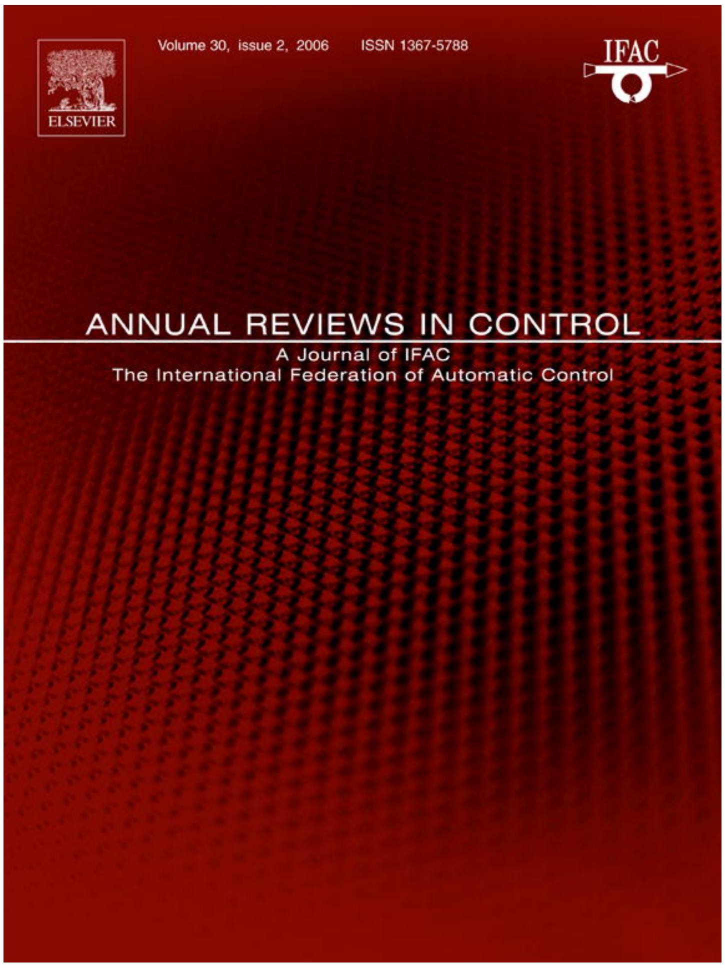
Volume 30, issue 2, 2006
ISSN 1367-5788
ELSEVIER
IFAC
ANNUAL REVIEWS IN CONTROL
A Journal of IFAC
The International Federation of Automatic Control

Annual Reviews in Control 30 (2006) 223–232

Annual Reviews in Control

www.elsevier.com/locate/arcontrol

From case-based reasoning to traces-based reasoning

Alain Mille

LIRIS UMR CNRS 5205, Université Lyon1, Insa-Lyon, Université Lyon2, ECL, France

Received 19 January 2006; accepted 19 September 2006

Abstract

CBR is an original AI paradigm based on the adaptation of solutions of past problems in order to solve new similar problems. Hence, a case is a problem with its solution and cases are stored in a case library. The reasoning process follows a cycle that facilitates ''learning'' from new solved cases. This approach can be also viewed as a *lazy learning* method when applied for task classification. CBR is applied for various tasks as design, planning, diagnosis, information retrieval, etc. The paper is the occasion to go a step further in reusing past unstructured experience, by considering traces of computer use as experience knowledge containers for *situation based* problem solving.

© 2006 Elsevier Ltd. All rights reserved.

Keywords: Problem solvers; Artificial intelligence; Knowledge-based systems; Knowledge representation

1. Context

This tutorial paper aims to present efficient technologies to amalgamate human operators and computer driven systems for tasks implying man–machine cooperation. The operator's experience and the computer's reliability have to be combined in a virtuous spiral. The ability to continuously capitalize on human experience while solving new problems in a technological context is what could be the key to softly amalgamate human operators and complex computer driven systems. The Case-Based Reasoning (CBR) paradigm, and further the Trace-Based Reasoning (TBR) approach are suitable for such a goal. In order to be able to understand the CBR AI paradigm, the paper recalls its cognitive foundations and details how it works technically. A number of useful pointers to technical papers will allow the reader to go further depending on her interest. This part of the paper borrows its formalization and especially its adaptation description from a common work between the team of LIRIS and LORIA (see Fuchs, Lieber, Mille, & Napoli, 2006). In the second part of the paper we take the opportunity to shortly present the TBR paradigm, a recent generalization of the CBR paradigm able to deal – with dynamic problem solving in episodes while CBR needs well-predefined context descriptions. In this way TBR-manages to dynamically build contexts, which can be very fruitful in man–machine cooperation for controlling complex systems.

E-mail address: alain.mille@liris.cnrs.fr.

1367-5788/$ – see front matter © 2006 Elsevier Ltd. All rights reserved.
doi:10.1016/j.arcontrol.2006.09.003

2. Case-based reasoning

2.1. CBR foundations

Minsky, Schank, Abelson and others gave general directions for reusing past problem solving schemes to solve new problems in new situations. In this paper, we focus on Minsky's and Schank's pioneering works. Interested reader will find biographical information about them in Crevier (1993).

2.1.1. Marvin MINSKY and ''stereotypes based reasoning''

Marvin Minsky argues that usual theoretical approaches in AI try to be too ''precise'', local and not really structured to face ''real world'' problems (see Minsky, 1975 for details[1]). He considers several approaches in Artificial Intelligence and Cognitive Psychology:

- The proposal, he made with Papert, to divide ''knowledge'' in structures they called ''microworlds'' (Minsky & Papert, 1974).
- The definition of ''problem spaces'' by Newell and Simon (1960).
- The expression of linguistic objects by Schank and Abelson (1977).

He describes these approaches as promising and contrasting to the classical approaches that attempt to describe knowledge

[1] http://web.media.mit.edu/~minsky/papers/Frames/frames.html.

as one set of unstructured knowledge pieces. Minsky proposes the notion of ‘‘Frame’’ as a convenient structure to support these theories. A frame is supposed to describe the ‘‘context’’ in which the reasoning process has to be done.

