Part I Basics: Preliminaries

Part I introduces the basic elements of case-based reasoning (CBR) in such a way that no previous knowledge about the topic is required. The intended reader is anyone interested in learning about CBR. Part I provides a good basis for anyone who will use, design, develop, modify, or make decisions about CBR systems. The overall audience is very broad and includes, among others, engineers, computer scientists, librarians, medical researchers and entrepreneurs.

Chapter 1 provides a background of CBR and an introduction to the book.

Chapter 2 provides a sound description of the methodology. It starts by explaining that CBR is intended to reuse previous experiences. These experiences are the cases that are in principle pairs of the form problem-solution. The reasoning determines how to make use of the experiences. For a new problem one searches for an experience that has a problem which is closely related to the new problem. This reasoning takes place in a systematic way in the form of a process model. The knowledge needed to perform the processes of the model is stored in knowledge containers.

We will explain two essential points for CBR:

- How to make use of an experience even if it differs from the actual problem.
- How to find a useful experience.

Chapter 3 extends the initial description for further use of CBR. It employs similarity to relate problems to possible solutions directly without going back to experiences. This is frequently used in e-commerce where the problem is a demand that is directly related to a product. Fortunately, almost all techniques developed for using experiences apply here as well.

Chapter 4 gives examples that support and complement the understanding of the previous chapters. The intention is to demonstrate the broad scope of CBR applications to the reader. There is almost no attempt to show the specific solution approach. This can only be done on the basis of the material given in the subsequent parts. It is strongly recommended therefore that you frequently consult these applications when reading the remaining parts.

1 Introduction

1.1 About This Chapter

This chapter introduces this book on case-based reasoning (CBR); it is hence recommended to all readers. No previous knowledge about CBR is needed; only an interest in learning about it. It explains what CBR is and presents the main highlights in its history. This chapter aims to explain how to better use this book. The actual contents start in Chap. 2, Basic CBR Elements.

1.2 General Aspects

Case-based reasoning (CBR henceforth) has been a flourishing field for over three decades. It attracts researchers and entrepreneurs as well as practitioners. In fact, many practical applications have been commercially successful. There are several reasons for CBR being of interest; a few examples are:

- The origin of CBR as well as its relevance fall at the intersection of several disciplines of rather heterogeneous nature. They are mainly Cognitive Science; Computer Science and Computer Systems Analysis; Business; Library and Information Science; Engineering; and Education. Each of these disciplines has its own roots, its own methods, and its own foundations. This broadness opens the door for plentiful and diverse applications.
- The CBR methodology provides a computational model that is very close to human reasoning. CBR is rather intuitive; it is easy to understand. Consequently, when implemented, it utilizes a human paradigm in a computational context; benefiting from vast memory and speed provided by computers.
- CBR is able to deal with informal questions. An extensive and complex formalization of the problems is not required for CBR to be used.

Despite its intuitive plausible ideas, CBR can be technically complex. This book tries to bring you to a level at which CBR methods are understandable.

In order to understand the CBR methodology, we invite you to think of simple, repeating problems one encounters on a daily basis, problems such as those of finding a spot to park a car, preparing a meal or finding a restaurant, suggesting strategies to the coach during a game, and interpreting people’s actions or words. Now we invite you to think about how often you have simply thought of a previous time you faced the same problem and ended up simply adapting a previous solution. These solutions can be as simple as following someone on foot in a parking lot in order to find a spot or going to the same restaurant you went to the previous week but remembering this time to ask for the dressing on the side, or even understanding your spouse by thinking “last time she said no, she actually meant yes!”

Many people have no real impression of what CBR may be concerned with. One reason is that the term *case-based reasoning* is not immediately understood in conversations everyday and even many technologists may have a problem grasping how to use a case for the intended reasoning. A simple way to make this clear is to refer to the use of analogy in law, in particular, in Common law[1]. A case is an event, for instance, a criminal event that occurred in the past and where someone got, say, a penalty as depicted in Fig. 1.1. If now a new case occurs, one looks for a case from the past that seems to be similar to the new one. Then one looks at the decision of the past case and tries to imitate it as closely as possible. This imitation is usually not a direct copying because the new situation is not totally identical to the old one. Therefore, some modification or adaptation has to take place: CBR includes the identification of the new problem, finding a similar one, recognising their differences, and adapting the old solution to solve the new problem. In CBR, such considerations have been extended widely and have been made accessible to be manipulated and executed computationally.

Fig. 1.1 Importance of learning from the past

1.2.1 A Bit of the History

The history of CBR interleaves with the history of the development of models of memory. Cognitive Science is the field in which memory models are studied and categorized. Associated with CBR are schema-oriented memory models, which have a long tradition and go back at least to Bartlett (1932).

Building upon schema-oriented memory models, Schank proposed the dynamic memory theory, or MOPs model of dynamic memory (Schank 1982). The dynamic memory uses a unit of representation, the memory organisation packet (MOP) — a dynamic structure used to represent patterns of situations in memory.

MOPs are stereotypical situations that tend to repeat. A computer program can become able to interpret situations if it knows the basic elements of such situations and can adapt. Examples of stereotypical situations are activities such as going on a trip, eating at a restaurant, going to class, purchasing a new car, and so on. These situations are characterised by actors, events, goals, scenes, certain instance types, and abstraction types.

Schema-oriented memory models and, in particular, MOPs had quite an impact on CBR. The schema-oriented memory models characterise the first period in the history of CBR. Systems like CYRUS, CHEF, and MEDIATOR, among others (see Riesbeck

[1] Common law is the system widely used in English-speaking countries such as England and the US that is mostly based on precedents.

and Schank 1989 and Kolodner 1993), characterise this period. Many of the applications in this period are proofs of concept for a number of novel tasks for artificial intelligence (AI). They are also discussed in Appendix B, Relations and Comparisons with Other Techniques.

Rather than being systematic engineering products, these systems are based on the kinds of complex tasks they can achieve and the lines of research they can motivate. The first workshops where research on CBR was presented took place in the US in 1988 (Kolodner 1988), 1989 (Hammond 1989), and 1991 (Bareiss 1991).

