At its most easy, CBR is based on the inspection that, when we answer a problem, we often base our solution on one that worked successfully for a similar problem in the past. An example of CBR would be someone lashing to work in the morning. That person doesn’t clearly plan his route; he just takes the route he usually takes. If that person meets a traffic jam, he may memorize how he avoided that jam in the past. If that person tries a new route to avoid the traffic jam and he is successful, he will remember it and possibly use it in the future. Another example would be a student doing a statistics assignment. When he sees a new problem, he will try to see if there is any resemblance between the current problem and the ones he learned in the class. If there exists one past problem whose solution traces fit onto the current one, the student would relate the same past method to solve the current problem. These examples show the major two steps in the process of CBR — retrieval and alteration, which will be discussed further.

A more formal definition for CBR: CBR is a simple problem-solving paradigm that involves matching a current problem against problems that were successfully solved in the past. The procedure can be amplified by adapting solutions so that they more closely match the current problem.

4.1 DESCRIPTIVE FRAMEWORK

Our framework for describing CBR methods and systems has two main parts:

- A process model of the CBR cycle
- A task-method structure for case-based reasoning
The CBR cycle

The two models are opposite and represent two views on case-based reasoning. The first is a dynamic model that identifies the main subprocesses of a CBR cycle, their interdependencies and products. The second is a task-oriented view, where a task decomposition and linked problem solving methods are described. The framework will be used in successive parts to identify and argue important problem areas of CBR, and means of dealing with them.

4.2 CBR CYCLE

As it was mentioned above, CBR consists of two main parts — retrieval and adaptation. But in fact and in detail, CBR is constituted with four RE’s

According to Kolodner [1996], the CBR working cycle can be described best in terms of four Processing stages:

- **Case retrieval:** after the problem situation has been assessed, the best matching case is searched in the case base and a rough solution are retrieved.
- **Case adaptation:** the retrieved solution is adapted to fit better the new problem.
- **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.
- **Case-base updating:** If the solution was confirmed as correct, the new case may be added to the case base.

Aamodt and Plaza [1994] give a slightly different scheme of the CBR working cycle comprising the four REs (Fig. 2):

- RETRIEVE the most similar case(s);
- REUSE the case(s) to attempt to solve the current problem;
- REVISE the proposed solution if necessary;
- RETAIN the new solution as a part of a new case.
The descriptions of above cycles are as follows:

**Retrieve**: Given a target problem, retrieve cases from memory that is relevant to solving it. A case consists of a problem, its solution, and, typically, comments about how the solution was derived. For example, suppose Fred wants to prepare blueberry pancakes. Being a novice cook, the most applicable experience he can recall is one in which he effectively made plain pancakes. The procedure he followed for making the plain pancakes, together with justifications for decisions made along the way, constitutes Fred’s retrieved case.

**Reuse**: Map the solution from the previous case to the target problem. This may involve adapting the solution as essential to fit the new situation. In the pancake example, Fred must adapt his retrieved solution to include the addition of blueberries.

![Fig. 4.1: CBR cycle (adapted from Aamodt and Plaza [1994])](image-url)
**Revise:** Having mapped the previous solution to the target situation, test the new solution in the real world (or a simulation) and, if necessary, revise. This step is often referred to as adaptation. Suppose Fred adapted his pancake solution by adding blueberries to the batter. After mixing, he discovers that the batter has turned blue - an undesired effect. This suggests the following revision: delay the addition of blueberries until after the batter has been ladled into the pan.

**Retain:** After the solution has been successfully adapted to the target problem, store the resulting knowledge as a new case in memory. It is in this way that the system (intelligent agents system) acquires new cases and is said to learn. Fred, thus, records his newfound procedure for making blueberry pancakes, thereby inspiring his set of stored experiences, and better preparing him for future pancake-making demands.

