Proceedings of the Twenty-Sixth International Florida Artificial Intelligence Research Society Conference

Trace-Based Reasoning — Modeling Interaction Traces for Reasoning on Experiences

Amélie Cordier, Marie Lefevre,
Pierre-Antoine Champin, Olivier Georgeon and **Alain Mille**
University of Lyon, CNRS
LIRIS, UMR5205, University Lyon 1, France

Abstract

This paper addresses Trace-Based Reasoning (TBR) by using Case-Based Reasoning (CBR) as a descriptive framework. TBR is a reasoning paradigm in which inferences are made on specific objects called traces. Traces are sequential records of events observed and stored during an interactive process. We report two contributions. First, we propose a review of the current researches related to TBR. Then, we compare CBR and TBR. From this comparison, we show that the exploitation of traces instead of cases as knowledge sources raises very specific challenges. More precisely, new methods for defining similarity measures and for performing adaptation of traces are required. These new methods have to take into account the sequential properties of traces. We emphasis the benefits of using traces as a knowledge container in a reasoning process and we pinpoint promising applications of TBR.

Introduction

Human activities are increasingly mediated by digital environments, providing the opportunity to capture users' experiences in the form of interactions traces. Interaction traces could become potential containers of knowledge that could be formalized, shared and reused provided that we develop appropriate tools and methods to exploit these traces. Given the growing importance of traces in our everyday lives, developing such tools has become a major challenge. Trace-Based Reasoning (TBR) is a paradigm of Artificial Intelligence in which inferences are performed on specific objects called *modeled traces*. Modeled traces are defined as a temporally situated record of events observed during an interactive process.

The term Trace-Based Reasoning was introduced by (Mille 2006). TBR was then described as an extension of the traditional Case-Based Reasoning (CBR) cycle proposed by (Aamodt and Plaza 1994). The main motivation of such an extension was to overcome the limitations of predefined structures used to represent cases in CBR. In this initial paper, the author stated that, when using such predefined structures, it is very difficult to take the context into account during the reasoning process. As opposed to CBR, he described TBR as a solution to harness unstructured experience captured in traces. As a consequence, each experience is kept along with its context which is then easily accessible at reasoning time. Therefore, TBR was described as a generalization of the CBR paradigm in order to deal with dynamic reasoning. Over the last years, TBR has been subject of an increasing number of researches. The initial description of TBR has evolved and news concepts and methods have been defined. The first contribution of this paper is to provide a review of the current state of researches on TBR.

CBR and TBR can be related in many ways. We propose a comparison showing that CBR and TBR share a lot. However, because TBR uses knowledge stored in traces, new challenges arise. More precisely, classical similarity measures used in CBR do not apply anymore. New ways of computing similarity have to be defined in order to take into account the sequential properties of traces. In the same way, new strategies for adaptation of past experiences are to be defined. We show that using traces as a source of knowledge enables more rich and dynamic reasoning

This paper is organized as follows. First, we review current work on Trace-Based Reasoning. We illustrate our definitions with an existing application implementing TBR. Next, we report our comparative study of CBR and TBR. We pinpoint the specificities of TBR and the challenges raised. Then, we discuss related work. In the last section, we draw conclusions and describe our plans for future work.

Trace-Based Reasoning

This section reports a review of the current advances in Trace-Based Reasoning research. More details can be found in (Mille 2006; Georgeon et al. 2012; Cordier, Mascret, and Mille 2009; Settouti et al. 2009; Cordier, Mascret, and Mille 2010). To illustrate the definitions, we draw our examples from the Kolflow project.[1] Kolflow is a web-based environment providing users with a collaborative interface to edit and enrich knowledge bases. Traces are collected when users interact with the system and are used to provide them with a suitable assistance (Champin et al. 2012).

[1]This work is supported by the French National Research agency (ANR) through the KolFlow project (code: ANR-10-CONTINT-025), part of the CONTINT research program. http://kolflow.univ-nantes.fr

Traces, Models and M-Traces

A **trace** is commonly defined as a set of temporally situated elements. Many sets of records meet this general definition, for example, log files and RSS feeds (Rich Site Summary). In the absence of explicit models, however, such traces can only be exploited through ad-hoc processes, implemented for a specific purpose and not reusable. Associating a model with a trace makes it possible to perform inferences on them.

