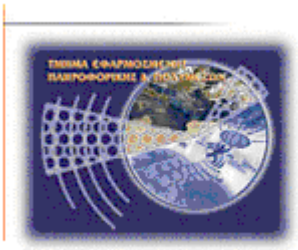
Τεχνολογικό Εκπαιδευτικό Ίδρυμα Κρήτης

Σχολή Τεχνολογικών Εφαρμογών

Τμήμα Εφαρμοσμένης Πληροφορικής & Πολυμέσων



Πτυχιακή εργασία

Survey για τεχνολογίες Case Based Reasoning στην εκπαίδευση

Παπαδόπουλος Βασιλης Α.Μ.818

Σαμπροβαλάκης Γιώργος Α.Μ.1103

Ηράκλειο – 30/11/2011

Επόπτης Καθηγητής: ΠΑΠΑΔΑΚΗΣ ΝΙΚΟΣ

Ευχαριστίες

Θα θέλαμε να ευχαριστήσουμε τους γονείς μας για την στήριξη τους όλα αυτά τα χρόνια.

Επίσης ευχαριστούμε τον καθηγητή κ. Παπαδάκη Νίκο για την πολύτιμη βοήθεια του και τις οδηγίες του στην εκπόνηση της πτυχιακής μας εργασίας.

Abstract

This survey is about Case Based Reasoning (CBR) in education. Case Based Reasoning is a technique of Artificial Intelligence (AI) based on human problem solving. CBR is an approach to problem solving that emphasizes the role of prior experiences during future problem solving.

In Chapter 1 an introduction of CBR is being described. Following in Chapter 2 a reference is being made in the history of CBR and CBR tools. In the next Chapter the CBR cycle is analyzed. In Chapter 4 are described some main software tools that use the CBR technique. Finally the last chapter focuses on CBR in education and how implementation of Case Based Reasoning in learning improves educational interaction.

Table of contents

Ευχαριστίες……	**ii**
Abstract……..	**iii**
Table of contents……...	**iv**
Image index……	**vii**
Chapter 1: Introduction…….....	**1**
1.1 What is CBR……	**1**
1.2 CBR term and examples……..……	**2**
1.3 CBR advantages and disadvantages……	**3**
1.3.1 Advantages……	**3**
1.3.2 Disadvantages……...	**5**
1.4 Why CBR…….	**6**
1.5 Comparing rule-based systems and model-based systems with CBR……	**8**
Chapter 2: History of CBR and CBR tools…….....	**11**
2.1 History of CBR in USA……..	**11**
2.2 History of CBR in Europe and Asia……	**13**
2.3 History of CBR tools in USA……	**14**
2.4 History of CBR tools in Europe……..	**15**
Chapter 3: CBR cycle……...	**16**
3.1 Introduction of CBR cycle……...	**16**
3.2 Popular CBR working cycle's……	**20**
3.2.1 Kolodner's cycle……...	**20**
3.2.2 Aamodt's and Plaza's cycle…..……	**23**
3.3 Relationship of problem and solution spaces…..……	**26**
3.4 CBR process example……..	**27**
3.5 Techniques and subtasks used in CBR tasks……...	**28**
3.5.1 Case retrieval…….	**28**
3.5.2 Case reuse……..	**31**
3.5.3 Case revise……	**32**
3.5.4 Case retain…….	**34**
Chapter 4: CBR software tools……	**36**
4.1 Generally……..	**36**
4.1.1 A classification of CBR applications……	**36**
4.1.2 The views of theorists and software vendors for CBR and CBR tools…….	**37**
4.2 CBR software tools and applications……...	**37**
4.2.1 ART*Enterprise……	**37**
4.2.2 Case-1……	**38**
4.2.3 CasePower…….	**38**
4.2.4 CBR2 (CBR Express, CasePoint, Generator and Tester)…….	**39**
4.2.5 Eclipse……...	**40**
4.2.6 ESTEEM……...	**41**
4.2.7 KATE……	**41**
4.2.8 ReCall……...	**42**
4.2.9 ReMind……	**43**
4.2.10 S3-Case……...	**44**
4.3 CBR shells and development environments……	**46**
4.3.1 CASPIAN……	**47**
4.3.2 CASUEL……...	**47**
4.3.3 CBR-Works……...	**48**
4.4 Case Based Reasoning object-oriented frameworks……	**49**

4.4.1 CBR*Tools……..	**50**
4.4.2 CAT-CBR…….	**51**
4.4.3 JColibri……..	**52**
4.4.4 CLAVIER…….	**53**
4.4.5 FormTool……..	**53**
4.5 Web-based CBR……...	**54**
4.6 Intelligent and Integrated Ticketing Management (I2TM) application example….	**55**
4.6.1 Data acquisition…….	**56**
4.6.2 Retrieve…….	**58**
4.6.3 Reuse…….	**59**
4.6.4 Revise……	**59**
4.6.5 Retain……	**60**
4.6.6 Evaluation…….	**60**
4.7 CBR application domains……	**61**
Chapter 5: CBR in education……...	**62**
5.1 Case Based Reasoning for Intelligent Tutoring System (ITS)…….	**62**
5.1.1 Introduction……...	**62**
5.1.2 Proposed system for distributed case based reasoning…….	**63**
5.1.3 Student modeling in distributed environment……...	**64**
5.2 Case-based tutoring in virtual education environments……..	**69**
5.2.1 Playing the DollarBay game…….	**70**
5.2.2 Overview of LambdaMOO……...	**71**
5.2.3 Virtual educational environments at NDSU…….	**71**
5.3 The application of CBR on Q&A system……	**73**
5.3.1 Introduction……...	**73**
5.3.2 Architecture of the Auto Q&A system…….	**73**
5.3.3 Case Authoring Module and case base construction……	**75**
5.3.3.1 Definition of the Question-Answer case……	**75**
5.3.3.2 The 3-Layer architecture of the Q&A case base……	**76**
5.3.4 Case study and experiment……...	**78**
5.4 Design of individualized instruction with learning object using case-based reasoning in WBI…….	**81**
5.4.1 Introduction……...	**81**
5.4.2 Theoretical framework……..	**82**
5.4.2.1 Individual differences……	**82**
5.4.2.2 Case Based Reasoning……...	**90**
5.4.2.3 Learning objects…….	**93**
5.4.3 Architecture of MLOID……	**97**
5.5 Looking towards the future……..	**99**
Chapter 6: Bibliography…….	**100**

Image index

Figure 1 - CBR compared with Rule-based and Model-based systems……..	**9**
Figure 2 - CBR cycle……	**18**
Figure 3 - CBR cycle according to Kolodner……..	**22**
Figure 4 - CBR cycle according to Aamodt & Plaza……..	**25**
Figure 5 - Relationship between problem and solution spaces in……	**26**
Figure 6 – Near Neighbor Retrieval (NNR) equation……..	**30**
Figure 7 - A classification hierarchy of CBR applications……..	**36**
Figure 8 – Overview of CBR software tools and applications……	**45**
Figure 9 - A summary of the major CBR shells……..	**46**
Figure 10 - A summary of the major CBR development environments……..	**48**
Figure 11 - CBR*Tools Object Model…….	**50**
Figure 12 - CAT-CBR process of developing a CBR system……..	**51**
Figure 13 - I2TM system architecture……..	**55**
Figure 14 - Overview of the data structure in I2TM and CBR-TM…….	**57**
Figure 15 - a: Influence of the database size of the CBR-TM system performance, b: Influence of the number of simultaneous customers on the CBR-TM system performance	**61**
Figure 16 - Communication model among Agents……..	**63**
Figure 17- Process of student modeling ……..	**65**
Figure 18 - Design architecture for CBR based ITS in distributed & agent environment...	**66**
Figure 19 - Teaching Agent component……..	**67**
Figure 20 - Architecture of the Q&A system……..	**74**
Figure 21 - Case representation with detailed features……	**75**
Figure 22 - Case Representation with natural language description……	**76**
Figure 23 - 3-Layer architecture of a Question-Answer case base……..	**77**
Figure 24 - Question submission in Q&A system……..……	**78**
Figure 25 - The returned similar question lists ranked by average scores……..	**79**
Figure 26 - The adjusted question list based on the learning network…….	**80**
Figure 27 - Plot of the mean errors convergence and 95% confidence interval along the CBR process……..	**80**
Figure 28 - Learning strategies used by each Kolb's learning style model…….	**85**
Figure 29 - The architecture of MLOID system……..	**97**

Chapter 1: Introduction

1.1 What is CBR

Case-based reasoning (CBR) is an intelligent-systems method that enables information managers to increase efficiency and reduce cost by substantially automating processes such as diagnosis, scheduling and design **[1]** but it is also a pervasive behavior in everyday human problem solving **[2]**. CBR is a kind of analogical reasoning that focuses on reasoning based on previous experience or accepted by the being actively exercising choice, most deeply explored in human cognitive science. A previous experience can play several roles, such as **[3]**:

- Suggesting a solution to a new problem or a way to explain a situation.
- Warning of a problem that will appear.
- Allowing the potential effects of a proposed solution to be predicted.

A case-based reasoner works by matching new problems to "cases" from a historical database and then adapting successful solutions from the past to current situation **[1]**. In CBR terminology, a case usually denotes a problem situation. A previously experienced situation, which has been captured and learned in a way that it can be reused in the solving of future problems, is referred to as a past case, previous case, stored case, or retained case. Correspondingly, a new case or unsolved case is the description of a new problem to be solved. Case-based reasoning is a cyclic and integrated process of solving a problem, learning from this experience, solving a new problem **[4]**.

Organizations as diverse as IBM, VISA International, Volkswagen, British Airways, and NASA have already made use of CBR in applications such as customer support, quality assurance, aircraft maintenance, process planning, and decision support, and many more applications are easily imaginable **[1]**.

1.2 CBR term and examples

Case-based reasoning is a problem solving paradigm that in many respects is fundamentally different from other major AI approaches. Instead of relying solely on general knowledge of a problem domain, or making associations along generalized relationships between problem descriptors and conclusions, CBR is able to utilize the specific knowledge of previously experienced cases. A new problem is solved by finding a similar past case, and reusing it in the new problem situation **[4]**. In case-based reasoning systems expertise is embodied in a library of past cases, rather than being encoded in classical rules. Each case typically contains a description of the problem, plus a solution and/or the outcome. The knowledge and reasoning process used by an expert to solve the problem is not recorded, but is implicit in the solution **[5]**. CBR also is an approach to incremental, sustained learning, since a new experience is retained each time a problem has been solved, making it immediately available for future problems.

Mentioned with the term problem solving is used here in a wide sense, coherent with common practice within the area of knowledge-based systems in general. This means that problem solving is not necessarily the finding of a concrete solution to an application problem, it may be any problem put forth by the user. So it can be used to justify or criticize a solution proposed by the user, to interpret a problem situation, to generate a set of possible solutions, or generate expectations in observable data are also problem solving situations**.**

Let us illustrate this by looking at some typical problem solving situations:

- A physician, after having examined a particular patient in his office gets a reminding to a patient that he treated two weeks ago. Assuming that the reminding was caused by a similarity of important, the physician uses the diagnosis and treatment of the previous patient to determine the disease and treatment for the patient in front of him.
- A drilling engineer, who have experienced two dramatic blow out situations, is quickly reminded of one of these situations (or both) when the combination of critical measurements matches those of a blow out case. In particular, he may get a reminding to a mistake he made during a previous blow-out, and use this to avoid repeating the error once again.
- A financial consultant, working on a difficult credit decision task, uses a reminding to a previous case, which involved a company in similar trouble as the current one, to recommend that the loan application should be refused.

The CBR field has grown rapidly over the last few years, as seen by its increased share of papers at major conferences, available commercial tools, and successful applications in daily use **[4]**.

1.3 CBR advantages and disadvantages

1.3.1 Advantages

Case-based reasoning provides many advantages for a reasoner :

- Case-based reasoning allows the reasoner to propose solutions to problems quickly, avoiding the time necessary to derive those answers from scratch.

The doctor remembering an old diagnosis or treatment experiences this benefit. While the case-based reasoner has to evaluate proposed solutions like any reasoner does, it gets a head start on solving problems because it can generate proposals easily. There is considerable advantage in not having to redo time consuming computations and inferences. This advantage is helpful for almost all reasoning tasks, including problem solving, planning, explanation, and diagnosis.

- Case-based reasoning allows a reasoner to propose solutions in domains that he/she/it doesn't understand completely.

Many domains are impossible to understand completely, often because much depends on unpredictable human behavior, e.g., the economy. Others nobody understands yet, e.g., how some medications and diseases operate. Other times, we simply find ourselves in situations that we don't understand well, but in which we must act anyway, e.g., choosing which graduate students to accept into a program. Case-based reasoning allows us to make assumptions and predictions based on what worked in the past without having a complete understanding.

- Case-based reasoning gives a reasoner a means of evaluating solutions when no algorithmic method is available for evaluation.

Using cases to aid in evaluation is particularly helpful when there are many unknowns, making any other kind of evaluation impossible or hard. Instead, solutions are evaluated in the context of previous similar situations. Again, the reasoner does his/her evaluation based on what worked in the past.

- Cases are particularly useful for use in interpreting open-ended and ill-defined concepts.

Attorneys are taught to use cases as precedents for constructing and justifying arguments in new cases. But it is also important in everyday situations.

- Remembering previous experiences is particularly useful in warning of the potential for problems that have occurred in the past, alerting a reasoner to take actions to avoid repeating past mistakes.

Remembered experiences can be successful or failure episodes, i.e., situations in which things did not turn out exactly as planned. Consider again the reasoner trying to plan a meal. He/she can be helped considerably, for example, if he/she remembers a meal that was supposed to be easy-to-prepare and inexpensive and instead was hard to make because some of the ingredients were hard to obtain in manufactured form and had to be made from scratch. The reasoner is warned, by this case, to avoid those ingredients or to make sure they are available before committing to a menu.

- Cases help a reasoner to focus its reasoning on important parts of a problem by pointing out what features of a problem are the important ones.

What was important in previous situations will tend to be important in new ones. Thus, if in a previous case, some set of features was implicated in a failure, the reasoner focuses on those features to insure that the failure will not be repeated. Similarly, if some features are implicated in a success, the reasoned knows to focus on those features. Such focus plays a role in both problem solving and interpretive case-based reasoning. In interpretive case-based reasoning, justifications and critiques are built based on those features that have proven responsible for failures and successes in the past. An attorney, for example focuses on those aspects of a new situation that mattered in previous cases. In problem solving, a reasoner might attempt to adapt his solution so that it includes more of what was responsible for previous successes and less of what was responsible for failures **[18]**.

1.3.2 Disadvantages

- A case-based reasoned might be tempted to use all cases blindly, relying on previous experience without validating it in the new situation.
- A case-based reasoned might allow cases to bias him or her or it too much in solving a new problem.
- Often people, especially novices, are not remembered of the most appropriate search of cases when they are reasoning.

Relying on previous experience without doing validation can result in inefficient or incorrect solutions and valuations. Retrieval of inappropriate cases can cost precious problem-solving time or lead to costly errors that can be avoided by more incremental methods.

People do find case-based reasoning a natural way to reason , however , and endeavor of explaining the processes involved in case-based reasoning might help us to learn how to teach people to reason better using cases. In addition, the case memory technology we develop is beginning to allow us to build decision adding systems that augment human memory by providing the appropriate cases while still allowing the human user to reason in natural and familiar way. And we can make sure our programs avoid negative types of behavior **[18]**.

1.4 Why CBR

The study of CBR is driven by two primary motivations:

- The first, from cognitive science, is the desire to model human behavior.
- The second, from artificial intelligence, is the pragmatic desire to develop technology to make AI systems more effective.

Interest in CBR as a cognitive model is supported by studies of human reasoning which demonstrate reasoning from cases in a wide range of task contexts. For example, studies support the importance of remindings of prior examples in learning a computer text editor (Ross - 1984), learning programming (Pirolli & Anderson - 1985), mathematical problem solving (Faries & Schlossberg - 1994, Ross - 1984), diagnosis by automobile mechanics (Lancaster & Kolodner - 1987) and physicians (Schmidt, Norman & Boshuizen - 1990), explanation of anomalous events (Read & Cesa - 1991) and decision-making under time pressure (Klein & Calderwood - 1988, 1989). Understanding these processes requires developing and testing theories of how humans store, retrieve, and apply prior cases.

Observations that people use case-based reasoning have also spurred interest in CBR as an AI technology. Humans are robust problem-solvers, they routinely solve hard problems despite limited and uncertain knowledge, and their performance improves with experience. All of these qualities are desirable for real-world AI systems. Consequently, it is natural to ask how CBR can advance AI technology.

Discussions of this question have identified five main problems that can be ameliorated by case-based reasoning:

1. **Knowledge acquisition**: A classic problem in traditional knowledge-based systems is how to provide the rules on which the systems depend. The rule acquisition process can be laborious and unreliable: it may be difficult to elicit rules, and there is no assurance that those rules will actually be sufficient to characterize expert performance. In some domains, rules may be difficult to formalize or the number of rules required may be unmanageably large.

Because case-based reasoners reason from complete specific episodes, CBR makes it unnecessary to decompose experiences and generalize their parts into rules. Some task domains are especially natural for CBR, with cases that are suitable for CBR already collected as part of standard problem-solving procedures. In those domains, the cost of knowledge acquisition for CBR is very low. Mark, Simoudis, & Hinkle describe their experience in one such domain, autoclave loading. Other reports corroborate comparatively rapid development times for other CBR applications (e.g., Simoudis & Miller - 1991). Of course, not all domains are natural CBR domains, cases may be unavailable, or may be available but in a hard-to-use form (e.g., cases described with natural language text). In these situations, applying CBR may depend on a significant "case engineering" effort to delimit the information that cases must contain, to define the representation for that information and to extract that information from available data. Likewise, applying CBR requires developing criteria for indexing and

reapplying prior cases. However, even if this initial process requires considerable effort, CBR can still provide overall benefits for knowledge acquisition. First, experts who are resistant to attempts to distill a set of domain rules are often eager to tell their "war stories" the cases they have encountered. This facilitates gathering the needed data for CBR. Second, as discussed in the following point, after the initial case engineering effort it is often simple to augment and maintain the knowledge a CBR system needs.

2. **Knowledge maintenance**: Defining an initial knowledge base is generally only the first step towards a successful AI application. Initial understanding of the problem is often imperfect, requiring system knowledge to be refined. Likewise, changes in task requirements and circumstances may render existing knowledge obsolete. Although refinement of case representations and indexing schemes may be required as a task becomes better understood, CBR offers a significant benefit for knowledge maintenance: a user may be able to add missing cases to the case library without expert intervention.

Also, because CBR systems do incremental learning, they can be deployed with only a limited set of "seed cases" to be augmented with new cases if (and only if) the initial case library turns out to be insufficient in practice. A CBR system needs only to handle the types of problems that actually occur in practice, while generative systems must account for all problems that are possible in principle.

3. **Increasing problem-solving efficiency**: People achieve satisfactory problem solving performance despite the fact that commonplace problems in everyday reasoning, such as explanation and planning, are NP-hard (Bylander et al. – 1991, Chapman - 1987). Reuse of prior solutions helps increase problem-solving efficiency by building on prior reasoning rather than repeating prior effort. In addition, because CBR saves failed solutions as well as successes, it can warn of potential problems to avoid.

4. **Increasing quality of solutions**: When the principles of a domain are not well understood, rules will be imperfect. In that situation, the solutions suggested by cases may be more accurate than those suggested by chains of rules, because cases reflect what really happens (or fails to happen) in a given set of circumstances.

In medical reasoning, for example, anecdotes about specific cases go beyond codified knowledge, serving as "the as-yet-unorganized evidence at the forefront of clinical medicine" (Hunter - 1986).

5. **User acceptance**: A key problem in deploying successful AI systems is user acceptance: no system is useful unless its users accept its results. To trust the system's conclusions, a user may need to be convinced that they are derived in a reasonable way. This is a problem for other approaches: neural network systems cannot provide explanations of their decisions, and rule-based systems must explain their decisions by reference to their rules, which the user may not fully understand or accept (Riesbeck - 1988). On the other hand, the results of CBR systems are based on actual prior cases that can be presented to the user to provide compelling support for the system's conclusions **[21]**.

1.5 Comparing rule-based systems and model-based systems with CBR

The method in which knowledge is stored and used is the major difference between CBR, rule-based and model-based systems (**figure 1**). CBR primarily reasons from examples rather than from rules which allows for easy knowledge acquisition and application development. CBR replaces generate and test mechanisms of rule-based systems with retrieve and adapt. Adaptation implies partial matching which is a technique rule-based systems cannot implement. CBR is sometimes mistakenly described as rule-based reasoning with big rules. More correctly, however, it is large chunks of domain knowledge indexed and stored in a manner which is unique. CBR also has the ability to learn. There are two components in the CBR learning process, success and failure. CBR systems learn new cases as a result of a success. If a failure results, CBR systems can learn the problems to anticipate and recovery strategies from new cases. Ultimately this implies that the system automatically refines its knowledge (Leake - 1995). Knowledge in rule-based and model-based systems is limited to those rules that have been identified and stored. Maintaining a rule-based system is normally a manual process requiring further knowledge acquisition. It is not capable of adapting its own knowledge to new situations whereas CBR has an in-built adaptive mechanism (Burstein & Smith - 1994, Althoff et al. - 1994).

The advantages of CBR over rule-based systems include its ability to:

- Handle domains where problems have many exceptions to rules.
- Handle domains where problems are not fully understood.
- Learn from experience, that is keep up with knowledge that workers learn in their daily experiences, indicating an ability to store temporal information.
- Represent the expert's knowledge more accurately (Schank suggests that experts usually tend to use knowledge in the form of particular experiences (cases) rather than in the form of rules).
- Provide methodologies for validating and maintaining the application.
- Offer cost-effective solutions to knowledge acquisition bottleneck problems (Slade - 1991).

Rule-based systems are more suited to solve a problem when it is difficult to gather case data (Althoff et al. - 1994). CBR is unsuitable when there is little or no case data available.

Studies have shown that CBR systems can be implemented more quickly than model or rule-based systems and are easier to maintain. This results in a reduction in the knowledge-elicitation effort, since CBR systems do not require an understanding of how to solve the problem. That is, to build a CBR system you only have to obtain past cases and their solutions, you do not have to elicit rules from experts. In rule-based systems the addition of a new rule can require the modification of several other rules, whereas the addition of cases to a case library rarely involves modification of the case base (Leake - 1995). Model-based reasoning is based on knowledge of the structure and behaviour of the devices the system is designed to understand.

Important differences between rule-based systems and CBR include :

- Rules-based systems provide answers and CBR provides precedents, or a set of precedents allowing the user to choose the most applicable precedent.
- CBR do not need all criteria to be fulfilled, that is, some fields may be left blank if they are unknown, without jeopardizing the result.

CBR results are improved after each iteration if a new case is added to the case-base, rule-based systems in comparison predict the same answers and the associated error level after each iteration.

Case-based reasoning	Rule-based reasoning	Model-based reasoning
Cases in case library are constants	Rules in rule bases are patterns	Models replicate a known process and usually consist of a database and rules.
Cases are retrieved that match the input partially or exactly. Cases are retrieved first, approximating the entire solution at once, then adapted and refined to a final answer.	Rules are retrieved that match the input exactly. Rules are applied in an iterative cycle of micro-events.	Is applicable when a causal model exists and the domain is understood.
Cases are chunks of domain knowledge.	Rules are independent, consistent pieces of domain knowledge.	Stores causal models of devices or domains.
Knowledge is in the form of cases. Shows from which cases solutions were derived.	Knowledge is extracted from experts and encoded in rules. Gives rules as explanations.	Provides a means of verifying solutions, but the solution generation is unguided.

Figure 1 - CBR compared with Rule-based and Model-based systems (Zeleznikow - 1995) **[22].**

Expert systems or knowledge-based systems (KBS) are one of the success stories of Artificial Intelligence (AI) research. In a recent survey the UK Department of Trade & Industry found over 2000 KBS in commercial. It has been around twenty years since the first documented KBS (the trinity of classic systems: DENDRAL, MYCIN and PROSPECTOR) were reported, yet in that time the basic architecture of KBS has changed little. The early KBS, and today's systems, are based upon an explicit model of the knowledge required to solve a problem (known as second generation systems) using a deep causal model that enables a system to reason using first principles. But whether the knowledge is shallow or deep an explicit model of the domain must still be elicited and implemented often in the form of rules or perhaps more recently as object models.

Despite the undoubted success of model-based KBS in many sectors developers of these systems have met several problems:

- Knowledge elicitation is a difficult process.
- Implementing KBS is a difficult process requiring special skills and often taking many years.
- Implemented model-based KBS are often slow and are unable to access or manage large volumes of information.
- Implemented model-based are difficult to maintain.

Solutions to these problems have been sought through better elicitation techniques and tools, better KBS shells and environments, improved development methodologies, knowledge modeling languages and ontologies, facilitating the co-operation between KBS and databases in expert databases and deductive databases, and techniques and tools for maintaining systems.

However, over the last few years an alternative reasoning paradigm and computational problem solving method has increasingly attracted more and more attention. Case-based reasoning solves new problems by adapting previously successful solutions to similar problems. CBR is attracting attention because it seems to directly address the problems outlined above:

- CBR does not require an explicit domain model and so elicitation becomes a task of gathering case histories.
- Implementation is reduced to identifying significant features that describe a case, an easier task than creating an explicit model.
- Applying database techniques into largely volumes of information can be managed.
- CBR systems can learn by acquiring new knowledge as cases thus making maintenance easier **[6]**.

Chapter 2: History of CBR and CBR tools

2.1 History of CBR in USA

The work Schank and Abelson in 1977 is widely held to be the origins of CBR. They proposed that our general knowledge about situations is recorded as "scripts" that allow us to set up expectations and perform inferences. Scripts were proposed as a structure for conceptual memory describing information about stereotypical events such as, going to a restaurant or visiting a doctor.

However, experiments on scripts showed that they were not a complete theory of memory representation - people often confused events that had similar scripts. For example, a person might mix up room scenes from a visit to a doctor's office with a visit to a dentist's office. Such observations fell in line with the theories of concept formation, problem solving and experiential learning within philosophy and psychology.

Roger Schank continued to explore the role that the memory of previous situations (cases) and situation patterns or memory organization packets (MOPs) play in both problem solving and learning **[6]**.

Other trails into the CBR field has come by Gentner from the study of analogical reasoning and - further back - from theories of concept formation, problem solving and experiential learning within philosophy and psychology (Wittgenstein - 1953, Tulving - 1972, Smith - 1981).

For example, Wittgenstein observed that "natural concepts", the concepts that are part of the natural world (such as bird, orange, chair, car, etc.) are polymorphic. That is, their instances may be categorized in a variety of ways, and it is not possible to come up with a useful classical definition, in terms of a set of necessary and sufficient features, for such concepts. An answer to this problem is to represent a concept extensionally, defined by its set of instances - or cases **[4]**.