Figure 4.1 illustrates the whole process of CBR. When user inputs a problem, the problem is interpreted and converted as a new case into the specific format of the reasoning system. Then the transformed new case enters the stage of RETRIEVAL where the new case is coordinated against the previous cases in the case library of the reasoning system. The retrieved case and the new case are both passed to next stage REUSE where the solution part of the retrieved case is applied to the new case, with the guidance of adaptation knowledge. The application of old solution involves substitution of solution features, structural modification of the solution and/or derivational replay of solution. These methods will be discussed at next section. After this stage, now the new case is along with the modified solution based on the old one. This modified solution is considered as a suggested solution, which is still incomplete because it is adapted according to the requirement of new case and this solution may have variation among its solution parts. To ensure the adapted solution is reliable, it will be passed to next stage REVISE, where the adapted solution will be further adapted based on the user feedback and additional meta-adaptation knowledge called repair. This final adapted solution is then returned as a complete solution to the new case but this does not come to an end. It
was discussed that CBR learns by accumulating new cases. However, should any newly derived case be accumulated in the case library? The answer is no. CBR should only store the cases that can contribute to future reasoning of solutions, which could not be done only by the cases in current case library. In other words, if the cases in a case library are competent enough to cover the newly adapted solution, this new solution should not be stored in order to avoid redundancy and discrepancy. Otherwise, it should be stored for future use. This leads to the final stage of CBR – RETAIN. The case considered as being able to contribute in the future is named “Learned case” and stored in the case library.

4.3 A HIERARCHY OF CBR TASKS

The process view just described was chosen in order to highlight on CBR as a cycle of sequential steps. To further decompose and describe the four top-level steps, we switch to a task-oriented view, where each step, or sub process, is viewed as a task that the CBR reasoned has to accomplish. While a process-oriented view enables a global, external view to what is occurrence, a task oriented view is suitable for describing the comprehensive mechanisms from the perspective of the CBR reasoned itself. This is coherent with a task-oriented view of knowledge level modelling. At the knowledge level, a system is viewed as an agent who has goals, and means to achieve its goals. A system description can be made from three perspectives: Tasks, methods and domain Knowledge models. Tasks are set up by the goals of the system, and a task is performed by applying one or more methods. For a method to be able to accomplish a task, it needs knowledge about the general application domain as well as information about the current problem and its context. Our framework and analysis approach is strongly influenced by knowledge level modelling methods, particularly the Components of Expertise methodology. The task-method structure we will refer to in subsequent parts of the paper is shown in figure 2. Tasks have node names in bold letters, while methods are written in italics. The links between task nodes (plain lines) are task decompositions, i.e. part-of relations, where the direction of the relationship is downwards. The top-level task is problem solving and learning from experience and the method to accomplish the task is case-based reasoning (indicated in a special way by a stippled arrow).
This splits the top-level task into the four major CBR tasks matching to the four processes of figure 1, retrieve, reuse, revise, and retain. All the four tasks are essential in order to execute the top-level task. The retrieve task is, in turn, partitioned in the same manner (by a retrieval method) into the tasks identify (relevant descriptors), search (to find a set of past cases), initial match (the relevant descriptors to past cases), and select (the most similar case). All task partitions in the figure are complete, i.e. the set of subtasks of a task is designed to be sufficient to accomplish the task, at this level of description. The figure does not show any control structure over the subtasks, although a rough sequencing of them is indicated by having put earlier subtasks higher up on the page than those that pursue (for a particular set of subtasks). The actual control is
specified as part of the problem solving method. The relation between tasks and methods (stippled lines) identify alternative methods applicable for solving a task. A method specifies the algorithm that identifies and controls the execution of subtasks, and accesses and utilizes the knowledge and information needed to do this. The methods shown are high level method classes, from which one or more specific methods should be chosen. The method set as shown is incomplete, i.e. one of the methods indicated may be sufficient to solve the task, several methods may be combined, or there may be other methods that can do the job. The methods shown in the figure are task decomposition and control methods. At the bottom level of the task hierarchy (not shown), a task is solved directly, i.e. by what may be referred to as task execution methods.

4.4 THE ADVANTAGES OF CASE-BASED REASONING

CBR suggests a model of reasoning that incorporates problem solving, understanding, and learning and integrates all with memory processes. The advantages of CBR are briefly discussed below:

- Reusing old cases is advantageous in dealing with situations that always persist. With reference to old cases, complexities of solving novel situations can be abridged.

- Traditional reasoning processes cannot recall a relevant case unless it understands the new situation it is in. This suggests that understanding or interpreting a situation is a necessary part of the reasoning cycle and both a prerequisite to problem solving and a co-requisite during problem solving. But the need for problem understanding is not specific to CBR.