The **model of a trace** is the formal description of the structure and the content of a trace. The model provides information about the properties of the trace (e.g. time unit used, max length, etc.), about the elements of the trace (e.g. obsels, see below), and about the relations between the elements, including the temporal relations (e.g. time stamp, interval, order, etc.).

An observed element, called **obsel**, refers to an element of the trace. An obsel is the digital counterpart to an event that occurred in the real world. We deliberately chose to use the term obsel instead of event to insist on the fact that obsels are stored in the trace on purpose. Obsel types are formally described in the trace model. Each obsel type is characterized by a name, a timestamp, and a set of properties (usually attribute-value pairs, but can be more complex).

An **M-Trace** (for Modeled Trace) is a trace associated with its model. Figure 1 gives an example of an M-Trace in Kolflow. On the left is an excerpt from the trace model. We see that we can find at least two types of obsels in the trace: *click* and *typeText*. On the right is the trace. Here, the trace shows that the user has clicked on the field ”search”, had typed the text ”apple”, and had clicked on the button ”Find”.

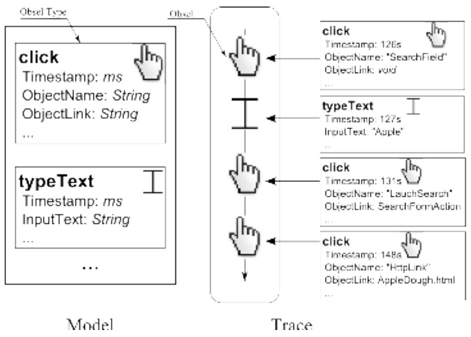

Figure 1: An example of M-Trace.

The **meta-model** of traces defines the properties that all the M-Traces must satisfy and ensures the interoperability between the traces. More details are available in (Settouti et al. 2009).

Processing traces

A **Trace-Base Management System** (TBMS) provides a set of tools to handle M-Traces. Figure 2 shows how M-Traces are stored and manipulated in the TBMS. The symbols in the traces represent different obsels chosen arbitrarily. Operations involved in the processing of traces are described below.

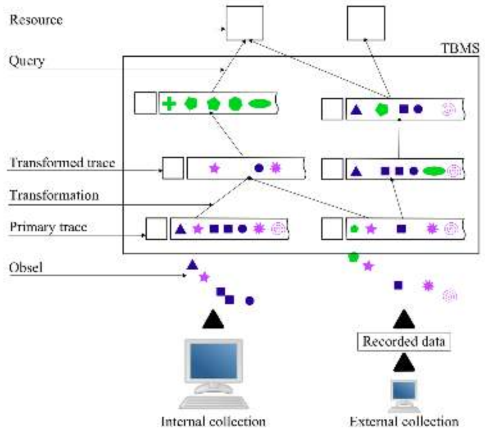

Figure 2: Storage and manipulation of M-Traces in a TBMS.

The **collection process** gathers obsels into *primary M-Traces* and stores them in the TBMS. We distinguish internal collection methods from external collection methods. **Internal collection methods** implement the collection process directly inside the observed system. **External collection methods** construct obsels from previously recorded data (e.g., log files). This method is useful when we want to collect traces from a system that we cannot modify.

A **query** operates on one or several M-Traces and produces a result of any kind (e.g. a value, a set of values, a text, an object, a graphical representation, an ontology, a visualization interface, etc.). The result of a query is not stored in the TBMS. It is called a *ressource*. However, it is always possible to keep a link between the result and the M-Traces it comes from (e.g. to keep provenance information).

A **transformation** applies to one or several M-Traces and produces an M-Trace. An M-Trace produced by a transformation is called a *transformed M-Trace*, to distinguish it from a primary M-Trace. That is, transformed M-Traces are constructed from previously existing M-Traces all the way down to primary M-Traces. The resulting transformed M-Trace is stored in the TBMS. Transformations enable the user to perform various manipulations on M-Traces, such as merging several M-Traces together, filtering an M-Trace, reformulating an M-Trace, etc. Transformations use *transformation knowledge* (TK). Transformations are stored in the TBMS (but they are not represented on figure 2).