As Minsky explains ‘‘*Here is the essence of the theory: when one encounters a new situation (or makes a substantial change in one's view of the present problem) one selects from memory a structure called a Frame. This is a remembered framework to be adapted to fit reality by changing details as necessary.*’’

A frame has three main parts:

- A part about its use (its goal and context of use).
- A part about what can occur after using this frame.
- A part about what to do if there is an unwanted result after the frame's application.

> ‘‘We can think of a frame as a network of nodes and relations. The ‘‘top levels’’ of a frame are fixed, and represent things that are always true about the supposed situation. The lower levels have many terminals—‘‘slots’’ that must be filled by specific instances or data. Each terminal can specify conditions its assignments must meet. (The assignments themselves are usually smaller ‘‘sub-frames.’’) Simple conditions are specified by markers that might require a terminal assignment to be a person, an object of sufficient value, or a pointer to a sub-frame of a certain type. More complex conditions can specify relations among the things assigned to several terminals’’ (see Fig. 1).

The complete *frame-system* works like a frame with ‘‘always true’’ things at the very top nodes and ‘‘sensors on the world’’ at the lowest level.

This theoretical approach has mainly led to Object Languages, such as Smalltalk and Frame Languages, such as KL-One, but the ‘‘analogical reasoning process’’ has not really been implemented.

2.1.2. Roger Schank: one of the firsts to speak of case-based reasoning

Understanding stories in natural language was one of the first objectives of Schank when he developed his theory on CBR see Schank (1982). The basic idea is that mental schemes are guiding the understanding of texts, allowing to complement

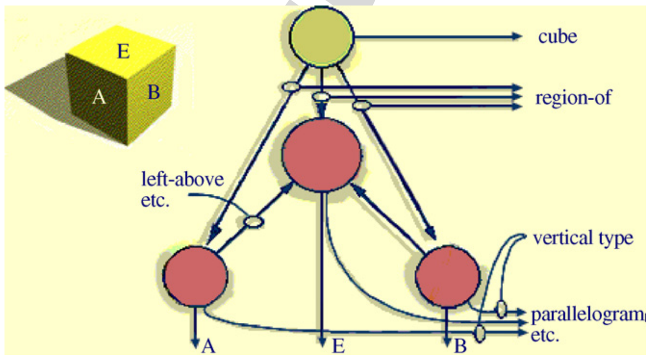

Fig. 1. Frame example (Minsky).

what was not said. ‘‘Understanding is Explaining’’: was claimed by Robert Schank[2].

Consider the following sentences: ‘‘John went to the restaurant. He took a piece of ham. It was good.’’ In order to understand this short text, we need to know, for example, that when we take a piece of ham in a restaurant, this is for eating. Nothing of that is said in the text, but we guess that John ate the piece of ham. Schank proposes to represent the behavior (here, in a restaurant) by a ‘‘script’’ splitting it in different steps which can be finer scripts and so on. See Fig. 2 for an illustration of a potential script structure.

So, a script describes an episode of a known behavior in the form of a sequence of events as they generally occur (experimented usual situations). When a new situation is encountered, the script is adapted to take this exception into account. In order to complement events with useful information, scripts contain other pieces of information, mainly:

- Actual goals.
- Current plans.
- Social links.
- Played roles.
- Character traits.
- And generally speaking, anything indicating the exhibited behavior by the script in a given situation.

Scripts and frames share many properties but what is really different is the status of ‘‘immediate experience’’. While Minsky argues that frames are ‘‘idealized stereotypes’’ of encountered situations, Schank proposes to keep ‘‘concrete episodes’’ in memory, i.e. such as these episodes occurred in reality. They are organized in a dynamic memory and reused by adaptation. This episodic memory is automatically organized by a simple generalization process as illustrated in Fig. 3.

Many researches complemented these pioneering works and the main concepts are established. AI technologies now integrate this paradigm of CBR for Problem Solving and Lazzy Learning.

2.2. Knowledge and CBR principles

Minsky and Schank were CBR pioneers but Janet Kolodner worked explicitly on CBR and wrote the first book on the subject in 1993 (Kolodner, 1993). From an engineering point of view, Agnar Aamodt and Enric Plaza (1994) proposed a CBR reasoning cycle and many research works and CBR applications have been developed since then [http://www.ai-cbr.org]. Useful references are given at the end of the principle section.

2.2.1. What is a case?

A case is the description of a problem solving episode. Its structure is designed to describe the context of the task: diagnosis, planning, decision making, help desk, configuring,

[2] http://www.edge.org/3rd_culture/bios/schank.html.

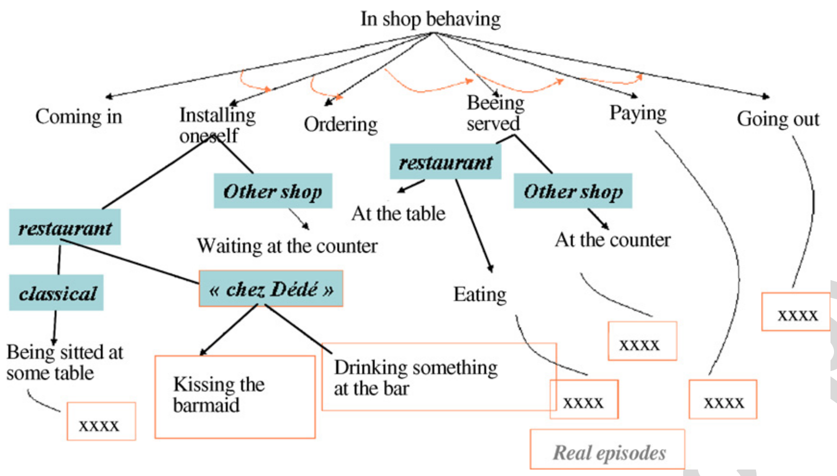

Fig. 2. Illustration of a script (Schank).

etc. In a pedagogical context, we consider a case as one list of descriptors, where each descriptor can as well be a complex structure.

A **case** is composed of a problem part and a solution part: *case* = (*pb*, *sol*(*pb*)). A **source case** *source_case* = (*source*, *sol*(*source*)) is a case of which the solution *sol*(*source*) will be reused in order to solve a new problem. The new problem is called a **target case** *target_case* = (*target*, *sol*(*source*)).

