FROM CASE-BASED REASONING TO TRACES-BASED REASONING

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

Abstract: CBR is an original paradigm to adapt solutions of past problems in order to solve new similar problems. A case is a problem with its solution and cases are stored in a case library. The reasoning process obeys to a cycle allowing to "learn" from new solved cases. This approach is also viewed as a lazy learning method when applied for classification. This AI technology is applied for various tasks as designing, planning, diagnosing, information retrieving, etc. The talk will be the occasion to go a step further in reusing past experience, by considering traces of computer use as experience knowledge containers for contextual and situation based problem solving.

Keywords: Problem solvers, Artificial intelligence, knowledge-based systems, knowledge representation

1 CASE BASED REASONING

1.1 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 and Schank pioneers works. Interested reader will find biographical information about them in (Crevier 1993) for example.

Marvin MINSKY draws main lines of what we could call "Stereotypes Based Reasoning"[1]

Marvin Minsky argues that usual theoretical approaches in AI try to be too "precise", local and not really structured to face "real world" problems. He considers several approaches through Artificial Intelligence and Cognitive Psychology:

- A common proposal, he made with Papert, to divide "knowledge" in structures they called "microworlds"
- The definition of "problem spaces" by Newell and Simon
- The expression of linguistic objects by Schank, Abelson and Norman.

He describes these approaches as promising by contrast with classical ones attempting to describe Knowledge as set of pieces of knowledge, with no particular structure. Marvin 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 occurs after using this frame
- A part about what to do if there is an unwanted result after frame 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*".

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.

Roger Schank: one of the first to speak of Case Based Reasoning

Understanding stories in natural language was one of the first objectives of Schank & al. when he

[1](Minsky 75) and http://web.media.mit.edu/~minsky/papers/Frames/frames.html

developed his theory on CBR. The basic idea is that mental schemes are guiding the understanding of texts, allowing filling the "gaps" of what was not said. "Understanding is Explaining" said Robert Schank.

Let's consider the following sentences: "John went to the restaurant. He got some ham. It was good.". Understanding this short text needs to know that when we got ham in a restaurant, it is for eating. Nothing of that is said in the text, but we guess that John ate the ham… Robert 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.

So, a script describes an episode according to a known behaviour by the way of a sequence of events as they are awaited (usual experimented situations). When a new situation is encountered, the script is adapted to fit with this exception. In order to complete events with useful information, scripts contain other information and mainly:

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

Scripts and schemas share many properties but what is really different is the status of "immediate experience". While Minsky argues that frames are "idealistic stereotypes" of encountered situations, Schank proposes to keep in memory "concrete episodes" as they occurred in reality. They are organized in a dynamic memory and reused by adaptation. Episodes memory is self organized by a simple generalization process.

1.2 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 Plazza (1994) proposed a CBR reasoning cycle and a lot of correlated research works and applications were developed since [http://www.ai-cbr.org]. A paper to appear in "Knowledge Engineering Review" makes the state of the art on the subject (Mantaras and al. 2005).

What is a case?

A case is the description of a solving problem episode. Its structure fits the situation of the task: diagnostic, planning, decision helping, design, etc. For pedagogical purpose, we consider a case as a list of descriptors (descriptors can be structured).

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 from which the solution $Sol(source)$ will be reuse in order to solve a new problem we call a **target case** $target\_case = (target, Sol(source))$.

A case is described by descriptors:

- source=$\{d^s_1..d^s_n\}$ where d^s_i is a source problem descriptor.
- Sol(source)=$\{D^s_1..D^s m\}$ where D^s_i is a source solution descriptor
- target=$\{d^t_1..d^t_n\}$ where d^t_i is a target problem descriptor.
- Sol(target)=$\{D^t_1..D^t_n\}$ where D^t_i is a target solution descriptor.

Example 1:

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

Problem part

d^s_1 = Flat surface (real)
d^s_2 = Flat location (a structure)
d^s_3 = Flat state (a list of defects)

Solution part

D^t_1 = Sale price of the flat (real)
D^t_2 = Sale conditions (payment facilities for example)

Example 2

Consider a task of car diagnosis.