In 1992, the first German workshop on CBR was organised, and it became a landmark of the beginning of the second period: An era of knowledge compiled into engineering methods that allowed the development of CBR systems much more systematically and reduced development times and costs by a large degree. The INRECA projects (see Bergmann 2001, 2002) were the main applications promoting this shift. This second period also marked a correspondingly stronger European influence. Europeans started to organize workshops periodically. Workshops in Germany and the UK were now held annually. In the US, multiple workshops were organised with the main American AI conference: AAAI. The community produced two influential publications: In 1993, Janet Kolodner published Case-Based Reasoning (Kolodner 1993), a thorough book describing all the work in the field done thus far; and in 1994, Agnar Aamodt and Enric Plaza published an article on CBR foundations where they introduced the CBR cycle (Aamodt and Plaza 1994). The CBR cycle is widely recognised as the CBR methodology process model. By 1995, an international conference on CBR had started to be held every two years. An example of an influential and representative application of this era is Cassiopee – a CBR system developed in the INRECA project for the diagnosis and troubleshooting of CFM56-3 engines in the Boeing 737 aircraft.

The greater attention attracted by an international conference influenced the community; the result was a third period during which CBR publications started to appear in multiple conferences and journals. By the end of the 1990s, maintenance was a new topic of interest; adaptation and retrieval continued to dominate. New areas starting to appear were conversational CBR, textual CBR, and knowledge management. A special track on CBR, first organised in 2001, became constant at the Florida Artificial Intelligence Research Society Conference: FLAIRS. Since 2002, the annual German workshop has been called the Workshop on Experience Management. This period is characterised by the use of CBR for more general methods and problems, which can be regarded as *experience mining*. This view of the CBR methodology extends the reuse of previous cases.

The third period started in the second half of the 1990s; by the mid-2000s the scenario had changed again. Recommender systems and diversity, which had become important topics in CBR, and data mining algorithms started to appear more often in CBR publications. This period may be seen as a transition to a stronger statistical influence, following the trends in AI in general. This period was well documented in a special issue called the Case-Based Reasoning Commentaries (Aha and Marling 2005). The issue included articles from the main research areas of CBR. This was the result of a community-wide effort that started in 2003 in a workshop in New Zealand with the

1.5 References

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

Aha DW, Marling C (eds) (2005) Special issue on case-based reasoning commentaries. Knowledge Engineering Review 20(3):201-328

Bareiss R (ed) (1991) Proceedings of a workshop on case-based reasoning. Washington, DC, 1991. Morgan Kaufmann, San Mateo, CA

Bartlett FC (1932) Remembering: a study in experimental and social psychology. Cambridge University Press, UK

Bergmann R (2002) Experience management: foundations, development methodology, and internet-based applications. Springer, New York

Bergmann R (2001) Highlights of the European INRECA projects. In: Aha DW, Watson ID (eds) ICCBR 2001: case-based reasoning research and development. 4th international conference on case-based reasoning, Vancouver, BC, Canada, July/August 2001. Lecture notes in computer science (lecture notes in artificial intelligence), vol 2080. Springer, Berlin, p 1

Greene F, Smyth B, Cunningham P (2008) An analysis of research themes in the CBR conference literature. In: Althoff K-D et al (eds) ECCBR 2008: advances in case-based reasoning. 9th European conference, Trier, Germany, September 2008. Lecture notes in computer science (lecture notes in artificial intelligence), vol 5239. Springer, Berlin, p 18

Hammond K (ed) (1989) Proceedings of a workshop on case-based reasoning. Pensacola Beach, FL, 1989. Morgan Kaufmann, San Mateo, CA

Kolodner JL (1993) Case-based reasoning. Morgan Kaufmann, San Mateo, CA

Kolodner JL (ed) (1988) Proceedings of a workshop on case-based reasoning. Clearwater Beach, FL, 1988. Morgan Kaufmann, San Mateo, CA

Richter MM (1995) The knowledge contained in similarity measures. Keynote at ICCBR-95: 1st international conference on case-based reasoning, Sesimbra, Portugal, October 1995

Riesbeck CK, Schank RC (1989) Inside case-based reasoning. Lawrence Erlbaum, Hillsdale, NJ

Schank RC (1982) Dynamic memory: a theory of reminding and learning in computers and people. Cambridge University Press, New York, NY

2 Basic CBR Elements

2.1 About This Chapter

This chapter introduces the basic concepts of case-based reasoning (CBR). It does not require any previous knowledge about the topic. The intention is that the reader should understand the principal ideas and follow the descriptions of the first examples, the remaining chapters in Part I, and Part II. This chapter provides the fundamental basis for understanding the remainder of the book. In most cases no formal definitions are used. In later chapters the concepts will be extended and put on a more formal basis, and they will be illustrated by examples.

2.2 General Aspects

Case-based reasoning (CBR) is a methodology for solving problems. These problems may be of a variety of natures. In principle, no problem type is excluded from being solved with the CBR methodology. The problem types range from exact sciences to mundane tasks. However, this does not mean that CBR is recommended for all problems. Throughout this book, we will give examples of problems and explain how CBR can be used. As a result of understanding these circumstances, the reader will develop a sense of when CBR is recommended.

Because CBR is essentially based on experiences, this chapter will discuss the main aspects of the CBR methodology and how it uses experiences in a specific way to solve problems. It is inherent in using and reusing experiences that they embed answers to problems or ways to get solutions. These answers can, for instance, help to solve difficult combinatorial problems, as an add-on they can also suggest or improve solutions where uncertainty is involved.

2.3 Case-Based Reasoning

The term case-based reasoning consists of three words and they need a short explanation. A **case** is basically an experience of a solved problem. This can be represented in many different ways. A case base is a collection of such cases. The term **based** means that the reasoning is based on cases, that is, cases are the first source for reasoning. The term most characteristic of the approach is **reasoning**. It means that the approach is intended to draw conclusions using cases, given a problem to be solved.

The kind of reasoning is, however, quite different from reasoning in databases and logic. The most important characteristic that distinguishes case-based reasoning from other kinds of reasoning is that it does not lead from true assumptions to true conclusions. This means that even if the solution in a recorded case were correct for its original problem, this may not be the case for a new problem. This possibility is based on the general fact that the situation in the recorded experience may not be exactly the same as that in the new problem. In fact, to be reused, it only has to be "similar". Therefore, the result of making use (or reuse) of the experience may only be "close" to the correct solution of the new problem. This means that applying CBR is a kind of approximate reasoning. Consequently, in order to more precisely describe its nature, we will investigate the concepts of being *similar* and *close* in more detail. In fact, CBR is essentially centred on these terms and most parts of this book are meant to describe this form of reasoning.