A case is described by its descriptors:

source $= \{d_1^s, \ldots, d_n^s\}$ where d_i^s is a source problem descriptor.
Sol(source) $= \{D_1^s, \ldots, D_m^s\}$ where D_i^s is a source solution descriptor
cible $= \{d_1^c, \ldots, d_n^c\}$ where d_i^c is a target problem descriptor.
Sol(cible) $= \{D_1^c, \ldots, D_n^c\}$ where D_i^c is a target solution descriptor.

Example 1. Consider the problem of finding the adequate price for a flat.

Problem part

d_1^s : flat surface(real).
d_2^s : flat location(a structure).
d_3^s : flat state(a list of defects).

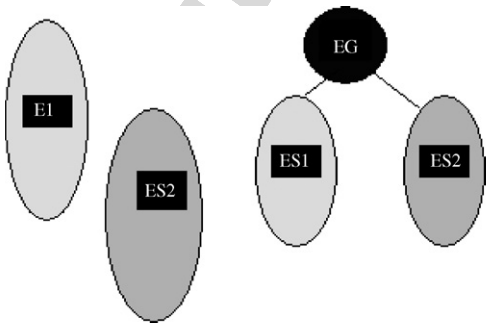

Fig. 3. Episodes E1 and E2 are combined in order to build their generalization EG. ES1 and ES2 are specializations of EG.

Solution part

D_1^c : salling price of the flat(real).
D_2^c : sale conditions(payment facilities for example).

Example 2. Consider the task of car breakdown diagnosis.

Problem part

d_1^s: noises (list of symbolic descriptors).
d_2^s: external symptoms (list of symbolic descriptors).
d_3^s: car model (symbolic descriptor).
d_4^s: first “put into circulation” date (date descriptor).

Solution part

D_1^c: mechanical parts to troubleshoot (list of symbolic descriptors).
D_2^c: diagnosed faults on the mechanical parts.

2.3. Ontology of description attributes

To match and compare cases, attribute values have to be compared for the purpose of similarity evaluation. Each attribute has a *type*. Knowing the type makes it possible to choose the appropriate comparison operators. It is useful to describe the ontology of the types of attributes to enable an efficient similarity measure but not for “describing the world”!

Table 1
A small case base on the “Flat sale problem”

Attribute label	Case 1	Case 2	Case 3	Attribute type
Pb_surface	55	35	55	Real
Pb_district_location	Rhône district	Rhône district	Ain district	Symbol
Sol_sale_price	20,000	45,000	15,000	Real
Pb_flat_type	F2	F2	F2	Symbol
Pb_town_location	Lyon	Lyon	Bourg en Bresse	Symbol

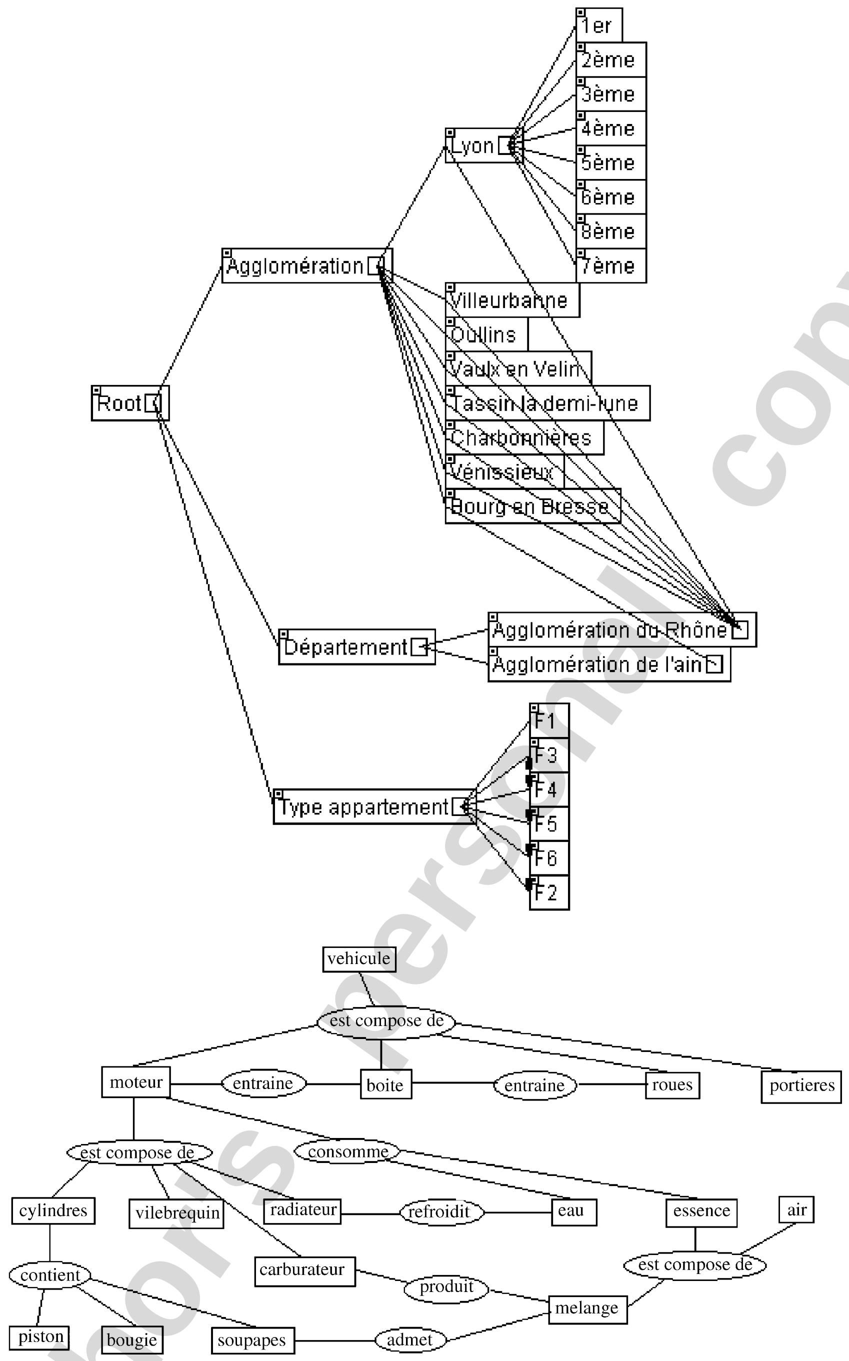

Fig. 4. Examples of domain ontologies for case descriptors.