Whilst the philosophical roots of CBR could perhaps be claimed by many what is not in doubt is that it was the work of Roger Schank's group at Yale University in the early 80's that produced both a cognitive model for CBR and the first CBR applications based upon this model **[6]**.

Schank's model was the basis for the development of the earliest CBR systems: Janet Kolodner's CYRUS and Michael Lebowitz's IPP **[7]**.

CYRUS was basically a question-answering system which contained knowledge, as cases, of the various travels and meetings of former US Secretary of State Cyrus Vance **[4]**. It was an implementation of Schank's dynamic memory model. Roger Schank's case-memory model later served as the basis for several other CBR systems including MEDIATOR (Simpson – 1985), CHEF (Hammond – 1986), PERSUADER (Sycara – 1987), CASEY (Koton – 1989) and JULIA (Hinrichs – 1992) **[6]**.

Other schools of CBR and closely allied fields emerged in the 1980's, investigating such topics as CBR in legal reasoning, memory-based reasoning (a way of reasoning from examples on massively parallel machines), and combinations of CBR with other reasoning methods **[7]**.

Another basis for CBR, and another set of models, were developed by Bruce Porter and his group (Porter - 1986) at the University of Texas in Austin. They initially addressed the machine learning problem of concept learning for classification tasks. This lead to the development of the PROTOS system (Bareiss - 1989), which emphasized on integrating general domain knowledge and specific case knowledge into a unified representation structure. The combination of cases with general domain knowledge was pushed further in GREBE (Branting - 1991), an application in the domain of law.

Another early significant contribution to CBR was the work by Edwina Rissland and her group at the University of Massachusetts in Amhearst. With several law scientists in the group, they were interested in the role of precedence reasoning in legal judgments (Rissland - 1983). Cases (precedents) are here not used to produce a single answer, but to interpret a situation in court, and to produce and assess arguments for both parties. This resulted in the HYPO system (Ashley - 1990), and later the combined case-based and rule-based system CABARET (Skalak - 1992). Phyllis Koton at MIT studied the use of case-based reasoning to optimize performance in an existing knowledge based system, where the domain (heart failure) was described by a deep, causal model. This resulted in the CASEY system (Koton - 1989), in which case-based and deep model-based reasoning was combined **[4]**.

CBR technology has produced a number of successful deployed systems, the earliest being Lockheed's CLAVIER, a system for laying out composite parts to be baked in an industrial convection oven. CBR has been used extensively in help desk applications such as the Compaq SMART system **[7]**.

2.2 History of CBR in Europe and Asia

In Europe, research on CBR was taken up a little later than in the US. The CBR work seems to have been stronger coupled to expert systems development and knowledge acquisition research than in the US **[4]**. Amongst the first cited European work is that of Derek Sleeman's group from Aberdeen in Scotland. They studied the uses of cases for knowledge acquisition, developing the REFINER system (Sharma & Sleeman - 1988). At a similar time Mike Keane, from Trinity College Dublin, undertook cognitive science research into analogical reasoning that has subsequently influenced CBR (Keane - 1988) **[6]**.

Among the earliest results was the work on CBR for complex technical diagnosis within the MOLTKE system, done by Michael Richter together with Klaus Dieter Althoff and others at the University of Kaiserslautern (Althoff - 1989) **[4]**. This has given rise to the PATDEX system (Richter & Weiss - 1991) and subsequently to the CBR tool S3-Case **[6]**.

At the University of Trondheim, Agnar Aamodt and colleagues at Sintef studied the learning aspect of CBR in the context of knowledge acquisition in general, and knowledge maintenance in particular. For problem solving, the combined use of cases and general domain knowledge was focused (Aamodt - 1989). This lead to the development of the CREEK system and integration framework (Aamodt - 1991), and to continued work on knowledge-intensive case-based reasoning **[4]**.

At Artificial Intelligence Research Institute (IIIA) in Blanes, Enric Plaza and Ramon Lopez de Mantaras developed a case-based learning apprentice system for medical diagnosis (Plaza - 1990), and Beatrice Lopez investigated the use of case-based methods for strategy-level reasoning (Lopez - 1990) **[4]**.

In the UK, CBR seems to be particularly applied to civil engineering. A group at the University of Salford is applying CBR techniques to fault diagnosis, repair and refurbishment of buildings (Watson & Abdullah - 1994). Yang & Robertson in Edinburgh are developing a CBR system for interpreting building regulations, a domain reliant upon the concept of precedence. Whilst another group in Wales are applying CBR to the design of motorway bridges (Moore - 1994) **[6]**.

Currently, the CBR activities in the United States as well as in Europe are spreading out **[4]**. However, the increasing number of CBR papers in AI journals and the increasing number of commercially successful CBR applications is likely to ensure that many more countries take an active interest in CBR in the future **[6]**. Germany seems to have taken a leading position in terms of number of active researchers, and several groups of significant size and activity level have been established recently.

From Japan and other Asian countries, there are also activity points, for example in India (Venkatamaran - 1993). In Japan, the interest is to a large extent focused towards the parallel computation approach to CBR (Kitano - 1993) **[4]**. As an indicator the British Computer Society Specialist Group on Expert Systems has held CBR workshops suitable for novices at both it last annual conferences **[6].**

2.3 History of CBR tools in USA

Case-Based Reasoning (CBR) research in the USA was boosted by a three year Defence Advanced Research Project Agency (DARPA - now renamed as ARPA) initiative in CBR in 1987, The project's aim was to combine the work of several American universities that were carrying out research in CBR with those of a commercial company in order to blend the universities research results into a generic CBR tool. The company was Cognitive Systems, Inc., headed at the time by Dr. Roger Schank, a pioneer in developing and promoting CBR technology as an alternative to other approaches.

By 1990, Cognitive Systems had developed a research prototype tool called “The CBR Shell” written in Common Lisp for Macintosh computers. This tool had facilities for representing, storing, indexing and retrieving cases. Applications in several domains were demonstrated, including battle planning, natural language understanding and telex message classification. After demonstrating the tool at several conferences and workshops in 1990, and having received a large degree of interest in it from both government and industry, Cognitive Systems began developing a commercial-strength, multi-platform version of “The CBR Shell” that would eventually be released in the spring of 1992 as REMIND.

As momentum began to build for the commercial use of CBR technology in 1990**,** other US companies began to develop CBR tools of various kinds. In mid-1991, Inference Corporation released a CBR-based help desk building tool called CBR EXPRESS. CBR EXPRESS had the unique ability to integrate natural language text into the case indexing and retrieval process, making it easier for end users to interact with the case base in the course of problem solving.

In addition, CBR EXPRESS had a custom user interface that was specifically geared towards building help desk applications. CBR EXPRESS rapidly became a front runner in knowledge-based help desk tools with, according to the supplier, over 13,000 copies sold world-wide. Also in late 1991, ESTEEM was released, integrated into the expert system shell Kappa PC. This provided a system able to integrate CBR with rule-based knowledge. ESTEEM provided CBR capabilities to end users at a significantly lower price than CBR EXPRESS or REMIND but with less functionality and limited capabilities for handling reasonably large databases.

After the initial three CBR took (REMIND, CBR EXPRESS and ESTEEM) were released, other tools for performing various CBR tasks emerged in the US, including:

- The Easy Reasoner from The Haley Enterprise, a tool for software developers that combines CBR EXPRESS-like natural language handling, with

REMIND-like induction and ESTEEM-like ability to integrate with an expert system shell (ECLIPSE).

- Induce-It (renamed Case-Power**),** which provides simple inductive indexing from Excel spreadsheets for case representation and storage **[8]**.

2.4 History of CBR tools in Europe

In Europe, up until early 1990's, there was little work performed under the heading of "Case-Based Reasoning", although several groups worked with CBR technology under different names. Donald Michie, at the Turing Institute in Glasgow, pioneered the closely-related field of induction in the early eighties and reported on many impressive applications. In 1988, KATE-INDUCTION, developed by Michel Manage, was made available and was recommended for use by Texas Instruments. This included some CBR features, although the term was not used at that time.

As regards CBR tools developed under that title, three have been developed by European companies:

- The first proper commercial European CBR shell appeared in mid-1991. Initially called CASE WORK, it was later renamed KATE-*CBR* and is now sold by AcknoSoft.
- A second tool, RECALL, appeared in August 1993 from another French company ISoft, which was founded in the early 1990s.
- The third tool, *S^3-CASE*, also appeared in 1993. Developed by the German company tecInno, it benefited significantly from the INRECA project (see below).

European CBR research was boosted significantly by the three and a half year INRECA project (ESPRIT 6322) that started in May 1992. INRECA aims at integrator induction and CBR techniques for holding decision support applications. As mentioned above, INRECA led to the development of S^3-CASE and to some of the most recent developments in KATE 4.0.

Another important CBR research project, which also started in 1992, is FABEL, funded by the German Ministry of Research. FABEL aims at integrating CBR with model-based approaches for design tasks. The cases consist of graphically-represented layout fragments from an architectural design domain.

It appears that the European CBR community now has a distinctive character in that the emphasis is on applications rather than on tools, and on the integration of CBR with other technologies. Work on induction and CBR undertaken by European companies has let to many applications **[8]**.

Chapter 3: CBR cycle

3.1 Introduction of CBR cycle

A CBR system solves new problems by adapting solutions that were used to solve old problems. CBR is to solve a new problem by remembering a previous similar situation and by reusing information and knowledge of that situation. The case base holds a number of problems with their corresponding solutions. Once a new problem arises, the solution to it is obtained by retrieving similar cases from the case base and studying the similarity between them. A CBR system is a dynamic system in which new problems are added to the case base, redundant ones are eliminated, and others are created by combining existing ones **[11]**.

The basic scenario for almost all CBR applications is: In order to find a solution of an actual problem, one looks for a similar problem in an experience base, takes the solution from the past and uses it as a starting point to find a solution for the actual problem **[14]**.

There are two main phases in a CBR system:

- **Initial development**: A number, usually large, of previous cases of faults and their known solutions are encoded into the systems. This is known as developing the case base.
- **Routine use**: A current problem, with unknown origin and unknown solution, is presented to the system, initially as a textual description. The system then searches the case base in an attempt to find historically known cases, which match this current problem as closely as possible. In situations in which there are several matches of a similar degree of closeness, the system may ask one or more questions to try to disambiguate these previous cases, and to narrow down the solution to one (or a few) which match the current problem closely **[11]**.

The CBR system has not the only goal of providing solutions to problems but also of taking care of other tasks occurring when it is used in practice **[14]**.
The processes involved in CBR can be represented by a schematic cycle (CBR cycle) **[6]**.

According to Kolodner, the CBR working cycle can be described best in terms of four processing stages:

1. **Case retrieval:** after the problem situation has been assessed, the best matching case is searched in the case base and an approximate solution is retrieved.
2. **Case adaptation:** the retrieved solution is adapted to fit better the new problem.
3. **Solution evaluation:** the adapted solution can be evaluated either before the solution is applied to the problem or after the solution has been applied. In any case, if the accomplished result is not satisfactory, the retrieved solution must be adapted again or more cases should be retrieved.
4. **Case-base updating:** If the solution was verified as correct, the new case may be added to the case base.

Aamodt and Plaza give a slightly different scheme of the CBR working cycle comprising the four RE's :

1. **Retrieve** the most similar case(s).
2. **Reuse** the case(s) to attempt to solve the current problem.
3. **Revise** the proposed solution if necessary.
4. **Retain** the new solution as a part of a new case.

Although they use different terminologies, the CBR working cycles denoted above are essentially the same **[16]**.

All case-based reasoning methods have in common the following process:

- **Retrieve** the most similar case(s) comparing the case to the library of past cases.
- **Reuse** the retrieved case to try to solve the current problem.
- **Revise** and adapt the proposed solution if necessary.
- **Retain** the final solution as part of a new case **[5]**.

In CBR cycle a new problem is matched against the cases furnishing the case base and one or more similar cases are retrieved. A solution suggested by the matching cases is then reused. Unless the retrieved case is a close match, the solution will probably have to be revised (adapted) and tested (evaluated) for success, producing a new case that can be retained ensuing, consequently, update of the case base **[16]**.

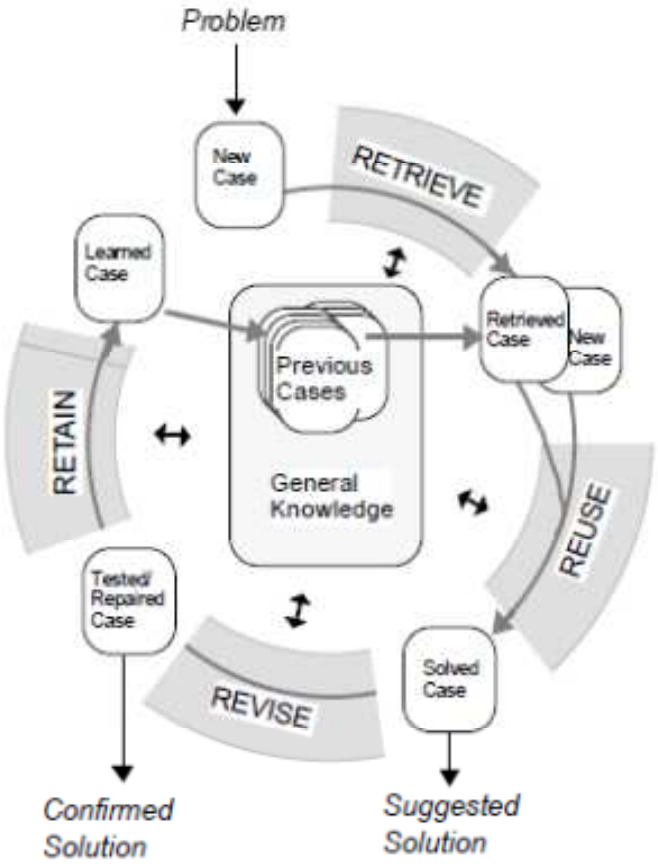


Figure 2 - CBR cycle

A new problem is solved by retrieving one or more previously experienced cases, generating a solution by reusing the case in one way or another, revising the solution by checking its correctness/usefulness - updating the solution if needed, and retaining the new experience by incorporating it into the existing knowledge-base (case-base).

The four processes each involve a number of more specific steps. In **figure 2**, this cycle is illustrated.

An initial description of a problem (top of figure) defines a new case. This new case is used to **Retrieve** a case from the collection of previous cases. The retrieved case is combined with the new case through **Reuse** into a solved case (i.e. a proposed solution to the initial problem). Through the **Revise** process this solution is tested for success (e.g. by being applied to the real world environment or evaluated by a teacher) and repaired if failed. During **Retain**, useful experience is retained for future reuse, and the case base is updated by a new learned case, or by modification of some existing cases **[13]**.

- The main advantage of this technology is that it can be applied to almost any domain. CBR system does not try to find rules between parameters of the problem, it just tries to find similar problems (from the past) and to use solutions of the similar problems as a solution of an actual problem. So, this approach is extremely suitable for less examined domains - for domains where rules and connections between parameters are not known. Furthermore, in more examined domains integration of CBR in classical rule-based reasoning systems brings some efficiency.
- The second very important advantage is that CBR approach to learning and problem solving is very similar to human cognitive processes - people take into account and use past experience to make future decisions **[14]**.

3.2 Popular CBR working cycle's

3.2.1 Kolodner's cycle

According to Kolodner "a case is a conceptualised piece of knowledge representing an experience that teaches a lesson fundamental to achieving the goals of the reasoned". Therefore, a case can be defined as a conceptualised part of knowledge representing past experience in general.

The importance of the represented knowledge in a case is because it supplies a wide range of contents in a form of the computer and/or human readable formats. Case representation encloses a detailed problem description and a detailed solution description. The detailed problem description consists of a new problem and a solved problem description. When the new problem issue arises, the retrieval process identifies the case with the most similar problem description in the past cases. If there is any stored problem description, it represents the description of detailed solution of that case. If it is necessary, adaptation occurs and a new solution is created.

Within a case representation, most types of data can be stored in a case. However it may be difficult to represent large amount of interrelated data. Therefore the functionality and acquisition of information need to be clarified first to decide what should be represented in cases **[12]**.

According to Kolodner the CBR working cycle can be described best in terms of these processing stages:

1. **Representation:** Given a new situation, generate appropriate semantic indices that will allow its classification and categorization. This usually implies a standard indexing vocabulary that the CBR system uses to store historical information and problems. The vocabulary must be rich enough to be expressive, but limited enough to allow efficient recall.

2. **Retrieval:** Given a new, indexed problem, retrieve the best past cases from memory. This requires answering three questions:

    - What constitute an appropriate case?
    - What are the criteria of closeness or similarity between cases?
    - How should cases be indexed?

    Indexing a case is essential in establishing similarity, because the indices help define the important elements of a problem, those that should be considered when studying the problem.

Thus, part of the index must be a description of the problem that the case solved, at some level of abstraction. Part of the case is also the knowledge gained from solving the problem represented by the case. Retrieving a case starts with a (possibly partial) problem description and ends when best matching cases found. The subtasks involve identifying a set of relevant problem descriptors, matching the case and returning a set of sufficiently similar cases, and selecting the best case from the set of cases returned.

3. **Adaptation**: Modify the old solutions to confirm to the new situation, resulting in a proposed solution. With the exception of trivial situations, the solution recalled will not immediately apply to the new problem, usually because the old and the new problem are slightly different.

   Reusing the retrieved case solution in the context of the new case focuses on:

   - Identifying the differences between the retrieved and the current case.
   - Identifying the part of a retrieved case that can be transferred to the new case.

   Generally the solution of the retrieved case is transferred to the new case directly as its solution case.

4. **Validation**: Determine whether the proposed solution is successful. Checking a solution can take many forms, depending on the domain.
   Whatever the means, after the system checks a solution, it must evaluate the results of this check. If the solution is acceptable, based on domain criteria, the CBR system is done with reasoning. Otherwise, the case must be modified again, and this time the modifications will be guided by the results of the solution's evaluation. Revising the case solution generated by the reuse process is necessary when the solution proves incorrect. This provides an opportunity to learn from failure.

5. **Update**: If the solution fails, explain the failure and learn it, to avoid repeating it. If the solution succeeds and warrants retention, incorporate it into the case memory as a successful solution and stop.
   The CBR system must decide if a successful new solution is sufficiently different from already-known solutions to warrant storage. If it does warrant storage, the system must decide how the new case will be indexed, on which level of abstraction it will be saved, and where it will be put inside the case-base organization **[10]**.

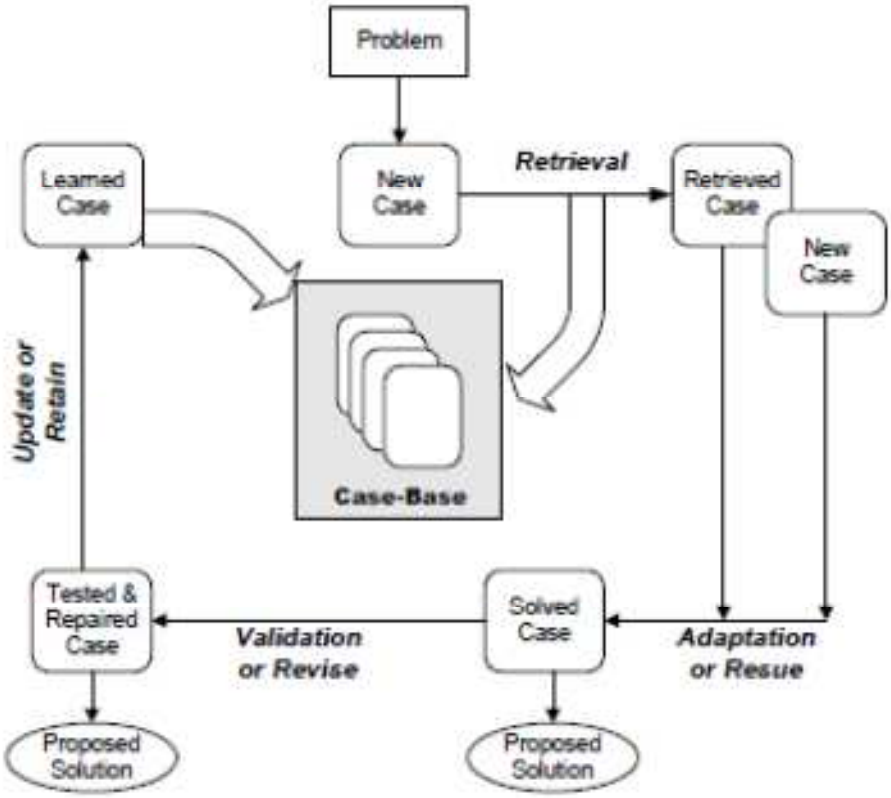


Figure 3 - CBR cycle according to Kolodner

3.2.2 Aamodt's and Plaza's cycle

Aamodt and Plaza (1994) have described CBR typically as a cyclical process comprising the four RE's:

1. The **retrieve** task starts with a (partial) problem description, and ends when a best matching previous case has been found. Usually, an initial identification subtask comes up with a set of relevant problem descriptors, a matching subtask returns a set of cases that are sufficiently similar to the new case, given a similarity threshold of some kind, and a selection subtask then works on this set of cases and chooses the best match (or at least a first case to try out). While some case-based approaches retrieve a previous case largely based on superficial, syntactical similarities among problem descriptors (e.g. the CYRUS system (Kolodner - 1983), ARC (Plaza & Lopez de Mantaras - 1990), and PATDEX-1 (Richter and Wess - 1991) systems), some approaches attempt to retrieve cases based on features that have deeper, semantical similarities (e.g. the Protos (Bareiss - 1989), CASEY (Koton - 1989), CREEK (Aamodt - 1991), and MMA (Plaza and Arcos - 1993) systems).
   Syntactic similarity assessment (sometimes referred to as a "knowledge-poor" approach) has its advantage in domains where general domain knowledge is very difficult or impossible to acquire beforehand.
   Semantically oriented approaches on the other hand, often referred to as (knowledge-intensive), are able to use the contextual meaning of a problem description in its matching, for domains where general domain knowledge is available

2. **Reuse** of the retrieved case solution in the context of the new case focuses on two aspects:

   - The differences among the past and the current case.
   - What part of a retrieved case can be transferred to the new case.

   In simple classification tasks the differences are abstracted away and the solution class of the retrieved case is transferred to the new case as its solution class. This is a trivial type of reuse.

   More typically, systems have to take into account differences in and thus the reused part, cannot be directly transferred to the new case but requires an adaptation process that takes into account those differences.

   There are two main ways to reuse past cases:

   - Transformational reuse: Reuse of the past case solution.
   - Derivational reuse: Reuse of the past method that constructed the solution.

3. Case **revision** consists of two tasks:

    - Evaluate the case solution generated by reuse (and if successful) learn from the success.
    - Repair the case solution using domain-specific knowledge.

    The evaluation task takes the result from applying the solution in the real environment (asking a teacher or performing the task in the real world). This is usually a step outside the CBR system, since it (at least for a system in normal operation) involves the application of a suggested solution to the real problem. The results from applying the solution may take some time to appear, depending on the type of application. In a medical decision support system, the success or failure of a treatment may take from a few hours up to several months. The case may still be learned, and be available in the case base in the intermediate period, but it has to be marked as a non-evaluated case. A solution may also be applied to a simulation program that is able to generate a correct solution.

    This is done in CHEF (Hammond - 1989), where a solution (i.e. a cooking recipe) is applied to an internal model assumed to be strong enough to give the necessary feedback for solution repair.
    Case repair involves detecting the errors of the current solution and retrieving or generating explanations for them.

4. The **retain** task takes care of the learning step. It incorporates into the existing knowledge what is useful to retain from the new problem solving episode. Learning from a successful or a failed problem solving attempt is triggered by the outcome of the revision and possible repair tasks.
    It involves:

    - Selecting what information from the experience that should be retained.
    - In what form it should be retained.
    - Whether a new case should be constructed.
    - How a new case should be indexed for later retrieval.
    - How it should be integrated in the memory structure and knowledge base in general.

    A new case may be built, or the old case may be generalized or strengthened to subsume the present case as well. If the problem was solved by other methods, including asking the user, a new case is constructed.
    The “indexing problem” is a central and much focused problem in case-based reasoning. It amounts to deciding what type of indexes to use for future retrieval, and how to structure the search space of indexes.
    Through interaction with the user, the general domain knowledge may also be updated (for example when holes or inconsistencies have been discovered during attempts to explain the new case) **[4]**.

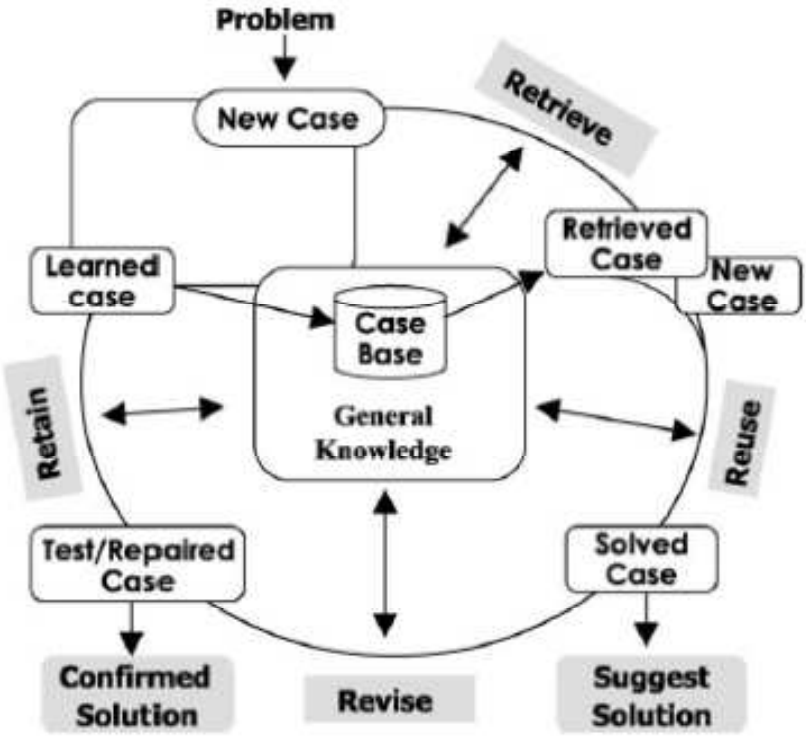


Figure 4 - CBR cycle according to Aamodt & Plaza

3.3 Relationship of problem and solution spaces

As illustrated in **Figure 5**, Leake (1996) expresses the role of similarity through the concepts of retrieval and adaptation distances. Also in Leake's diagram is captured the relationship between problem and solution spaces in CBR.