A transformation is **deterministic**, meaning that it gives the same result each time it is run with the same input. With this mechanism, it is always possible to reproduce a transformation. It therefore provides a smooth and easy access to provenance information of each M-Trace. Notably, transformation knowledge may evolve over time. Handling this evolution is a real challenge. It is discussed at the end of this paper.

The **transformation graph** represents the relations between M-Traces. In this graph, nodes represent M-Traces, and edges represent transformations from M-Traces to other M-Traces. The transformation graph is a central feature of the TBMS enabling users to navigate between M-Traces. This feature is a consequence of the deterministic properties of traces enabling to keep track of provenance information. Provenance information is valuable, particularly when the user needs more context information.

In Kolflow, the collection process is implemented inside the web environment, it is therefore internal. It would have been possible to use the log files of the web server to collect traces through an external collection process, but this method gives less information about the user interactions. As stated above, Kolflow traces record interactions of the user with the web environment (click, typeText, etc.). A query gives, for example, the number of clicks a user made during a given period. Transformations are used to exploit traces in many ways. For example, transformations are used to rewrite traces at a more ”user-friendly” level. By applying such a transformation to the trace presented in figure 1, we get a trace containing a new obsel which is ”Search for apples”. As stated above, a transformed trace is also an M-Trace and therefore has its own model.

Dynamic and interactive reasoning

TBR is defined as a reasoning paradigm that produces new knowledge by performing inferences on modeled traces. We claim that, given the properties described above, TBR enables a more dynamic and interactive reasoning process. A TBR process usually follows three steps: elaboration, retrieval, and reuse.

The goal of the **elaboration step** is to produce an episode signature. An **episode** is an M-Trace representing a given task or experience. An **episode signature** is a specification of the constraints that an episode must satisfy (pattern, duration, etc.). There are many ways of expressing an episode signature (rules, sets of constraints, M-Trace and constraints, automata, finite states machines, etc.). The process of building an episode signature may vary according to the users needs. For example, if the goal is to retrieve a past situation based on its similarity with the current situation, in order to provide user assistance, then the episode signature can be created from a fragment of the current trace of the user. In this case, the user visualizes his or her current trace and specifies what elements are significant to generate the episode signature. In Kolflow, episodes signatures are defined in a different way. They consist of a subset of a model of a trace, and a set of constraints. If we want to retrieve all the episodes that correspond to a modification of a page over the last three hours, we build an episode signature as a transformation looking subsets of traces starting with an obsel of type ”edit page” and ending with an obsels of type ”save page”, with obsels of type ”typeText” in between.

Given an episode signature, the **retrieve step** consists in finding a set of episodes that can be used to solve the current problem. Constraints may apply on the number of episodes to retrieve, the search algorithm to use, the certainty with which the proposed solutions match the signature, etc.

The **reuse phase** uses the retrieved episode to tackle the situation at hand. During this step, the user uses query and/or transformation operations on the retrieved traces to exploit their content. Again, depending on the task addressed, there are many ways to reuse an episode. The system can simply display the episode to the user. The system can also replay the episode as a form of macro, or even possibly adapt the episode to replay it differently. The user can also apply specific queries on the episode to produce new knowledge (e.g. statistical information, information about users, etc.).

During all the reasoning process, TBR exploits traces and transformations stored in the TBMS, but also additional knowledge represented in the TBR knowledge base (similarity measures, adaptation strategies, etc.). The three reasoning steps of TBR are strongly interrelated. Moreover, users are in the center of the reasoning loop. They can interact with the reasoning process at any time and get immediate feedback through traces. They can go back and forth in the reasoning process if adjustments are needed.

Figure 3 gives an overview of an application implementing TBR from a knowledge point of view. The user interacts with a system (called *observed system*) to perform a given task. Interaction traces are collected and stored in the TBMS (figured by the thin arrows on figure 3). TBR relies on all the available knowledge bases: the TBMS consisting of traces and transformations, the observed systems knowledge base, if any, and the TBR knowledge base (similarity measures, adaptation strategies, etc.). Thick lines show where TBR applies. TBR can be used by the observed system to perform: task automation, recommendations, user assistance, traces visualization, etc. (see (1) on the figure). Moreover, TBR can be used to perform dynamic an online enrichment of the knowledge bases of the system (2). It can facilitate tasks such as classification, pattern matching, knowledge discovery and knowledge mining. Similarly, TBR can contribute to enrich external knowledge bases (3). This external knowledge can be used for different purposes, including reengineering of the whole system. Last but not least, users may learn new knowledge produced by the TBR process (4). This process is called reflexivity. See (Ollagnier-Beldame 2011) for more information.