The ontology can be shared by a whole case base, but it is not mandatory to build such an ontology. Each attribute can have a ‘‘facet’’ explaining how to manage the similarity measure for each specific case. Theoretically, ‘‘pure CBR’’ contains all necessary knowledge in cases.

2.3.1. *What is a case base?*

A case base is a collection of solved cases for a class of problems. For example, there are separate and different case bases for the ‘‘Flat sale problem’’ and for the ‘‘Card diagnosis problem’’. For the ‘‘Flat sale problem’’, a case is the description of a sale episode and descriptors obey the corresponding ontology. In the following table (Table 1), the blank rows stand for problem descriptors and the pink row stands for the solution description (here, the sale price).

The district location descriptor Pb_Disrtict_Location can be easily inferred from the ontology (see Fig. 4). Even if it seems that building a case base is easier than building a set of rules, there still exist problems with respect to knowledge engineering. Most of the industrial CBR applications propose forms to establish a library of cases. A case base can be small (if different possible types of cases are well represented and the

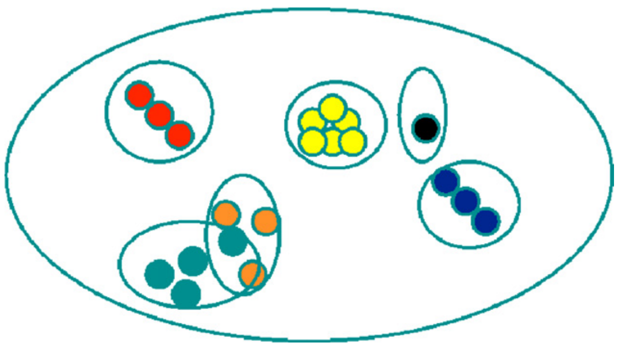

Fig. 5. Clustering cases by ‘‘type of adaptation process’’.

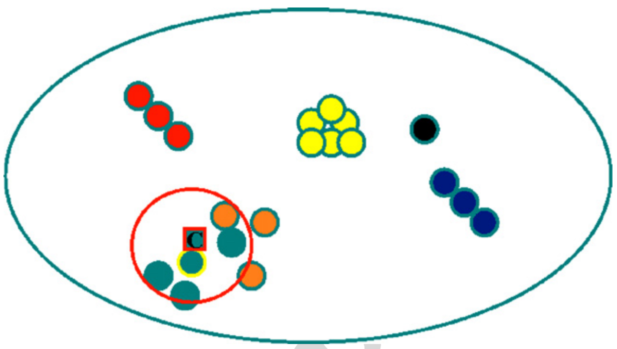

Fig. 6. Resolution process.

domain knowledge is rich) or very large (if it exists a wide variety of cases and the domain knowledge is poor).

For each case base, there exists an associated metric that, when combined with values of problem descriptors, allows to project case on the ‘‘solutions plan’’. *Similar cases* are cases that have *similar solutions* for *similar problems*.

2.3.2. *What is the resolution process? How to choose a source case in the case base?*

In order to decide if it is possible to adapt a source case to solve a target case, a similarity threshold between problem descriptions has to be reached. Moreover, there is no chance to use the same adaptation process for different kind of problems (for example, adapting the price of an old flat is not the same as adapting the price of a new one, even if everything else is very similar). Consequently, similarity measures are used to build dynamic clusters of cases in order to choose which kind of adaptation method has to be chosen for a given new problem.

The resolution process is illustrated in Figs. 5 and 6: similarity of the new target case (C) is computed with all the other cases[3]. The algorithm chooses the type of adaptation which is the most significant and the most represented in the cluster of neighbors. (C) has been assessed to be associated with a ‘‘blue’’ adaptation process.

Case-Based Reasoning needs a case base on which a metric and a measure of similarity have been defined.

2.3.3. *CBR cycle*

Aamodt and Plaza (1994) were the first who proposed a CBR cycle to make evident the knowledge engineering effort needed when developing CBR applications. This general cycle has been complemented by an ‘‘Elaborate’’ step which was not part of the original cycle (Fig. 7). This step is sometimes called ‘‘indexing’’ step.

Each step has its own method to exploit the knowledge base and the case base but ‘‘retrieve’’ and ‘‘adapt’’ steps explain how to build the knowledge representation for the domain and the cases.

[3] The case base can be structured in order to cut the number of matches to do.

2.3.4. *Elaborate*

Elaborating a new case is a method that decides which descriptors are useful for finding ‘‘adaptable’’ cases in the case base. Similarity is here a synonym of ‘‘adaptability’’. Adaptability depends directly on the supposed effort to adapt a source case solution in the context of the target case problem.

A general elaboration method completes or s filters the raw description of a problem on the basis of domain knowledge, and then infers new descriptors and importance weights. Dependencies are very important to be explicitly available at this step. This ‘‘elaborate’’ step is illustrated in Table 2 while Fig. 8 illustrates how the domain knowledge can be used to deduce a new descriptor from others.

2.3.5. *Retrieve*

The ‘‘retrieve’’ step is the key step in CBR because the quality of the adaptation depends on the quality of the retrieval. We have to keep in mind that we are searching for ‘‘similar’’ solutions by matching source and target problems. It is necessary to define a similarity measure, which takes into account dependencies between problem and solution descriptors and to verify the availability of adaptation operators for the observed differences. Numerous similarity measures are described in literature (coming from data analysis for example) that take into account specificities of descriptors (time, space, complex structures, plans, sequences, etc.). It is often possible

Table 2
Elaboration of a problem

Attribute label	Attribute type	Attribute value **(raw)**	Attribute value **(elaborated)**
General status	Symbol (inferred)	??	**Good**
Nb of kms	Real	198,000	198,000
Nb of years of the car	Real	10	10
Car manufacturer	Symbol (inferred)	??	**Peugeot**
Car model	Symbol	206	205
Car type	Symbol	Break	Break
Defects	List of symbols	(superficial problems)	(superficial problems)
Sale price (solution)	Real	???	???