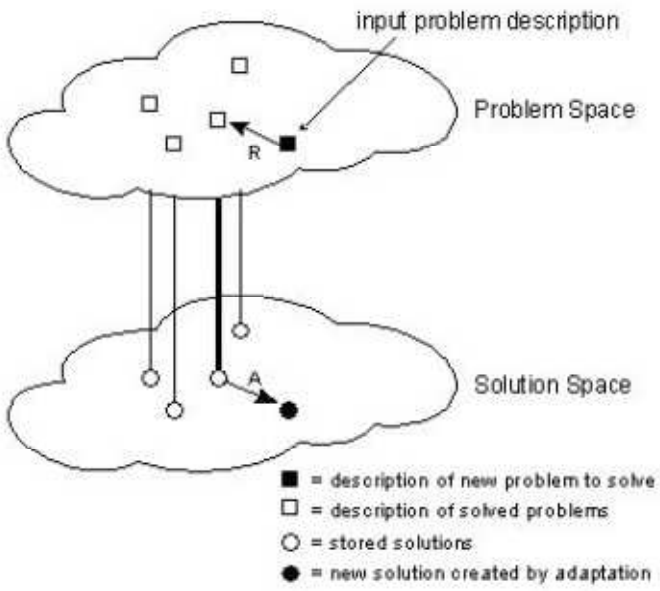


Figure 5 - Relationship between problem and solution spaces in CBR (Leake - 1996).

In **Figure 5**, the retrieval distance *R* increases as the similarity between the input problem description and a stored problem description decreases (i.e., lower similarity means greater distance).
A common assumption in CBR is that the retrieval distance *R* is commensurate with *A*, the adaptation distance (or effort).
However, several authors have questioned this assumption and its implication that the most similar case is the easiest to adapt **[15]**.

3.4 CBR process example

The CBR process is best illustrated by an example. Consider the problem of a project manager predicting how many resources to allocate for the development of different software components.

Knowledge or memory of the past is the basis for predicting future effort. Here the case is a software component. Each case will comprise a vector of features to describe each component. Examples of features might include the programming language (categorical), the number of interfaces (discrete) and the time available to develop, since severe schedule compression may adversely affect the development effort (continuous).

Notice how the vector can comprise features of different types. This adds some complexity to the way in which distance is measured. The choice of features is arbitrary and may be driven by both pragmatic considerations (what is easily available) and domain considerations which feature best characterize the problem.

One constraint is that the values for the features must be knowable at the time the prediction is required which will usually militate against the use of features such as code length.

For effort prediction the solution part of the case is trivial, merely a single value denoting the actual effort consumed. For our effort prediction problem, the case-base will grow as components are completed and the solution, i.e. the actual required amount of effort in person hours, or whatever, becomes known.

When a new prediction problem arises, the new component must be described in terms of the feature vector so that it can be viewed as the target case. The problem then becomes one of retrieving similar cases from the case base and using the known effort values as a basis of the prediction for the target case. The prediction may be modified by the application of rules, typically obtained from a domain expert such as an experienced project manager, or by a simple procedure such as finding the mean.

Once the component has been completed and the true effort value is known the case can be added to the case-base. In this way the case-base is enlarged over time and can also follow trends or changes in the underlying problem domain, such as the introduction of new technologies and programming languages. For this reason some similarity measures explicitly include a notion of recency so that newer cases are preferred **[4]**.

3.5 Techniques and subtasks used in CBR "tasks"

3.5.1 Case retrieval

Case retrieval subtasks: Case retrieval's subtasks are referred to as:

- Identify Features.
- Initially Match.
- Search.
- Select.

Identify Feature

To identify a problem may involve simply noticing its input descriptors, but often (and particularly for knowledge-intensive methods) a more elaborate approach is taken, in which an attempt is made to "understand" the problem within its context. Unknown descriptors may be disregarded or requested to be explained by the user.
In PROTOS, for example, if an input feature is unknown to the system, the user is asked to supply an explanation that links the feature into the existing semantic network.

To understand a problem involves filtering out noisy problem descriptors, to infer other relevant problem features, to check whether the feature values make sense within the context, to generate expectations of other features. Other descriptors than those given as input, may be inferred by using a general knowledge model, or by retrieving a similar problem description from the case base and use features of that case as expected features. Checking of expectations may be done within the knowledge model (cases and general knowledge), or by asking the user.

Initially Match

The task of finding a good match is typically split into two subtasks:

- An initial matching process which retrieves a set of plausible candidates.
- A more elaborate process of selecting the best one among these.

The latter is the Select task, described below. Finding a set of matching cases is done by using the problem descriptors (input features) as indexes to the case memory in a direct or indirect way. There are in principle three ways of retrieving a case or a set of cases:

- By following direct index pointers from problem features.
- By searching an index structure.
- By searching in a model of general domain knowledge.

PATDEX implements the first strategy for its diagnostic reasoning, and the second for test selection. A domain-dependent, but global similarity metric is used to assess similarity based on surface match. Dynamic memory based systems takes the second approach, but general domain knowledge may be used in combination with search in the discrimination network.

PROTOS and CREEK combines one and three, since direct pointers are used to hypothesize a candidate set which in turn is justified as plausible matches by use of general knowledge. Cases may be retrieved solely from input features, or also from features inferred from the input. Cases that match all input features are good candidates for matching, but (depending on the strategy) cases that match a given fraction of the problem features (input or inferred) may also be retrieved.
PATDEX uses a global similarity metric, with several parameters that are set as part of the domain analysis. Some tests for relevance of a retrieved case is often executed, particularly if cases are retrieved on the basis of a subset of features. For example, a simple relevance test may be to check if a retrieved solution conforms with the expected solution type of the new problem. A way to assess the degree of similarity is needed, and several "similarity metrics" have been proposed, based on surface similarities of problem and case features. Similarity assessment may also be more knowledge-intensive, for example by trying to understand the problem more deeply, and using the goals and constraints from this elaboration process to guide the matching. Another option is to weigh the problem descriptors according to their importance for characterizing the problem, during the learning phase.
In PROTOS, for example, each feature in a stored case has assigned to it a degree of importance for the solution of the case. A similar mechanism is adopted by CREEK, which stores both the predictive strength (discriminatory value) of a feature with respect to the set of cases, as well as a features criticality, i.e. what influence the lack of a feature has on the case solution.

Select

From the set of similar cases, a best match is chosen. This may have been done during the initial match process, but more often a set of cases are returned from that task. The best matching case is usually determined by evaluating the degree of initial match more closely. This is done by an attempt to generate explanations to justify non-identical features, based on the knowledge in the semantic network. If a match turns out not to be strong enough, an attempt to find a better match by following difference links to closely related cases is made. This subtask is usually a more elaborate one than the retrieval task, although the distinction between retrieval and elaborate matching is not distinct in all systems. The selection process typically generates consequences and expectations from each retrieved case, and attempts to evaluate consequences and justify expectations. This may be done by using the system's own model of general domain knowledge, or by asking the user for confirmation and additional information. The cases are eventually ranked according to some metric or ranking criteria. Knowledge-intensive selection methods typically generate explanations that support this ranking process, and the case that has the strongest explanation for being similar to the new problem is chosen. Other properties of a case that are considered in some CBR systems include relative importance and discriminatory strengths of features, prototypicality of a case within its assigned class, and difference links to related cases.

Case retrieval techniques

The two most widely used techniques of case retrieval are:

- Nearest-neighbor retrieval.
- Inductive retrieval.

Nearest-neighbor retrieval (NNR) is a technique to measure how similar the target case is to a source case (Watson - 1997). It processes retrieval of cases by comparison of a collection of weighted attributes in the target case to source cases in the CBR library. If there is no matched case in the CBR library, CBR system will return the nearest matched source case. The return of the nearest case match can be represented by the following equation (Watson - 1997):

$$\text{Similarity}(T, S) = \sum_{i=1}^{n} f(Tj, Si) \times Wi$$

where

T is the target case
S is the source case
n is the number of attributes in each case
I is an individual attribute from 1 to n
f is a similarity function for attribute I in cases T and S
w is the importance weighting of attribute I

Figure 6 – Near Neighbor Retrieval (NNR) equation

The equation of the NNR represents the sum of similarity of the target case to the source case for all attributes multiplied by the importance weighting of individual attributes. The CBR system therefore retrieves a meaningful case that may provide a detailed solved problem description to a new problem. However, the NNR technique is not efficient. This is because whenever new cases are introduced, indexing needs to be performed and it could affect efficiency.

Inductive retrieval (IR) is a technique to extract rules or construct decision trees from the past cases (Watson - 1997). This technique processes a target case based on indexed source cases. The source cases are normally indexed by keywords and stored into a set of cases. The set of cases are divided into a decision tree structure. If target case is not found in the decision tree at runtime, the CBR system may not retrieve a source case. Aamodt and Plaza (1994) and Watson (1997) suggest the use of a combination of these two techniques in which inductive indexing is used to retrieve a set of matching cases and then the nearest neighbour retrieval is used to rank the cases in the set according to their similarity to the target case. Indexes are commonly used in file and database systems to speed up retrieval and optimize accessibility of data.

3.5.2 Case reuse

Case reuse subtasks:

The reuse of the retrieved case solution in the context of the new case focuses on two aspects:

- The differences among the past and the current case.
- What part of a retrieved case can be transferred to the new case.

Copy

In simple classification tasks the differences are abstracted away (they are considered non relevant while similarities are relevant) and the solution class of the retrieved case is transferred to the new case as its solution class. This is a trivial type of reuse.
However, other systems have to take into account differences in (a) and thus the reused part (b) cannot be directly transferred to the new case but requires an adaptation process that takes into account those differences.

Adapt

There are two main ways to reuse past cases:

- **Transformational reuse**: Reuse the past case solution.
- **Derivational reuse**: Reuse the past method that constructed the solution.

In **transformational reuse** the past case solution is not directly a solution for the new case but there exists some knowledge in the form of transformational operators {T} such that applied to the old solution they transform it into a solution for the new case. A way to organize this {T} operators is to index them around the differences detected among the retrieved and current cases. An example of this is CASEY, where a new causal explanation is built from the old causal explanations by rules with condition-part indexing differences and with a transformational operator {T} at the action part of the rule. Transformational reuse does not look how a problem is solved but focuses on the equivalence of solutions, and this requires a strong domain-dependent model in the form of transformational operators {T} plus a control regime to organize the operators application.

Derivational reuse looks at how the problem was solved in the retrieved case. The retrieved case holds information about the method used for solving the retrieved problem including a justification of the operators used, subgoals considered, alternatives generated, failed search paths, etc. Derivational reuse then re-instantiates the retrieved method to the new case and “replays” the old plan into the new context (usually general problem solving systems can be seen here as planning systems). During the replay successful alternatives, operators, and paths will be explored first while filed paths will be avoided, new subgoals are pursued based on the old ones and old subplans can be recursively retrieved for them.

An example of derivational reuse is the Analogy/Prodigy system that reuses past plans guided by commonalties of goals and initial situations, and resumes a means-ends planning regime if the retrieved plan fails or is not found.

Case reuse techniques:

Case storage, which is often referred to as case-memory or memory organisation, is used in the case reuse phase of the CBR cycle. It replicates the conceptual view of case representation in most storage devices. Cases are often reproduced to increase the quality of the solutions of the already solved problems in a given set of circumstances. Stored cases are also used for future reference (Leake - 1995).
It is worth noting that reusable case is more user-acceptable because its solution has already been accepted and convinced by the previous user. Watson (1997) suggests that the case representation should be characterised using indexes. The intention of characterised case representation is to balance between the storing methods and their indexes in order to simplify accessibility and retrieval of relevant cases (Watson - 1997).

3.5.3 Case revise

Case revise subtasks:

When a case solution generated by the reuse phase is not correct, an opportunity for learning from failure arises. This phase is called case revision and consists of two tasks:

- Evaluate the case solution generated by reuse. If successful, learning from the success.
- Repair the case solution using domain-specific knowledge.

Evaluate solution

The evaluation task takes the result from applying the solution in the real environment (asking a teacher or performing the task in the real world). This is usually a step outside the CBR system, since it (at least for a system in normal operation) involves the application of a suggested solution to the real problem. The results from applying the solution may take some time to appear, depending on the type of application.
In a medical decision support system, the success or failure of a treatment may take from a few hours up to several months. The case may still be learned, and be available in the case base in the intermediate period, but it has to be marked as a non-evaluated case.

A solution may also be applied to a simulation program that is able to generate a correct solution. This is used in CHEF, where a solution (i.e. a cooking recipe) is applied to an internal model assumed to be strong enough to give the necessary feedback for solution repair.

Repair fault

Case repair involves detecting the errors of the current solution and retrieving or generating explanations for them. The best example is the CHEF system, where causal knowledge is used to generate an explanation of why certain goals of the solution plan were not achieved. CHEF learns the general situations that will cause the failures using an explanation-based learning technique.

This is included into a failure memory that is used in the reuse phase to predict possible shortcomings of plans. This form of learning moves detection of errors in a post hoc fashion to the elaboration plan phase were errors can be predicted, handled and avoided. A second task of the revision phase is the solution repair task. This task uses the failure explanations to modify the solution in such a way that failures do not occur. For instance, the failed plan in the CHEF system is modified by a repair module that adds steps to the plan that will assure that the causes of the errors will not occur.

The repair module possesses general causal knowledge and domain knowledge about how to disable or compensate causes of errors in the domain. The revised plan can then be retained directly (if the revision phase assures its correctness) or it can be evaluated and repaired again.

Case revise techniques:

Adaptation is used in the case revision phase of the CBR cycle. It is a technique to alter the retrieved case to reproduce a new solution to new problem. The retrieved case can be changed so that it can be presented to suit new use. The purpose of case adaptation is to improve the CBR system's overall problem solving ability using newly introduced cases for future use.

The two most widely used techniques of case adaptation are: structural adaptation and derivational adaptation.

- **Structural adaptation** (SA): Is a technique to apply adaptation rules or formulas directly to the stored solution in the CBR library. Once a case has been applied by adaptation rules or formulas, the CBR system adapts the case as a match with the new problem.
- **Derivational adaptation** (DA): Is a technique to reuse the rules or formulas that generated the original solution to produce a new solution to the current problem (Watson - 1997). The retrieved solution then must be stored as an additional case in the CBR library so that it reproduces a new solution to the new case **[12]**.

3.5.4 Case retain

Case retain subtasks:

This is the process of incorporating what is useful to retain from the new problem solving episode into the existing knowledge. The learning from success or failure of the proposed solution is triggered by the outcome of the evaluation and possible repair. It involves:

- Selecting which information from the case to retain.
- In what form to retain it.
- How to index the case for later retrieval from similar problems.
- How to integrate the new case in the memory structure.

Extract

In CBR the case base is updated no matter how the problem was solved. If it was solved by use of a previous case, a new case may be built or the old case may be generalized to subsume the present case as well. If the problem was solved by other methods, including asking the user, an entirely new case will have to be constructed. In any case, a decision need to be made about what to use as the source of learning. Relevant problem descriptors and problem solutions are obvious candidates. But an explanation or another form of justification of why a solution is a solution to the problem may also be marked for inclusion in a new case.

In CASEY and CREEK, for example, explanations are included in retained cases, and reused in later modification of the solution. CASEY uses the previous explanation structure to search for other states in the diagnostic model which explains the input data of the new case, and to look for causes of these states as answers to the new problem. This focuses and speeds up the explanation process, compared to a search in the entire domain model. The last type of structure that may be extracted for learning is the problem solving method, i.e. the strategic reasoning path, making the system suitable for derivational reuse. Failures, i.e. information from the revise task, may also be extracted and retained, either as separate failure cases or within total problem cases. When a failure is encountered, the system can then get a reminding to a previous similar failure, and use the failure case to improve its understanding of - and correct - the present failure.

Index

The “indexing problem” is a central and much focused problem in case-based reasoning. It amounts to deciding what type of indexes to use for future retrieval, and how to structure the search space of indexes. Direct indexes, skips the latter step, but there is still the problem of identifying what type of indexes to use. This is actually a knowledge acquisition problem, and should be analyzed as part of the domain knowledge analysis and modeling step.

A trivial solution to the problem is of course to use all input features as indices. This is the approach of syntax-based methods within instance-based and memory-based reasoning. In the memory-based method of CBR-Talk, for example, relevant features are determined by matching, in parallel, all cases in the case-base, and filtering out features that belong to cases with few features in common with the problem case.

In CASEY, a two-step indexing method is used. Primary index features are general causal states in the heart failure model that are part of the explanation of the case. When a new problem enters, the features are propagated in the heart failure model, and the states that explain the features are used as indices to the case memory. The observed features themselves are used as secondary features only.

Integrate

This is the final step of updating the knowledge base with new case knowledge. If no new case and index set has been constructed, it is the main step of Retain. By modifying the indexing of existing cases, CBR systems learn to become better similarity assessors. The tuning of existing indexes is an important part of CBR learning.

Index strengths or importances for a particular case or solution are adjusted due to the success or failure of using the case to solve the input problem. For features that have been judged relevant for retrieving a successful case, the association with the case is strengthened, while it is weakened for features that lead to unsuccessful cases being retrieved. In this way, the index structure has a role of tuning and adapting the case memory to its use.

PATDEX has a special way to learn feature relevance: A relevance matrix links possible features to the diagnosis for which they are relevant, and assign a weight to each such link. The weights are updated, based on feedback of success or failure, by a connectionist method.

In knowledge-intensive approaches to CBR, learning may also take place within the general conceptual knowledge model, for example by other machine learning methods or through interaction with the user. Thus, with a proper interface to the user (whether a competent end user or an expert) a system may incrementally extend and refine its general knowledge model, as well as its memory of past cases, in the normal course of problem solving. This is an inherent method in the PROTOS system, for example. All general knowledge in PROTOS is assumed to be acquired in such a bottom-up interaction with a competent user. The case just learned may finally be tested by re-entering the initial problem and see whether the system behaves as wanted **[4]**.

Chapter 4: CBR software tools

4.1 Generally

4.1.1 A classification of CBR applications

In 1995, Althoff and his colleagues suggested a classification method of CBR application as shown in **Figure 7**. Under this classification scheme, CBR applications can be classified into two categories:

- Classification tasks.
- Synthesis tasks.

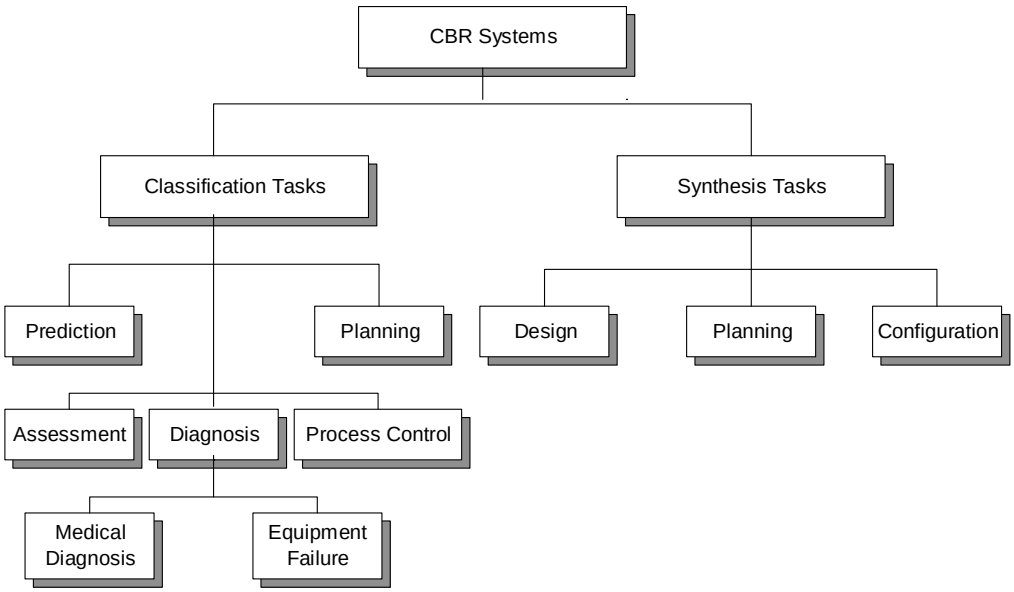


Figure 7 - A classification hierarchy of CBR applications (Althoff et al. - 1995)

Classification tasks are very common in business and everyday life. A new case is matched against those in the case-base from which an answer can be given. The solution from the best matching case is then reused. In fact, most commercial CBR tools support classification tasks.

Synthesis tasks attempt to get a new solution by combining previous solutions and there are a variety of constraints during synthesis. Usually, they are harder to implement. CBR systems that perform synthesis tasks must make use of adaptation and are usually hybrid systems combining CBR with other techniques (Watson - 1997) **[19]**.

4.1.2 The views of theorists and software vendors for CBR and CBR tools

Theoreticians might argue that the current surge in interest in CBR is due to the intuitive nature of CBR and because it may closely resemble human reasoning. Software vendors might argue that it is because CBR tools have made the theory practically feasible. There is truth in both views but certainly the tools have made a contribution. This section reviews most of the currently available major CBR tools. The tools are dealt with in alphabetical order. The section concludes with a table that summarises the functionality of the tools reviewed **[17]**.

4.2 CBR software tools and applications

4.2.1 ART*Enterprise

ART*Enterprise is the latest incarnation of ART Inference Corporation's flagship development product (currently in Version 2.0 beta). Inference Corporation1 based in California are one of the oldest established vendors of AI tools. Inference market ART*Enterprise as an integrated, object-oriented applications development tool designed for MIS developer's offering a variety of representational paradigms including:

- A procedural programming language.
- Objects supporting multiple inheritance, encapsulation and polymorphism.
- Rules.
- Cases.

This is all packaged with a GUI builder, version control facilities, and an impressive ability to link to data repositories in proprietary DBMS formats conforming to the ODBC standard for developing client-server applications. Moreover, ART*Enterprise offers cross-platform support for most operating systems, windowing systems and hardware platforms.

The CBR component in ART*Enterprise is essentially the same as that in CBR2 (or rather vice-versa since CBR2 uses code from ART to provide its CBR functionality). However, because developers have direct access to the CBR functionality ART*Enterprise is more controllable than in CBR2.

In conclusion, ART*Enterprise is perhaps the ideal tool for embedding CBR functionality within a corporate wide information system. Although the CBR functionality itself is more limited than some tools (i.e., cases are flat value: attribute pairs and there is no support for inductive indexing) the proven knowledge representational abilities of ART will make it a good tool for performing complex case adaptation. It can be assumed that since ART*Enterprise uses similar code to CBR2 that its case retrieval times will be as fast (or faster) than those recorded in Althoff and al's experiments. A small word of warning is needed: although ART*Enterprise is available on the PC platform under MS Windows it is very demanding on resources - a fast 486 or Pentium with a minimum of 32MB of RAM is required.

4.2.2 Case-1

Case-1 is a new CBR tool from Astea International (a beta release of Version 1.0 was reviewed). The company has a background in providing integrated sales, support and service systems. Case-1 was obviously developed with CBR Express in mind and it shares many of its features. Cases are represented as free form text describing a problem, a set of weighted questions that can confirm or reject a case and a set of solutions. As with CBR2 cases can be authored by people who have no programming experience. Cases are stored in a relational database (Watcom) and the interface is developed using Visual Basic (Case-1 runs under MS Windows). Case-1 does score over CBR Express in letting case authors have easy access to the lexicon of words ignored during text matching, however the product is not as mature as CBR2 and does not seem to offer any significant functional improvements. However, the tool is well integrated with Astea's other customer support tools and therefore if you are already a client of Astea you may be advised to use Case-1.

4.2.3 CasePower

Formerly called Induce-it from Inductive Solutions Inc. CasePower builds its cases within the spreadsheet environment of Microsoft Excel. CasePower is a specialised tool for constructing Excel spreadsheets that can be analysed using CBR. Within the limited confines of Excel it provides basic CBR functionality mainly suitable for numeric applications. Symbolic data can be represented as ordered hierarchies that are mapped to numerical ranges. However, for more complex non-numerical applications another CBR tool may be preferable.

CasePower uses nearest neighbour retrieval and it reduces search time by calculating an index for each case in advance. This can be a lengthy process for a large case-base but it does reduce retrieval times. The system simply calculates the index for the new case and compares it against the pre-calculated indices of the case-base. If a new case is to be retained, the entire set of case indices must be recalculated. Adaption can be performed using Excel formulae and macros. Similarly all the other features of Excel are available such as graphing, reporting and DDE.

4.2.4 CBR2 (CBR Express, CasePoint, Generator & Tester)

Produced by Inference Corporation, the CBR2 family of products are certainly the most successful CBR products to date with over 500,000 licences sold world-wide. CBR2 is specifically tailored to the vertical market of customer support help desks. The CBR2 family of tools having the following roles:

- **CBR Express** is a development or authoring environment for cases, it also features a customer call tracking module.
- **CasePoint** is a search engine for case-bases developed using CBR Express.
- **Generator** is a tool that automates the creation of cases-bases from sets of MS Word or ASCII files.
- **Tester** is a tool that provides a variety of metrics for case-base developers using CBR2 .CBR2 uses a simple case structure of flat records. Cases comprise a title, a description, a set of weighted questions (effectively value: attribute pairs), and a set of actions. Cases can be stored in almost any proprietary database format, although the Raima database format is supplied as a default. CBR2 is network ready and case-bases can be shared across an organisation's network CBR2 uses nearest neighbour matching to initially retrieve cases by matching a users free text query against the title and descriptions of cases in the case-base. A key feature of CBR2 is its ability to handle free-form text. This was felt to be vital to the help desk market since it lets customers describe their problems in their own words rather than being taken through a decision tree style question and answer session. CBR2 ignores words such as: and, or, I, there, etc., it can use synonyms, and represents words as a set of trigrams. The trigram for cartridge is: CAR, ART, RTR, TRI, RID, IDG, DGE. The use of trigrams means that CBR2 is very tolerant of spelling mistakes and typing errors such as letter transpositions. The trigrams for cartridge and cartrigde will still match closely. Although there are obvious problems with this lexical approach it is nonetheless surprisingly powerful and very useful for CBR2's market.
  After an initial set of relevant case are retrieved using the textual matching retrieval then becomes knowledge guided as questions are be asked to focus case retrieval. Developers using CBR2 use an interface (CBR-Express) that deals with all programming elements of case creation and editing resulting in a syntax free environment that lets people without programming experience quickly develop case-bases. The interface of CBR Express is constructed using Asymetrix ToolBook version 1.5, and developers with the authoring version of

ToolBook can obtain access to the source code of the interface to customise it. However, this is a non-trivial task and should only be attempted by experienced ToolBook developers. Otherwise there is a real risk of compromising the functionality of the system (this advice also applies to KATE).

During retrieval CBR2 examines a user's free form text entry and matches this against cases' titles and descriptions. This results in the retrieval of a set of cases. A list of ranked solutions with likelihood values is generated from the cases and the user is offered these along with a set of questions. Answers to these questions help narrow the number of cases that match leading to a more accurate solution that is presented to the user. In the event of a solution not being reached (CBR2 has a customisable threshold value) or if a solution is not satisfactory the CBR cycle is closed by using the concept of an unresolved case (CASE-1 has borrowed this concept from CBR2). An unresolved case saves the entire transcript of the consultation so the case-base administrator can subsequently find out what that case's solution was and modify the unresolved case to create a new case.

If you want to integrate or embed CBR2 it is probably more efficient to use CasePoint as a DDE server application (it is also now available as a DLL). As a delivery vehicle CasePoint has many advantages over CBR Express. A criticism of nearest neighbour matching is that if a case base were large and if cases had many features it is not an efficient process. However, the matching algorithm that CasePoint uses is extremely fast. CasePoint also supports the use of a rule-file that identifies keywords in the query text and automatically answers certain questions. It can also order questions so that they best discriminate between cases under consideration.