- CBR does not require exact description of problem to carry on the reasoning process while traditional reasoning (rule-based reasoning) cannot work with unfinished problem description.

- CBR emphasizes the use of concrete instances over abstract operators because they can provide more guidance and operational knowledge in solving a new problem than can abstract operators. Furthermore, they can show application and use of knowledge that abstract operators do not supply.
CBR emphasizes manipulation of cases over composition, decomposition, and re-composition processes. Though sometimes composition tasks still occur in reasoning, reasoning using concrete cases will come at first and then composition of operators next.

- It is usually necessary to adapt an old solution to fit a new situation because no old case is ever exactly the same as a new one. Adaptation compensates for the differences between an old situation and a new one, while most parts of the old case can still be reused.

- Learning of CBR occurs as a natural consequence of reasoning by accumulating new cases. Successful past cases can give information or method to solve current problem while unsuccessful past cases can warn that certain undesired results, happened in the past, should be avoided in current situation.

- Evaluation of cases can be based on feedback and follow-up procedures to judge whether the cases are useful and give contribution to learning. If cases are learnt (stored) without any follow-up analysis and evaluation, the reasoning using past cases will be unreliable.

4.5 DISADVANTAGES OF CBR:
- Use old cases blindly
- Bias towards old cases for new problems
- New user may not reminded of most appropriate set
- Does not fully explore the solution space. So most optimal solution may not be found

4.6 TYPES OF CASE BASE REASONING

1. Textual CBR: Here in this type of Case Base Reasoning Free text is used
2. Conversational CBR: List of question and answers are used in conversational type of Case Base Reasoning. Here no common case structure are used
3. Structural CBR: Database like illustration is done in structural Case Base Reasoning Ashley et al. [2003].
4.7 KNOWLEDGE REPRESENTATION

A case is a *unique knowledge entity* describing a problem and solution. It can be represented as a single database.

Usually representation is:

- A problem point to one or more case.
- A case has a single solution.
- A question can influence one or more case.

4.8 LOCAL SIMILARITY

Similarity between two cases is based on the similarity between the two cases features. The local similarity calculation depends on the type of the feature.

Following are common methods to calculate local similarity:

- **Numeric**: \( \text{sim}(a, b) = \frac{|a-b|}{\text{Range}} \)
- **Sumbolic**: \( \text{sim}(a, b) = \begin{cases} 1 & \text{if } a=b \\ 0 & \text{if } a \neq b \end{cases} \)
- **Multi-valued**: \( \text{sim}(a, b) = \frac{\text{card}(a \cap b)}{\text{card}(a \cup b)} \)
- **Taxonomy**: \( \text{sim}(a, b) = \frac{h(\text{common node}(a, b))}{\min(h(a), h(b))} \)

Where

- \( \text{Card} \) is the cardinality (size) of the set
- \( \text{Range} \) is the absolute value of difference between the upper and lower boundary of the set.
- \( h \) is the height (number of levels) of the taxonomy tree.

4.9 CASE INDEXING

A CBR system's ability to retrieve relevant cases quickly and correctly from its case base is its main power. It builds a structure that will return the most suitable case(s) at high speed. Case base indexing minimizes the number of cases that have to be evaluated at run time and is required for a large set of cases as linear searched will yield a probability long retrieval time. Within the CBR community, an explicit formal specification (i.e., ontology) of what the terms “indices” and “indexing” actually mean in
terms of a CBR system has not been established yet. Kolodner [1996] identifies indexing with an convenience problem by J. Kolodner [1996], that is, with the whole set of issues inherent in setting up the case base and its retrieval process so that the right cases are retrieved at the right time. Thus, case indexing involves assigning indices to cases to facilitate their retrieval. CBR researches proposed several guidelines on indexing I. Watson and F. Marir [1994]. Indexes should be:

- predictive of the case relevance;
- recognizable in the sense that it should be understandable why they are used;
- abstract enough to allow for widening the future use of the case base;
- Concrete (discriminative) enough to facilitate efficient and accurate retrieval.