2.4 Experiences and Cases

Experiences are essential for CBR. In general, an experience is a recorded episode that occurred in the past, such as "Remember, last year in Italy we had a similar problem with our car. The hint the mechanics gave us worked pretty well. We had this problem quite often; and our usual way for fixing the problem always worked somehow". Such experiences are used to help solve future problems or make future decisions. However, not every recorded episode will be useful in this respect. For this reason, we consider two recorded experiences of the same event, in Table 2.1.

Table 2.1 Two recorded experiences of the same event

Experience Report 1.	Experience Report 2.
April 10 10.45: Problem reported 11.00 Maintenance arrived 11.50 Expert from group C was called 12.10 Expert arrived 13.15 Exchange part arrived 14.15 Part was built in 14.30 Machine is running	April 10 Morning: Strange loud noise, dust came out, speed of machine is slowing down. 10.45 Machine stopped 11.15 Test 35 and test 45 failed 12.15 Pressure was detected not working 12.25 Valve A was working improperly, after exchange the problem was solved.

Both experiences in Table 2.1 record what happened in the past. But the first one is almost useless for advice on fixing such problems in the future. It may, however, be used for administrative purposes. The second report offers some advice for a procedure that can be useful, although no detailed description is provided. This shows that past events can be viewed from many different perspectives.

To model a real-world situation there is no need to consider all aspects of a problem. The task is to identify those that are relevant and useful for solving the problem. In complex and unclear situations this may sometimes be difficult.

Suppose you go on a trip with your colleagues, driving your new used car, used but new to you – the cheapest you could afford. The car has been working fine, except that it will not start once the engine is warmed up. You are running out of gas and stopping at a gas station and waiting for the engine to cool down would delay your trip. As you share the problem with your colleagues, one of them is quickly reminded of a time when his cousin had a similar problem starting her car, so she would only park or stop the car on downhill slopes. Given that her car has a stick shift, she could simply let it ride to get it started. Because your car is also a stick shift, you can reuse the same solution, which is feasible because you are in a hilly area.

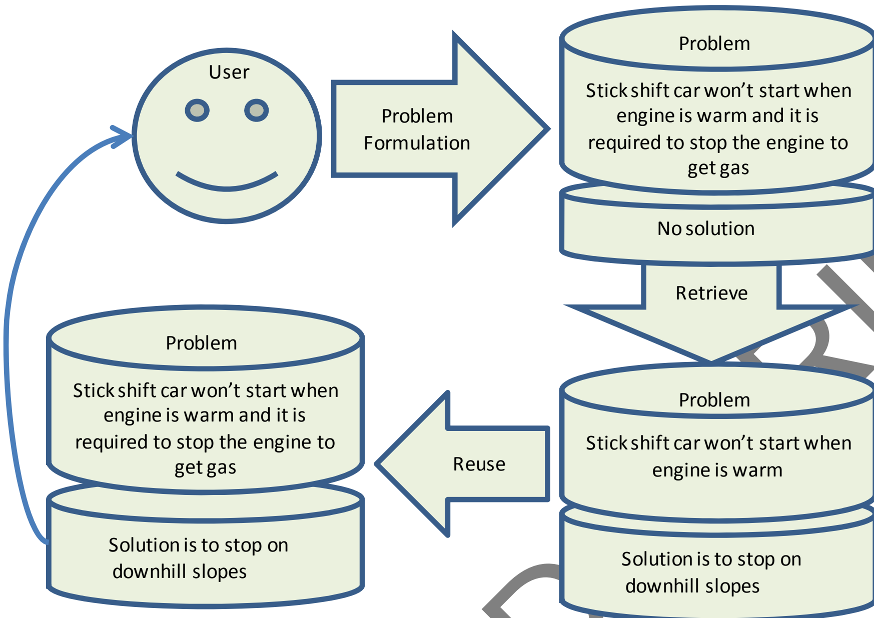

Fig. 2.1 Steps of the experience

In Fig. 2.1 the diagram breaks down the CBR methodology into steps, which helps explain how CBR uses experiences to solve problems. These steps are executed around the concepts of problems and solutions. Problems, solutions, and these steps will cover a major part of this chapter.

2.4.1 Parts of a Case

Cases can be quite complex and consist, as mentioned, of whole stories. CBR uses them for solving problems; therefore, there must be something in the experience that talks about a problem and its solution. In a simple view, CBR divides an experience into two parts:

- A problem part (or a description of a problem situation),
- A solution part that describes how one has reacted.

Often one restricts CBR to solutions that have been successful, but that is by no means necessary or adequate. A failed solution is also an important piece of information that states what one has to *avoid*. The coexistence of both successful and failed experiences leads to the following definition.

Definition 2.1

a) Positive experiences (cases) implement successful solutions and lead to the advice: Do it again!

b) Negative experiences (cases) implement failed solutions and lead to the advice: Avoid this!

When positive and negative cases occur one can introduce two sets of cases: C^+ (positive) and C^- (negative) cases. Negative cases occur often in the context of decision making where one has to choose from different alternatives or when advice has to be given. Negative cases have to be distinguished from cases that contain errors.

Major types of experiences occur in:

a) Classification: Decide the class to which an object belongs. For instance, classify mushrooms into the two classes “edible” and “poisonous”.
b) Diagnosis: Decide what the diagnosis of a problem is. For instance, determine whether what causes a car to malfunction is lack of gas (see also the example given below).
c) Prediction: Decide what happens tomorrow. For instance, for predict expenses for a firm for a given month in a given year.
d) Planning: Decide on a sequence of actions to reach a given goal. For instance, make travel plans.
e) Configuration: Decide which elements to include. For instance, decide how to select technical features and components of equipment.

In Chap. 4, Application Examples, there are several problems in these categories given and it is shown how they can be approached by CBR. Chapter 4 defines and discusses the different forms of reasoning that are embedded in different experiences. Because experiences can perform different reasoning tasks, it is important that a CBR system be uniquely designed to tailor each type of experience. Consequently, one considers one type of problem at a time, that is, one reasoning task. A CBR system is typically designed to perform one reasoning task. These systems offer an extended view of CBR.

Recall we mentioned that, to be reused, a recorded experience needs only to be similar to the new problem. This form of approximate reasoning generates an additional, though optional, component in cases. This third component is usually regarded as case outcome. Case outcomes do not retain knowledge about the experience itself, but they can be seen as a place to record meta-experiences, that is, information about uses of an experience. For example, how often a case is used, how successful it has been, and so on.