4.2.5 Eclipse

From Haley Enterprises Eclipse is a close relative of ART. The forward chaining functionality of ART, written in LISP, was re-implement in C by NASA, entering the public domain as the language CLIPS. In the late eighties Paul Haley, the former Chief Scientist of Inference, developed a new ART-like language compatible with CLIPS. This became Eclipse. Like ART, Eclipse offers objects, only this time fully compatible C++ objects, and optimised forward chaining using the Rete algorithm (an efficient pattern matching algorithm for implementing production rule systems). Eclipse is available for the DOS operating system, MS Windows, UNIX platforms and certain mainframe environments. The Easy Reasoner is a module within Eclipse that offers CBR functionality similar to that of the Inference products. Eclipse is only available as a C library (i.e., there is no development interface) and is therefore only suitable for experience C programmers. Eclipse supports the usual range of variable types and offers similar text handling facilities to ART (i.e., ignoring noise words and using trigrams to cope with spelling mistakes). Interestingly Eclipse also uses stems to identify, for example, that *magnetically* and *magnetic* all stem from *magnet*. Once cases have been retrieved they can be asserted as Eclipse objects for adaptation by its rule-base.

4.2.6 ESTEEM

ESTEEM, from Esteem Software Inc., was originally in written in Intellicorp's Kappa-PC. Version 1.4 is now written in C++ and has its own inference engine enabling developers to create adaptation rules. It supports case hierarchies that help narrow the search. It also supports applications that access multiple case-bases and nested cases. This means that one can reference another case-base through an attribute slot in a case. ESTEEM also provides control of the induction process (ID3) through feature counting, weighted feature computation, inferred computation. Nearest neighbour matching is also supported.

ESTEEM runs on PC Windows and represents exceptionally good value for money ($495). The developers interface comprises five simple editors that define cases, customise similarity assessment and retrieval, define adaptation rules, import data from ASCII files of databases, and create simple form-based user interfaces. Version 1.4 now supports multimedia as a feature type and can be used as a MS DDE server for application embedding.

The source Kappa-PC (KAL) code is available from ESTEEM and thus lets developers embed CBR functionality within the Kappa environment. Moreover, Kappa-PC code can be exported as C code which can then be compiled (using a C compiler) into a stand alone .EXE file.

4.2.7 KATE

KATE, produced by Acknosoft in Paris, is made up of a set of tools sometimes referred to as CASECRAFT (i.e., KATE-INDUCTION, KATE-CBR, KATE-EDITOR and KATE-RUNTIME). Development should be on PC Windows (as the interface components are made with ToolBook) but deployment can be on PC Windows, Mac or SUN.

KATE-INDUCTION is an ID3 based induction system that supports an object representation for cases. Cases can be imported from many database and spreadsheet formats. The induction algorithm is very tolerant of missing data and can make use of background knowledge. Retrieval using trees generated by induction algorithm is extremely fast.

KATE-CBR is the nearest neighbour component of the suite. Users can customise similarity assessments and since it supports the same object hierarchies as KATE-INDUCTION the two techniques can be combined.

KATE-EDITOR is a set of C DLL's that are integrated with ToolBook to form a customisable developer's interface. In particular easy forms can be developed to assist case entry.

KATE-RUNTIME is another set of interface utilities that can be customized with ToolBook to deliver an application. KATE can also be delivered as embedded C code. KATE tools are a powerful set of well integrated CBR tools. Retrieval is extremely fast (even with large case bases) and can be customised by experienced developers. KATE is one of the few tools to include automatic testing routines.

4.2.8 ReCall

ReCall is a CBR trademark of the Paris based AI company ISoft. This tool offers a combination of nearest neighbour and inductive case retrieval. ReCall is coded in C++ and is available on the PC under Windows 3.1, on UNIX Workstations under Motif, for: SUN, IBM RS6000, BULL DPX20, HP series 700, and DEC Alpha. It is designed on an open architecture allowing users to add CBR functionality to their own applications.

Recall presents an object-oriented language with taxonomies, inheritance and multiple-inheritance mechanisms, typed descriptors, facets, deamons, and relationships between objects (individual cases are represented as instances). This allows users to specify complex domain knowledge in a structured but modular way, and to describe cases having noisy, incomplete and uncertain descriptions since feature values can be inherited. Recall provides multiple hierarchical indices that are used for organisation purposes and for efficient case retrieval. ReCall provides different methods for automatically analysing the case base providing for selection of indices as well as their organisation. However, experienced developers can impose their own organisation.

Automatic procedures are based on inductive techniques. The automatic procedures takes into account the domain knowledge defined in the cases, helping users to develop applications interactively. Similarity functions take into account both the properties and values of descriptors, as well as structural differences between cases. ReCall uses a variant of a nearest-neighbour algorithm that improves similarity computations.

ReCall supports two different adaptation mechanisms: a default adaptation mechanism based on a voting principal, and user defined adaptation rules. As ReCall is based on C++, external function calls can provide more complex adaptation. ReCall can be interfaced to external applications in particular with data bases and since ReCall is available as a C++ library, CBR functionality can be integrated with other applications. Through the use of specialised graphic editors, the developer can define objects, relationships between objects, taxonomies, deamons and adaptation rules. A tree editor allows the user to interact directly on the case organisations in order to control and modify indices. A user mode allows developers to write adaptation rules or daemons, whilst a developer mode gives access to an interpreted language. ISoft have also recently released a product called AC2 that uses case representation and induction facilities of ReCall for data mining.

4.2.9 ReMind

Produced by Cognitive Systems Inc. ReMind was developed with support from the US DARPA programme. It was originally developed for the Macintosh and has since been ported to MS Windows and some UNIX platforms. ReMind offers template, nearest neighbour, inductive, and knowledge-guided inductive retrieval.

The template retrieval supports simple SQL-like queries returning all cases that fall within set parameters. The nearest neighbour retrieval is informed by user defined importance weightings that can be placed on case features. Inductive retrieval can be done automatically by ReMind with no user involvement or the user can create a qualitative model to guide the induction algorithm (based on CART).

Qualitative models (Q-models) are created graphically to indicate which concepts (case features) are dependent on other concepts. Qualitative weightings can be placed on these dependencies and ReMind then uses the Q-model to guide the induction algorithm (hence knowledge-guided induction) resulting in decision trees that more closely reflect the causal relationship of concepts in the cases. Interestingly, different qualitative models can be created to explore different theories about the domain or to allow what-if questions to be asked.

In ReMind case adaptation is provided by creating adaptation formulae that adjust values based on the difference between the retrieved and the new case. These are also created graphically using a visual programming technique. Although this takes a little getting used to the extremely close typing of case features combined with the close typing of the operators does reduce syntax errors.

ReMind is available as a C library, for embedding in other applications, and as an interactive development environment. ReMind is a flexible CBR tool offering a wide range of case-retrieval methods along with interesting concepts such as Q-models and visual adaptation formulae. It does not have the powerful text handling features of Inference’s products and Eclipse, though it does provide an elementary natural language capability via a lexicon of terms that can be mapped to an ordered symbol hierarchy. However, in general users are forced to select rather than describe a situation. ReMind is perhaps the most flexible of CBR tools currently on the market.

In particular the use of background knowledge in the form of Q-models and prototypes to inform the induction algorithm can greatly improve the efficiency of cluster trees. ReMind, however has two major limitations:

- Cases are stored in an entirely hidden way, they cannot be exported or viewed by any other application.
- The nearest neighbour algorithm in ReMind is slow, and is only suitable for small case-bases. Inductive retrieval is very much faster, but building a large cluster tree is extremely slow taking several hours for a several thousand cases.

4.2.10 S3-Case

S3-Case is part of the German company techInno's S3 environment for systems maintenance running on PC Windows, Mac, OS/2 and various UNIX platforms. Written in SMALLTALK it supports an object oriented model with inductive (ID3) and nearest neighbour retrieval and adaptation via forward chaining inference engine. Rule can also be used to prune the search space before retrieval. A simple user interface can be customised to suit user needs. Experienced SMALLTALK developers can embed or extend the functionality of S3-CASE **[17]**.

Product *Vendor*	Platforms	Representation	Retrieval	Adaptation	Interface	Other Comments	Price (approx.)
ART*Enterprise *Inference Corp.*	a wide variety of PC, workstation, DEC, IBM HP and mainframe environments	flat value attribute pairs supporting a full range of variable types	nearest neighbour but can be augmented using ART's programming environment	functions, rules and other knowledge-based techniques	fully featured GUI builder	excellent data integration with most DBMS formats and version control	PC version $12,000
CASE-1 *Astea International*	PC Windows	flat records supporting text and weighted questions	nearest neighbour and knowledge-guided retrieval	no adaptation features	interface cannot be customised	designed for help desks Version 1.0 is now shipping	???
CasePower *Inductive Solutions Inc.*	PC Windows, Macs OS/2	MS Excel Spreadsheet ordered symbol hierachies and nested cases	nearest neighbour	via Excel functions and macros	Excel interface	Excel must be bought to use this product	less than $1000
CBR2 *Inference Corp.*	PC Windows and MVS version	flat records supporting text and weighted questions	nearest neighbour and knowledge-guided retrieval	no adaptation features	ToolBook interface of CBR-Express can be customised	runtime, testor and generator modules available	PC version CBR Express $10,000
Eclipse *The Haley Enterprise*	any ANSII C environment	flat value attribute pairs full range of variable types	nearest neighbour	functions, rules and other knowledge-based techniques	no interface it is only supplied as a C library	this ART-like product is very fast	???
ESTEEM *Esteem Software Inc.*	PC Windows UNIX/X Motif	cases can be ordered hierarchically and can be nested	nearest neighbour and inductive (ID_3)l retrieval	functions and rules	GUI Builder	now supports DB access and multimedai	$495
KATE *AcknoSoft*	PC Windows & UNIX	hierachical cases	nearest neighbour and induction (ID_3)		ToolBook interface can be customised	available as a C library for embedding	???
ReCall *ISoft*	PC Windows & UNIX	hierachical cases with relationships	nearest neighbour and induction	deamons	graphical development environment	available as a C++ library for embedding	$9,000
ReMind *Cognitive Systems Inc.*	PC Windows Mac & UNIX	nested flat cases & ordered symbol hierarchies	nearest neighbour, induction (CART) & template retrieval	formulae	development interface can be customised	available as a C library for embedding	$6,000
S_3-CASE tecInno GmbH	PC Windows Mac & UNIX	object oriented hierarchical cases	nearest neighbour and induction (ID_3)	rules	customisable interface	written in SMALLTALK	???

Figure 8 - Overview of CBR software tools and applications

4.3 CBR shells and development environments

CBR shells are kind of application generators with graphical user interface. They can be used by nonprogrammer users but the extension or integration of new components in these tools is not possible. There is a clear difference between a CBR application and a CBR shell. A CBR application is a direct implementation of CBR methodology to a specific domain problem in order to solve this problem. On the other hand, a CBR shell is an application that enables developers to develop a domain specific CBR application.

In the late 1980's, the U.S. DARPA program funded a series of workshops on CBR and the development of a CBR tools (DARPA – 1991). This tool became Cognitive System's ReMind and marked the transition of CBR from purely academic research in cognitive science and artificial intelligence into the commercial area.

Many CBR shells have been developed to make the theory practically feasible (Watson – 1997). Figure 9 shows a summary of the key features of the major CBR shells. In addition to the previous tools there are three major CBR development environments CASPIAN, CASUEL and CBR-Works.

Product	Platform	Representation	Retrieval	Interface
ART Enterprise	PC, Workstation	flat attribute:value pairs supporting a full range of variable types	Nearest-neighbor	Fully featured GUI builder
CaseAdvisor	PC Windows	Flat records supporting text and weighted questions	Nearest-neighbor and knowledge-guided	Use Netscape
CBR3	PC Windows	Flat records supporting text and weighted questions	Nearest-neighbor and knowledge-guided	CasePoint available as a DLL or API and CGI scripts
Eclipse	Any ANSI C environment	Flat attribute	Nearest-neighbor	No interface, only supply as a C library
ESTEEM	PC Windows	Case can be nested	Nearest-neighbor with inductive weight generation	Simple form-based GUI builder
KATE	PC Windows and UNIX	Hierachical cases	Nearest-neighbor and induction	ToolBook interface can be customized

Figure 9 - A summary of the major CBR shells (Watson – 1997)

4.3.1 CASPIAN

CASLPIAN (Pegler & Price – 1996) is CBR tool in the public domain developed at the University of Aberystwyth in Wales. It was used as the CBR component of the Wayland system. It has a simple command line interface, but can be integrated with a GUI front end if required. CASPIAN is written in C and can run on MS-DOS, MAC or UNIX but without the GUI. CASPICAN performs nearest-neighbour matching and used rules for case adaptation. It stores a case-base, including adaptation rules, in ASCII file. An individual case comprises a series of attributes and a solution. CASPIAN has an internal engine sophisticated enough to allow its use in industrial applications.

4.3.2 CASUEL

CASUEL (Manago & al. - 1994), the Common Case Representation language developed by the European INRECA project (Integrated Reasoning from Cases), is the interface language between the INRECA component systems. It is also intended to serve as the interface language between the INRECA integrated system and the external world, and as a standard for exchanging information between classification and diagnostic systems that use cases. CASUEL is a flexible, object-oriented, frame-like language for storing and exchanging descriptive models and case libraries as ASCII files. It is designed to model naturally the complexities of real cases. CASUEL represents domain objects in a class hierarchy using inheritance, slots being used to describe the objects, with typing constraints on slot values, as well as different kinds of relationships between objects. CASUEL also supports rule formalism for exchanging case completion rules and case adaptation rules, as well as a mechanism for defining similarity measures. CASUEL is more concise than flat feature values vectors for representation of objects with a large number of potentially relevant attributes of different types, only a few of which are applicable to any given case. Its use reduces the number of information-gain calculations needed for induction systems or similarity computations required for case-based reasoning. CASUEL does not require applications to use all of them. CASUEL is a keyword-driven language that allows different system components to ignore irrelevant definitions. CASUEL is also open in the sense that new features can be defined, if necessary for a particular kind of application of component (Watson – 1997).

4.3.3 CBR-Works

CBR-Works can be seen as a CBR-shell providing all necessary tools to model, maintain, and consult a case base (Schulz – 1999). CBR-Works comes from the German company TECINNO, running on MS Windows, Mac, OS/2, and various UNIX platforms. Written in SMALLTALK, it supports an object-oriented model and flexible retrieval methods. It also supports the definition of concept and type hierarchies to help define similarity of symbolic concepts. CBR-Works includes an attribute editor, a rule editor, similarity criteria editor, distributed processing support and is easily integrated to existent applications. CBR-Works can import case-bases from Microsoft Excel and in the CASUEL case format. **Figure 10** shows a summary of the three discussed CBR development environments.

Product	Platform	Representation	Retrieval	Interface
CASPIAN	DOS, MAC, or UNIX	Attribute-Value for feature representation	Nearest-neighbor	Can be integrated with a GUI
CASUEL	Portable	Frame-like language for storing and exchanging descriptive models as ASCII files	Not Applicable	Not Applicable
CBR-Works	MS Windows, MAC, or UNIX	Flat records supporting text and weighted questions	Nearest-neighbor with support of feature weights	Fully featured GUI

Figure 10 - A summary of the major CBR development environments

4.4 Case Based Reasoning object-oriented frameworks

Most of the CBR tools presented in scientific papers aim to provide Application Programming Interfaces (APIs) which provide a set of functions that deal with CBR algorithms and methodologies. They intended to help programmers to embed these APIs in their application development (Jaczynski & Trousse – 1998). Usually these APIs can be extended by the programmer to modify the provided algorithms. However, none of these tools are designed to provide an open development environment that lead users to more uniform tool at the level of design. The concept of object-oriented frameworks has been introduced in the late 80's and has been defined as "a set of classes that embodies an abstract design for solutions to a family of related problems, and supports reuses at a larger granularity than classes." (Johnson & Foote - 1988).
The goal of a framework is to capture a set of concepts related to a domain and the way they interact. In addition a framework is in control of a part of the program activity and calls specific application code by dynamic method binding. A framework can be viewed as an incomplete application where the user only has to specify some classes to build the complete application.
Frameworks allow the reuse of both code and design for a class of problems, giving the ability to non-expert to write complex applications quickly. Frameworks also allow the development of prototypes which could be extended further on by specialization or composition. A framework once understood, it can be applied in a wide range of domain, and can be enhanced by the adding of new components.
Before exploring the CBR frameworks, there are some points inside the framework that need to be addressed (Jimenez-Diaz & Gomez-Albarran – 2004):

- Users must know the type of application which the framework can be used. Users should understand whether or not the application could be developed based on their choice of the framework.
- The mapping between application domain concepts and framework classes should be well studied to avoid the normal indirect mapping between domain entities and framework class.
- The framework users need to know behaviour of elements within the framework in order to identify the hierarchy of classes that will be involved in the design of the application.
- Users need to study carefully the communication between the classes of the framework in order to avoid the integrity problem of the framework.
- Some problems like the duplication of functionality and extension of some parts of the framework can be avoided by the knowledge of framework architecture.

The following is a discussion of three object-oriented CBR frameworks CBR*Tools, CAT-CBR, and JColibri. Their architecture and how CBR methodologies are applied in them, is being discussed.

4.4.1 CBR*Tools

CBR*Tools is an object-oriented framework for CBR which is specified with the Unified (Jaczynski – 1998). Modeling Language (UML) notation (Booch – 1994) and written in Java. It offers a set of abstract classes to model the main concepts necessary to develop applications integrating case-based reasoning techniques: case, case base, index, measurements of similarity, reasoning control. It also offers a set of concrete classes which implements many traditional methods (closest neighbours indexing, Kd-tree indexing, neuronal approach based indexing, standards similarities measurements). CBR*Tools contains more than 220 classes divided in two main categories: the core package for basic functionality and the time package for the specific management of the behavioural situations. The programming of a new application is done by specialization of existing classes, objects aggregation or by using the parameters of the existing classes. CBR*Tools delegates each CBR step retrieve, reuse, revise or retain to a different object. Each class defines an abstract interface to a step of the reasoning while the Reasoner class defines how to control the reasoning. The step classes must be specialized to implement a specific reasoning. The Reasoner class allows the implementation of different reasoning control methods. In order to ensure that the reasoning step implementations and the reasoning object are consistent, the ReasonerFactory class is provided. **Figure 11** shows the class diagram of CBR*Tools object model.

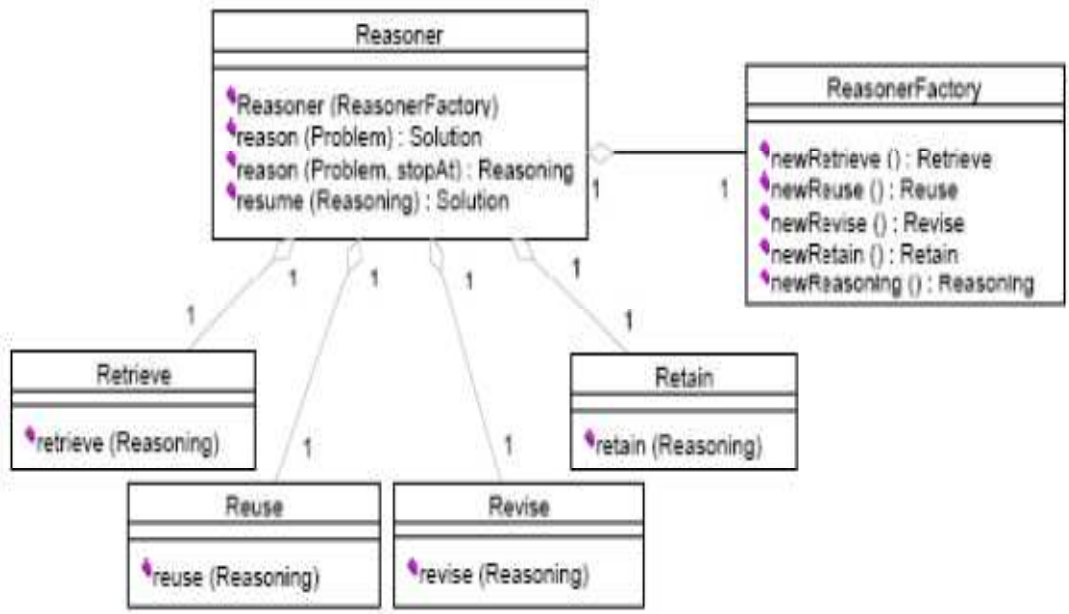


Figure 11 - CBR*Tools Object Model

4.4.2 CAT-CBR

CAT-CBR platform uses a library of CBR components to guide the user in the development of a CBR application (Abasolo & al. - 2002). These components describe the different tasks that can appear in a CBR system and also the problem solving methods that can be applied to these tasks. The CAT-CBR platform has been developed on Noos platform (Arcos – 1997). Noos uses feature terms as representation language.

Universal Problem-solving Methods Language (UPML) has been used to describe the CBR components used inside the framework (Abasolo et al., 2002). Two levels can be differentiated in a component description: a specification level in which UPML is used and an operational level in which the Noos is used.

CAT-CBR uses two processes to enable users to develop a CBR application the configuration process and the operationalization process. The configuration process focuses on selecting different components and connecting them in order to specify an application. CAT-CBR has an interactive tool where users choose the components that need to be included in an application. This tool is built over a CBR system that guides and gives support to users during the configuration process. The operationalization process takes an application specification and generates an executable application. The platform generates a file that links with Noos methods following the structure of the configuration of components. **Figure 12** shows the process of developing a CBR system. It is done in three steps: Configure, enable, and enact.

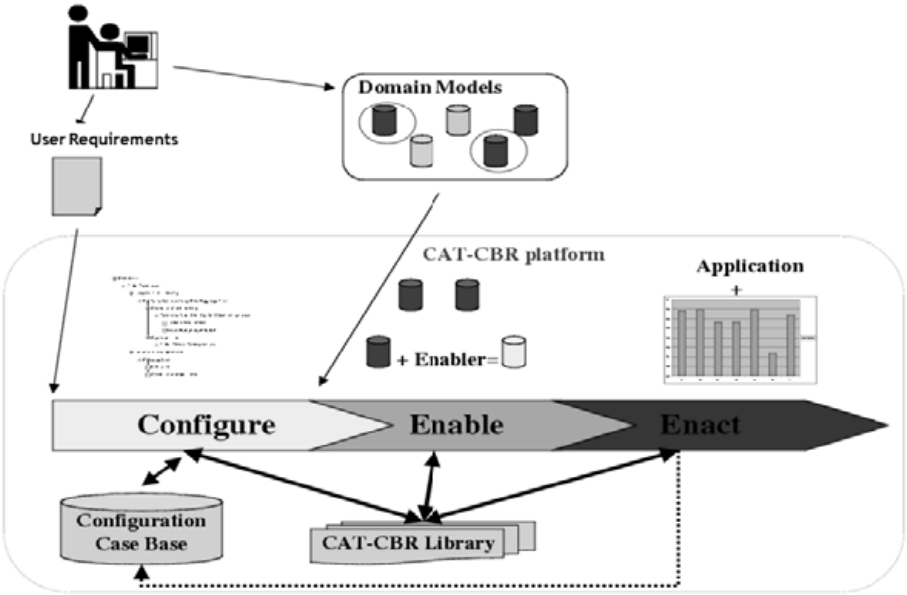


Figure 12 - CAT-CBR process of developing a CBR system (Abasolo & al. – 2002)

The goal of the Configure step is to decide which technique will be used in the CBR system. Only general information about the desired CBR system is required, this information is about general objectives (i.e. classify), or performance characteristics (i.e. noise tolerance). As result of the configure step, users get a configured CBR system, this configured CBR system is a task-method decomposition of components from the CAT-CBR library. This configured system specifies also which models will be used by each method. The goal of the Enable step is to link the configured system with the concrete domain. In this step user have two options, first, they can assign the concrete models that the configuration needs to be carried out, second, they can use methods to acquire these models that the configuration needs and they are not currently available. Finally in the Enacting step, the configuration and models will be translated into an executable code. As the platform is developed over Noos framework the resultant code will be Lisp functions. Once the configuration is operationalize, the application can run to solve new problems.

4.4.3 JColibri

The application framework of JColibri (Bello, Tomas et al. - 2004) comprises a hierarchy of JAVA classes plus a number of XML files. The framework is organized around four main elements: tasks and methods, casebase, cases, and problem solving methods.

Tasks and Methods: XML files explain tasks supported by the framework and the methods to solve these tasks.

Tasks are the key elements that represent the method goal and can identify it by name and description in an XML file. Users can add task to the framework at anytime.

Case Base: JColibri has a memory organization interface that assumes that whole case-base can be read into memory for the CBR to work with it. It is not feasible for big size. JColibri implemented a new interface who allows retrieving cases enough to satisfy a SQL query. A second layer of case base is a data structure which will organize cases after they loads into memory. The two layer approach is efficient enough to allow different strategies for retrieving cases.

Cases: JColibri represent cases in a very simple way. A case is individual which has number of relationships with other individuals. Framework is supported by different data types which define any simple case.

Problem Solving Methods: JColibri deals with the CBR methodology as follows:

- **Retrieval:** Main focus of methods in this category is to find similarity between cases. Similarity function can be parameterized through system configuration.
- **Reuse:** A complete design where case-based and slot-based adaptation can be hooked is provided.
- **Revise:** It is not supported by jColibri framework.
- **Retain:** Process of updating the case base is totally based on implementation of the case base **[10]**.

4.4.4 CLAVIER

CLAVIER is a CBR system developed at Lockheed Missiles and Space Company that was one of the first commercially fielded CBR applications (Hennessy and Hinkle – 1992). This CBR system is used to help autoclave operators in arranging composite aircraft parts for curing in a convection autoclave. CLAVIER acts as a collective memory for Lockheed and as a uniquely useful way of transferring expertise between autoclave operatives (Watson – 1997).

4.4.5 FormTool

FormTool is a CBR system has been used since 1995 by General Electric (GE) for determining what colorants to use for producing a specific color of plastic. FormTool helps GE save lots of money and has become the primary tool for creating color matches at GE Plastics sites (Cheetham and Graf – 1997) **[19]**.

4.5 Web-based CBR

With the advent of the World Wide Web, the Internet provides a powerful approach to assist users in finding information and carrying out interactive access to the useful source. Currently, most CBR systems are loaded on the local machine and run under the individual operating system. This creates problems of inconvenience and high cost (Watson & Gardingen – 1999). Web-based CBR applications offer a promising alternative. The Internet permits access a CBR system with a web browser without time and location limits. It also provides a more powerful interactive style. Web-Based CBR also suggests several other advantages over the traditional CBR systems:

- Platform-independence. Users need not worry about the platform since the Internet permits a single shared web-based application to be used on any platform.
- It has response-time advantages for the user (Doyle et al. – 1998). The web-based CBR can speed up, assist users in finding or configuring the distributed on-line information irrespective time and location.
- It is a great tool for business, especially in the realm of e-commerce. The web-based CBR provides expert advice over a network and acts as assistants for e-commerce and sales support service.
- It is much easier to gather new experience and update the case library. It cuts the cost of system maintenance since there is a single, centralized case library.