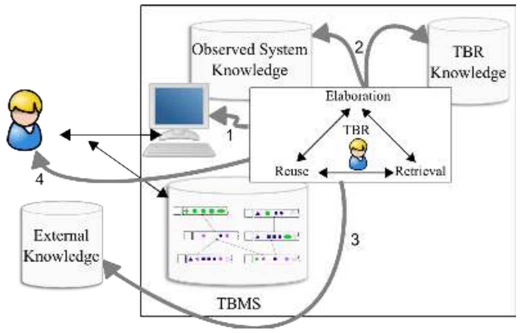

Figure 3: Implementing Trace-Based Reasoning.

Implementation of traces in Kolflow fits this framework. In Kolflow, traces are mainly used to provide user assistance. Traces of previous users are displayed to help new users performing complex tasks. Traces are also used to enrich the main knowledge base of the system. See (Champin et al. 2012) for a detailed example.

TBR from a CBR point of view

This section aims at comparing CBR and TBR. For that, we adopt a CBR point of view, and we follow the cycle proposed by Aamodt & Plaza in (Aamodt and Plaza 1994). We also take into account the knowledge containers used in CBR, as identified by Richter in (Richter and Aamodt 2005).

A comparison from the processes point of view

The first step of the traditional CBR cycle is to retrieve, among a case base a case (or several cases) similar to the *target case*. The target case figures the description of the problem for which we want to find a solution. Usually, a target case has the same structure as any other case, but the solution part is missing. In TBR, this definition does not apply. The nature of the ”target case” is very dependent of the task at hand (user assistance, knowledge acquisition, problem solving, etc.). For that reason, the notion of episode signature is introduced. An episode signature describes the properties of the episodes we want to find in traces as well as constraints on the retrieval process. The episode signature is the outcome of the elaboration step of the TBR process. This elaboration step, often avoided in classical CBR systems, becomes of major importance. As discussed in the previous section, several approaches can be used to build an episode signature, depending on the type of problem. These approaches have to take into account the characteristics of the traces, among which sequentiality, weak structure, and large number of elements. Furthermore, these approaches benefit from transformation mechanism provided by the TBMS.

Once the episode signature is elaborated, TBR can proceed to the retrieve step, as in CBR. In CBR, retrieval is based on the use of similarity measures that compare the target case to the source cases in order to determine what the most similar cases are. In TBR, similarity measures do not apply either, because it is not possible to compare an episode signature and a trace. Here, we need new mechanisms in order to find, in the trace, episodes that satisfy the given signature. Again, new challenges arise.

The second step of the CBR cycle is the reuse step, during which the retrieved case is reused, and possibly adapted is order to solve the current problem. The process is similar in TBR. The retrieved episode is reused to provide a solution to the current problem. However, given the large number of situations TBR can be applied to, the reuse step can be done in many ways. The episode can be display to the user, it can be transformed to achieve a particular goal, it can be adapted to match a particular context, etc. Depending on the way the episode is reused, different strategies apply and different knowledge containers are involved (visualization strategies, adaptation strategies, etc.).

CBR also implies a revise and a retain steps. During the revise step, the solution is corrected if it is not satisfactory enough. Once the solution is repaired, the case is considered as a solved case and is then memorized in the case base for future reuse. In TBR, these steps do not exist. Indeed, the whole reasoning process is traced. Therefore, the revision of possible solutions is traced as well, and naturally, the solution is memorized in the trace. However, additional indexing mechanisms can be implemented in order to improve retrieval efficiency later on.

A comparison from the knowledge point of view

In (Richter and Aamodt 2005), four knowledge containers are identified: vocabulary, similarity measure, case base and adaptation rules. He insisted on the fact that knowledge can be shifted between containers through a learning process or manually. In TBR, we identify the following knowledge containers: M-Traces, transformation knowledge (TK) and TBR knowledge.