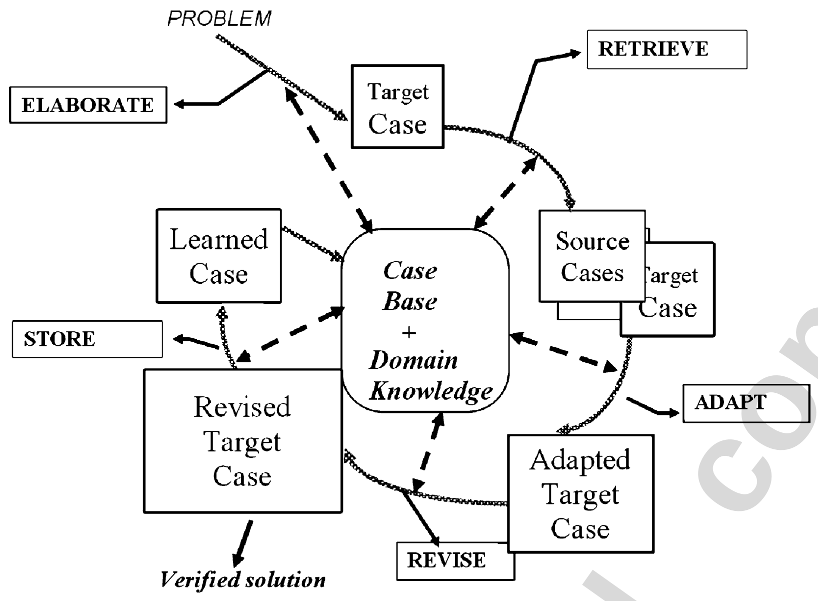

Fig. 7. CBR cycle steps.

to translate these "special" similarity measures into simpler ones by transforming complex descriptors in a set of simpler ones. Intuitively, we understand that we have to give a high weight for problem descriptors exhibiting a high dependency with solution descriptors and for which, at the same time, no simple adaptation operators exist (see Table 3 for an example of such weights). Conversely, we can put low weights on problem descriptors exhibiting little dependency and for which it is easy to adapt dependent descriptors of the solution.

For simplicity, we consider here the following measure of distance: $d = \sum_i p_i \times d_i / \sum_i p_i$, where the distance between two problem descriptions is constituted by the weighted sum of distances between attributes. Weights represent the knowledge about the importance of the "influence" of problem descriptor values on the solution.

The retrieval step utilizes these weights for choosing the best case, i.e. the easiest case to adapt. The classical algorithm is the KNN algorithm (K nearest neighbors) (Cover & Hart, 1967).

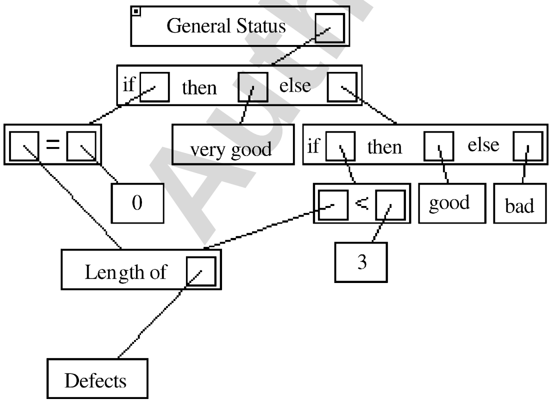

Fig. 8. General knowledge to infer the value of "general status" from the value of "defects".

2.3.6. Adapt

Adaptation is the last step of the analogical inference. It computes what could be a target solution by adapting the solution of the most similar case. Adaptation rules have to express the management of differences between source and target problems for guiding the adaptation of the source solution. The schema in Figs. 9 and 10 illustrates relevant knowledge and the inference process of adaptation:

$$D_k^t = D_k^s \pm \Delta D_k; \quad D_k^t = D_k^s \pm I\left(\frac{D_k^s}{d_i^s}\right) \times \Delta d_i \tag{1}$$

$$D_k^t = D_k^s \pm \Delta D_k; \quad D_k^t = D_k^s \pm \int_{i=\{j,m\}} \left(I\left(\frac{D_k^s}{d_i^s}\right) \times \Delta d_i \right) \tag{2}$$

The formula (Eq. (1)) expresses that D_k^t is computed by "adding" the influence $I(D_k^2/d_i^s)$ "in proportion of" the difference Δd_i between source and problem descriptors. "Addition" operator and "in proportion of" operator can be

Table 3
Attribute weights = influence importance of the attribute on the solution

Attribute label	Attribute type	Influence weight of the attribute on the solution
General status	Symbol (inferred)	20%
Nb of kms	Real	35%
Nb of years of the car	Real	25%
Car manufacturer	Symbol (inferred)	5%
Car model	Symbol	5%
Car type	Symbol	10%
Observed defects list	List of symbols	No importance
Sale price (solution)	Real	???

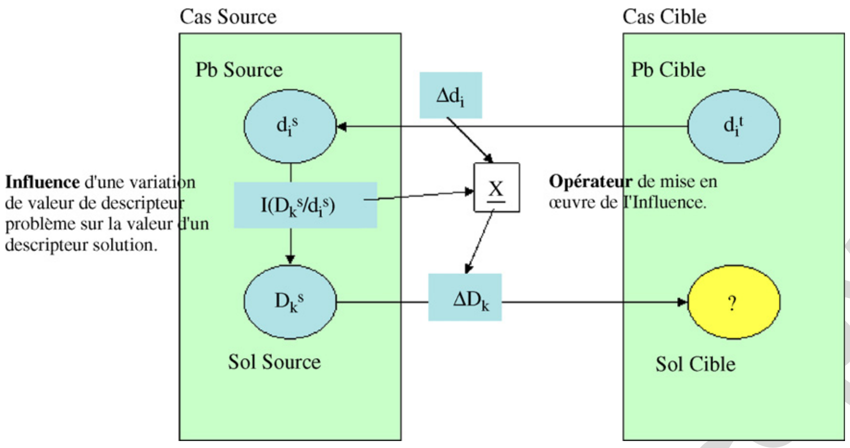
Cas Source
Cas Cible
Pb Source
Pb Cible
Δd_i
d_i^s
d_i^t
Influence d'une variation de valeur de descripteur problème sur la valeur d'un descripteur solution.
$I(D_k^s/d_i^s)$
X
Opérateur de mise en œuvre de l'Influence.
D_k^s
ΔD_k
?
Sol Source
Sol Cible

Fig. 9. Illustration of a single simple adaptation.

very specifically coined for the types of descriptors and to the context of the case (the type of adaptation to process).