To further investigate the web-based CBR application, this thesis describes an implementation of a web-based CBR shell, I-NaCoDAE (Internet-Navy Conversational Decision Aids Environment). I-NaCoDAE is an extension of NaCoDAE, a non-web based conversational CBR system (Aha & Breslow – 1997). The goals of the thesis project were the following:

- Design and implementation of a web-based CBR shell.
- Design and development of a web-based Graphic User Interface for the CBR shell.
- Development of a generic client-server model for the CBR shell that attempts to balance the load between the client and the server according to the current network conditions.

I-NaCoDAE was implemented in Java, an object-oriented, multithreaded and platform-independent programming language. Businesses have recognized the potential of Java on the server that Java is inherently suited for large client/server applications. The cross-platform nature of Java is also extremely useful for organizations. Based on the above considerations, Java was used as the programming language to develop the client-side application (applet) and the server-side application, and to implement the CBR shell system design **[19]**.

4.6 Intelligent and Integrated Ticketing Manager (I2TM) application example

The Spanish company TISSAT S.A. runs a Technology Management Center (TMC) that offers customer support, communication and Internet services for public administration organisms and private companies. TISSAT works either with problems related to computer errors or with other domains, such as the international emergency phone 112 of Valencia (Spain), which covers the emergencies of over four and a half million of citizens. TISSAT attends to customer requests via a call center. This call center can receive queries via phone, e-mail, Internet or fax. There is a maximum time to provide a correct solution for each query. This time is agreed between TISSAT and its customers in the Service Level Agreements (SLA's). When the maximum time to solve a problem is exceeded, the company is economically penalized. In order to efficiently manage its call center, TISSAT has developed a help-desk toolkit called I2TM (Intelligent and Integrated Ticketing Management). I2TM manages customer requests, integrates the available channels to make a request and manages the inventory. The system also helps operators to solve new problems by searching for solutions successfully applied to similar problems in the past. This will ease their work and thus, they will be able to provide quicker and more accurate answers to customer problems. In order to cope with this functionality, we have developed a tool called CBR-TM (Case-Based Reasoning for Ticketing Management). This tool works as a separate module of the I2TM system, which allows making changes in the I2TM implementation without affecting the CBR-TM module and vice versa. **Figure 13** shows the overview of the entire system. I2TM and CBR-TM communicates and synchronises their data via web service calls.

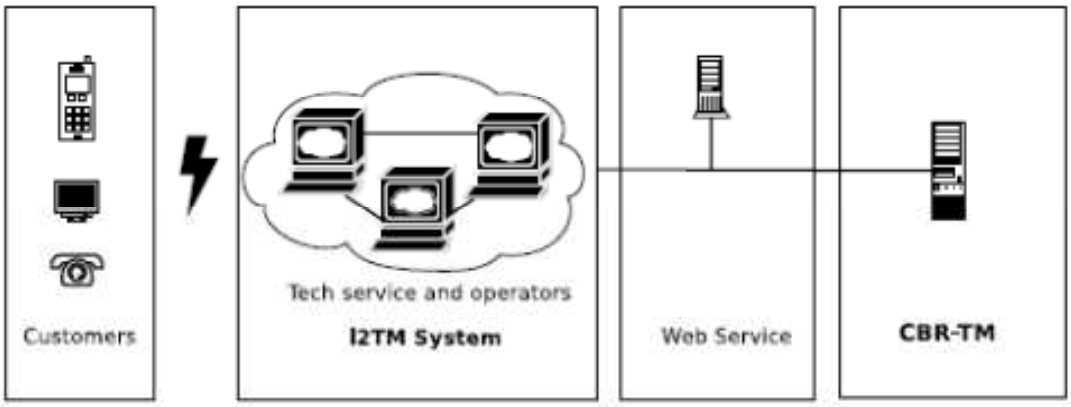


Figure 13 – I2TM system architecture

Before the implementation of the CBR-TM module and the new I2TM system itself, some weaknesses to improve in the call center operation were identified. On one hand, it was necessary to save the knowledge and experience of the operators in an appropriate format (previously it was simply written in hand-written notes or in reference manuals that were usually out of date). This would avoid losing valuable information whenever the operators leave the company and it may also be used to train new operators. Moreover, the information about problems that had been already solved by other operator was not available on-line and the operators lost time solving them again. On the other hand, the information to manage comes from a wide range of domains and data types. In order to facilitate the update of the CBR-TM module, each phase of the reasoning cycle (Retrieve, Reuse, Revise and Retain) is implemented as a plugin algorithm. Thus, CBR-TM is a flexible system and any change in the algorithms that implement the phases, or even the introduction of new algorithms, does not affect the entire CBR-TM system. The specific algorithm that has to be used in each phase is specified in a XML configuration file. The following sections describe with more detail the reasoning phases of the CBR-TM module.

4.6.1 Data acquisition

An important task in this project has been to obtain a test database to validate our CBR system during its development. In order to extract this information, we analysed the old call center database. The registers of the database (tickets) contain information about previously solved problems. Therefore, a ticket in our system is a new case to solve. The data structure in CBR-TM and the relations with the structure of the new databases of I2TM is shown in **Figure 14**.

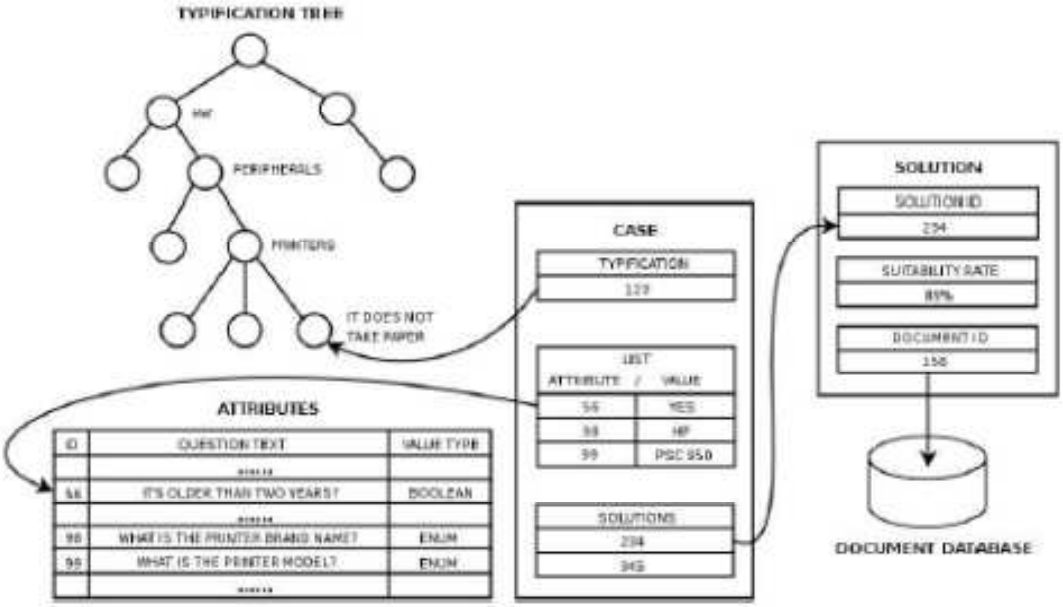


Figure 14 – Overview of the data structure in I2TM and CBR-TM

TISSAT maintains a non-disjoint tree (Typification Tree) that contains the taxonomy of the problem types (categories) in a hierarchical order (from less to more specific categories). These categories are set by TISSAT depending on the application domain of each project managed by the company. The first level nodes of the tree represent projects and the nodes below them are the categories of those projects. The CBR-TM module is able to reread the tree whenever a new project is added or any category is modified. In this sense, CBR-TM is a multi-domain system able to work with different types of problems. TISSAT also maintains a database of answers to questions that the operators ask to the customer when a query is made. These answers are saved as attributes in a database and they provide more specific information about the problem represented by the categories. In addition, TISSAT registers successfully applied solutions in a document database. In CBR-TM, a case is the prototyped representation of a set of tickets sharing the same categories and attributes. Each case has one or more associated solutions. One solution of the document database can also be associated with more than one case. CBR-TM stores the cases in a case-base.

4.6.2 Retrieve

The first step when CBR-TM is asked to solve a new ticket is to retrieve a set of cases from the case-base that are related to the same problem as the ticket. I2TM uses a web service call named GetSolutions to start this process in the CBR-TM module. The call needs as parameters the values of the ticket attributes and its categorisation. The retrieval process comprises three steps: Indexation, Mapping and Similarity calculation. At the end of the retrieval phase, a list of cases sorted by similarity with the ticket is obtained. This phase is implemented through three different types of plugin algorithms: the Indexer, the Mapper and the Similarity algorithms.

The Indexer algorithm hierarchically organises the cases of the case-base in order to facilitate their retrieval. Currently, the operators perform the indexation by categorising manually the ticket.

The Mapper algorithm explores the Typification Tree to retrieve the category nodes of the ticket and its predecessors (since upper categorisations represent more generic problems, but they are also related with the current problem and their solutions might also be suitable). Then, the algorithm searches in the case-base and retrieves all the cases with either the same categorisation as the ticket or a more generic one.

Once the set of similar cases has been selected, it is sorted by similarity with the ticket. The Similarity algorithm performs this arrangement. Here arises the problem of finding the similarity between cases that share some attributes and have different ones. Note that the cases associated with different categories of the Typification Tree can have different attributes. Moreover, there are many possible attribute types. The attributes can also have missing values, which makes more complicated the calculation of the similarity between cases. In order to test the CBR-TM module, we have adapted and implemented some similarity measures: two similarity measures based on the Euclidean distance (classic Euclidean and Normalized Euclidean) and a similarity measure based on the ratio model proposed by Tversky. In addition, we have implemented a set of distance metrics that allow us to work with different attribute types (numeric, nominal and enumerated). The Similarity algorithms use the distance metrics to compute local distances between the attributes of the cases, and the similarity measures to compute global distances between the cases (the similarity between the cases). Finally, the set of retrieved cases is sorted by means of a k-nearest neighbour algorithm.

4.6.3 Reuse

The reuse phase is implemented by means of the Solution Selection plugin algorithm. At the end of the reuse phase, we obtain a sorted list of solutions to apply to the ticket. First, the Solution Selection algorithm proposes the solutions of the most similar case to the ticket, sorted from higher to lower degree of suitability. Next, it proposes the solutions of the second most similar case, and so on. Note that the solutions themselves are not adapted, but proposed directly in a specific order to use them to solve the current ticket. When this process is finished, CBR-TM answers the GetSolutions web service call and returns it with the list of proposed solutions and their associated suitability for the ticket.

4.6.4 Revise

In the revision phase, the I2TM system uses the CloseQuestion webservice call to report to the CBR-TM module the customer degree of satisfaction with the proposed solution. The tickets that were not requested to CBR-TM, but solved directly by the operator, are also reported. This phase, implemented by means of the Rewarder plugin algorithm, helps CBR-TM to improve its performance. When CBR-TM is reported a solved ticket, it performs the retrieval phase in order to discover whether this ticket has already a prototype case in the casebase. If such case exists and the solution applied to the reported ticket is already associated with this case, the degree of suitability of this solution is increased. Otherwise, the new solution is associated with the case. If there is not a similar enough case in the case-base, a new case with its solution is created. The similarity threshold has been found experimentally and it can be changed to any desired value. Note that the retrieval phase would be avoided here if we were able to know which case was used to propose its solution to solve the ticket. Moreover, this solution could be penalized if it does not fit the ticket. However, we consider that in the current implementation of our system this is not appropriate. On one hand, the CBR-TM module may be reported a ticket that was not requested previously to the module. In this situation we have to perform the retrieval phase in order to check if there is a similar case in the case-base or, otherwise, to create a new one. On the other hand, it is possible that CBR-TM had proposed an invalid solution but it had not made any mistake, since this is not a completely automated system and, for instance, the operators can fail in their categorisations. Moreover, do not use a proposed solution does not necessary mean that this solution is erroneous, but the operator may have chosen other solution for any reason.

4.6.5 Retain

As it is explained above, each time that a ticket is solved, the I2TM system reports back to the CBR-TM module. The retention phase is also done by means of the Rewarder algorithm, which checks if it is necessary to create a new prototype case for the ticket. Therefore, the retention phase can be viewed as a consequence of the revision phase. If the ticket that has been reported to CBR-TM is not similar enough to any case of the case-base (it exceeds the similarity threshold), a new case will be added to the case-base.

4.6.6 Evaluation

Using the Ticket Database, has been run several tests to validate the CBR-TM module. The tests have been performed using a cross-partition technique, separating the ticket database into two databases for training (loading the case-base) and testing the system. This test has been made to check on the computer error domain the behaviour of the similarity measures implemented. Therefore, the tests have been repeated setting the system to work with a different similarity measure each time. First of all, the system performance has been checked. This performance may be influenced by the size of the database or by the number of customers performing simultaneous requests.

Figure 15 shows that as the number of tickets in the database used to create the case-base of CBR-TM increases the mean error in the answers to the requests decreases. Note that, is being performed a supervised learning, it is considered an error when CBR-TM does not propose the same solution as the one we have recorded in the Ticket Database for the ticket that has been requested. It demonstrates that, the more problems CBR-TM solves, the more it increases its knowledge to solve new ones.

Figure 15 shows the response time of the CBR-TM module when the number of customers performing simultaneous requests increases. Although in this test it is considered that the customers are making the requests almost at the same time, CBR-TM is able to answer all of them quickly. With regard to the behaviour of the different similarity measures, their performance in this domain is almost the same **[20]**.

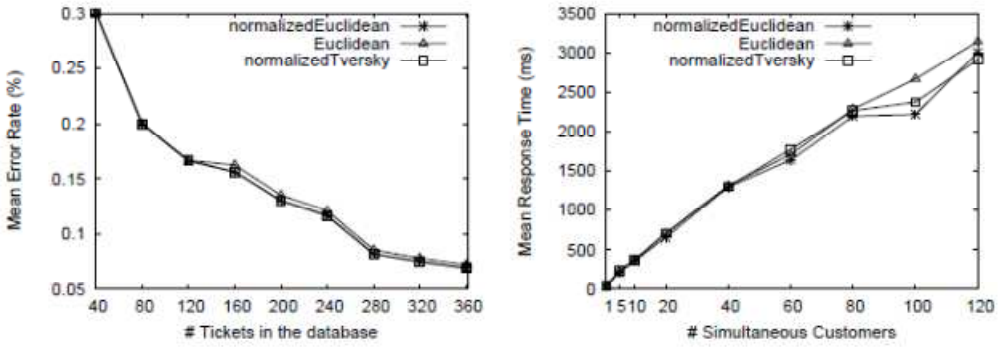


Fig 15 – a: Influence of the database size of the CBR-TM system performance,
b: Influence of the number of simultaneous customers on the CBR-TM system performance

4.7 CBR application domains

Although CBR is a relatively new AI methodology, numerous successful applications exist in the academic as well as in the commercial domain. Already in 1994, Watson and Marir reported over 100 commercially available CBR applications. The domains of these numerous CBR systems reported in the literature are the following:

- **Interpretation** as a process of evaluating situations/problems in some context (e.g., HYPO for interpretation of patent laws proposed in 1991, KICS for interpretation of building regulations proposed in 1994, LISSA for interpretation of non-destructive test measurements proposed in 1999).
- **Classification** as a process of explaining a number of encountered symptoms (e.g., CASEY for classification of auditory impairments proposed in 1989, CASCADE for classification of software failures proposed in 1992, PAKAR for causal classification of building defects proposed in 1994, ISFER for classification of facial expressions into user defined interpretation categories proposed in).
- **Design** as a process of satisfying a number of posed constraints (e.g., JULIA for meal planning proposed in 1992, Déjà Vu for control-software production proposed in 1996, CLAVIER for design of optimal layouts of composite airplane parts proposed in 1996, EADOCS for aircraft panels design proposed 1997).
- **Planning** as a process of arranging a sequence of actions in time (e.g., BOLERO for building diagnostic plans for medical patients proposed in 1993, TOTLEC for manufacturing planning proposed in 1993).
- **Advising** as a process of resolving diagnosed problems (e.g., DECIDER for advising students proposed in 1987, HOMER – a CAD/CAM help desk proposed in 1998) **[16]**.

CHAPTER 5: CBR in education

5.1 Case Based Reasoning for Intelligent Tutoring System (ITS)

5.1.1 Introduction

Online learning with Intelligent Tutoring System (ITS) is becoming very popular where the system models the student's learning behavior and presents to the student the learning material (content, questions-answers, assignments) accordingly. In today's distributed computing environment, the tutoring system can take advantage of networking to utilize the model for a student for students from other similar groups. Next it is presented a methodology where using Case Based Reasoning (CBR), ITS provides student modeling for online learning in a distributed environment with the help of agents. It is described the approach, the architecture, and the agent characteristics for such system. This concept can be deployed to develop ITS where the tutor can author and the students can learn locally whereas the ITS can model the students' learning globally in a distributed environment. The advantage of such an approach is that both the learning material (domain knowledge) and student model can be globally distributed thus enhancing the efficiency of ITS with reducing the bandwidth requirement and complexity of the system.

The major challenge in teaching is to improve both instructional productivity and learning quality for large and diverse population of students under real world constraints such as limited financial resources and insufficient number of qualified instructors. Different researches in the field suggest that students who are engaged in learning through intelligent tutoring processes are more likely to achieve success. Since mid 90's, few educational system models were web based and used ITS with student modeling in distributed style. In past decade, researchers from different disciplines have come out with systems which define and classify different teaching and learning styles in distributed environment.

Following it is conceptualized a Case Based Distributed Student Modeling (agent based) ITS architecture to support student-centered, self-paced, and highly interactive learning. In this system the first step in building an effective learning environment is building a case base where the system maintains a rich set of cases (scenario) of student's learning pattern, and employs an efficient and flexible case retrieval system. This maximizes the interactivity between the ITS and the students and customizes the learning process to the needs of an individual student. The system must use the student's learning profile such as learning style and background knowledge in selecting, organizing, and presenting the learning material to support case based learning. It also supports personalized and more intensive interaction between the student and the ITS.

Distributed CBR based student modeling enables adaptive delivery of educational contents and facilitates automatic evaluation of learning outcomes. The system also incorporates a new approach to course content organization and delivery, which can be developed, based on distributed and agent based instructional components. Instructional components represent the customized interactive presentation of any topic of a subject or different subjects.

5.1.2 Proposed system for distributed Case Based Reasoning

The proposed system is based on finding a case that is similar to the learning domain of a past student in a distributed environment. The system consists of a number of specialized agents with different expertise. Each student has a unique *Personal Agent* (student profiler) that manages the student's personal profile, including knowledge background, learning style, interests, courses enrolled in, etc. The other two agents in the system are *Teaching Agent* and *Course Agent* and they communicate with each other through different communication channels situated in a distributed environment.

The model of communication between agents is shown in **Figure 16**. A web based course is supported combinedly by a *Teaching Agent* and C*ourse Agent* that manage course material and course-specific teaching techniques and student modeling strategy. Multiple course agents exist on distributed sites to provide greater efficiency, flexibility, and availability.

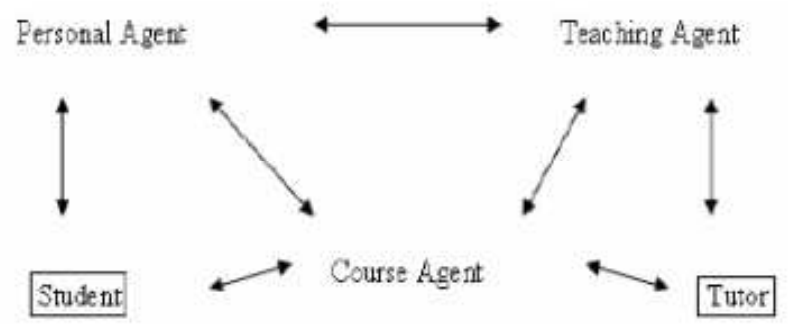


Figure 16 - Communication model among Agents

The *Teaching Agents* can talk to any *Course Agent*, and often choose one nearby for better performance. The *Course Agents* also act as mediators for communication among students and tutors. A *Teaching Agent* interacts with a student and serves as an intelligent tutor of a topic or course. From a *Course Agent*, each *Teaching Agent* obtains course material and course specific teaching techniques and then tries to teach the material in the most appropriate form and pace based on the background and learning style of the student.

Lecture notes, presentations, multiple examples, with different difficulty level are used to make difficult concepts and operations easy to understand.

The following characteristics specify the attractiveness of this CBR based distributed student modeling:

1) In the learning environment students, tutors and corresponding student learning material are distributed across the entire network. Similarly, the potential users as student or as tutor are also widely distributed.
2) The student's behavior, corresponding background knowledge, and skills are dynamic, and accordingly the learning material and teaching methodology of the ITS are also required to be dynamic in nature i.e. they depend on case to case basis.
3) Students have different backgrounds, learning attitude and personalities. Students generally attempt to register in various courses at the same time. In this case coordinating learning on different topics for each student enriches the learning experience with in different environment. This is possible using CBR based distributed student modeling through personal agent **[23].**

5.1.3 Student modeling in distributed environment

The process of student modeling is shown in **Figure 17.** The following activities take place during the student modeling when the student interacts with the system.

1) Selection of topic by the student and getting student‘s background by presenting problems to the student.
2) Analyzing the student's response by the system.
3) Selection of case by the system based on the response.
4) Adaptation of the case by the system (if new then leave the case).
5) Achieving the knowledge component of the student model through case retrieval.
6) Generation of teaching strategy by the system.
7) Presenting the next problem/content to the student.

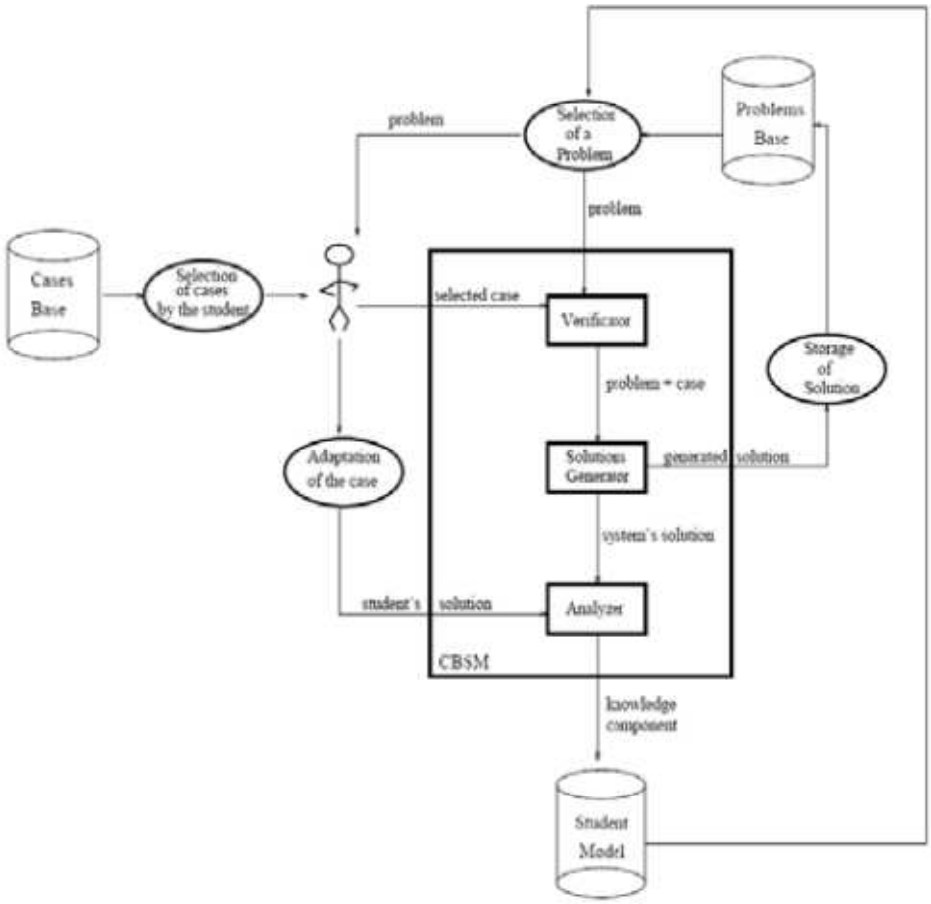


Figure 17 - Process of student modeling

CBR based student modeling system can be divided into three parts. A case based ITS (Intelligent Tutoring System), a student interface module and the instruction processing engine or agent communication system. The ITS is divided into a student module, a tutor module with expert module (human-tutor interaction) and an interface. The student module consists of a case base, which is based on an overlay or bug part library system.
The interfaces of student module, tutor module and case based student modeling reside on every node of the network. *Teaching Agent* indexes all the cases globally. *Personal Agent* or student profiler maintains static background profile (such as GPA or percentage, majors, interests and course taken) and the student's dynamic profile based on their interaction. The agent is responsible for indexing all the registered or aspirants students (learners) with their personal profiles globally. The *Course Agent* manages the course material as per student's requirement and *Teaching Agent* decides the topic or lessons, to be supplied to the students according to student's performance. The student modeling keeps the record in case base for future reference and manages the student performances and course material accordingly. The nodal architecture of the proposed system is shown in the **Figure 18**.