M-Traces act as cases in the sense that they record problem solving experiences. The model part of the trace is similar to the structure of a case in CBR. It gives information on what one can expect in the trace and therefore enables the definition of inferences to be performed on traces. However, the main difference is that a case is usually represented according to a predefined structure, whereas in traces, experiences are not explicitly formalized in the trace, they are embedded among other contextual information. This makes experiences more difficult to retrieve as it is not anymore a matter of similarity measures between two structurally identical objects, but involves more complex operations. However, the benefits are manifold. First, it enables a better and easier access to contextual information (as the problem solving experience is kept in context) at any time. Next, it enables to dynamically build the definition of the problem solving experience we want to retrieve, which enables us to retrieve different kinds of experiences in the same set of traces.

Similarity measures do not really apply in the TBR context, even if they can be used as a subpart of the retrieval process. In order to retrieve episodes given an episode signature, TBR uses a retrieval strategy and retrieval knowledge stored in the TBR knowledge base. TBR can also exploit transformations available on traces for that purpose. As explained in the previous section, retrieval strategies are manifold (e.g. rules, automata, finite states machines, M-Traces, sets of constraints, etc.). Adaptation in TBR can make use of many types of adaptation knowledge. As for retrieval knowledge, adaptation knowledge can be available in the form of transformations, hence stored in the TBMS or in specific adaptation knowledge stored in the TBR knowledge base.

It is important to emphasis on the fact that, due to the dynamicity of the TBR process, users can interact with the reasoning at any time and thus, help with the enrichment of the different knowledge containers.

Related work

Automated management of sequential data is definitely an important topic and has already leaded to researches in many fields among which computer sciences, informatics, mathematics, human-computer interaction (HCI), and cognitive sciences. In these fields, terms such as sequences, footprints, traces, logs, protocols or streams, are used. Traces have been studied from a theoretical point of view, see (Diekert and Rozenberg 1995) for a review.

In the CBR field, several approaches make use of sequential information during the reasoning process. Leake used provenance information to improve reasoning and explanation in CBR (Leake 2010). Weber and Ontañon use annotated traces recorded when a human user played video games in order to feed a case-based planner (Weber and Ontañón 2010). In (Minor et al. 2010), authors report on their work on adaptation of workflow which can be considered as specific sequences. In (Gundersen 2012), the author studies the particular properties of sequences in order to design more efficient similarity measures. All these approaches implements forms of reasoning on traces.

Several studies have implemented TBR in real-world application or have tried to apply TBR principles to other research areas. Most of these studies have led to evaluations of the benefits of using a TBR approach. For example, Mathern *et al.* have applied TBR to Stream Mining. Stream Mining is the process of extracting knowledge from continuous records. They have shown that continuous records can be transformed into M-Traces. The outcome of a TBR process applied on M-Traces may be used to improve the results of the mining process. See (Mathern, Bellet, and Mille 2010) for a discussion. In (Georgeon et al. 2012), the authors report on the implementation of a trace-based system to model the car-driving activity from traces collected with an instrumented vehicle. They demonstrate how TBR can be used to facilitate activity analysis and modeling. TBR also proved to be an efficient tool for user assistance, as it is shown in (Cordier, Mascret, and Mille 2010). In this paper, the authors demonstrate that, first, traces are reflexive objects: users tend to find their own traces remarkably intuitive. As an example, Zarka *et al.* showed that merely replaying a trace provided an efficient form of assistance (Zarka et al. 2011). Second, traces can be shared between users, which facilitates experience sharing. Third, traces can be transformed, which make them usable at different levels and in various processes. Fourth, traces act as rich knowledge containers. They allow collection, management and restitution of knowledge. TBR for user assistance has been specifically studied in the case of human learning systems (Settouti et al. 2009). Intelligent tutoring systems and collaborative learning tools are, by nature, designed to provide assistance to learners. These tools often use different forms of traces as an input for the assistance. The Visu application, for example, provides a good example of the use of traces in a collaborative learning space (Bétrancourt, Guichon, and Prié 2011).