Problem part

d^s_1 = Noises (list of symbolic descriptors)
d^s_2 = External symptoms (list of symbolic descriptors)
d^s_3 = Car model (symbolic descriptor)
d^s_4 = First circulation date (date descriptor)

Solution part

D^t_1 = Mechanical pieces to troubleshoot (list of symbolic descriptors)
D^t_2 = Diagnosed faults on the mechanical pieces

Ontology of description attributes

To match and compare cases, attribute values have to be compared for similarity evaluation purpose. Each attribute has a *type*. Knowing the type allows to choose adequate comparison operators. It is useful to describe the ontology of the types of attributes to enable efficient similarity measure (not for "describing the world")!

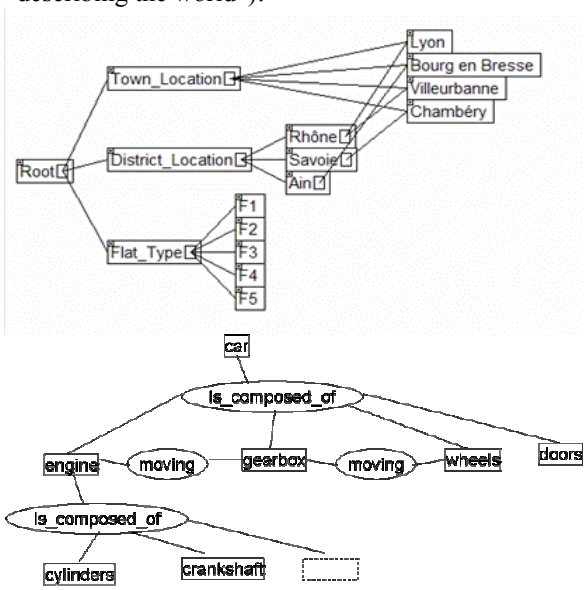


Figure 1 Examples of domain ontologies for Case descriptors

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. "Pure CBR" embodies any knowledge in cases.

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 fit the corresponding ontology. On the following table, white lines stand for problem descriptors and grey line stands for the solution description (here, the sale price).

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	20000	45000	15000	Real
Pb_Flat_Type	F2	F2	F2	Symbol
Pb_Town_Location	Lyon	Lyon	Bourg en Bresse	Symbol

The district location can be easily inferred from the ontology. Even if it seems that building a case base is easier that building a set of rules, Knowledge Engineering problems are often encountered. Most of industrial CBR applications propose forms to fill the case base. The case base can be small (if different possible types of cases are well represented and that the domain knowledge is rich) or very large (if there exists a wide variety of cases and that the domain knowledge is poor).

For each case base, there is an associated metric allowing to project cases on the "solution plan". *Similar cases* are cases that have *similar solutions* for *similar problems*.

How to choose a source case?

There is a threshold of similarity to take into account when attempting to adapt a past case for a new one. 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 thing than adapting the price of a new one, even if anything 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.

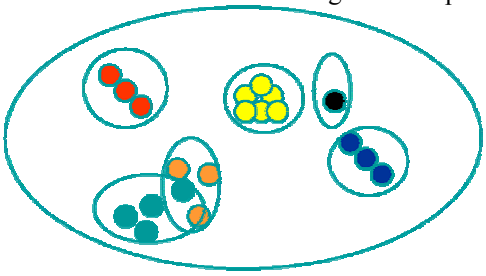


Figure 2 Clusterng cases by "type of adaptation process"