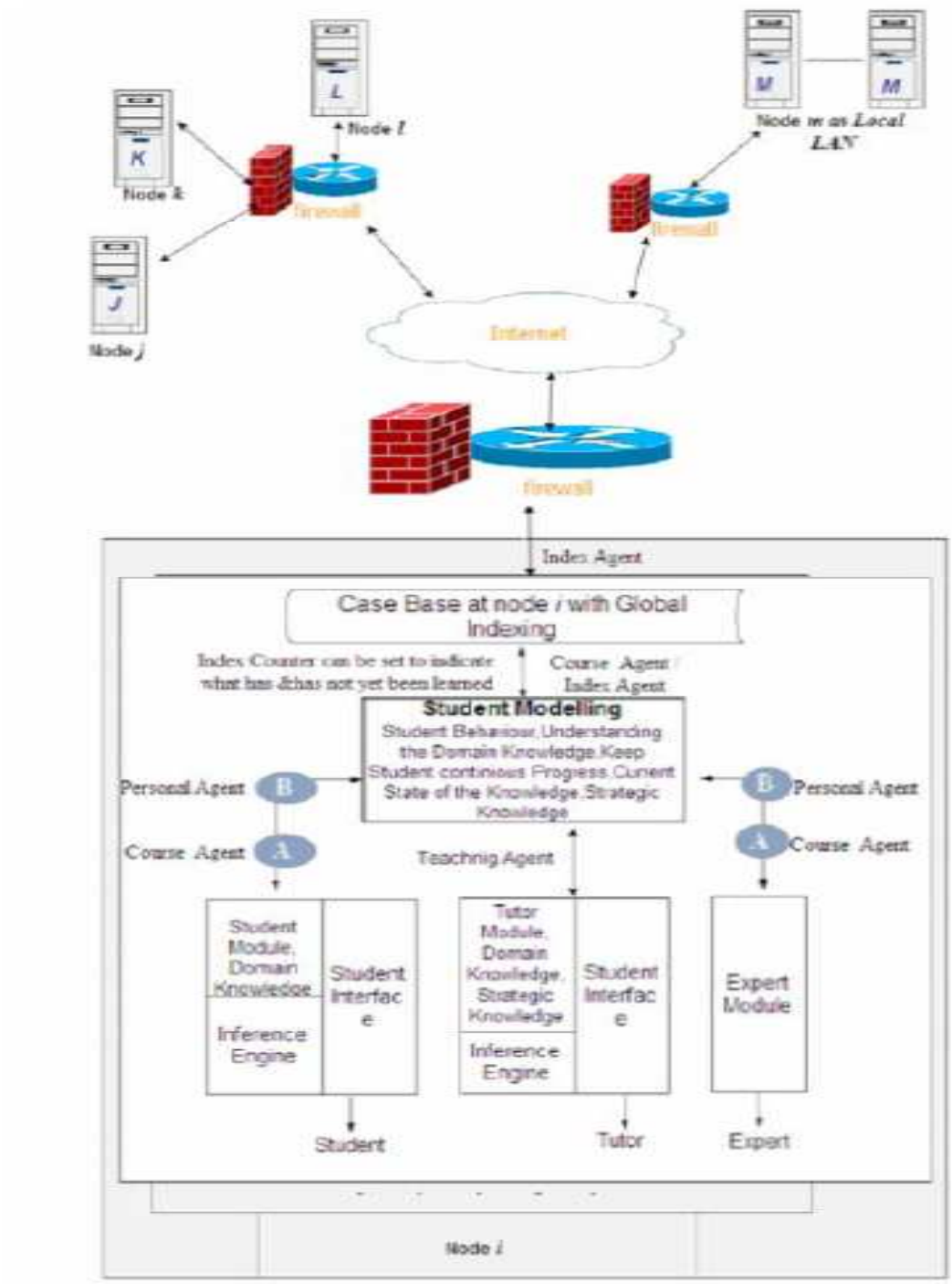


Figure 18 - Design architecture for CBR based ITS in distributed & agent environment

The design of such a case based student modeling system requires algorithms for the following operations:

1) Case creation in active case base.
2) Case indexing in active case base.
3) Update case in case base.
4) Adaptation of cases in the case base.
5) Case storage.
6) Pattern matching of case in case base or retrieval of cases from case base.
7) Checking duplicate case from case base.
8) Case deletion from case base and storing the deleted casein to archive for future reference.

Let us think of a scene of leaner at node *i* interacting with the ITS. Node *i* can be a standalone user or can be a school with LAN. If the student is the first time user then he/she is registered with the system. The *Personal Profile Agent* maintains the user profile. The session starts and student modeling activities initialize. The past performance is evaluated first and thereafter *Teaching Agent* suggests the topic. The component of T*eaching Agent* is shown in **Figure 19**.

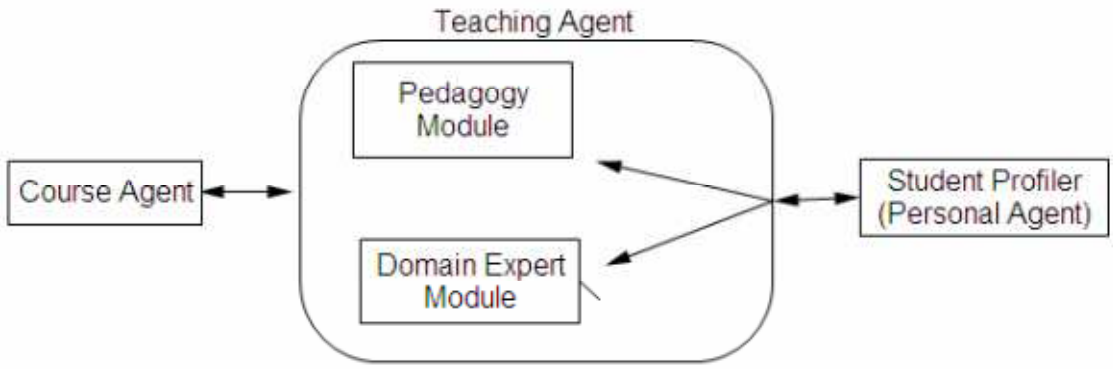


Figure 19 - Teaching Agent component

The *Course Agent* retrieves the appropriate topic first at the local node *i* and presents the material to the student. If the appropriate topic content is not found at node *i* the *Teaching Agent* then *searches* the appropriate content from the entire network with the help of the course index's. The students' learning is accessed by presenting questions to the student. The response is analyzed and the *Personal Agent* matches the leading style with the cases present in the case base which is stored in distributed manner with the help of global case index. If the case is new i.e. there is no similar case stored within entire network then this case is stored as new at node *I* and its one more copy is stored at some another node for fault tolerance. This is the responsibility of the *Course Agent* to manage this storage. Thus the whole system is managed in the distributed environment with just three agents:

1) Student profiler or Personal Agent.
2) Teaching Agent.
3) Course Agent.

The major characteristics of the system are:

- The system is fully distributed (not bounded with any network topology) i.e. domain knowledge and strategic knowledge both are distributed.
- Reduces the need of large storage spaces at the user's site, to store all the cases.
- Redundancy (duplicacy) is maintained for fault tolerance. Load balancing of the cases is achieved at each node by the storage management. Later one can research on optimum redundancy parameter.
- Case indexing is fully redundant on all nodes.
- If the node is a LAN, the domain knowledge can be localized and case base can be global **[23].**

5.2 Case-based tutoring in virtual education environments

Virtual education environments are gaining popularity as tools to enhance student learning. These environments are often used to allow students to experience situations that would be difficult, costly, or impossible in the physical world. North Dakota State University (NDSU) provides students with environments to enhance their understanding of geology (Planet Oit), cellular biology (Virtual Cell), retailing (DollarBay), and history (Blackwood). In order to maximize the learning potential of each individual student, an ideal environment needs to provide customized lessons to that student based on his or her individual performance. One method to address this requirement is the use of case-based reasoning software. This software is used to monitor student performance, track progress throughout an environment, compare the student to other students in the same environment, and create customized tutor dialogs to communicate this information to the student in the form of individual tutor lessons. An example of case-based lesson building software that meets the above requirements can be observed in the current DollarBay retailing environment.

The traditional teaching environment is usually thought to be that of a classroom: a single teacher giving lectures to a group of students who are expected to use their notes and textbook to prepare for periodic examinations and demonstrate that they have learned. Technology provides an alternative to this scenario. One of the ways technology can be used to supplement learning is through the construction of virtual education environments to simulate scenarios that may be difficult for students to experience in the physical world. Using the Internet, students can access these worlds remotely, be it in a classroom or in the solitude of their own dwelling. At North Dakota State University (NDSU), the World Wide Web Instructional Committee (WWWIC) is engaged in research aimed at developing virtual education environments to assist in the education and growth of students. Some of the key factors that lead to success of these environments at NDSU are the use of graduate and undergraduate students in the development process, the use of the environments in actual classes, and the application of knowledge from one environment to the others. One of the major goals of WWWIC research is to find ways to provide tutoring agents to communicate “expert stories” to students as they progress through the environment. These agents should monitor the student and send advice on an “as needed” basis while being careful to never insist upon or block any course of action (Slator et al. - 1999).

Although several of the virtual education environments at NDSU provide some basic tutoring functionality to students, none had implemented a completely functional case-based system except DollarBay. This system provides complete analysis of student behavior based on selected attributes and a message delivery mechanism. Following it is described the design and functionality of the case-based tutoring system implemented in DollarBay. This system provides the means to generate personalized lessons for each student participating in the DollarBay environment. In addition, it provides a framework that may be used to implement similar functionality in the other virtual education environments at NDSU with a minimum of coding. The DollarBay simulation is based upon a client/server paradigm.

The server side of DollarBay consists of a server program and a database. It is permanently connected to the Internet and allows other users to connect at any time, from any location, to the DollarBay environment. The database for DollarBay is implemented in LambdaMOO (Curtis – 1997). The database contains representations of all of the objects in the DollarBay environment, including the MOO programs necessary to give the objects their specific behavior. On the client side, the student interacts in the DollarBay environment by using a graphical user interface (GUI). The GUI window is generated by a Java applet and serves as the connection to the LambdaMOO server. This allows the student to interact with the environment by pointing, clicking, and selecting objects as well as typing text **[24].**

5.2.1 Playing the DollarBay game

As players engage themselves in the DollarBay game, they are assigned a location and must decide what to sell, what level of service to offer, how much to spend on advertising, how much to stock, who to buy from, and what price to set to appear attractive to the customer agents (Slator & Farooque - 1998). In order to simulate an economic environment, time is divided into "virtual weeks". Each week the customer agents are given a shopping list representing a week's worth of demand for various products. After each week has concluded and the shopping lists are exhausted, each agent assigns new attractiveness ratings to each store based upon the past week's experience and new shopping lists are created for the upcoming week (Slator & Farooque – 1998). At the end of each virtual week the weekly calculation charges players for their weekly expenses, recalculates the customer agent motivations as described above, and updates each of the player cases.

At the end of a player's life, they are retired to the Hall of Fame. The Hall of Fame is a place where players are moved when they graduate from the game by reaching a profit goal or are inactive for a long period of time. Players are moved to the Hall of Fame by the reaper, who is sent out periodically to retire graduated and inactive players. The reaper is responsible for recycling all of the objects related to the player, such as store, company, ads, and products. Recycling makes object numbers available for reuse at a future date as new objects are created. The reaper also moves the player's active case to historical cases for future reference by case-based tutors.

5.2.2 Overview of LambdaMOO

MOO is an abbreviation for a Multi User Dungeon Object Oriented or Multiuse Object Oriented system. It is designed to be a network accessible, programmable, interactive, multi-user system that is well suited to the construction of text-based collaborative software most commonly used as multi-participant low bandwidth virtual reality (Curtis - 1997). It contains a small, simple language that is designed to be easy to learn and supports expressions, looping, control structures, and built-in functions. LambdaMOO models virtual reality by representing virtual world entities (tutors, cases, stores, players, etc) as objects. It supports other value types as well (integers, strings, etc) but the objects are the backbone of any LambdaMOO database (Zelenak – 1999). A property is used to store an arbitrary MOO value. Verbs are LambdaMOO programs associated with a particular object.

In addition to the functionality described above, LambdaMOO also includes a built in command parser. Any time the command parser is invoked, the first word is always taken to be a verb. After identifying the form of the command, the parser may then proceed to execute it using one of the built-in commands, or by looking up the MOO objects representing the direct and indirect objects and then executing the command (Curtis - 1997). LambdaMOO is also an object-oriented language.

5.2.3 Virtual educational environments at NDSU

There are several environments currently implemented at NDSU. The Virtual Cell (White et al. – 1999) is a 3-D rendering of an interactive biological cell. Students are given assignments by a virtual lab instructor, and sent out to explore cells in tiny submarines. Planet Oit (Schwert et al. – 1999), is a part of the Geology Exploration Project and is used by geology students at NDSU. Students, playing the role of geologists on a field exploration of the mythical Planet Oit, are asked to acquire a set of field instruments. Blackwood (Slator et al. – 2001), is a simulation of a 19th century western town populated with intelligent agents who simulate the economic environment representative of the time.

Players accept a role in the environment and are forced to compete with other players and agents holding the same role (Slator – 1999). The DollarBay world allows students to simulate owning and operating their own retail store in the city of DollarBay. Students who quickly figure out how to best serve the needs of the shoppers rapidly rise to be profitable. The overall goal of intelligent tutoring is to focus on developing and employing intelligent agents within multi-user distributed simulations to help provide effective learning experiences (Slator – 1999).

Examples of diagnostic tutoring may be seen in Planet Oit. For example, the science tutor looks at the decision making process that a player follows while trying to properly identify a material, and what experiments were performed on the material, the tutor is able to identify students who have made “lucky guesses” and let them know that they did not follow the proper process in getting to their answer. Rule-based tutoring at NDSU was at one time functional in DollarBay. A rule-based tutor functions by knowing a set of rules about a domain, monitoring student action for any indication of breaking one of the rules, and then visiting the student to present a warning (Slator & Brian M. – 1999). For example, one of the rules that the now defunct rule-based tutor in DollarBay monitored was if a student had set their prices to an excessive markup. In such an instance, the tutor would send a message to the student informing them that they may be setting their prices too high (Slator, Brian M. - 1999).

An example of case-based tutoring outside the NDSU realm is the Georgia Tech Case-Based Intelligent Tutoring System (GTCBITS). This system is used to demonstrate the importance of critical information to airplane pilots by using stories of difficult situations or incidents encountered by other pilots. The topics presented can range from problems arising from the complex nature of aircraft, the dynamic nature of the aviation environment, or new/changing features of an aircraft (Palmer - 2002). Case-based tutoring at NDSU has presently only been implemented in the DollarBay virtual world **[24].**

5.3 The Application of Case Based Reasoning on Q&A system

5.3.1 Introduction

Question and Answer (Q&A) system is one of the most important components in e-Learning environment which aims at answering the questions asked by the student during their study processes. Accuracy and efficiency are the main two criteria used to evaluate the Q&A systems. Many Q&A systems have already developed based on email-solution, keyword-matching or word-segmentation techniques. With the growth of the number of users and questions, the process time of these systems will become longer and the matching accuracy will become lower due to different presentations of the question and variable interests of the user.

In order to overcome the above disadvantages, we introduce CBR (Case Based Reasoning) into traditional Q&A system. CBR, representing a new generation of expert system technology, has enjoy tremendous success as a technology for solving problems related to knowledge reuse. Below it is filled an interactive Q&A engine based on CBR. This engine uses keywords of the question to trigger case and sorts the results by the relationship and can modify the weights of the keywords dynamically depending on the feedbacks from the user. It is also presented a new feature-weight maintenance algorithm to increase the accuracy. At last it is extended the 2-layer architecture of CBR to a 3-layer structure to make the system more scalable and maintainable **[25].**

5.3.2 Architecture of the auto Q&A system

This Q&A system is based on the CBR technology. The whole system is divided into two separate modules, with the first one called Case Authoring Module and the second one introspective Q&A Engine.
The Case Authoring Module is to represent the unstructured field knowledge structurally based on empirical expert knowledge and application background. All these structural representations can be transferred into question-answer instances and stored in the system case base. The introspective Q&A Engine is the kernel of our system. It is triggered by the keywords or description of the problems and returns the similar problems related to the description ranked by the score. So the user can select the most similar problems and get the answer. Furthermore, the system provides a feedback module to adjust the weights of keywords according to the user's score. The architecture of the Q&A system is shown as **Figure 20**:

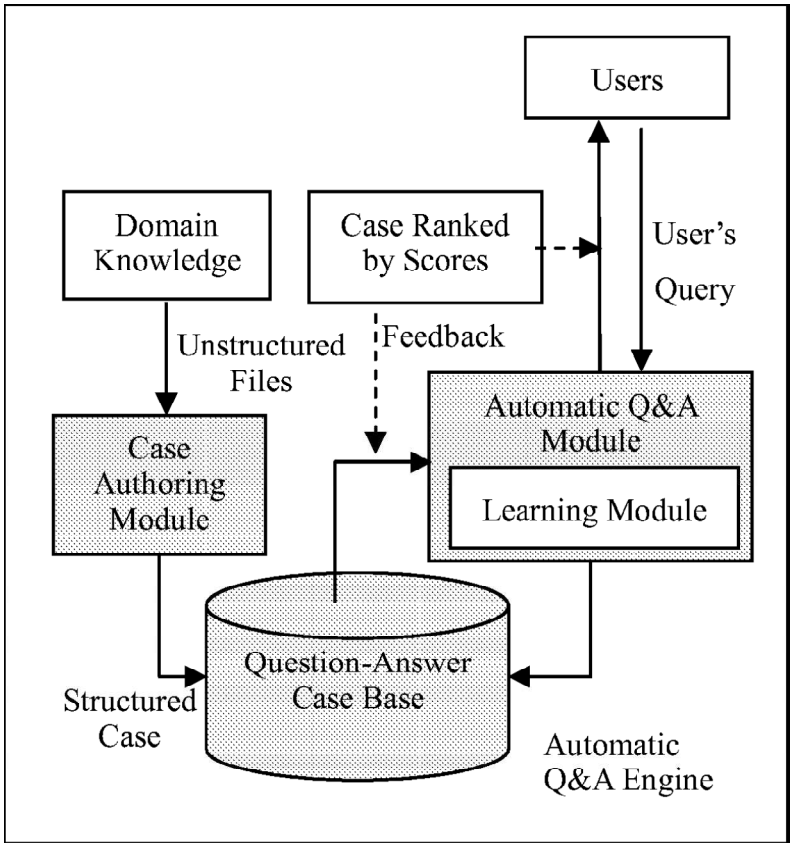


Figure 20 - Architecture of the Q&A system

The system has been used in the professional e-learning site of Shanghai Jiao Tong University (www.nec.sjtu.edu.cn). So all the questions, answers and the relativity between them are accessed through the standard web interfaces. The users, especially the students, produced a great number of questions and potential answers during the learning process. All the questions and answers are assembled in log files. So we can train the index architecture of the relationship between questions and answers based on the log files. This process is running during the life cycle of the system, which makes the Q&A system become a closed-loop system.

5.3.3 Case Authoring Module and case base construction

5.3.3.1 Definition of the Question-Answer case

The description and definition of case is the foundation of a CBR system, and there isn't a uniform standard for it so far since it has strong domain characteristic. In the Case Authoring Module, we define our case description based on the e-Learning domain characteristics and organize the unstructured domain knowledge in a structural way. The cases in the Q&A system are the description of question and answer. The representation of a case is as follows:

- **Keywords:** Short description of the case, which can be used in fuzzy string matching with the user's initial free-form text input.
- **Attributes:** The features that present the main content and characteristics of the question.
- **Question Description:** This is a more detailed textual description of the question's object or content used to confirm the general problem area.
- **Answer:** The answer provides a solution to the case in either textual format or any multimedia format.

Figure 21 and **Figure 22** give an example of the case representation for the Q&A system. The cases in **Figure 21** are fairly refined, down to the detailed features and their values, while the cases in **Figure 22** have only two major parts: problem description and answer.

Type	Keyword1	Keyword2	Answer
Concept & Difference	Switcher	Router	Describe the concepts and difference of Switcher and Router.

Figure 21 - Case representation with detailed features

No.	Problem Description	Answer
1	What's the difference between the Switcher and Router?	The function of switcher is quite different from the router …… (Just present the difference between them)
2	What are the concepts of Switcher and Router?	Switcher is the …… Router is the …… (Just list the concepts of them)

Figure 22 - Case Representation with natural language description

5.3.3.2 The 3-Layer architecture of the Q&A case base

After the description of the case has been defined, the next essential task is to construct the case base and feature index. Each case is associated with a set of feature-value pairs. These pairs are combinations of important descriptors of a case, which distinguish it from other cases. So a case base could naturally be viewed as 2-Layer architecture comprised of the feature-value layer and case layer. Using the weights assigned to the connections between the feature-value pairs and the case, a CBR system determines the most relevant cases ranked by the feature information submitted by the user and then returned the results to the user for considerations. However, in practical implementations we find that the definition of the 2-Layer structure sometimes does not work well since when the number of cases becomes too large the efficiency and scalability of the system will decrease dramatically. So we expand the 2-Layer architecture into a 3-Layer by dividing the case layer into question layer and answer layer. Additionally, we introduce a second set of weights, which attaches to the connections between questions and their possible answers. This second set of weights represents how important an answer is to a particular question.

In order to describe the situation of user's query, the traditional 2-level case base architecture is extended into a 3-level network structure and take the 2-layer structure as a special case. Consider each case as presented as a triple <F, P, S>, where F corresponds to the feature values, P are the problem description and S are the answers.

This representation can be split into three levels: a feature level corresponding to feature values F, a problem description level corresponding to P and an answer level corresponding to S. With this model, when a user enters the feature-values at the first level, the system ranks problem descriptions for users' consideration at the middle layer. The user then selects one intended problem description and the system provide the possible answers which is also ranked according to the second set of weights. New system architecture is shown in **Figure 23**.

An important motivation for this separation in a case is to reduce the redundancy. Given N cases and M solutions, a case base of size N x M is now reduced to the one of size N + M, which eases the scale-up problem and helps make the case base maintenance task easier. A solution can now be shared by several cases and will only need to be revised once if needed.

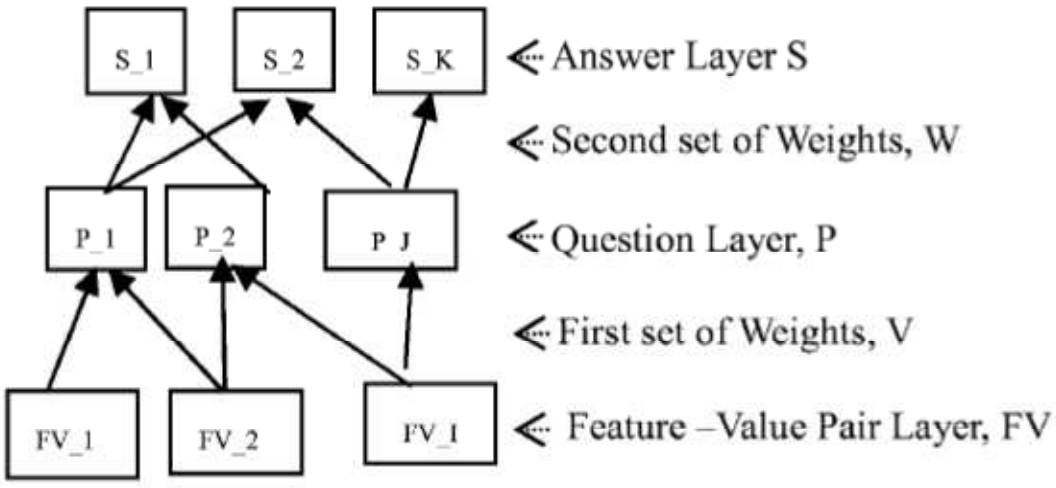


Figure 23 - 3-Layer architecture of a Question-Answer case base

5.3.4 Case study and experiment

The introspective Q&A system has been implemented based on a professional distance learning web sites. **Figure 24 to Figure 26** show an example of the process to solve a problem (since it is a system for Chinese users, so even the English version still has some Chinese information). As can be seen in **Figure 24**, user can first enter a problem description to identify the context under which the problem is solved. User can describe the problem with any natural description language as he wishes.

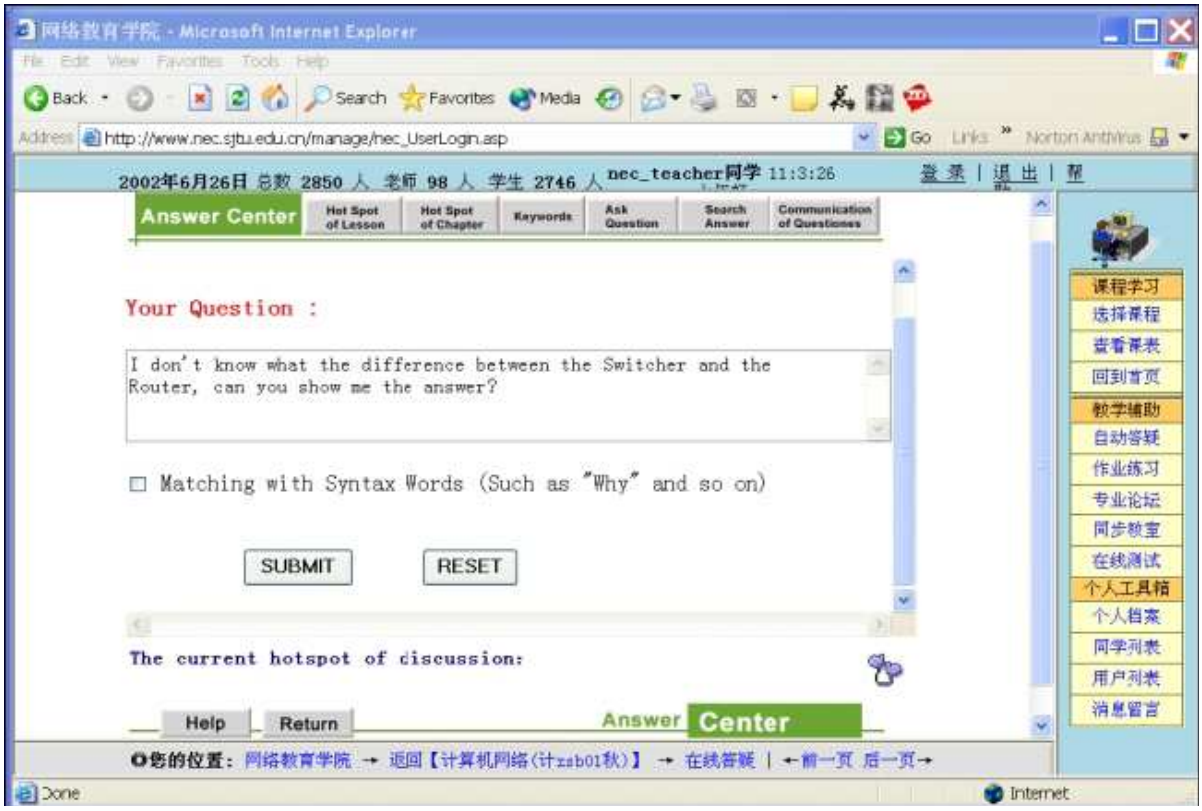


Figure 24 - Question submission in Q&A system

After submitting the question, a collection of initial cases which match the partial description are retrieved and returned to the user. These cases serve as the candidate answers for subsequent problem solving. Questions that are associated with the candidate cases are then presented to the user (as shown in **Figure 25**).

The system will return the questions ranked by the average score. Then user can select any similar question to see the answer. If they think the ranking and the score of the question is not fit for him, he can adjust the score in the answer showing column. And the system will then adjust the corresponding weights based on the score given by the user and use the new weights to calculate score next time (As shown in **Figure 26**).

In order to evaluate the efficiency and validity of our algorithm, we give the test results based on the Network Course database from the NEC Question/Answer Repository. The Network Course database contains 300 instances and 28 attributes. The main case base is divided into several incremental sub-case base, containing 50, 100, 150, 200, 250 instances respectively. Firstly these databases are converted into the case bases that the algorithm can handle by converting all rows into cases and all columns into features.