Discussion

In this paper, we reviewed recent research on TBR from a CBR point of view. TBR is defined as a reasoning paradigm that uses knowledge available into traces to perform inferences on these traces. TBR can be related to Schank's definition of CBR as an approach reusing past problems solving schemes to solve new problems in new situations (Schank 1982). Schank's notion of Dynamic Memory that reorganizes itself continuously and stores instances of events is appealingly similar to the notion of TBMS. The notion of episode can be reminiscent of Schank's scripts that are grouped according to MOPS (episode signatures) and explained by TOPS (stable domain knowledge). So while the definition of TBR does not fit well with the more recent, and widely accepted, definition of CBR (Aamodt and Plaza 1994), both approaches obviously share roots in Schank's seminal proposal.

Here, we argue that it is valuable to formalize a trace as a specific digital object (M-Trace) to facilitate its use during a reasoning process. Similarly, we argue that TBR can provide a handy framework to design and reuse trace-based reasoning systems. TBR offers valuable properties, notably: engineering of dynamic knowledge, dynamic reasoning and reflexivity. These properties are discussed below.

The originality of TBR is that it tackles the problem of **engineering dynamic knowledge**. In many approaches, knowledge is a mere by-product of the traces, abstracted away from the initial interactions from which that knowledge was drawn. Such approaches make it impossible (or very difficult) to explain or revise the used knowledge in an informed way. On the contrary, modeled traces offer an elegant solution to capture user experience. They act as a rich knowledge container. TBR makes it possible to dig as deep as required in the formation of any chunk of knowledge, or to "zoom out" in order to put it back in context. Therefore, TBR enables us to "keep experiences in context". As transformed episodes are always linked to their original primary traces, it is possible to retrieve, at any time, explanations on the provenance of an episode.

Furthermore, through the flexible mechanism of transformations, TBR enables **dynamic reasoning**. This feature can be related to the notion of Agile CBR introduced by Craw in (Craw 2009). In this paper, Craw suggested that "CBR should achieve more agility by exploiting the knowledge in cases in an opportunistic way to update the commitment to parts of an evolving solution". A similar idea was developed by Cordier in (Cordier 2008) for improving adaptation knowledge acquisition. In TBR, the underlying mechanism of transformations supports dynamicity of reasoning. In addition, TBR is not limited to the transformations envisioned during design time; new transformations can be applied to existing traces at any time to accommodate new requirements. This flexibility comes, of course, with a number of challenges. A first challenge is met by knowledge engineers, who have to carefully define trace models, so that they are rich enough to support multiple, and possibly unforeseen, uses. This requirement obviously meets technical and practical obstacles when it comes to collecting and storing such rich traces. A second challenge is related to transformations and their reproducibility: in order to ensure this desirable feature, it may be necessary to embed a large amount of transformation knowledge in the system, rather than relying on external sources, prone to change without notice. Whether those changes can be captured themselves as traces is of course an interesting question. Conversely, the status of different kinds of resources produced from traces through queries should be investigated. This raises the question of how to keep a link between those resources and the originating traces. It also raises the issue of the boundaries of TBR: does it include the production of such resources from traces, or is it restricted to producing and handling traces?

But the major challenge that TBR has to face is to enable **reflexivity**. While off-line processing of traces is relatively well understood, only through on-line integration with the user's activity will traces become first-class citizens in the design of applications. Just like databases are nowadays easily integrated in a system to provide storage and querying, we envision that traces should become as easily integrated to provide reflexivity and user assistance.

Evaluation of TBR is not in the scope of this paper. However, several projects have implemented TBR approaches and have provided several forms of evaluations, as it is pointed out in the third paragraph of the related work section. In future work, we plan to investigate how some CBR approaches using various forms of sequential data could fit (or not) with this framework, and what lesson can be learned from this study. We will focus in priority on the researches described in the second paragraph of the related work section.

This review suggests that TBR is an attractive solution to support dynamic reuse of previous experiences. TBR seems more suitable than CBR in various contexts, notably user assistance, where we often face unordinary problem-solving situations. It seems that TBR raises many challenges but also offers promising research directions.

Acknowledgments

The authors gratefully thank the reviewers of this paper for their constructive comments. They also thank Agnar Aamodt for his advice and comments on the preliminary version of this paper.

References

Aamodt, A., and Plaza, E. 1994. Case-based reasoning; foundational issues, methodological variations, and system approaches. *AI Communications* 7(1):39–59.