While humans can understand accounts of experiences told in everyday language, computers require some formality. Although natural to humans, the recognition of similarity and the consequent ability to reuse experiences requires an analogy when using a computer. This is a formal system that is intended to represent experiences so they can be reused.

Sometimes experiences are not given in such a suitable, formal way because they may rely on experiences that are informally described, for instance, in a textual form. Part IV is dedicated to situations such as when experiences are available in textual, visual, or conversational forms.

2.4.2 Problems

Problems are central to CBR because the main purpose of the methodology is problem solving. The formulation of a problem is sometimes difficult because it refers to the context in which it is stated. So, each problem formulation requires a different kind of solution. For example:

What is the price of this car?

- One answer could be: Too expensive for us.
- Another answer could be: $252,600.

It is obvious that one has to know the context in which the problem is stated in order to find out which answer is appropriate. In other words, for a precise statement the context has to be included in the problem formulation.

Part of the context is often the inherited culture. Consider for instance the Roman and the Anglo-Saxon laws. In the Roman law there are rules that say that in such and such a situation the decision is in favour of the defendant. In the Anglo-Saxon law the decision is traditionally based on the relationship (i.e., analogy) between an event that occurred in the past and the actual event. This latter kind of decision making is what CBR applies.

Another cultural point is what is considered as important in planning. For instance, what counts more, building a street or a school? Depending on the culture, laws may be different in different areas. Other cultures are provided by different sciences such as medicine, business and engineering; even large companies have developed their own culture. The CBR context has to take this into account because transferring solutions across cultures is problematic. For example, each bank has developed its own policy for giving loans to customers. The same bank may interpret the policy differently in each different country it operates; this becomes apparent during financial crises.

There are two types of problems we discuss in the context of the CBR methodology. The problems in the cases recorded as experiences are usually referred to as problems in CBR. The cases in the case base can sometimes be distinguished as candidate cases, as they are candidates for reuse. However, the entire CBR process is triggered by a problem. This is the new problem, or the actual problem that motivates a user to find a problem-solving method. To make this distinct from other uses, we henceforth refer to this as the query problem or, simply, the problem.

This section introduces problems and problem types, where the latter are more general. Next, we distinguish between a problem and a solution. These simple and intuitive notions are intended to eventually have formal definitions. Alternatively, we will use the term query instead of problem, and answer instead of solution.

2.4.3 Solution Types

The possible ways of representing a solution vary:

- It can be just a solution in the narrow sense.
- It can contain in addition:
    - Comments, illustrations, explanations.
    - Advice how to use the solution.
    - The effect by describing what occurred with the solution in the past.
    - Remarks on the strategy with which the solution was obtained.

In simple cases the solution contains a name or simple data, for instance, an object or an expected temperature. It may also be a project with values given to predefined attributes, such as jogging three times a week for 45 minutes. Solutions may also have a complex object-oriented structure as a technical object. Even more complex are solutions for planning and those in textual or image form.

In a complex situation the solution is a decision for performing an action or even a process. Here one has to distinguish the decision from the action; the action refers to an implementation and run of a strategy that may change states of variables. While the decision is usually clearly formulated, the outcome of the action may be uncertain. Suppose, for instance, that we have the choice between the different lotteries $L_1, \ldots, L_n$ and we want to choose a lottery that has maximal expected win. Then our solution can only present us a certain lottery; the win is represented as a probability distribution. Hence the computed probability has to be mentioned in the solution description. Another example is if we decide to fly to Toronto. The execution may fail or be postponed because of various unforeseen events. The latter means that the result of using a solution is uncertain because of unexpected external results like bad weather or an earthquake. If these are likely to happen one should extend the solution by an entry "effect" for describing what really happened. The user who sees the solution does not know this. If it is added then the user may get a hint for some possible adaptation. Finally, there are situations where the usefulness of the solutions can only be judged if they are executed in reality. This is the case with decisions for organising city traffic, or, more generally, with making predictions.

2.5 Case Representations

Now we know that cases are experiences and that such experiences have a context. We also know that cases include problems and solutions. The next step in introducing the CBR methodology is to explain how a case is explicitly represented and how cases are organised. Note that most of the formalisms used are not new. In fact, they can be considered quite common and they are used in many other problem-solving methodologies.

2.5.1 How Cases Are Represented

The simplest way to represent a case is by using feature-value pairs. A feature value pair is used to represent a state of an entity, for example, colour of an entity, "Jessica's car is red", where the feature is the colour of the car and the value is red, and the entity is Jessica's car. Instead of the word "feature" the word attribute is often used, and we will freely switch between these.

Features need to be identified for both problem and solution. Suppose someone has a headache and needs a diagnosis indicating what problem may be causing the headache. In Table 2.3, we find several cases for this.

A set of features has to be selected to represent cases. Each patient is represented in a case. Table 2.2 depicts one case with feature-value pairs for problem and solution.

Cases have to be described in some language. In principle, such a language can have an arbitrary character. This is only a preliminary view; more details and variations are given in Chap. 5, Case Representations. Feature value representations are, in fact, just an attribute-value vector.

Definition 2.2

(i) For a given set U of objects, an attribute A assigns to each object $O \in U$ some value taken from a set dom (A), the domain of A.

(ii) An attribute-value description is a finite vector of attributes.

This means the represented object is just an attribute-value vector. The problems and the solutions are described in this way. It is, however, a very simple definition of a concept; it will be extended later in various ways; see Chap. 5, Case Representations.

The need for more complex representations originates from the fact that such a representation cannot entail everything we can see and that is of interest. Consider the example in which we want to describe a car failure in order to represent it as a case. A case description limits the scope from the potentially infinite properties of a car to only a small part. Out of the thousand parts a car may have, many are irrelevant to most problems. This leads to the question: Which attributes should one take? The question cannot be answered universally, not even for cars. The point is that the chosen attributes should be relevant for the problem type in question. Our early example was for diagnostic purposes, typical when a fault occurs. In order to sell a car however different attributes would be relevant; as we see in Table 2.2.

A case would be a description of the car problem together with a description of the solution. Here, a problem is just a case without a solution, in this example unusual car noises. Table 2.2 presents the example through its attributes. One must realise that there are numerous car failures that refer to many different aspects of a car. However, within a certain type of failure the diversity is rather restricted. Hence, one has either to know what the possibly relevant attributes are or where one can find them. Table 2.2 shows some attributes.