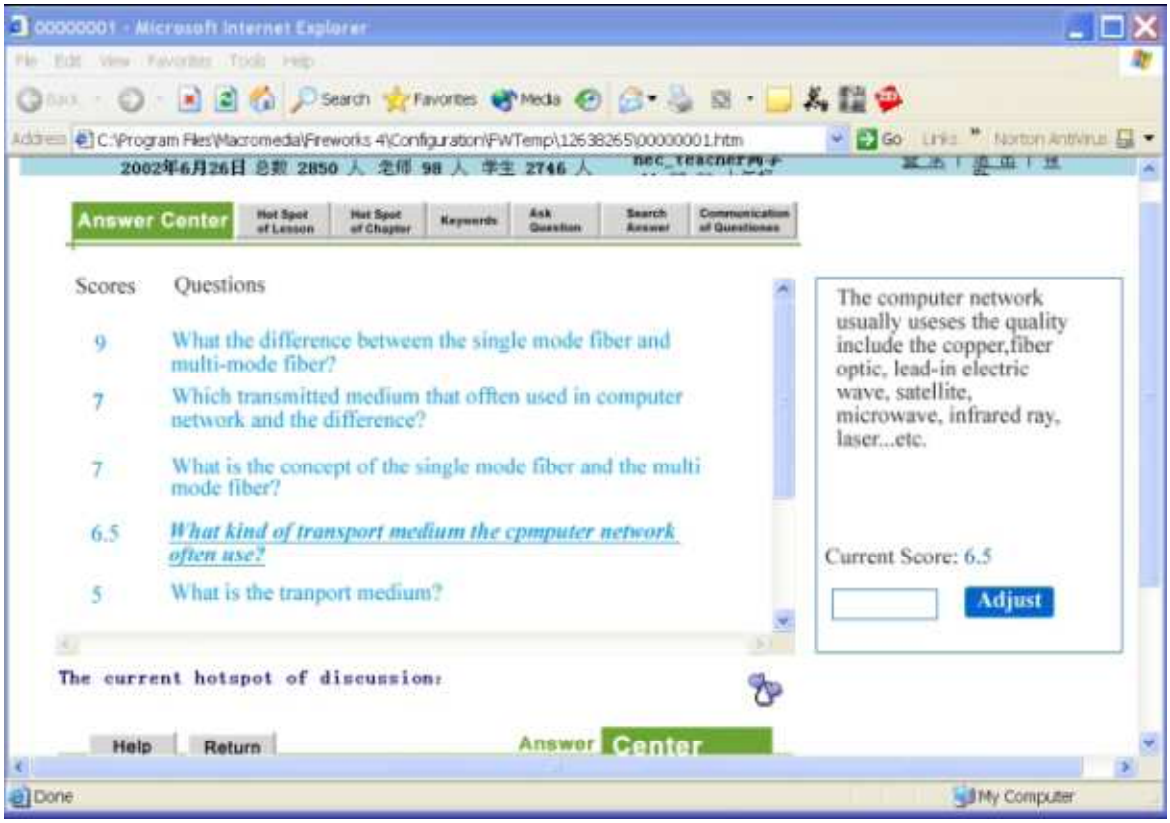


Figure 25 - The returned similar question lists ranked by average scores

In these tests, the score of a case or an answer is between 0.0 and 1.0, meanwhile, suppose the initial values of the weight are 0.5. Based on the 3-Layer architecture, ten training based on the sub-case bases are being performed. Finally, we can get the mean-errors convergence plot (As shown in **Figure 27**).

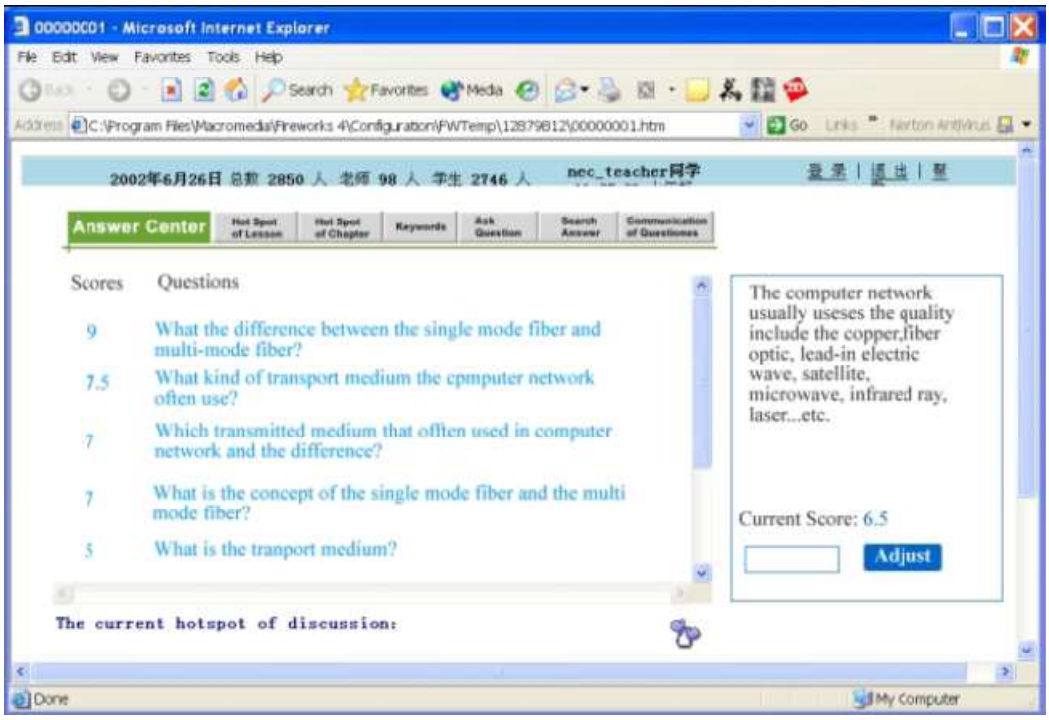


Figure 26 - The adjusted question list based on the learning network

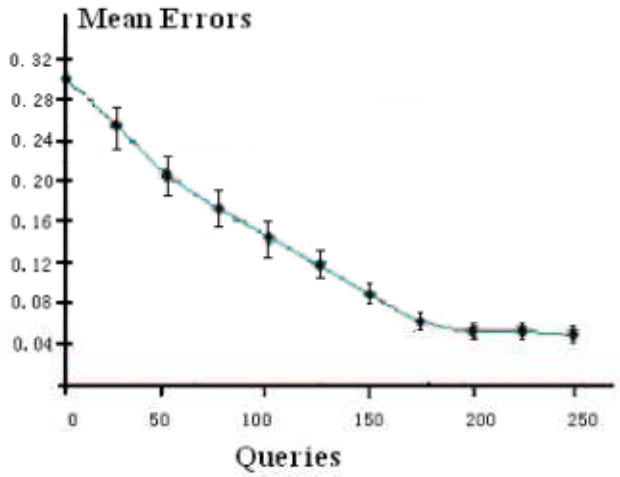


Figure 27 - Plot of the mean errors convergence and 95% confidence interval along the CBR process

In this figure, the 95% confidence interval is also shown on each datum point, where the size of the interval indicates the fluctuation around the mean values, and the average processing time is approximately 1.8 seconds **[25].**

5.4 Design of individualized instruction with learning object using case-based reasoning in WBI

5.4.1 Introduction

Internet-based technology is becoming an important tool of instructional delivery, and distance education is providing education with an increasingly important phenomenon. However, most Internet-based instruction has not considered individual differences. Many scholars have studied learning styles, and those inventories had been used to help teachers select appropriate strategies in the classroom. Beck (2001) analyzed three widely used learning style inventories, which are 4MAT System, Dunn's LSI, and Renzulli and Smith's LSI, and matched them to the most compatible teaching strategies. In Web-Based Instruction (WBI), determination of individual differences or preference of learners can be helpful to meet learning objectives.

Case-Based Reasoning (CBR) is one of the most successfully applied Artificial Intelligence (AI) technologies of recent years. The concept of CBR is derived from cognitive science. Schmidt (2003) described that "CBR is a generation of multiple learning techniques that uses remindings of similar situations in order to solve new problems" CBR is being applied by helpdesks and diagnostic systems in business or medical areas. Researchers are developing an Intelligent Tutoring System (ITS) for the educational field, and some are attempting to diagnose a student's learning difficulties using CBR. From the educational point of view, cases in real life related to curriculum can help learners easily establish their own knowledge. On the other hand, a Web-based system can accumulate cases of whether learners accomplished objectives, what their individual differences were, and what kinds of objects were used. These cases will be a good tool of reasoning.

Below it is suggested a prototype of Matching Learning Object to Individual Differences (MLOID) which contains learning objects based on Sharable Content Object Reference Model (SCORM). The MLOID system is designed to determine individual differences of a learner and offer appropriate learning objects to a student in accordance with his learning preference using case-based reasoning. In addition, the learners can receive feedback depending on the evaluation. The system will record the case of the learner and learning style and individual differences. This case will be the reference for case-based reasoning. Individual differences in Web-based instruction, CBR theory and learning objects are being described. Also, how CBR and learning objects affect adaptive instruction is stated. Moreover, the architecture of the MLOID is illustrated.

5.4.2 Theoretical framework

This system is built on a combination of three theories: individual differences, learning objects, and case-based reasoning (CBR).

5.4.2.1 Individual differences

What are individual differences in Web-based adaptive instruction?

In Web-based adaptive instruction, individual differences are primary points to decide how to adapt instruction. Student learning differs because student learning traits and thinking process differ depending on what the student is trying to learn (Jonassen & Grabowski - 1993). Individual differences are important to the design of Web-based instruction because all learners do not use Web-based instruction effectively (Chen & Paul - 2003). Many studies have been conducted to identify the relationship between learning success and individual differences, such as learning style, cognitive style, and prior knowledge (Hsu & Dwyer – 2004, Foster & Lin - 2003). In addition, these individual differences have significant effects on student learning in Web-based learning environment (Chen & Paul - 2003).

Individual differences have been studied by many scholars and there are many classifications. For example, in Aptitutde-Treatment Interaction (ATI) research, two aptitudes are described by Snow and Swanson in 1992 (Park & Lee - 2003). One is cognitive aptitude and the other is conative and affective aptitude. Cognitive aptitude comprised of intellectual ability, cognitive and learning style, and prior knowledge. Conative and affective aptitude are divided into motivational construct and volitional or action-control. Four kinds of individual differences are described, because recently these individual differences are often used to implement Web-based adaptive instruction.

1. Cognitive style

Cognitive style is the most popular method to distinguish individual differences. There are many different definitions of cognitive styles. Jonassen and Grabowski (1993) used cognitive control and cognitive style. Cognitive control means individual's perception of environmental stimuli, and cognitive style is learner traits or individual's perceptual habits. But many scholars used these two conceptions in terms of cognitive style. Liu and Ginther (2001) described that cognitive styles refer to the individual's consistent and characteristic predispositions of perceiving, remembering, organizing, processing, thinking, and problem solving. Cognitive style is individual preferred and habitual approach to organizing and representing information.

Cognitive styles and learning styles are often used interchangeably. Generally, cognitive styles are more related to theoretical or academic research, while learning styles are more related to practical applications (Liu & Ginther - 2001). A major difference between these two terms is the number of style elements involved. Specifically, cognitive styles are more related to a bipolar dimension while learning styles have more than two characteristics.

- **Field independence vs. field dependence:** One of various dimensions of cognitive styles is field dependence which describes the degree to which a learner's perception or comprehension of information is affected by the surrounding perceptual or contextual field (Jonassen & Grabowski - 1993). Field-dependence-independence is value-neutral and is characterized as the ability to distinguish key elements form a distracting or confusing background (Witkin, Moor, Goodenough & Cox - 1977). Field-independent persons are likely to be self-motivated and more autonomous in relation to the development of cognitive restructuring skills. Conversely, field-dependent persons are likely to be concerned with what others think and more autonomous in relation to the development of high interpersonal skills (Park & Lee - 2003, Liu & Ginther - 2001). According to Witkin et al. (1977), field independent persons tend to be intrinsically motivated and enjoy individualized learning, while field dependent ones tend to be extrinsically motivated and enjoy cooperative learning. Jonassen and Grabowski (1993) suggested different learning strategies of each style. Field dependent learners acquire knowledge using concentration on information and repetition or rehearsal of information to be recalled. In contrast, filed independent learners acquire knowledge selecting information sources, searching for and validating information, transferring knowledge, generating metaphors and analogies and analyzing structurally.
- **Holist vs. serialist:** Another cognitive style was identified by Pask (1976 as cited in Jonassen and Grabowski - 1993) as holist and serialist. He mentioned that these styles are a measure of bipolar information processing strategy that describes the way that learners select and represent information. According to Park and Lee (2003), holist prefers a global task approach, a wide range of attention, reliance on analogies and illustrations, and construction of an overall concept before filling in details. In contrast, serialist prefers a linear task approach focusing on operational details and sequential procedures. Jonassen and Grabowski (1993) suggested different learning strategies of each style. Holist uses effectively learning strategies: paraphrasing, comparing new knowledge with existing knowledge, concept mapping, inferring causes, and predicting outcomes. Serialist uses learning strategies, such as searching for information, validating sources or authenticity, evaluating current information, repeating material to be recalled, and analyzing key idea.

- **Sensory preference-visual, verbal and kinesthetic:** A sensory modality is a system that interacts with the environment through one of the basic senses (Bissell, White & Zivin - 1971 as cited in Liu & Ginther - 2001). The most important sensory modalities are visual, auditory, and kinesthetic. Visually oriented individuals acquaint themselves with the environment through their vision. Visual discrimination refers to one's ability to differentiate one object from another. The ability to visually discriminate letters and words is essential in learning to read. In contrast, the individual with kinesthetic tendencies is more concerned with body sensations experienced through a tactile and/or kinesthetic mode. Conversely, kinesthetic style means to experience sensation through the reactions and movement of muscles, tendons, and joints. Verbalizers prefer to process information from words, either by reading or listening, rather than through images (Kirby, Moore & Shofield - 1988). Some students are equally comfortable using either visual or verbal information for learning. In fact, the differences between the visualizer and verbalizer are often not as vast as researchers have found with other cognitive styles.

2. Learning style

Learning style is also one of the most important characteristics to take into consideration when developing adaptive hypermedia (Chen & Paul – 2003, Magoulas et al. – 2003, Hsu & Dwyer - 2004). Various learning style inventories have been implemented to help teachers select the most appropriate teaching strategies to meet the learning styles of their students (Beck - 2001).

- **Kolb's learning style model:** Kolb's learning style model shows one's preferred methods for perceiving and processing information, and characterizes four basic learning modes and four learning styles. He identifies four adaptive learning modes: concrete experience (CE), reflective observation (RO), abstract conceptualization (AC), and active experimentation (AE). The learning process is seen not only active and passive, but also concrete and abstract (Jonassen & Grabowski – 1993, Liu & Ginther - 2001). Based on these four learning modes, Kolb (as cited in Jonassen & Grabowski - 1993) suggests four ways of learning styles: divergers, assimilators, convergers, and accommodators. Divergers refer to the one who relies on concrete experience and reflective observation. Assimilators depend on abstract conceptualization and reflective observation. Convergers combine abstract conceptualization and active experimentation. Finally accommodators are characterized with the combination of concrete experience and active experimentation. Learning strategies used by each learning style are presented as shown the following **figure 28**.

Learning style model	Learning strategies
Diverger	▪ Searching for information, evaluationg current information ▪ Generating metaphors, generating examples ▪ Imaging or illustrating knowledge, inferring causes ▪ Evaluating implications
Assimilator	▪ Selecting information sources, validating source or authority, ▪ Analyzing key ideas, predicting outcomes, ▪ Inferring causes, evaluating implications
Converger	▪ Setting learning goals, validating sources or authenticity, ▪ Repeating material to be recalled, outlining ▪ Predicting outcomes
Accommodator	▪ No specific learning strategies are evident from the characteristics of this type of learner

Figure 28 - Learning strategies used by each Kolb's learning style model (Jonassen & Grabowski, 1993)

- **Dunn and Dunn's learning style:** Dunn and Dunn (as cited in Jonassen & Grabowski - 1993) classified learning stimuli into four categories: environmental, emotional, sociological, and physical. They identified several learning styles within each category. The environmental variable includes four factors: sound, temperature, light, and seating/furniture design. The sociological variable includes three general factors for adults and four factors for children consisting of learning groups, presence of authority figures, learning in several ways, and for children, motivation from adults. The emotional variable consists of four factors including motivation, responsibility, persistence, and need for structure. Finally, the physical variable is comprised of four overall factors: modality preferences, intake, time of day, and mobility.

3. Prior knowledge

A student's background knowledge can influence learning outcomes (Far & Hashimoto - 2000). The prior knowledge comprises knowledge of learning content and familiarity with computer or Internet. As level of prior knowledge rises, the need for instructional support decreases, conversely, as level of prior knowledge decreases, the need for instructional support rises (Jonassen & Grabowski - 1993). On the other hand, students have different degrees of familiarity with computer hypermedia applications, which can affect their learning in an online setting (Magoulas et al. - 2003, Muir - 2001). The value of prior knowledge in predicting the student's achievement and needs of instructional supports has been demonstrated in many studies (Ross & Morrison - 1988). In addition, they argued that prior achievement measures relate directly to the instructional task, therefore, the measures should provide a more valid reliable basis for determining adaptations than other aptitude variables. Jonassen & Grabowski (1993) also stated that the existence of prior knowledge will be prerequisite for using learning strategies such as self-monitoring, search for information, integration of knowledge, paraphrasing-summarizing, comparing new knowledge with existing knowledge, beliefs, generating metaphors, generating examples, and elaboration of knowledge.

4. Motivation

Entwistle's (1981, as cited in Jonassen & Grabowski - 1993) classification of student-motivation orientation provides more hints for adapting instruction to the student's motivation state. He identified three types of students based on motivation orientation styles:

- Meaning-oriented students, who are internally motivated by academic interest
- Reproducing-oriented students, who are extrinsically motivated by fear of failure
- Achieving-oriented students, who are motivated primarily by hope for success.

On the other hand, Miltiadou and Savenye (2003) classified six motivational construct into three general families. The first is individuals' perceptions about their ability to accomplish a task. It includes constructs such as self-efficacy, locus of control, and attributions. The second family pertains to individuals' reasons or purposes for engaging in a task. It encompasses constructs such as goal orientation and intrinsic versus extrinsic motivation. The third family refers to individuals' techniques and strategies for accomplishing a task and includes self-regulation.

How has Web-based adaptive instruction been achieved?

Many researchers argued that individual differences have significant effects on student learning in Web-based instruction (e.g., Rourke & Lysynchuk – 2000, Martinez – 2001, Hsu & Dwyer - 2004). Martinez (2001) stated that individuals did best in the environment which best suited their learning orientation. The findings suggest that learning environments influence learning outcomes depending on how it matches the learning orientation. Rourke and Lysynchuk (2000) classified learning styles as assimilator, accommodator, converger, and diverger based on Kolb's Learning Style Inventory. They failed to demonstrate that the achievement of assimilators would be the highest. However, the accommodators' achievement was lower than others. This result supports that achievement in hypertext is different across learning styles. Hsu and Dwyer (2004) used two cognitive styles, field-independence (FI) and field-dependence (FD), to measure individual differences. Their study revealed that FI students were more successful than FD students on the comprehension type questions, and FD students who received the adjunct questions performed significantly better than those who didn't receive questions. They claimed "Different types of adjunct questions are differently effective for student processing different learning styles".

In other studies the Web has been proven to be an equally effective learning environment for students regardless of cognitive style (Aragon, Johnson & Shaik – 2000, Shih & Gamon - 2002). Aragon et al. (2000) used Curry's (1991) model of learning style components and effects to distinguish learning style preference: motivational level, task engagement level, and cognitive control. In their study, there was no relationship between learning style and course performance in online instruction. Shih and Gamon (2002) used Group Embedded Figure Test (GEFT) to classify learning styles as field-dependent or field-independent. They examined relationships between student achievement and learning style, pattern and strategies. The result revealed that only learning strategy had significant effect on student achievement.

What have individual differences been used in Web-based adaptive instruction?

Adaptive systems use a mechanism called "model" to understand individual needs and what a student does know and does not know (Park & Lee - 2003). These systems have used various individual differences depending on researchers. Even though individual differences are divided into four styles above, some studies used a different kind of individual difference or more than two styles.

Miltiadou and Savenye (2003) suggested the methods for ensuring students' success in the online environment based upon two parts of motivation: students' self-efficacy beliefs, goal orientation and self regulation. Using prior knowledge, Brusilovsky (2003) provided evidence that users with different knowledge level of the subject may appreciate different adaptive navigation support technologies. On the other hand, Mitchell, Chen and Macredie (2005) considered two types of prior knowledge: domain expertise and system expertise. The study revealed that the student with lower domain knowledge gained more

benefits from a hypermedia tutorial than those with higher prior knowledge. In addition, the students who enjoyed the Web and Web-based learning are more able to cope with the non-linear interaction. Specially, Liu and Ginther (2001) suggested using many models of individual differences in their adaptive design guidance: field independence-dependence, holistic-analytic, sensory preference, hemispheric preference, and Kolb's learning style model. Martinez (2000) asserted that learning orientation should be used in adaptive instruction, because cognitive perspectives particularly lack adequate consideration of how people want or intend to learn online.

What have adaptive methods been used in Web-based adaptive instruction?

Different adaptive techniques or methods have been used Web-based adaptive instruction. There are two main classes of adaptation: adaptive presentation and adaptive navigation (Brusilovsky – 2003, Chen & Paul - 2003). Adaptation at content level is called adaptive presentation, while adaptive navigation support is based on the adaptation structure level (Chen & Paul - 2003). Furthermore, Inan and Grant (2004) classified adaptive techniques into eleven methods: adaptive content, navigation and orientation, sequencing, support and feedback, facilitation, interaction, assessment, collaboration, interface, individualization level, social context. These adaptive methods can be categorized into four categories: adaptive presentation, adaptive navigation, adaptive control level and adaptive interaction. While most implemented adaptive system used one of the four methods (e.g., Brusilovsk – 2003, Hsu & Dwyer - 2004), some studies, which were not implemented but gave some guidelines or principles to develop an adaptive system, proposed to use combined methods (e.g., Liu & Ginther – 2001, Muir - 2001).

1. Adaptive presentation

This method is personalization of courses or contents to match with student's characteristics. Adaptive layout or interface can be involved in this method. Different users will receive different text as a context of the same page according to their preference or styles (Chen & Paul - 2003). Hsu and Dwyer (2004) used this method using adjunct questions. Interface for instruction can be adapted by changing layout of page: font, color or location. Oliver and Herrington (1995) stated that improper usage of color, graphics or images enhances material aesthetic and conversely decrease learning by distracting user from instructional task.

2. Adaptive navigation

This method aims to provide students with effective paths through learning materials by changing visible links, guiding students to the next item to be learned in accordance with individual differences. Unlike the linear structure of books and traditional computer-assisted learning, hypermedia presents information with non-linear format (Khalifa & Lam -2002). Students are provided with freedom of navigation in non-linear structure. In this way, learners can construct their own individualized knowledge structures by cross-referencing the related topics in the subject domain. (Mitchell et al, 2005). The most popular techniques for adaptive navigation support include direct guidance, sorting, hiding, annotation and generation (Brusilovsky - 2001 as cited in Brusilovsky - 2003). He demonstrated that different links for navigation are effective for students with different knowledge level.

3. Adaptive control level

There are opposed approaches to computer assisted learning systems: more directive tutor-centered style and more flexible student-centered browsing approach. The level of adaptation can be ranged from full system-control to full student-control. Diverse adaptive control levels to accommodate individual differences would enhance learner's learning autonomy on the Web-instruction. Magoulas et al. (2003) implemented several levels of adaptation in their system.

4. Adaptive interaction

Online contact, support and feedback can be included in adaptive interaction. Student interaction, such as student-student contact and student-teacher contact, should be provided in order to reduce learning anxiety and maximize learning performance (Liu & Ginther - 2001). Feedback and support can also play important role in student learning and attitude by providing review, hint or tips.

What are the limitations in previous studies?

The limitations in previous studies can be summarized: few implementations, limited adaptive methods, fixed adaptation, and no profile information. First, there is little empirical evidence for the effectiveness of adaptive hypermedia system (Park & Lee - 2003). Many previous studies have aimed to analyze an effect of individual differences on Web-based instruction or simply suggested guidelines to develop adaptive systems. There is not much research regarding adaptations in real environments.
Second, even though a few adaptive systems have implemented, most systems used only one or two adaptive methods. There is no single approach to adapt in order to accommodate individual differences (Magoulas et al. - 2003). The adaptive content is not the only part of adaptation. Various adaptive methods which are effectively combined to accommodate individual differences are needed. Third, the adaptive methods used in previous studies are fixed, meaning that a certain adaptive technique was predetermined

by researchers. DeBra (2000) pointed out that if prerequisite relationships in adaptive hypermedia systems are omitted by the user or just wrong, the user may be guided to pages that are not relevant or that the user cannot understand. Bad guidance is worse than no guidance. Therefore, more flexible and changeable adaptation should be implemented to efficiently match the needs of individuals.

Last, as Magoulas et al. (2003) mentioned that further research is needed to exploit the information stored in the student model, a profile of student should be accumulated in database. The profile may include the learning progress, the result of assessment, or the attitude to the instruction. The information would be useful in order to provide additional materials or measure effectiveness of the adaptive system.

5.4.2.2 Case Based Reasoning

What is case-based reasoning?

Artificial Intelligence (AI) can be used in building intelligent educational software, because it can provide an excellent methodology for learning and reasoning from the human experience (Salem - 2000). The field of AI and education has traditionally been technology focused. Salem (2000) classified six AI fields in education: Knowledge representation (KR), Case-base reasoning (CBR), Natural Language Processing (NLP), Intelligent Tutoring System (ITS), Intelligent Tutoring Systems Authoring Sheets (ITSAAs) and Distributed Artificial Intelligence (DAI). Among these fields, CBR receives increasing attention within the AI and the education communities.

CBR is based on human intuition in which new problems are often similar to previously encountered problems; therefore, past solutions may be used in the current situation. Kolodner (2004) illustrated CBR reasoning as that "We naturally bring our previous experience and knowledge to bear in interpreting new situations we encounter, draw conclusions based on explanations and on similarities between situations, and anticipate when this new thing we've just learned might be applicable.". CBR is a generation of multiple learning techniques that use the way of applying experiences in problem solving by retrieving previous cases that are similar to the current problem situation (Schmidt - 2003, Mamaghani - 2002). According to Kolodner (2004), CBR is a kind of analogical reasoning that focuses on reasoning based on previous experience. A previous experience can play roles:

- Suggest a solution to a new problem or a way of interpreting a situation.
- Warn of a problem that will arise.
- Allow the potential effects of a proposed solution to be predicted.

The root of case-based reasoning in AI is found in the works of Roger Schank on dynamic memory and the central role that is a reminding of earlier situations and situation patterns in 1982. Other trails into the CBR field have come from the study of analogical reasoning by Gentner in 1983 (Aamodt & Plaza - 1994). The first system that might be called a case-based reasoner was the CYRUS (Kolodner, 1983 as cited in Aamodt & Plaza - 1994). It was basically a question-answering system with knowledge of the various travels and meetings of former US Secretary of State Cyrus Vance. Another basis for CBR was made by Bruce Porter, the machine learning problem of concept learning for classification tasks. Edwina Rissland and her group also made the role of precedence reasoning in legal judgments in 1983 (Aamodt & Plaza, 1994).