Eq. (2) generalizes the previous one by "integrating" the effect of several differences on problems parts.

Δd_i: difference between values of source and target problem descriptors according to a specific matching function.

$I(D_k^s/d_i^s)$ = influence of a differenced_i^son the value ofD_k^s.

$\times$ = operator to compute "influence" according to the observed differences between problem descriptors.

$\int_{i=\{j,m\}}\left(I\left(\frac{D_k^s}{d_i^s}\right)\times\Delta d_i\right)$ sums individual influence effects of the differences between problem descriptors for an "individual" source solution descriptor (there is no general equation for several source solution descriptors).

$\pm$ = operator of "addition" of the integrated computed influence to a source solution descriptor to propose a value for the corresponding target solution descriptor.

Consider the following "car sale problem" (Table 4)

He adaptation rule could be the following:

In this rule (Fig. 11), we consider only two influences of problem descriptors on the price: *number of kilometers* and *car status*. Each positive (negative) difference of 1 km on the first descriptor adds (subtracts) € 0.1 to the price while the fact to go from "bad status" to "good status" (or vice-versa) adds (or subtracts) € 1000.[4]

2.3.7. Revise

Revising is sometimes necessary when the adapted solution did not match the current situation and needs "revisions" to correspond. In order to revise, we can:

- Try the adapted solution in the "real" world (for example, we try to sell our car with the adapted selling price...).
- Introspect the case base with the complete case in order to verify how similar complete cases worked when applied (for example, we could verify that similar cars were really sold with a similar price).
- Use another problem solving process (simulator, expert system...)

In each option we can observe differences between what the system proposed and what would have been a correct proposal. After the revising step, we could use these differences as starting points to revise the domain knowledge and to learn about the retrieval/adaptation process.

2.3.8. Memorize (learn)

Adding a new solved case to the case base is the basic "learning" mechanism of CBR. Other important entities can be capitalized:

- Similarity measure.
- Influence knowledge (functions?).
- New dependencies, etc.

It is possible to learn from the "trace" of the "reasoning process" which led to the new case. For example, if we keep trace of the adaptation process, we can consider the traces as cases of adaptation usable during the "adaptation" step.

2.4. CBR knowledge engineering

We briefly summarize important steps usually found during knowledge engineering for a CBR application:

Table 4
A "car sale problem"

Attribute label	Attribute type	Attribute-value	Elaborated value
General status	Symbol (inferred)	??	**Good**
Nb of kms	Real	198,000	198000
Nb of years of the car	Real	10	10
Car manufacturer	Symbol (inferred)	??	**Peugeot**
Car model	Symbol	206	206
Car type	Symbol	Break	Break
Defects	List of symbols	(superficial problems)	(superficial problems)
Sale price (solution)	Real	???	???

[4] In order to take into account the "car status", it would be, for example, possible to express it by a "mark" between 1 and 10 and to use it in classical metrics.

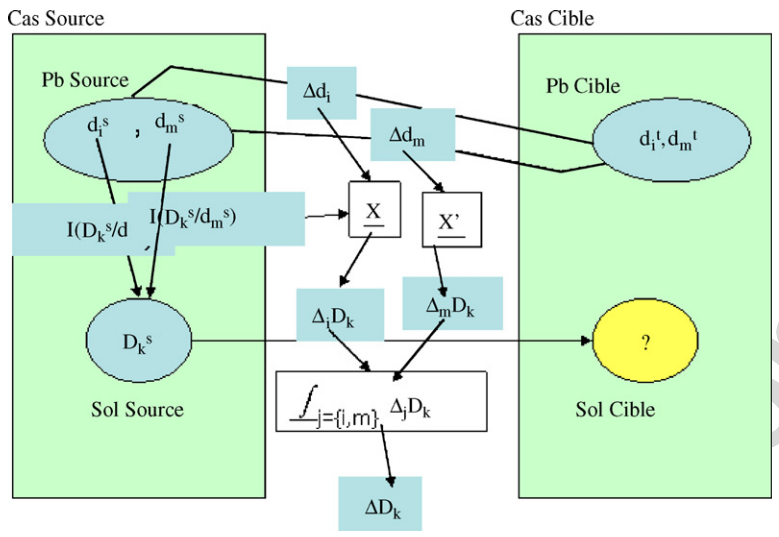

Fig. 10. General adaptation.

- Collecting potential cases.
- Describing the case descriptors.
- Testing the structures of cases with users and experts.
- Trying to build an ontology of descriptors attributes.
- Observing the reusing of cases by users and experts for real concrete problems.
- Focusing on the adaptation process.
- Eliciting dependencies and influences as they have to be used in adaptation.
- Building a measure of similarity on the base of known dependencies and influences.
- Testing the measure of similarity with the set of solved cases.
- Building the adaptation rules according to dependencies and influences.
- Testing the adaptation process on the set of known cases.
- Building the first cases with experts'' (we call this period the learning period of the system).
- Revising the whole system.
- Delivering the CBR system with an initial case base, useful for the reusing... and with continuous learning possibilities!

2.5. *Organized bibliography in order to go further*

In order to be able to understand deeply the CBR paradigm, there exists a rather comprehensive **body of literature:**

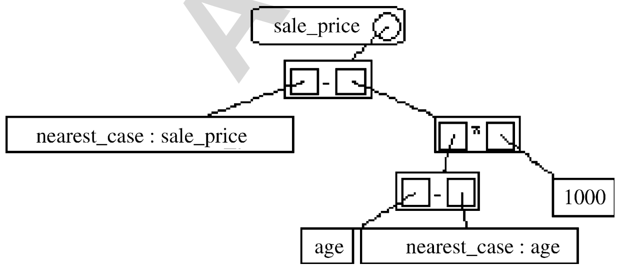

Fig. 11. Adaptation rule for sale price.