Table 2.2 Attributes of the case

Attributes and their values	
Symptom	Unusual car noise
Observations	Knocking engine
Since last inspection (month)	3
Rhythmic pounding	No
Related to car speed	No
Oil pressure light flickering	Yes
Leaking oil	No

In order to make use of experiences for solving the problem of finding a diagnosis and a repair we need a collection of many experiences to choose from. These will be a collection (or set) of cases. As a general advice, we can look at humans describing a failure. That means we did not invent something new here. The CBR goal is just to support the usage of those experiences. Even if all attributes used are of interest, it is not guaranteed that they all have the same importance. This will be considered later when similarity is discussed.

2.6 Case Bases

A case base is a memory; it contains a collection of cases that is used in the context of the CBR methodology for the purpose of performing a reasoning task.

Definition 2.3

A *case base* is a collection of cases.

A case base is a data source and usually it is finite. What is *specific to CBR* is how a case base is used. In Appendix B, Relations and Comparisons with Other Techniques, we contrast case bases with databases. The usage for CBR requires special ways of utilizing the case base. The word “memory” is heavily used in cognitive sciences too; this will also be discussed in Appendix B.

2.6.1 How Are Cases Organised?

We have three main types of case organisation: flat, structured, and unstructured (e.g., text, images). Figure 2.2 illustrates the three basic types of case organisation. Note that these forms already suggest different programming paradigms, but we will only get into the programming aspects in Chap. 5, Case Representations.

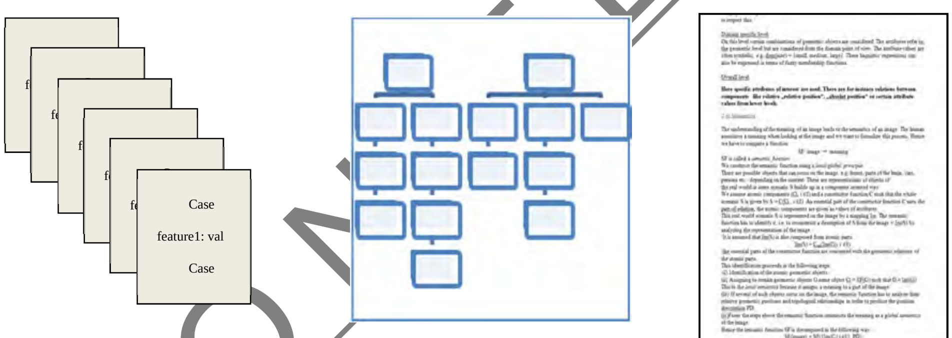

Fig. 2.2 Three types of case organisation: flat, structured, unstructured text

2.6.1.1 Flat Organisation

The flat organisation is the simplest to design and implement, and most suitable for a small number of cases. Table 2.3 shows an example of cases in a flat organisation. Attributes are listed in Table 2.3 in the leftmost column. They are used for representing case problem and solution. The six cases are represented through different values. There are no relationships between the cases. No one case has any relationship to another that needs to be represented, which means that the representation is complete. This is an example of a case base, that is, a collection of six cases that can be used in the context of the CBR methodology to diagnose the potential source of a headache.

2.6.1.2 Structured and Unstructured Organisation

Cases can be organised into structures such as hierarchies and networks. To be structured, however, cases do not necessarily require a hierarchical organisation. The relationship between two cases will have specific characteristics. An object-oriented organisation is structured. The structured organisations can be beneficial when the number of cases is very large.

Table 2.3 Six diagnoses cases

Case id Attributes	Case 1	Case 2	Case 3	Case 4	Case 5	Case 6
Nausea	Yes	Yes	Yes	No	No	Yes
Fever	Yes	No	No	No	No	No
Malaise	Dizzy	Dizzy	Dizzy	No	Listless	Listless
Blood pressure	Normal	Normal to low	High	Normal to high	Normal	Normal to high
Vision changes	No	Yes	No	No	No	No
Shortness of breath	No	No	Yes	No	No	No
Patient name	Bart	Marge	Lisa	Homer	Maggie	Ned Flanders
Diagnosis	Influenza	Migraine	Heart problem	Stress	Vitamin deficiency	Hangover

Cases are very commonly hidden within texts or images. At this point we will not yet discuss other concerns pertinent to unstructured organisations. We have entire chapters dedicated to them, namely, Chaps. 17, 18, and 19.

2.7 Similarity and Retrieval

The purpose of retrieval is to retrieve the case from the case base (i.e., a candidate case) that is so similar to a given new problem that their solutions can be swapped. One of the implications of this concept of similarity within the CBR methodology is that CBR's similarity is not a general concept, but a polymorphic concept that varies for each case base. One can, for instance, use the same case base with different measures for different purposes.

The purpose of any problem-solving method is to obtain a good solution, ideally even the best. The meaning of this is given by the user. Now that we understand that the CBR methodology uses a memory (i.e., case base) of experiences represented as cases, the next step is to understand how to select the experience, that is, case, and to reuse it properly. The question that needs to be answered is this, "What case in this memory has the most suitable solution I can reuse to solve my new problem"? The way it is answered in CBR is by relating the problem and the problems in the cases that make up the case base in such a way that the notion of "most suitable" is reflected. This relation was earlier referred to as *similarity*. The user will identify a problem in the base as very similar to the query problem if its solution is very useful.

Now suppose we have is a query problem and a case base to choose experiences from. There are many experiences in the case base but we do not know which one to take. The main difficulty we face is that the query problem may not be recorded in the case base because one cannot store all possible situations. Therefore, CBR has developed intelligent techniques to take advantage of the experiences even if they do not exactly match the query problem.

In order to illustrate how this is done we extend the example of car faults (Table 2.2) and look at a query problem and a small case base, containing just two cases, shown in Table 2.4. What we do is to compare the query problem with the problems of the stored cases. This comparison is a crucial step and known as *similarity assessment*. The goal is to find a case that helps in solving the problem. In other words, the case should be useful for this purpose. The reason is that a case is useful if its problem description is close to the query problem. Similarity is just a word for this. The goal is the cases are analogous in such a way that their solutions can be reciprocally reused.