The advantage and drawbacks of CBR were presented by Cook (1997). CBR can make effective decisions, improved accuracy and increased capability. However, the first drawback is that adapting solutions of old experiences to a new situation might not be an optimal solution. Secondly, the solution quality relies heavily on the number of previous experiences cataloged in the case library. Lastly, the decision may not be enough to apply to real situation when many new situations occur.

What makes up case-based reasoning?

Kolodner (2004) described that the key to CBR is insertion and retrieval. Insertion is that while engaging in an experience, a reasoner interprets the situation that can be productively applied in the future. The case is labeled according to its applicability conditions. Retrieval is that while engaging new situation, a reasoner looks for previous cases which are usefully similar to the new case based on understanding of the new situation. The process model of the CBR cycle comprises four Re's (Aamodt & Plaza - 1994, Mamaghani - 2002):

1. Retrieve the most similar case.
2. Reuse the case to attempt to solve the problem.
3. Revise the proposed solution if necessary.
4. Retain the solution as part of a new case.

A case describes one particular diagnostic situation and records several features and their specific values occurred in that situation. It is important that a case is not a rule. In the step of retrieving, a system should determine a similarity between current situation and cases saved in a database. The purpose of calculating a similarity is to select cases that can be adapted easily to the current problem. A Degree of similarity means a utility of reusability of solution. Efficient case retrieval is essential for large case bases. There are various indexing and retrieval algorithms in this area: nearest-neighbor retrieval, inductive retrieval, and question-based retrieval. After retrieval step, the system recalls the best matching cases. Next, cases are adapted to form a new solution by testing and repairing those in the revise step, and then new case is retained in a database. Wang, Moore, Wedman & Shyu (2003) stated that case retrieval is closely related to case representation and indexing.

Case-Based Reasoning in education

CBR has been applied to educational realms as a case-based learning assistant system in Physics (Clasp , Fung & Kemp - 2000), stories to support problem solving (Jonassen & Hernandez-Serrano - 2002), an answering system in distance learning (Tsinakos - 2003), problem-solving emulation (Sormo & Admodt - 2002) and the Knowledge Innovation for Technology in Education (KITE) project (Moore, Means & Kim - 2004).
Fung and Kemp (2000) developed CLASP to support students taking physics class with providing solutions for studying and exercises with answers. The style of presenting the case will follow the user's wishes but only two modes of interaction: solution studying and guided-problem solving.

In Jonassen and Hernandez-Serrano's system (2002), the rationale and means for analyzing, organizing, and presenting stories to support problem solving are defined by case-based reasoning. The stories as cases are playing a role of exemplars of concepts, principles, or theories being taught by direct instruction. Furthermore, the stories can be advice for students for helping them learn to solve problems. Stories can be selected by students based on their perceived relevance, or they may be automatically selected by the learning environment using a CBR algorithm.

Sormo and Aamodt (2002) developed a problem-solving emulation module based on CBR in ITS system. The feature of the module is communication between a case-based reasoner and students. The system consists of a student model and an expert model. If a student selects how to solve a problem, the student model system will compare it with cases solved by other students, and send a data to an exercise selector. The selector also compares it with other cases solved by experts in the expert model.

Tsinakos (2003) used CBR techniques in the process of answering students' question in distance learning. He named the Process of Identification of Similar Question (PISQ). The system provides relevant answers to questions of students using two search methods. He described a controlled vocabulary search and free text search among the contents of a database which is Educational Knowledge Base. The PISQ automatically search previous cases and send an appropriate answer to a student. If the system fails to find a relevant case, the question is posted directly to a tutor, and he responds and identify if the question can be considered a new case. The new case of question is retained depending on the tutor's decision.

Moore, Means and Kim (2004) conducted the Knowledge Innovation for Technology in Education (KITE) project funded by the U.S. Department Education. They described "The project seeks to assist teachers in learning how to integrate technology into their teaching by presenting cases containing technology stories". Nearly 1000 real stories which are experience of in-service teachers across the U.S.A. were stored in a case library. The KITE project recalls many cases with a number which means how much percentage related to current problem and suggests adaptation of existing solutions.

In addition to retaining of new case as the last step of CBR, new solution is also created and stored in the library. The Technology Integration Learning Environment (TILE) is a part of the KITE project. This Web-based tool is used to show how to integrate technology in instructional activities and support the case based learning process for pre-service teachers.

In previous studies using CBR, not much empirical researches have been conducted in education area because CBR technology is relatively new. In addition, most CBR systems have been developed based on mostly computer science's view. For example, in Reyes and Sison's (2000) CBR tutor, there is no consideration about instructional theory and strategies. Implementation of CBR system based on robust educational basement is required.

5.4.2.3 Learning objects

What are learning objects?

Learning objects, which make it possible to combine and reproduce instructional components, are digital resources having reusability to support learning (Gibbon, Nelson & Richards – 2000, Martinez & Bunderson - 2000). The IEEE Learning Technology Standard Committee (LTSC) defines learning objects as "any entity, digital or non-digital that can be used, re-used, or referenced during technology supported learning" (LTSC - 2000). Wiley (2000) also defined learning objects as the smallest units within a larger instructional structure. However, his definition of leaning objects was somewhat different from LTSC's. Wiley described learning objects as "any digital resource that can be reused to support learning". Martinez and Bunderson (2000) also claimed that learning objects can be called "only if the objects are used for instructional purposes, meaning that learning objects are content meaningfully presented to accomplish specific objectives related to learning, they are designed using a conceptual framework embedded with instructional theory, strategies, and methodology".

Wiley offered the following taxonomy of learning objects that can be combined into an instructionally useful content: fundamental, combined-closed, combined-open, generative-presentation, and generative-instructional. The following taxonomy differentiates between five learning objects types. A Fundamental type is an individual digital source resource uncombined with any other (e.g., JPEG file). A Combined- closed type is a small number of digital resources combined at design time by the learning object's creator, whose constituent learning objects are not individually accessible for reuse from itself (e.g., a video clip). A Combined-open type is a larger number of digital resources combined by a computer in real-time when a request for the object is made, whose constituent learning objects are directly accessible for reuse from itself (e.g., a Web-page). A Generative-presentation type is a logic and structure for combing or generating and combing lower-level learning objects (e.g., a JAVA applet).

A Generative-instructional type is a logic and structure for combing or generating learning objects (Fundamental, Combined-closed types, and Generative-presentation) and evaluate student interactions with those combinations.

Granularity is the most difficult problem facing the designers of learning objects (Wiley - 2000). The ways in which learning objects can be combined with one another to facilitate learning are entirely dependent upon their structure. Gibbon et al. (2000) stated that learning objects must be seen in terms of their place in an architectural hierarchy capable of finding, comparing, and selecting them. When creating learning objects for instructional use, two design issues are preeminent: granularity and combination. Wiley, Gibbons and Recker (2000) described granularity with two views: The first view is the size of media ranged from a full course being the largest grain size to a single element of instructional media being the smallest grain size. The second view is the complexity of the content whose learning the object is meant to support. That is, granularity is the degree to which elements of domain content are combined within learning objects. This is a content or message-centric definition of granularity.

Gibbon et al. (2000) attempted to generalize layers of the purposive design process based on Brand's (1994) description of the principle of layering in designs: Site, structure, skin, services, space plan, and stuff. They enumerated the following list of instructional design layers: Model, problem, strategy, message, representation, and media-logic. According to Gibbons et al., there are two types of definitions of the term granularity, "simple definition" and "robust definition". Simple definitions are focusing on the state of combination of elements within a single design layer (e.g., media) of learning objects. Robust definitions are considering the state of combination of elements at each of the layers of instructional design. The framework of Gibbons et al. provides evidence that learning technology specifications and standards efforts are ignoring critical instructional design issues. That is current simple definitions of granularity are linked to the design layers furthest removed from instructional design: media and message.

What is metadata?

Metadata is data about learning objects. It is a resource description about content, quality or features that allow us to locate an item very quickly without investigating all the individual items on the Web (Dillon – 2000, Wiley – 2000, Jones - 2003). According to Martinez and Bunderson (2000), learning objects are indeed a good idea, but as long as they lack instructional value, we will be unable to use them effectively. Metadata standards define what data needs to be collected and stored to provide descriptive information about a content object. Metadata standards theoretically should also enable the appropriate use of a content object as learning objects.

Many groups are working together to define common international standards that the world can adopt for describing learning objects that can be interoperable, reusable, and effectively managed and presented. Their common interest is to fine minimum set of metadata standards that will support the world-wide development of learning objects for multiple purposes. Hodgins (2000) classified metadata into objective metadata and

subjective metadata. Objective metadata is factual data and most of which can be generated automatically, such as, physical attribute, date, author, operational requirements, costs, identification numbers and ownership. Subjective metadata is more varied and valuable attributes of learning objects determined by the person or group who creates the metadata. Your opinion of the product, for example, whether it worked as well as a fresh ingredient he argued that "Personalization becomes the key element of learning, subjective metadata become increasingly important. The value of the learning objects goes up as its associated metadata increase in richness and completeness, the value of the data objects also goes up as it approaches its smallest potentially useful size". Standardization enables information which originates in one context to be used in another in ways that are as highly automated as possible (Duval - 2001). There are three important accredited standardization organizations: IEEE LTSC (Learning Technology Standardization Committee), CEN ISSS LTWS (Learning Technologies Workshop) and ISO/IEC JTC1 SC36. He also described two main standardized metadata for education: IEEE LTSC LOM and Dublin Core-Education. LTSC has developed an elaborate metadata scheme with hierarchical structure. Data elements are regrouped under categories.

The elements under educational category are as the follows (Duval - 2001):

- Interactivity type (active versus expositive).
- Learning resource type (exercise, simulation, questionnaire, etc.).
- Interactivity level (from very low to very high).
- Semantic density (idem).
- Intended end user role (teacher, author, learner, manager).
- Context (primary education to vocational training).
- Typical age range.
- Difficulty (from very low to very high).
- Typical learning time.
- Description.
- Language of the typical intended user.

On the other hand, the Dublin Core (DC) is a set of 15 metadata elements intended to facilitate discovery of electronic resources in a general sense. LOM was originally developed specifically for the domain of education and training, whereas the Dublin Core was originally developed for general resources, and is now being adapted for the specific field of education and training.

There are many different consortia (ADL, AICC, ARIADNE, IMS, etc) that work on actual implementation of metadata and standards (Duval - 2001). One of the promising standards is SCORM (Sharable Content Object Reference Model) released by ADL (Advanced Distributed Learning organization) (Jones - 2003). This reference model applied LOM to raw media (7 mandatory items), content (15 mandatory items) and courses (15 mandatory items). Since SCROM version 1.0 was released in January 2000, SCORM 2004 was released in January 2004.

CBR and learning objects in adaptive learning

How can CBR affect adaptive instruction?

CBR techniques can increase accuracy in adaptive instruction. The key point of adaptive instruction is offering the most appropriate individualized instruction to students according to their differences or needs. CBR utilizes the advantages of using previously experienced cases to enhance the tutoring capability of the system (Reyes & Sison - 2000). In many previous CBR systems within education, the referred cases were instructional contents: stories, experience or problem solutions (e.g., Sormo and Aamodt – 2002, Tsaganou, Grigoriadou & Cavoura - 2002, Tsinakos – 2003, Moore et al. - 2004). Therefore, the purpose of CBR is to select the best reference to help learners solve problem. From the other point of view, the focus of cases can be moved to how to adapt instruction to each student. The case-based reasoner can play an important role in making instruction individualized because it can make effective decisions based on previous cases. The most effective decision of adaptation can be made by CBR.

CBR technique can give more flexibility to adaptive instruction. Fixed or constant adaptive methods or strategies have been used in most adaptive systems. Those adaptations were predetermined by researchers, so it was impossible for the predetermined strategies to be changed even if the adaptation was insufficient. Adequate design models must flexibly adjust to individual's characteristics (Bastiaens & Merrienboer - 2002). CBR can provide excellent methodology for flexibility and variation. Based on previous case of instruction for each student, the system can adjust or change adaptive level or method. The database of case or profile can be used as a basis of reasoning to decide whether the same kind of instructional unit will be offered. Adaptive system can be evolved and be flexible by CBR.

How can learning objects be used in adaptive instruction?

Learning objects can be sequenced for dynamically individualized lessons. Learning objects have potential for reusability, generativity, adaptability and scalability (Gibbon et al. - 2000). They can be joined together to perform an orchestrated instructional function. Various learning objects in terms of small units can be used for different adaptive methods: adaptive presentation, navigation or interaction. In addition, metadata offers the cataloguing information necessary to share, use and reuse learning objects. Therefore, metadata provides the mechanisms necessary to match learning objects to an individual's characteristics (e.g., Holzinger, Kleinberger & Muller - 2001). The effective adaptation can be accomplished by selecting the best adaptive strategy based on reasoning with previous cases and customizing instruction with learning objects.

5.4.3 Architecture of MLOID

The focus of the Matching Learning Object to Individual Differences (MLOID) system is to offer a student appropriate learning objects. In current Web-based learning, the learners are not the center of an instruction, nor do most systems consider learning characteristics of individuals. The MLOID system will individualize instruction in accordance with each learner. In order to be able to focus on the individualization, the system must have an algorithm of matching a various individual differences and a characteristic of various learning objects. Using the algorithm, the MLOID diagnoses individual differences of a student and selects relevant learning objects. As a basic CBR process, the system retrieves a successful learning case from the database, reuses the case as learning objects in the new customized instruction, revises the case by evaluation after the instruction, and retains the information in a individual differences table as well as students' record in a database.

As can be seen in below **figure 29**, the MLOID consists of diagnosis, case-based reasoner, learning model, learning object, and evaluation. The other attempt of the system is feedback process. After an instruction, a student will be asked to take an evaluation test. The system will recommend the student receive feedback depending on the score of the test. When a student requires feedback, the student can select whether he will use the same kinds of objects in previous instruction or different kinds of objects.
This process will help a student reach the objective of the instruction.

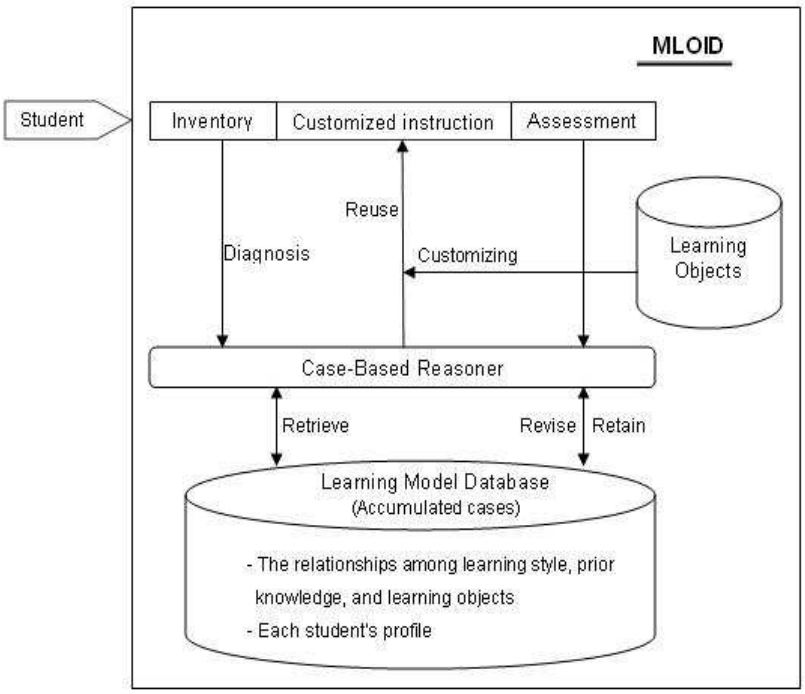


Figure 29 - The architecture of MLOID system

Diagnosis

A student is asked to take a diagnostic test in order for the system to identify the student's individual differences. The process sends a result to the CBR. For example, from the VARK (visual, auditory, reading/writing, kinesthetic) inventory, a student's learning preferences are identified along with prior knowledge on the content. These are sent to the CBR and stored in the Learning Model database.

Case-based reasoner

In our design, the task of the case-based reasoner is to select appropriate learning objects for a student to accomplish a learning objective. When the diagnosis identifies a student's individual differences, the reasoner refers to cases in the Learning model. If a student uses the system at first time, the reasoner will retrieve learning objects which are applicable to the student's individual differences. If the student already took some instructions, the reasoner refers to the Learning model as well as a profile of individual. The reasoner analyzes records of the student's performance and decides whether it will continue to use the same kind of learning objects. In selecting process, the reasoner sorts out learning materials and systemically decides most appropriate objects. Furthermore, the reasoner adds a case to individual differences and profile of each student in learning model.

Learning model

The learning model database, which accumulates cases, is a basis of reasoning. The database consists of two tables: the individual differences table and the profile of learners table. In the individual differences table, cases of relationship between identified individual differences and characteristics of learning objects are stored. In the profile of learners table, the records of how a student has performed within the instruction along with the learning objects used are saved to each student's case.

Learning object database

The learning object database contains the sharable objects with associated metadata. The standard of metadata follows SCORM.

Evaluation and feedback

At the end of instruction, a student can receive feedback depending on a score of an evaluation and request to the system different kinds of objects. All records are stored in the Learning Model database **[26]**.

5.5 Looking towards the future

It is believed that CBR-inspired educational approaches will be making their way more into the e-learning mainstream. For example, a curriculum for computer science education has been designed based on CBR principles at Carnegie Mellon University West in California. All learning is through challenges, and learners have the opportunity to test what they are doing in simulations of the real world. A review by Chapman (2003) describes more than 50 simulation-based e-learning vendors, reflecting a growing interest in constructivist learning environments, perhaps due to what Aldrich (2004) reports: The first generation of content-based e-learning systems were not as successful as predicted.

While this state of the world has opened the future to simulation-based e-learning, we feel it important to caution that not all simulation-based learning is equal. CBR warns us that unless learners can interpret outcomes well, they will not be able to learn well from their experiences. This suggests that the feedback given by simulations needs to be specific enough about outcomes for learners to be able to clearly identify their misconceptions and reasoning errors. It suggests, too, that simulations need to be embedded into learning environments that provide access to help with identifying those misconceptions. CBR suggests, too, that learning from simulation will happen best when the simulation is used as a tool to help the learner achieve an appealing challenge or mission.

Related efforts focus on game-based learning systems (Prensky - 2001, Gómez-Martín *et al*. - 2004) that embed simulations into serious games to promote motivation in the learner. Also related to motivation and amenable for the application of CBR ideas is the use of animated pedagogical agents (Lester *et al*. - 2001) – lifelike computer characters inhabiting a virtual world for pedagogical purposes who can tell the learner the story she needs to hear for solving the problem at hand. CBR ideas, and in particular, goal-based and design-based approaches, are especially relevant to simulation-based learning, and synergies should be explored in the coming years (Gómez-Martín *et al*. - 2005). Kolodner (1997) predicted "The use of such a software environment (simulations) acrossmany different problem-solving and design experiences will promote learning of a full range of cognitive, social, and self-directed learning skills..." In many ways, this prediction has become reality. The huge evolution of graphics technology, both in software and graphics cards, mainly driven by the game industry, has led to a situation where simulations are relatively cheap to build and use within e-learning systems (Jiménez-Díaz *et al*. - 2005). We encourage those who are designing simulation-based learning technologies **[27]**.

Chapter 6: Bibliography

[1]http://www.elsevier.com/wps/find/bookdescription.cws_home/680525/description#description , Applying Case-Based Reasoning: Techniques for Enterprise Systems, by Ian Watson, 1997.

[2]http://wiki.lawguru.com/index.php/Case-based_reasoning

[3]http://www.answers.com/topic/instructional-design-case-based-reasoning , Education Encyclopedia: Instructional Design: Case-Based Reasoning

[4]http://www2.iiia.csic.es/People/enric/AICom.html , Published in: AICom - Artificial Intelligence Communications, IOS Press, Vol. 7, pp. 39-59, by A. Aamodt & E. Plaza , 1994.

[5]http://www.aiai.ed.ac.uk/links/cbr.html

[6]http://www.ai-cbr.org/classroom/cbr-review.html , Case-Based Reasoning: A Review, by Ian Watson & Farhi Marir, 1994.

[7]http://en.wikipedia.org/wiki/Case-based_reasoning

[8]http://www.iis.uni-hildesheim.de/files/staff/althoff/Publications/CBR-REVIEW_gesamt.pdf , A review of industrial case-based reasoning tools, p18-19, by Klaus-Dieter Althoff - Eric Auriol, Ralph Barletta & Michel Manago, 1995.

[9]http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.66.1581 , Case-based Reasoning and Software Engineering, Citation: Managing Software Engineering Knowledge, New York : Springer, pp. 181- 198, by Martin Shepperd, 2003.

[10]http://www.foibg.com/ibs_isc/ibs-04/IBS-04-p05.pdf , Case-based reasoning tools from shells to object-oriented frameworks, Published in: ICCOMP'08 Proceedings of the 12th WSEAS international conference on Computers, by Essam Abdrabou & AbdEl-Badeeh Salem , June-July 2008.

[11]http://techlab.bu.edu/files/resources/articles_tt/Integration%20of%20ART-Kohonen%20neural%20network%20and%20case-based%20reasoning%20for%20intelligent%20fault%20diagnosis.pdf , Integration of ART-Kohonen neural network and case-based reasoning for intelligent fault diagnosis, Published: Expert Systems with Applications, Vol.26, Issue 3, pp. 387–395 , by Bo-Suk Yang, Tian Han & Yong-Su Kim, April 2004.

[12]http://www.dsl.uow.edu.au/content/files/techreports/kang03intelligent.pdf , , Intelligent knowledge acquisition with Case-Based Reasoning techniques , Published: Technical Report, by Seung Hwan Kang & Sim Kim Lau, May 2002.

[13]http://www.verdandetechnology.com/images/stories/verdande/publications/knowledge%20acquisition%20and%20learning%20from%20experience%20-%20the%20role%20of%20case-spesific%20knowledge.pdf , Knowledge acquisition and learning by experience - The role of case-specific knowledge, Book: Machine Learning and Knowledge Acquisition: Integrated Approaches, Ch. 8, 99, pp. 197-245, by Agnar Aamodt, 1995.

[14]http://www.emis.de/journals/NSJOM/Papers/38_3/NSJOM_38_3_219_226.pdf , Case-Based Reasoning framework for generating decision support systems, Published: Novi Sad J. Math. Vol. 38, No. 3, 219-226, by Vladimir Kurbalija & Zoran Budimac, 2008.

[15]http://www.iiia.csic.es/~mantaras/RRRR.pdf , Retrieval, reuse, revision, and retention in case-based reasoning, Published: The Knowledge Engineering Review, Vol. 20:3, 1–2, by Ramon López de Mántaras, David McSherry, Derek Bridge, David Leake, Barry Smyth, Susan Craw, Boi Faltings, Mary Lou Maher, Michael T. Cox, Kenneth Forbus, Mark Keane, Agnar Aamodt & Ian Watson, 2005.

[16]http://ibug.doc.ic.ac.uk/media/uploads/documents/courses/syllabus-CBR.pdf, Introduction to Machine Learning & Case-Based Reasoning , by Maja Pantic.

[17]www.cs.auckland.ac.nz/~ian/papers/cbr/cbr_tools.zip , Case-Based Reasoning tools: An overview, Proceedings of the 2nd UK Workshop on Case Based Reasoning, University of Salford April 10th, Published: AI-CBR/SGES, pp. 71-88, by Ian Watson, 1996.

[18]http://home.cc.gatech.edu/ccl/45 , Paper 1: JL Kolodner, Chapters 1 of Case-Based Reasoning, Morgan Kaufmann Publishers, *by* Janet L. Kolodner, 23-24 page, 1993

[19]www.karlbranting.net/papers/xijun.wang/chapt1.doc

[20]http://books.google.gr/books?id=01cvDWrxEdUC&pg=PA663&lpg=PA663&dq=CBR+Model+for+the+Intelligent+Management+of+Customer+Support+Centers&source=bl&ots=z0xklZrfNW&sig=FUYNn-EHh9SlrPyFVHYef-OuYuE&hl=el&ei=I4LJTrT0CsbLhAeQiJXTDw&sa=X&oi=book_result&ct=result&resnum=3&ved=0CDMQ6AEwAg#v=onepage&q=CBR%20Model%20for%20the%20Intelligent%20Management%20of%20Customer%20Support%20Centers&f=false , CBR Model for the Intelligent Management of Customer Support Centers, Book: Intelligent data engineeing and automated learning – IDEAL 2006, Published: Springer, by Stella Heras Barberá, Juan Ángel García-Pardo, Rafael Ramos-Garijo, Alberto Palomares, Vicente Julián, Miguel Rebollo, Vicente J. Botti , pp. 663-670, 2006.

[21]http://www.cs.indiana.edu/~leake/papers/p-96-01.pdf , CBR in context: The present and future, Book: Case-Based Reasoning: Experiences, lessons, and future directions, Published: AAAI Press/MIT Press, pp. 1-35, by David B. Leake, 1996.

[22]http://lab.geog.ntu.edu.tw/course/ginformation/hw/kids/Applying%20case-based%20reasoning%20techniques%20in%20GIS.pdf , Applying case-based reasoning techniques in GIS, Published: Int. j. geographical information science, Vol. 13, No. 1, 9-25 , by A. Holt & G. L. Benwell, pp. 12-14, 1999.

[23]http://www.waset.org/journals/waset/v29/v29-50.pdf , Distributed Case Based Reasoning for Intelligent Tutoring System: An Agent Based Student Modeling Paradigm, Published: World Academy of Science, Engineering and Technology 29, pp. 273-276, by O. P. Rishi, Rekha Govil & Madhavi Sinha, 2007.

[24]http://dbay.ndsu.edu/~mooadmin/docs/cve-02-paper2.pdf , Case-based Tutoring in Virtual Education Environments, Published: ACM Collaborative Virtual Environments, by Patrick M. Regan & Brian M. Slator, pp. 1-7, 2002.

[25]http://www.cs.ust.hk/~qyang/Docs/2002/australianaiconf.pdf , The Application of Case Based Reasoning on Q&A System, Published: AI '02 Proceedings of the 15th Australian Joint Conference on Artificial Intelligence: Advances in Artificial Intelligence, by Peng Han, Rui-Min Shen, Fan Yang & Qiang Yang, pp. 704-713, 2002.

[26]http://memphis.academia.edu/michaelmgrant/Papers , Design of individualized instruction with learning object using case-based reasoning in WBI, Published: Association for Educational Communications and Technology, October 18-22, by Jongpil Cheon & Michael Grant, p1-13, 2005.

[27]http://journals.cambridge.org/action/displayAbstract?fromPage=online&aid=435261 , Published: The Knowledge Engineering Review, Vol. 20:3, by Janet L. Kolodner, Michael T. Cox & Pedro A. Gonzalez-Calero, pp. 299-303, 2005.