Three different methods used in case indexing are nearest neighbor, induction and knowledge guide. In Nearest Neighbor the system would simply prefer cases that match more features to a case that matched fewer using a statistical method to determine best set of features. Each new case is compared with all other cases in the database. Induction-based system use a decision tree for retrieval as compared to nearest-neighbor indexing which is more associative, and induced decision tree is hierarchical and static. A simplest knowledge-guided approach uses human knowledge to the induction process by manually identifying known case features that are considered important and useful for case retrieval. Cases are reviewed for their important features and the appropriate questions are passed to query the user about the existence or absence of features. Different methods for case retrieval reuse, solution testing, and learning are summarized, and their actual realization is discussed in the light of a few example systems that represent different CBR approaches. It is a problem solving paradigm that in many respects is fundamentally different from other major AI approaches. Instead of relying only on general knowledge of a problem domain, or making associations along comprehensive relationships between problem descriptors and conclusions, CBR is able to utilize the specific knowledge of before experienced, concrete problem situations (cases).

4.10 CASE RETRIEVAL
Case retrieval is a process that a retrieval algorithm retrieves the most similar cases to the current problem. Case retrieval requires a combination of search and matching. In general, two retrieval techniques are used by the major CBR applications: nearest neighbor retrieval algorithm and inductive retrieval algorithm.

4.10.1 Nearest-Neighbor Retrieval

Nearest-neighbor retrieval is a simple approach that computes the similarity between stored cases and new input case based on weight features. A typical evaluation function is used to compute nearest-neighbor matching Kolodner [1994] as shown as

\[
similarity(Case_1, Case_2) = \frac{\sum_{i=1}^{n} w_i \times sim(f_i^1, f_i^2)}{\sum_{i=1}^{n} w_i}
\]

Where \( w_i \) is the importance weight of a feature, \( sim \) is the similarity function of features, and \( f_i^1 \) and \( f_i^2 \) are the values for feature \( i \) in the input and retrieved cases respectively. This formula displays a simple scheme for nearest-neighbor matching.

![Figure 4.3 Inductive Retrieval](image)

Inductive retrieval algorithm is a technique that determines which features do the best job in discerning cases and generates a decision tree type structure to organize the cases in
CASE BASE REASONING

memory. This approach is very useful when a single case feature is required as a solution, and when that case feature is dependent upon others. Here is a completed decision tree (see Figure 4.4) generated from the data in Table 4.1 The task is to predict the status of a loan from features of the loan applicant (income, job status and repayment).

Table 4.1 Four loan cases

<table>
<thead>
<tr>
<th>Case No.</th>
<th>Loan Status</th>
<th>Monthly Income</th>
<th>Job Status</th>
<th>Repayment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Case 1</td>
<td>Good</td>
<td>$2000</td>
<td>Salaried</td>
<td>$200</td>
</tr>
<tr>
<td>Case 2</td>
<td>Very bad</td>
<td>$4000</td>
<td>Salaried</td>
<td>$600</td>
</tr>
<tr>
<td>Case 3</td>
<td>Very good</td>
<td>$3000</td>
<td>Waged</td>
<td>$300</td>
</tr>
<tr>
<td>Case 4</td>
<td>Bad</td>
<td>$1500</td>
<td>Salaried</td>
<td>$400</td>
</tr>
</tbody>
</table>

If an objective case were presented as shown in Table 4.2, to determine the loan status of the target case, the algorithm would traverse the decision tree and seek for the best matching case in the case-base. For the given the loan repayment, the algorithm first selects the left branch. After this, the algorithm traverses to the node (Income>$1500) and selects the left branching according the monthly income. We can therefore predict that the best matching case is Case 4. Nearest-Neighbor Retrieval vs. Inductive Retrieval
Nearest-neighbor retrieval and inductive retrieval are widely applied in CBR applications and tools. Table 4.3 shows strengths and weakness of two techniques. The choice between nearest-neighbor retrieval and inductive retrieval in CBR applications requires experience and experimentation. Usually, it is a good choice using nearest-neighbor retrieval without any pre indexing.