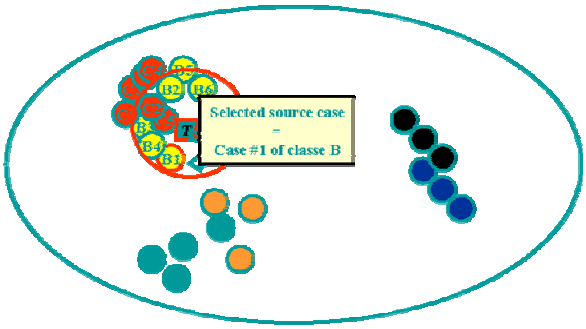


Figure 3 Source Case selection

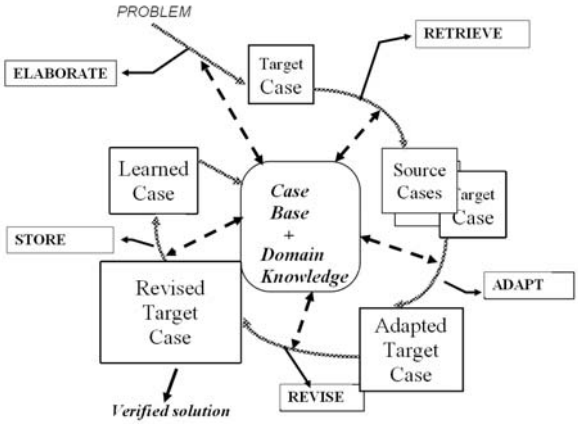
The resolution process is illustrated in **Figure 3**: similarity of the new target case (T) is computed with all other cases[2]. The algorithm chooses the type of adaptation which is the significant and the most represented in the cluster of neighbors. (T) has been assessed to fit with a "B" adaptation process.

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

CBR Cycle

Aamodt and Plaza (1994) proposed a first CBR cycle to make evident the knowledge engineering effort in CBR. We completed this general cycle by an "Elaborate" step which was not specified at first.

Each step has his proper way to use knowledge base and case base but "retrieve" and "adapt" steps explain how to build knowledge representation for domain and cases.



Elaborate

Elaborating a new case consists to decide what descriptors are useful for finding "adaptable" cases in the case base. Similarity is 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 method consists to complete or to filter the raw description of a problem on the basis of domain knowledge, inferring new descriptors and importance weights. Dependencies (β) are very important to be explicitly available at this step. This step "elaborate" is illustrated in Table 1 while Figure 4 illustrates how

[2] Case base can be structured in order to cut the number of matches to do.

the domain knowledge can be used to infer a new descriptors from an other one.

Att label	Att type	Att-value	Elaborated value
General status	Symbol (inferred)	??	**Good**
Nb kms	Real	198000	198000
Nb of years of the cas	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	???	???

Table 1 Elaboration of problem descriptors

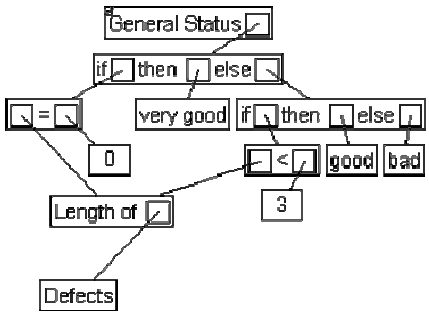


Figure 4 Domain knowledge to infer « general status » value from « list of defects »

Retrieve

The "retrieve" step is the key step in CBR because the quality of the adaptation depends on the quality of the retrieval. Do not forget that we are looking for "similar" solutions by matching source and target problems. It is necessary to define a similarity measure which will take into account dependencies between problem and solution descriptors and adaptation operators availability for observed discrepancies. There are numerous similarity measures in literature (coming from data analysis for example) taking into account specificities of descriptors (time, space, complex structures, plans, sequences, etc.). It is often possible to translate these specific similarity measures in 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 there is no simple adaptation operators. Conversely, we can put low weights for problem descriptors exhibiting little dependency and for which it is easy to adapt corresponding dependent solution descriptors. For reason of simplicity, we consider the following distance measure: $d = \frac{\sum_i p_i \times d_i}{\sum_i p_i}$ the distance between two problem descriptors is constituted by the weighted sum of attributes distances. Weights p_i hold the knowledge on the scale of "influence" of the problem descriptor d_i 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%
Manufacturer	Symbol (inferred)	5%
Car Model	Symbol	5%
Car type	Symbol	10%
Observed defects list	List of symbols	No importance
Sale Price (solution)	Real	???