Table 2.4 Attributes in query problem, Case 1 and Case 2

Query problem	Attributes labels	Case 1	Case 2
Unusual car noise	Symptom	Unusual car noise	Unusual car noise
Knocking engine	Observations	Knocking engine	Knocking engine
3	Since last inspection (month)	4	14
No	Rhythmic pounding	Sometimes	No
No	Related to car speed	No	No
Yes	Oil pressure light flickering	No	Yes
No	Leaking oil	No	Rarely
What is to be determined	Solution	Loose transmission torque converter	Oil burning

Assessing similarity between two cases represented with attribute-value pairs entails two concepts.

1) Similarity between attributes.
2) Relative relevance of each attribute.

When we compare two such cases it is natural to compare them attribute by attribute. This is the best way particularly when cases are represented through attributes. For this a similarity notion between attributes is needed. Each attribute requires its own similarity function. As a general rule, a similarity value 1 is given when the values of two attributes are the same; and a similarity value of 0 is given when the values are not the same. Along these lines, for two values that are different from each other but that may be considered medially similar, one can use a value of 0.5. This can, of course, become much more complex. These issues are further discussed in Chaps. 6 and 7, both about similarity.

In the example, such a similarity function would define for attribute “Oil pressure light flickering” a value of 1if both cases we have the same value for this attribute, and 0 otherwise. The attribute “Leaking oil” could have a similarity function return a value of 1 if in case both cases we have the same value for this attribute; a value of 0 if one case has a value of Yes and the other has a value of No; a value of 0.9 if one case has a value of Rarely and the other case has a value of No; and a value of 0.1 otherwise. These numbers resulting from the similarity function denote the degree of similarity.

The second concept within similarity assessment is the relative relevance of each attribute. In practice, each attribute is not equally relevant, and this has to be represented in the similarity assessment. In addition, the problem of describing the importance of the attributes is denoted by a number too. Larger numbers denote a greater importance because more important attributes play a larger role. The comparison is described in Table 2.5 and will later be described in more detail.

Table 2.5 Comparisons between query problem and cases

Attributes labels	Sim (Query problem, Case 1)	Sim (Query problem, Case 2)	Importance of attributes
Observations	1	1	8
Since last inspection (months)	0.9	0.2	2
Rhythmic pounding	0.6	1	7
Related to car speed	1	1	2
Oil pressure light flickering (binary)	0	1	8
Leaking oil	1	0.9	3

The comparison uses two parameters based on the two concepts already introduced:

a) Similarities between the values of the attributes between the two cars. These similarities are called local similarities.
b) The importance of the attributes. This is expressed in terms of integers where larger means more important. The numbers denoting the importance are called weights.

Presently, we assume that both parameters are given. Later we will better explain their meaning and potential sources.

Next, we want to compute the overall similarities of the query problem to the two cases. A simple and plausible way to do this is by taking weighted sums of local similarities with the weights as coefficients. If sim denotes the intended similarity, we get:

$$\mathrm{sim}(act, Case1) = \frac{1}{30} \cdot (1 \cdot 8 + 0.9 \cdot 2 + 0.6 \cdot 7 + 1 \cdot 2 + 0 \cdot 8 + 1 \cdot 3) = 0.633$$

$$\mathrm{sim}(act, Case2) = \frac{1}{30} \cdot (1 \cdot 8 + 0.2 \cdot 2 + 1 \cdot 7 + 1 \cdot 2 + 1 \cdot 8 + 0.9 \cdot 3) = 0.936$$

Therefore, we choose to reuse the solution from Case 2, which is oil burning. The reason is that Case 2 is more similar to the problem than Case 1 and therefore, according to our motivation, more useful. This can be formulated as a general principle.

Definition 2.4

Be *CB* a set of objects and *p* be an object; then some *s* of *CB* is a nearest neighbour to *p* if there is no object in the *CB* that has a higher similarity to *p* than *s*.

The principle is that no case is more useful than a nearest neighbour. The advantage of having degrees of similarity is that we can compare them and have a way to determine which experience is closer to the new problem. In particular, it allows us to say which recorded experience is the most similar one. This brings up the usefulness of the concept of nearest neighbours.

For an investigation of the cars example we had to define an adequate similarity measure. For our method, this looks as follows for objects with the attribute-value vectors of an arbitrary domain: $x = (x_1,..., x_n)$ and $y = (y_1,..., y_n)$:

Attributes: Attribute$_1$, Attribute$_2$,...., Attribute$_n$

Similarity between values of attributes (local similarities):

$sim_1(x_1,y_1)$, $sim_2(x_2,y_2)$,...,$sim_n(x_n,y_n)$; $x_i.y_i$ ε dom(Attribute$_i$).

Overall (global) similarity:

$$\sum_{i=1}^{i=n} \langle \omega_i \mathrm{sim}(x_i, y_i) | 1 \leq i \leq n \rangle$$

In the example, we did not discuss the origin of two numbers:

1) The similarity functions for each attribute.
2) The weights that should reflect the importance of each attribute.

The (local) attribute similarities are easier to get because they deal only with the domain of a single attribute and are therefore easier to estimate adequately. The weights, however, are of global character because they relate attributes to each other and are therefore much more difficult to determine. Intuitively importance means that an important attribute has a large influence on the choice of which case is the nearest neighbour for a query.

The case base in the example is very small but sufficient for illustrating these main concepts. We see that none of the cases has exactly the same problem as our query problem but the provided solution is still useful. There are two aspects that are over-simplified in the example:

1) We have only two cases and the decision between them is easy. If we have hundreds or thousands of cases, more sophisticated techniques need to be used.
2) Although the reused case problem is close to the query problem, it is not exactly the same. In the example we were lucky because the old situation could be the used unchanged. But suppose we had in the case a problem with front lights and in the actual situation exactly the same problem with back lights. Then it makes no sense to operate on the front lights. The advice is rather to do the same repair on the back.

The latter refers to an adaptation of the solution provided by the nearest neighbour. In general, *adaptation* takes place when one wants to reuse a solution with some modification.

2.8 Reuse and Adaptation

The use of cases is a reuse of previous experiences in a new situation. If the new problem situation is exactly like the previous one (which is supposed to have been successful) then the reuse is simple: Just copy the old solution. The general reuse principle for a selected case is shown in Fig. 2.3.

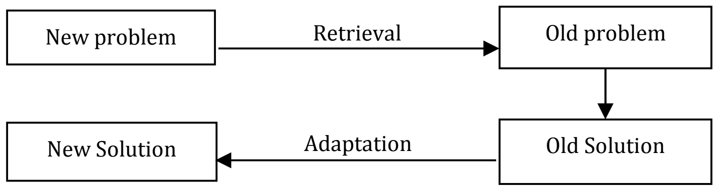

Fig. 2.3 Reuse principle

It is rare be able to use a solution exactly as it is recorded. This happens if the new problem situation is not too different in essential aspects from the nearest neighbour selected from the case base. Then the recommendation is to adapt the recorded solution before reusing it to best suit the new problem. This can be done either manually or automatically. CBR presents formal adaptation methods, which we will introduce next.