In some CBR tools, both techniques are used: inductive indexing is used to rescue a set of matching cases, and then nearest-neighbor is used to rank the cases in the set according to the similarity to the target case.
Table 4.3 Assessment between nearest-neighbor retrieval and inductive retrieval

<table>
<thead>
<tr>
<th>Retrieval Techniques</th>
<th>Strength</th>
<th>Weakness</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nearest Neighbor Retrieval</td>
<td>Simple</td>
<td>Slow retrieval speed when the case base is large</td>
</tr>
<tr>
<td>Inductive Retrieval</td>
<td>Fast retrieval speed</td>
<td>1. Depends on pre-indexing which is a time-consuming process</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2. Impossible to retrieval a case while case data is missing or unknown</td>
</tr>
</tbody>
</table>

4.11 APPLICATION OF CASE BASE

Case based reasoning first appeared in commercial tools in the early 1990's and since then has been used to create numerous applications in a wide range of domains:

**Diagnosis**: case-based diagnosis systems try to retrieve past cases whose symptom lists are similar in nature to that of the new case and suggest diagnoses based on the best matching retrieved cases. The majority of installed systems are of this type and there are many medical CBR diagnostic systems.

**Help Desk**: case-based diagnostic systems are used in the customer service area dealing with handling problems with a product or service.

**Assessment**: case-based systems are used to determine values for variables by comparing it to the known value of something similar. Assessment tasks are quite common in the finance and marketing domains.

**Decision support**: in decision making, when faced with a complex problem, people often look for analogous problems for possible solutions. CBR systems have been developed to support in this problem retrieval process (often at the level of document retrieval) to find
relevant similar problems. CBR is particularly good at querying structured, modular and non-homogeneous documents.

**Design:** Systems to support human designers in architectural and industrial design have been developed. These systems help the user in only one part of the design process, that of retrieving past cases, and would need to be combined with other forms of reasoning to support the full design process. There have been a number of successful CBR applications, but my particular favorites (aka Bob's top ten) are named in the above file (surprisingly) called "Bob's Top Ten." Let me know, if you would like to suggest another application to add to my list or displace one of my top ten. In summary, it can be seen that the application of CBR technique in the areas of SRM such as supplier evaluation/selection is a new approach, which can be used in integrating with CRM through the process of new product development and supply chain management. Since CBR is an advanced reasoning technique simulating human reasoning to retrieve a relative case, modify it and find a solution for the new coming problem, it can be used to supplement the conventional measures, which mainly rely on experts such as the purchasing manager or procurement engineer, to make decision on outsourcing matters.

### 4.12 COMPARING EXPERT SYSTEM AND CBR

Expert system generates new knowledge where as in Case Base Reasoning it searches for similar case and adapting these if necessary. In Expert system knowledge is stored implicitly while in CBR knowledge is stored explicitly. ES is hard to maintain as unpredictable implication by model change and extension. In CBR it is easier to maintain and update. however, there are a number of limitations with CBR applications.

1. When using past experiences to solve problems, it is quite difficult to find out whether the solutions to past experiences have been successful over time.
2. Also, with the case expanding through the addition of new cases it is possible that a lot of cases within the case base may become redundant.
3. Case adaptation can be a very complex process in attempting to derive modification rules.

4. 13 METHODOLOGIES FOR BUILDING CBR SYSTEMS

Full acceptance of CBR by industry depends on establishing software development methodologies for CBR, to define how to organize and develop CBR projects. Lessons from CBR applications form a foundation for defining such methodologies [9]. One fundamental principle revealed by many experiences is the value of an iterative development process. Since CBR systems can provide useful results even with a partial case library, systems can be set up with a set of seed cases that is augmented as gaps are revealed during use. Basically the process for developing a CBR application follows three phases:

**Case-base design:** a general representation for cases is developed using sources at hand (e.g., documentation, database records, and written accounts by experts). This is accomplished by a synchronized effort involving users, managers, and the developers of the system. A dictionary of terms used to describe problem features; the selection of appropriate features for case indexing and the specification of database schemas used to store cases are defined at this stage.

**Initial case-base development:** a small case base is initially developed to provide a base for the application. The initial case base is then extensively reviewed both by developers and users and is repeatedly refined until a valid case structure and case base covering a large portion of the application area is complete.

**Ongoing development and maintenance:** The case base is managed in a customary manner by a database administrator. Statistical quality control techniques may be used to monitor case accuracy and utility. Current research of maintenance of case library focuses on the RETAIN stage of CBR process which skip the storing of a case if the case is not covering any novel situation.