Table 2 Attribute weights = influence importance of the attribute on the solution

Retrieval step consists to use these weights to choose the best case to adapt. The classical algorithm is the KNN algorithm (K nearest neighbors

Adapt

Adaptation is the end of the analogical inference by computing which could be a target solution by adapting the solution of the most similar case. Adaptation rules have to express how to manage discrepancies between source and target problems to guide adaptation of the source solution. The following schema illustrates knowledge and inference process of adaptation:

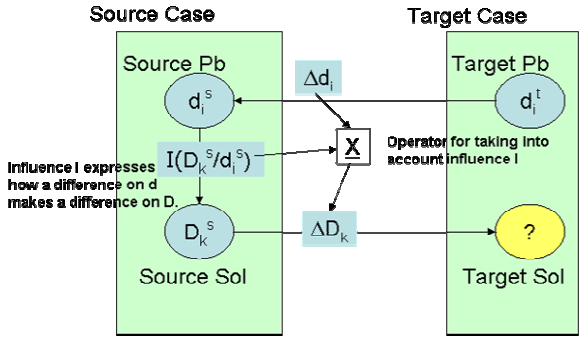


Figure 5 Simple adaptation process

Equation 1

$$D_k^t = D_k^s \pm \Delta D_k ; D_k^t = D_k^s \pm \mathrm{I}(D_k^s / d_i^s) \times \Delta d_i$$

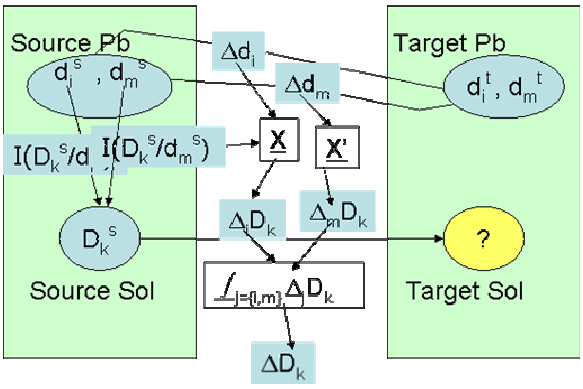


Figure 6 Global adaptation process

Equation 2

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

The formula (**Equation 1**) expresses that D_k^t is computed by "adding" the influence $I(D_k^s/d_i^s)$ "in proportion of" the difference Δd_i between source and problem descriptors. "adding" operator and "in proportion of" operator can be very specific to the types of descriptors and to the context of the case (the type of adaptation to process).

The formula (Equation 2) generalizes the previous one. There is a new operator « integrating » the effect of several discrepancies on problems parts.

- Δd_i = discrepancy between source and target problem descriptors values according to a specific matching function.
- $\mathrm{I}(D_k^s / d_i^s)$ = influence of a discrepancy of d_i^s on the value of D_k^s.
- $\times$ = operator to compute Influence according to the observed problem descriptors discrepancies.
- $\int_{i=\{j,m\}} \left(\mathrm{I}(D_k^s / d_i^s) \times \Delta d_i \right)$ sums individual influence effects of problem descriptors discrepancies 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 »

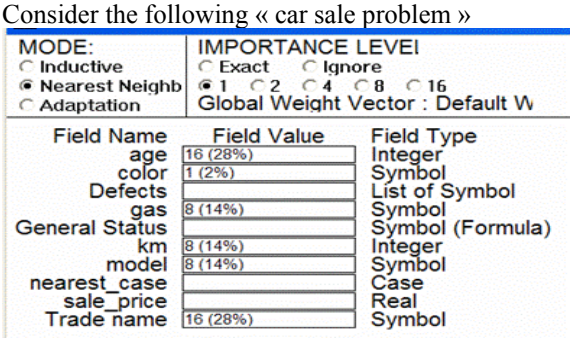


Figure 7 Importance (influence) values

The adaptation rule could be the following:

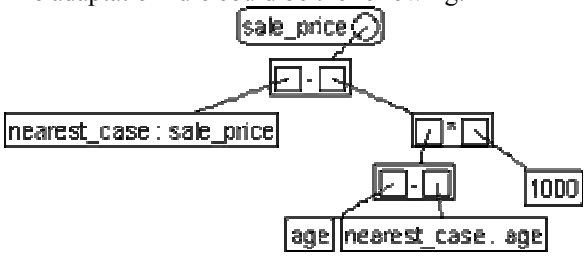


Figure 8 Simple adaptation rule

In this (too simple) rule, we consider only 1 influence of problem descriptors on the *price*: *age of the car*. Each positive (negative) discrepancy of 1 year on the descriptor *age* adds (subtracts) 1000 euros to the *price*.