Adaptation can be performed on different levels of granularity. One extreme case is reusing the solution strategy. An example is reusing a travel plan. Another extreme case is using the solution itself. Both are called solution adaptation.

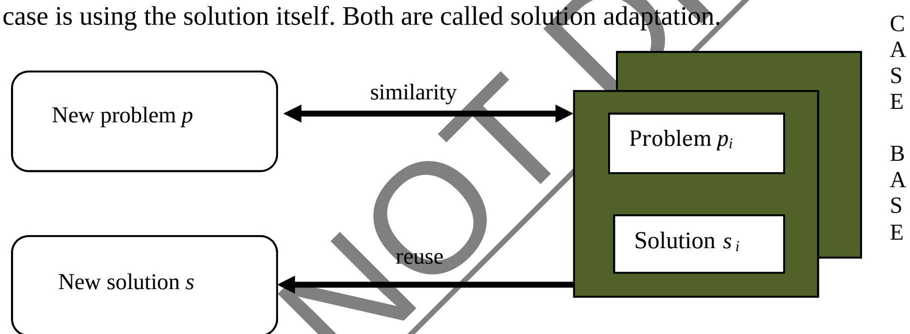

Fig. 2.4 Abstract CBR procedure

Suppose we have to design exercise plans for people who need to increase their endurance. The simplest way would be to create a weekly plan for running. Now suppose there is a person who is not allowed to run because of knee problems. The previous plan can still be used but running has to be replaced by swimming or bicycling. The elements of the procedure are shown in Fig. 2.4.

In an abstract way we can describe the CBR problem-solving procedure by the following steps:

a) First describe the problem formally.
b) Search in the case base for the nearest neighbour and select it.
c) Make use of the retrieved solution by copying or adapting it appropriately.

The approximate nature of case-based reasoning has the consequence that there is no guarantee that the chosen case provides a good solution. For instance, the case base may not even contain a good solution for the new problem. Sometimes this can be easily seen, as in symmetric problems. Take for instance an experience of a car problem with the solution "exchange the left bulb" when we have the same problem with the right bulb. It is not necessary to record this problem because we can simply adapt the presented solution. There are other situations where this is not so easy and a systematic evaluation is needed. This will be discussed in Chap. 9, Adaptation.

After adaptation, the adapted solution has to be tested in reality and possibly modified further. If the solution obtained in this way is satisfactory, then one may decide to add the case (new problem, final solution) to the case base in order to improve it. This last step can be interpreted as a learning step. More learning methods will be discussed in Chap. 10, Evaluation, Revision, and Learning. Adaptation allows case bases to be smaller than if no adaptation could be done. Furthermore, adaptation can be also extended by reusing a strategy when the solution is given because strategies can also be adapted. These methods are found in game playing. In chess, for instance, strategies are what is reused most often (but hard to formulate!)

If we solve a problem using experiences, there are many ways of doing it. For systematic reasons it is desirable to have a general process model for CBR problem solving. This will be discussed next.

2.9 Models of CBR

We now combine the understanding gained in the previous sections into two views on CBR. The first view considers the processes that take place when CBR is applied, that is, problem formulation, retrieve, reuse, revise, and retain. The second view considers the knowledge organisation in CBR. CBR systems store their knowledge in knowledge containers. The next two sections will describe these views in more detail.

2.9.1 CBR Process Model

Figure 2.5 introduces the first view we present of CBR, i.e., the main tasks the CBR methodology implements. In a more abstract way they will now be extended to a general process model that applies to the entire CBR methodology.

Figure 2.5 lays out the main tasks of this process. Although we have already described the main concepts of the process, in this section, we present these tasks in the context of the process model.

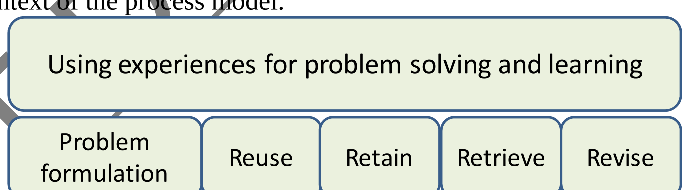

Fig. 2.5 Tasks in the CBR process

2.9.1.1 Problem Formulation

Problem formulation is a task that starts from the need to obtain a new problem from a user. Ideally, users should enter the new problem using the same representation and level of detail as those of the cases in the case base. Often, this is not the case. As an example, it may happen that the user knows what to achieve but cannot express a precise problem. Consider a user who wishes to find a “comfortable chair” for a living room. The problem formulation would need a description of chair parts and their properties that may not be available. Therefore, one cannot immediately describe such problems as cases.

This can be done in different ways. This is also known as the query generation problem. A somewhat oversimplified view is that the problem is stated exactly and complete with all details. In fact, it can be costly to acquire in an attribute-value representation the values of the attributes for the query problem.

An essential point therefore is to acquire as little information as possible for solving the query problem but enough to provide an answer. There are two major ways to proceed:

1) Use a specific, possibly standardized formulation of the problem.
2) Perform a dialogue with the user. This is discussed in Chap. 20, Conversational CBR.

After the content of the new problem is obtained, there are still different ways to formulate it physically. It can be typed into a computer; it can be spoken, or it can be represented as an image or diagram. These variations will be discussed in later chapters.

2.9.1.2 Retrieve

As previously mentioned, the goal of Retrieval is to determine the case that is most similar (i.e., most useful) to the new problem. Retrieval starts when the new problem is readily available and completes when a case is retrieved, becoming available for the next task of the process: reuse.

For purposes of simplification, we assume that only one case is retrieved. Variations are, of course, possible. They are further discussed in the chapters dedicated to retrieval, Chaps. 8 and 14.

Retrieval is comparable with a search, where the new problem is used for guidance and the case base is the search space. Retrieval is a demand and this demand has to be formulated. In order to formulate it one needs a set of search paths to select a successful one. The description of the paths is called an index structure. These indices are basic for the search. Depending on the search structure there are many indexing methods, for instance:

- For searching a book, the index is a page.
- Search for data entry uses a pointer to a record.
- Searching for a record in a database is a pointer to a record, realised by a key.