- Aamodt and Plaza (1994) give a ''knowledge engineering'' point of view on the CBR cycle which is now broadly accepted.
- Despite the fact that given URLs are no more working (Aha, 1998) gives a good idea of very different areas of CBR applications.
- Bergmann (2004) is the second edition of a book reporting the INRECA project[5] which aimed to define a knowledge Modeling Framework for CBR.
- Hanney, Keane, Smyth and Cunningham (1995) is a foundational article written by main CBR researchers. It focuses on adaptation which is probably the key problem in CBR.
- Kolodner (1993) makes the synthesis between cognitive approaches and computer models. A rather pedagogical book.
- Kolodner and Leake (1996) is a nice introduction to CBR. It is a tutorial taking place in Leake (1996), a book gathering experiences and lessons of research on CBR.
- Kolodner, Simpson, and Sycara-Cyranski (1985) is considered a seminal paper on the CBR process.
- Lenz (1998) is a book trying to link foundations to applications, and could be useful for people willing to develop CBR applications.
- Lòpez De Màntaras et al. (2006) is the most recent synthesis on the subject of CBR. It is a collective paper bringing together 13 important researchers of the field.
- Schank and Riesbeck (1989) is probably the first book on the subject. A seminal document. This book completes Schank (1982) which is the classical reference on CBR.
- Watson (1997) gathers many interesting enterprise applications giving good ideas on what can be done concretely.
- Renaud et al. (2006) is the first book presenting 11 CBR applications in French (to appear).

CBR has been widely studied for **design tasks:**

[5] http://www.wi2.uni-trier.de/de/cms/projects/INRECA/.

- Maher and Pu (1997) makes an interesting and valuable synthesis on Case-Based Design, completing the influential book on the subject Maher, Balachandran, and Zhang (1995).
- Faltings and Sun (1996) propose an original use of CBR in design for supporting innovation.

Help Desk is a very successful CBR application (Thurman et al., 1997):

- For customer technical support (Simoudis, 1992).
- For recommender systems (McSherry, 2005).

Diagnosis is a well known application of CBR, combined with other kinds of reasoning process (Portinale, Magro, & Torasso, 2004).

Planning has been widely studied by the CBR community Veloso, Munoz-Avila, and Bergmann (1996).

Knowledge management is a very promising domain of CBR application:

- Bergman (2002) founds the notion of "Experience Management".
- Champin, Mille, and Prié (2003) proposes a bridge between CBR and TBR for experience management and experience reusing.

Capitalizing on human expertise to support control tasks is an active area for CBR application:

- Brann, Thurman, and Mitchell (1995) focus on the accumulation of human expertise for discrete system control. In the same domain, Jurisica and Glasgow (1996) propose formal ways to learn control from cases.
- Fuchs, Mille, and Chiron (1995) propose an original way to help human decision in industrial supervision.

3. Trace-based reasoning

We share the idea that human experience, temporally situated by definition, is well represented by a temporal recording or by a trace of events that describe an underlying implicit process. CBR also claims that by addressing problem solving episodes, even if, de facto, CBR systems exploiting the temporal dimension of cases are not so numerous, case descriptors are not compulsorily described with time stamps. Moreover, a problem solving episode is considered independently of the different "stories" (contexts) where this episode occurred. A case is described with a fixed granularity, in a specific temporality and is constrained by an intangible description vocabulary. According to our intuition, we proposed to exploit use traces of a computer environment as possible indirect recordings of knowledge, which emerged while the user did her tasks with the help of the computer environment Champin et al. (2003). This theory defines what we call a "trace", how it can be represented and which kind of computations can be performed in order to retrieve useful past sequences of events for new uses.

When traces are exploited on the basis of pattern similarities, allowing some adaptations to new situations, we propose to call this kind of computation "Trace-Based Reasoning" (TBR). TBR is a kind of generalization of CBR principles as illustrated in Fig. 12.

The CBR cycle handles cases, which are stored in a case base, under a predefined form while the TBR cycle dynamically elaborates episodes which could be potentially useful in available traces according to some "task signature". The target episode is built on the base of other proposed episodes under control of the user. The target episode belongs to the current trace and will be stored inside without particular indexing. Stored traces are containers of potential episodes which will be shown up in new situations.

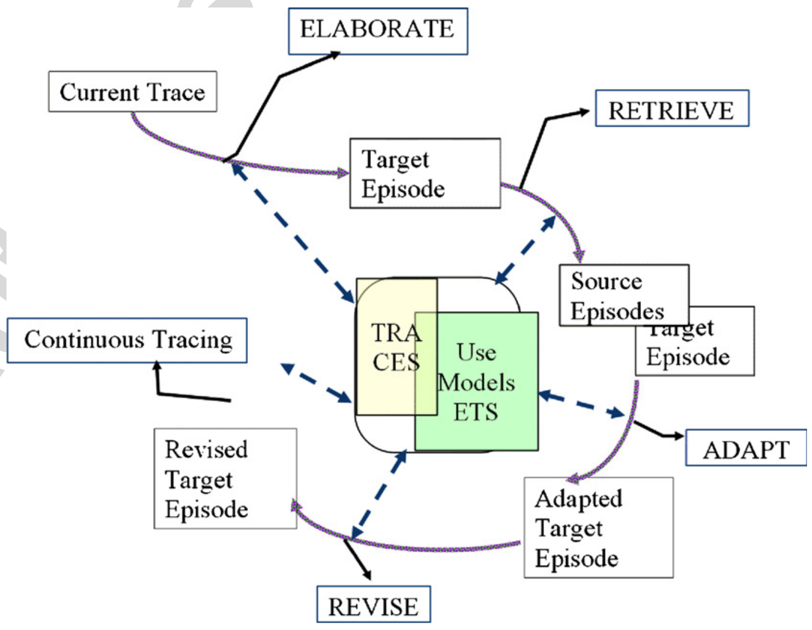

Fig. 12. The TBR cycle.

In the same way as in CBR, we consider that most steps of the reasoning cycle in TBR can be realized by the computer environment or/and by the user himself.

4. Conclusion

Case-Based Reasoning is an efficient AI paradigm for problem solving. Its robustness comes from its ability to learn from experience. Despite its big success, it suffers from the ‘‘frame problem’’ which means that new case structures are very difficult to manage with different structures of past cases. A case has to describe its ‘‘context’’ of use, which is difficult to decide before any reuse and can change in time and space. We propose an extension of the CBR paradigm by considering problem solving episodes as they can be found in computer use traces. Traces offer the possibility to build dynamically new case structures and to extend the context of cases if necessary.

Acknowledgment

The authors wish to thank Frank Nack and Amélie Cordier for their comments and remarks.

References

Aamodt, A., & Plaza, E. (1994). Case-based reasoning foundational issues, methodological variations and system approaches. *AI Communication, 7.1*, 39–59.

