR and traces gives a central role to users. They are involved in elaborating a problem, in retrieving a previous experience, and in the adaptating the solution. As a result, they provide the system continuous knowledge acquisition abilities by elaborating pieces of knowledge during each interaction. Using traces leads to a new engineering of CBR systems. Users permanently "re-conceive" applications depending on their needs and their knowledge.

One of the challenges related to the use of traces lies on human-computer interactions. Visualising our own traces is easy. Traces are reflexive. We "know" what we have done and we are able to understand the meaning of our actions. However, using traces in order to "learn" from experience is a more complex task even more so if we do not limit the concept of "experience" to one's experience, but if we extend it to shared experiences. Indeed, a trace could be the result of several users' interactions. In a similar way, a user might want to benefit from traces coming from other users. In this context, we stress the importance of sense negotiation which has been introduced and developed by Stuber in [Stuber2007]. Moreover, learning from traces requires additional interfaces providing tools transforming the trace and interacting with it. Designing efficient interfaces supporting learning from experience is a guarantee of success for continuous learning CBR systems.

With the increasingly important place of software in our daily lives, human-computer interactions will increase in importance as well. Hence, the ability to support interactions in order to efficiently reuse experience is a major challenge for future systems. Traces enable us to envision more interaction modalities in systems and especially to combine several interaction modalities in a single system. Thus, various users with different abilities or habits will be able to share their experience. With our approach, we plan to experiment two main application domains: namely technology enhanced learning and assistive technologies. However there are many other application domains. We believe that trace-based reasoning will have a significant impact on experience sharing

applications, particularly when they are web-based.

We hope that, thanks to interaction traces, next generation CBR systems will be real experience-based reasoning systems and will be shared by communities of heterogeneous users.

References

[Abbott1884] Abbott, E. A. 1884. *Flatland, a romance of many dimensions*. E.Books, Second revised edition.

[Aha, Breslow, and Munoz-Avila2001] Aha, D. W.; Breslow, L. A.; and Munoz-Avila, H. 2001. Conversational CBR. *Applied Intelligence*.

[Allen1984] Allen, J. 1984. Towards a general theory of action and time. *Artificial Intelligence* 23(2):123–154.

[Bergmann et al.1998] Bergmann, R.; Breen, S.; Göker, M.; Manago, M.; Schmitt, S.; Schumacher, J.; Stahl, A.; Tartarin, E.; Wess, S.; and Wilke, W. 1998. The inreca-ii methodology for building and maintaining cbr applications. In *Proceedings of the 6th German Workshop on CBR*.

[Briggs and Smyth2008] Briggs, P., and Smyth, B. 2008. Provenance, trust, and sharing in peer-to-peer case-based Web search. In *Advances in CBR proceedings*, 89–103. Springer-verlag. 9th European Conference on CBR, Trier, Germany.

[Cordier et al.2007] Cordier, A.; Fuchs, B.; Lieber, J.; and Mille, A. 2007. Interactive Knowledge Acquisition in CBR. In *Proceedings of the workshop on Knowledge Discovery and Similarity (at ICCBR'07)*, 85–94. Workshop of the seventh International Conference on CBR Workshop.

[Cordier2008] Cordier, A. 2008. *Interactive and Opportunistic Knowledge Acquisition in CBR*. PhD Thesis, Université Claude Bernard Lyon 1.

[Diekert and Rozenberg1995] Diekert, V., and Rozenberg, G. 1995. *The Book of Traces*. World Scientific Publishing, Singapore.

[Karacapilidis, Trousse, and Papadias1997] Karacapilidis, N.; Trousse, B.; and Papadias, D. 1997. Using CBR for argumentation with multiple viewpoints. In *CBR Research and Development (ICCBR'97), volume 1266 of Lecture Notes in AI*, 541–552. Springer.

[Leake and Dial2008] Leake, D., and Dial, S. A. 2008. Using case provenance to propagate feedback to cases and adaptations. In *Advances in CBR proceedings*, 89–103. Springer-verlag. 9th European Conference on CBR, Trier, Germany.

[Leake, Kinley, and Wilson1996a] Leake, D. B.; Kinley, A.; and Wilson, D. 1996a. *CBR: Experiences, Lessons and Future Directions*. AAAI Press, MIT Press. chapter Learning to improve Case Adaptation by Introspective Reasoning and CBR, 185–196.

[Leake, Kinley, and Wilson1996b] Leake, D. B.; Kinley, A.; and Wilson, D. 1996b. Linking adaptation and similarity learning. In *Eighteenth Annual Conference of the Cognitive Science Society*.

[Mascret2008] Mascret, B. 2008. Apprendre à assister l'utilisateur à partir de l'expérience tracée ? Master's thesis, Université Claude Bernard Lyon 1.

[M.Gaber, A.Zaslavsky, and Krishnaswamy2005] M.Gaber; A.Zaslavsky; and Krishnaswamy, S. 2005. Mining data streams: a review. *Sigmod Rec*.

[Penelope and Fisher1994] Penelope, S., and Fisher, C. 1994. Exploratory sequential data analysis: foundations. *Human Computer Interaction* 9(3):251–317.

[Settouti et al.2009] Settouti, L.; Prié, Y.; Champin, P.-A.; Marty, J.-C.; and Mille, A. 2009. A TBMS framework: Models, languages and semantics. Research Report, http://hal.inria.fr.

[Stuber2007] Stuber, A. 2007. *Co-construction de sens par négociation pour la réutilisation en situation de l'expérience tracée* . PhD Thesis, Université Claude Bernard Lyon 1.