Bétrancourt, M.; Guichon, N.; and Prié, Y. 2011. Assessing the use of a Trace-Based Synchronous Tool for distant language tutoring. In *Computer Supported Collaborative Learning 2011*, 486–493.

Champin, P.-A.; Cordier, A.; Lavoué, E.; Lefevre, M.; and Skaf-Molli, H. 2012. User Assistance for Collaborative Knowledge Construction. In *WWW 2012 SWCS'12 Workshop*, 1065–1073. ACM DL.

Cordier, A.; Mascret, B.; and Mille, A. 2009. Extending Case-Based Reasoning with Traces. In *Grand Challenges for reasoning from experiences, Workshop at IJCAI'09*.

Cordier, A.; Mascret, B.; and Mille, A. 2010. Dynamic Case Based Reasoning for Contextual Reuse of Experience. In Marling, C., ed., *Provenance-Awareness in Case-Based Reasoning Workshop. ICCBR 2010.*, 69–78.

Cordier, A. 2008. *Interactive and Opportunistic Knowledge Acquisition in Case-Based Reasoning*. Thèse de doctorat en informatique, Université Lyon 1.

Craw, S. 2009. Agile case-based reasoning: A grand challenge towards opportunistic reasoning from experiences. In *Proceedings of the IJCAI-09 Workshop on Grand Challenges in Reasoning from Experiences*, 33–39.

Diekert, V., and Rozenberg, G. 1995. *The Book of Traces*. World Scientific.

Georgeon, O.; Mille, A.; Bellet, T.; Mathern, B.; and Ritter, F. 2012. Supporting activity modelling from activity traces. *Expert Systems, 29(3), 261-275*.

Gundersen, O. 2012. Toward measuring the similarity of complex event sequences in real-time. In Agudo, B., and Watson, I., eds., *Case-Based Reasoning Research and Development*, volume 7466 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg. 107–121.

Leake, D. B. 2010. Case-based reasoning tomorrow: Provenance, the web, and cases in the future of intelligent information processing. In Shi, Z.; Vadera, S.; Aamodt, A.; and Leake, D. B., eds., *Intelligent Information Processing*, volume 340 of *IFIP Advances in Information and Communication Technology*, 1. Springer.

Mathern, B.; Bellet, T.; and Mille, A. 2010. An Iterative Approach to Develop a Cognitive Model of the Driver for Human Centred Design of ITS. In *European Conference on Human Centred Design for Intelligent Transport Systems*, Proceedings of European Conference on Human Centred Design for Intelligent Transport Systems, 85–95. HUMANIST publications.

Mille, A. 2006. Traces Based Reasoning (TBR) Definition, illustration and echoes with story telling. Research Report RR-LIRIS-2006-002.

Minor, M.; Bergmann, R.; Grg, S.; and Walter, K. 2010. Towards case-based adaptation of workflows. In Bichindaritz, I., and Montani, S., eds., *Case-Based Reasoning. Research and Development*, volume 6176 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg. 421–435.

Ollagnier-Beldame, M. 2011. The use of digital traces: a promising basis for the design of adapted information systems? . *International Journal on Computer Science and Information Systems* (Speci).

Richter, M., and Aamodt, A. 2005. Case-based reasoning foundations. *The Knowledge Engineering Review* 20(03):203–207.

Schank, R. C. 1982. *Dynamic Memory: A Theory of Reminding and Learning in Computers and People*. New York, NY, USA: Cambridge University Press.

Settouti, L. S.; Prié, Y.; Cram, D.; Champin, P.-A.; and Mille, A. 2009. A Trace-Based Framework for supporting Digital Object Memories. In *1st International Workshop on Digital Object Memories (DOMe'09) in the 5th International Conference on Intelligent Environments (IE 09)*.

Weber, B. G., and Ontañón, S. 2010. Using automated replay annotation for case-based planning in games. In *ICCBR Workshop on CBR for Computer Games (ICCBR-Games)*.

Zarka, R.; Cordier, A.; Egyed-Zsigmond, E.; and Mille, A. 2011. Rule-Based Impact Propagation for Trace Replay. In Ram, A., and Wiratunga, N., eds., *International Case-Based Reasoning Conference (ICCBR 2011)*, LNAI 6880, 482–495. Springer-Verlag Berlin Heidelberg.