Aha, J. (1998). The omnipresence of case-based reasoning in science and application. *Knowledge-Based Systems, 11*(5–6), 261–273.

Bergman, R. (Ed.). (2002). *Experience Management: Foundations, Development Methodology, and Internet Based Applications*. Berlin: Springer.

Bergmann, R. (Ed.). (2004). *Developing Industrial Case-Based Reasoning Applications: The Inreca Methodology* (2nd ed.). Springer Verlag.

Brann, D. M., Thurman, D. A., & Mitchell, C. M. (1995). Case-based reasoning as a methodology for accumulating human expertise for discrete system control. In *Proceedings of the 1995 International Conference on Systems, Man and Cybernetics*.

Champin P.-A., Mille A., & Y. Prié. (2003). MUSETTE: Modelling USEs and Tasks for Tracing Experience. ICCBR’03: Workshop ‘‘From structured cases to unstructured problem solving episodes’’ ICCBR’03: NTNU, 279–286.

Cover, T. M., & Hart, P. E. (1967). Nearest neighbor pattern classification. *IEEE Transactions on Information Theory, IT–13*(1), 21–27.

Crevier, D. (1993). *AI, the tumultuous history of the search for artificial intelligence*. Harper-Collins: Basic Books.

Faltings, B., & Sun, K. (1996). FAMING: supporting innovative mechanism shape design. *Computer Aided Design, 28*(3), 207–216.

Fuchs B., Lieber J., Mille A., & Napoli A. (2006). ‘‘A general Strategy for Adaptation in Case-Based Reasoning’’. Research Report. http://liris.cnrs.fr/alain.mille/publications/cbr-journal-paper-250806.pdf.

Fuchs, B., Mille, A., & Chiron, B. (1995). Operator decision aiding by adaptation of supervision strategies. *Lecture Notes in Artificial Intelligence vol 1010, Firs International Conference, ICCBR’95* (pp. 23–32).

Hanney, K., Keane M. T., Smyth B. & Cunningham P. (1995). ‘‘Systems, tasks and adaptation knowledge’’, in *International Conference on Case-Based Reasoning*, Manuela Veloso and Agnar Aamodt Eds, Sesimbra, Portugal. Lecture Notes in Artificial Intelligence, pp. 461–470, Springer Verlag.

Jurisica, I., & Glasgow, J. (1996). A Case-Based Reasoning Approach to Learning Control. In *Proceedings of the Fifth Conference on Data an Knowledge Systems for Manufacturing and Engineering, DSKME-96*.

Kolodner, J. (1993). *Case-based reasoning*. Morgan Kaufman Publishers.

Kolodner, J., & Leake, D. B. (1996). A tutorial introduction to case-based reasoning. *Case-based reasoning: experiences, lessons & future directions*. MIT Press. 420 p..

Kolodner, J., Simpson, R., & Sycara-Cyranski, K. (1985). A process model of case-based reasoning in problem solving. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence (IJCAI-85)* (pp. 284–290).

Leake, D. B. (Ed.). (1996). *Case-based reasoning: experiences, lessons and future directions*. AAAI Press.

Lenz, M. (1998). *Case-based reasoning technology: from foundations to applications*. Berlin: Springer.

Lòpez De Màntaras, R., Mc Sherry, D., Bridge, D., Leake, D., Smyth, B., Craw, S., et al. (2006). Retrieval, reuse, revision and retention in case-based reasoning. *Knowledge Engineering Review, 20*(3), 215–240.

Maher, M.-L., Balachandran, B., & Mei Zhang, D. (1995). *Case-Based reasoning in design*. Lawrence Erlbaum Associates.

Minsky, M. (1975). *‘‘A framework for representing knowledge’’. The Psychology of Computer Vision Ed.*. Patrick Winston Mc Graw Hill.

Minsky, M., & Papert, S. (1974). *Artificial intelligence, condon lectures*. Univ of Oregon Press.

Newell A. (1960). J.C. Shaw and Herbert Simon Report on a general problem-solving program. In *Proceedings of the International Conference on Information Processing* (pp. 256–264).

Portinale, L., Magro, D., & Torasso, P. (October 2004). Multi-modal diagnosis combining case-based and model-based reasoning: a formal and experimental analysis. *Artificial Intelligence, v.158*(n.2), 109–153.

Maher, M.-L., & Pu, P. (Eds.). (1997). *Issues and applications of case-based reasoning to design*. LEA Publisher.

Renaud J., Chebel Morello B., Fuchs B., Lieber J., (Eds.), Raisonnement à partir de cas: principes, méthodes et applications industrielles, Hermès, Paris, Collection Lavoisier Information et Commande (to appear).

Schank, R. C. (1982). *Dynamic memory. A theory of reminding and learning in computers and people*. Cambridge University Press.

Schank, R. C., & Abelson, R. P. (1977). *Scripts, plans, goals and understanding: an inquiry into human knowledge structures*. Hillsdale, NJ: L. Erlbaum. (Chap. 1–3).

Schank, R. C., & Riesbeck, C. (1989). *Inside case-based reasoning*. Hillsdale, New Jersey: LEA Publishers.

McSherry, D. (2005). Explanation in recommender systems. *Artificial Intelligence Review, 24*(2), 179–197.

Simoudis, E. (1992). Using case-based reasoning for customer technical support. *IEEE Expert, 7*(5), 7–13.

Thurman S D.A., Tracy, J. S., & Mitchell, C. M. (1997). Design of an Intelligent Web-Based Help Desk System. In *Proceedings of the 1997 IEEE International Conference on Systems, Man and Cybernetics*.

Veloso, M. M., Munoz-Avila, H., & Bergmann, R. (1996). General purpose case-based planning: methods and systems. *AI Commun., 9*(3), 128–137.

Watson, I. (Ed.). (1997). *Applying case-based reasoning: techniques for enterprise systems*. San Francisco: Morgan Kaufmann.

Prof. Alain Mille is professor in Information Systems at the University Claude Bernard Lyon 1 in France. He is in charge of the ‘‘Cognition, experience and situated agents’’ team of the LIRIS CNRS Lab. His research work is based on artificial intelligence models allowing to reuse concrete experience in order to help users in their tasks. He applies this approach in several industrial domains as robotics, industrial supervision, computer assisted design, ...