Revise

Revising is sometimes necessary when the adapted solution did not fit the current situation and needs "revisions" to fit it. 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 sale 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 an other problem solving process (simulator, expert system, …)

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

Memorize (learn)

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

- As noticed for the "revise" step, retrieval and adaptation knowledge:
  - Similarity measure
  - Influence knowledge
  - New dependencies, etc
- The "trace" of the "reasoning process" as it was done for the current new case. For example, if we keep trace of the adaptation process, we can consider these adaptation traces as "adaptation cases" usable in a CBR cycle for improving adaptation knowledge.

1.3 CBR Knowledge engineering

Very shortly, we can summarize important steps we usually find during knowledge engineering for a CBR application:

- Collecting potential "cases"
- Describing case descriptors
- Testing cases structures with "users/experts"
- Building an ontology of descriptors attributes
- Observing the reusing of cases by users/experts for real concrete problems.
- Focusing on the adaptation process
- Eliciting dependencies and influences as they are used in adaptation
- Building a similarity measure on the base of known dependencies and influences
- Testing similarity measure with the set of solved cases
- Building adaptation rules according to dependencies and influences
- Testing adaptation on the set of known cases

- Building new cases with “normal” users with “observers/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 reusing…and continuous learning possibilities!

2 TRACES BASED REASONING

We share the idea that human experience, temporally situated by definition, is well represented by a temporal record or trace describing an implicit underlying process. CBR claims also that property 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 time stamped. 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 contains intangible description terms.

We propose to exploit use traces of a computer environment as possible indirect records of knowledge which emerged while the user did his/her tasks with the help of the computer environment. We propose a theory defining what we call a "trace", how it can be represented and which kind of computations can be done in order to retrieve useful past sequences 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 "Traces Based Reasoning" (TBR). TBR is a kind of generalization of CBR principles.

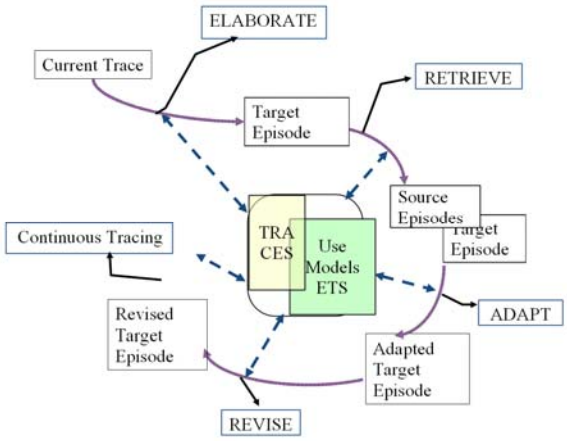


Figure 9 The CBR cycle handles cases, which are stored in a case base, under a predefined form; the TBR cycle dynamically elaborates episodes which could be potentially useful in available traces according to some "task signature"; the target episode is built with the help of other proposed episodes under the user control. The target episode belongs to the current trace, it will be stored in it without particular indexing. Stored traces are containers of potential episodes which will be revealed in new situations.

As for CBR, we consider that most of the reasoning cycle steps can be realized by the computer environment or/and by the user himself.

3 CONCLUSION

Case-Based Reasoning is an AI paradigm for problem solving. This approach is very efficient and 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 when their structure has to change because of new ways to solve problems for example. 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 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.

De Màntaras et al. (2005) Retrieval, reuse, revision and retension in case-based reasoning. *The Knowledge Engineering Review* **0.**1-2 (2005) (to appear)

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

Crevier D. (1993) . *AI, The tumultuous history of the search for Artificial Intelligence*. Basic Books, Harper-Collins, 1993.

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

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

Aamodt, A. and Plaza E.. Case-Based Reasoning foundational issues, methodological variations and system approaches *AI Communication* **7**.1 (1994): 39-59.