As previously mentioned, retrieval methods are not general; they have to be designed for each system. This is because of the complexity of cases and the inexact matching that CBR implements.

2.9.1.3 Reuse

Reuse is the step of the process when one case is selected for its solution to be reused. It is completed when the new solution is proposed for the next task of the process: revision. Reuse is about proposing a solution for solving the new problem by reusing information and knowledge in the retrieved case(s).

Reuse is quite simple when the new problem is identical to the retrieved case problem. When they differ, they require adaptation. This is a general theme; details are in Chap. 9, Adaptation.

2.9.1.4 Revise

Revise starts when a solution is proposed to solve the new problem, and it is completed when it is confirmed. Revise aims to evaluate the applicability of the proposed solution. Evaluations can be done in the real world or in a simulation. Simulation is easier and cheaper but may neglect practically important aspects. In the real world, evaluation aspects may be present that one might not have considered in the model. In fact, this is an old phenomenon in Artificial Intelligence called the frame problem. It says that one can never completely formulate all possible facts that may occur in the real world.

2.9.1.5 Retain

When revising generates a new case, updating the case base with the new (learned) case for future problem solving takes place. Nevertheless, a confirmed solution may or may not be retained. Some systems learn new solutions adapted through use; others accept only actual cases. Revise and retain are discussed in Chap. 10, Evaluation, Revision, and Learning.

This model is detailed and extended in various ways. The usefulness of having a process is that such improvements can smoothly be integrated. For instance, the learning aspect is much more complex; in the cycle, cases can only be added but not forgotten. This is connected with the maintenance issue that is discussed in Chap. 11, Development and Maintenance, together with the problem of developing a CBR system.

This view of CBR lists the main tasks the methodology entails. Another perspective on CBR is given by the knowledge containers it requires to be successfully implemented, which is discussed next.

2.9.2 CBR Knowledge Model

The knowledge container view of the CBR methodology is based on the perspective that CBR is a knowledge-based system. Knowledge-based systems are a class of intelligent systems that are designed by having a knowledge base in an independent module. In CBR, we extend this notion to emphasize how the methodology utilizes different kinds of knowledge in distinct repositories: the knowledge containers. While the tasks listed in the previous Sect. 2.9.1 look at CBR from the process point of view, one may also ask what kind of knowledge is represented and where it can be found. Knowledge can either be represented explicitly or be hidden in an algorithm. In any case, there must be some way to formulate the knowledge; we say that knowledge is presented in some formulation. The formulation is stored in what is called a knowledge container.

For the knowledge containers described next we state what kind of knowledge could be contained in them. We say little about how the knowledge is formally represented. In CBR we identify four major knowledge containers. They are presented in Fig. 2.6.

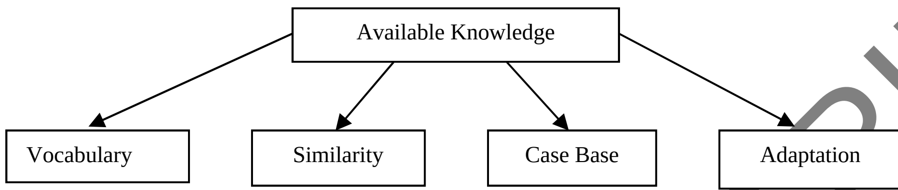

Fig. 2.6 Knowledge in CBR

The knowledge containers represent one view of a CBR system; they are not modules that can perform certain subtasks. They contain certain knowledge units that in combination help solve a problem. Next, we give a short overview of the containers that will be extended in the following chapters.

2.9.2.1 The Vocabulary Container

The vocabulary is basic for any knowledge-based system. This is not special to CBR. The vocabulary determines what one can discuss explicitly.

The vocabulary plays a role in all levels of abstraction, which is illustrated by very simple examples:

1. If we do not know the word heart rate we cannot talk about it. It is knowledge that this term plays a role.
2. If the term tax cost is missing one cannot compute the tax correctly. Again, this is knowledge. This aspect plays a major role in different countries, where different tax regulations are involved.

The vocabulary container retains knowledge about how to explicitly describe the knowledge elements being used. This does not depend on the types of descriptions, ranging from logical constructs to free text. It is a classical observation in science that the solutions of difficult problems have been found only after some person introduced a new crucial notion.

Therefore, there is usually much knowledge contained in the chosen vocabulary. For a real-world object there are in principle infinitely many terms that have something to do with the object but only a few are relevant for a specific task. That means an object can (and should) have different description terms for different tasks.

In the vocabulary container one can identify various sub-containers that are useful for technical purposes as retrieval, input or output. These are, for example, names of employees, companies, products in a supermarket, and so on. These sub-containers are frequently defined and used in application domains.

2.9.2.2 The Similarity Container

The knowledge in the similarity container consists of all knowledge needed to determine what makes a case similar to another such that their solutions can be reciprocally reused. There are multiple ways to ensure similarity knowledge accomplishes this: From the use of simple symbolic similarities where the values are either equal or not, through the use of weights to represent relative importance of the attributes, through the use of systems where relevance is computed at runtime, to the use of fuzzy algorithms that consider all attributes and their importance at once.

The similarity is used for retrieval purposes. This means that something has to be known about the problem and what is required for the solution. As an example we consider the task of squaring numbers and assume we are unable to multiply and do not want to learn how to do so. Suppose we have a base of solved problems, say

$$Squ = \{(2, 4), (2.5, 6.25), (-3, 9), (-5, 25), \ldots\}.$$

As a special problem we take square (3) = ? The answer is not in our list; therefore we have to look for the nearest neighbour of “3”. A first try is to take the Euclidean distance, which gives 2.5, and the answer 6.25. A much better method is to equip the similarity measure with the knowledge square(x) = square($-x$) for all x. Then we would retrieve −3 which gives the correct answer. The similarity measure is much easier to use than it is to learn multiplication. In Chaps. 6 and 7, similarity concepts are studied in detail.

For CBR and retrieval purposes it is important to quantify similarities. This is done by similarity measures, which can be defined as a mapping

$$\text{sim}: U \times U \to [0, 1]$$

where U contains the objects to be compared.

Not all aspects of a problem situation may be of equal importance. For example, the price of a car may be more important than the colour. If the similarity knows this then it would pay more attention to the price attribute than to the colour attribute. A way to make this possible is to assign weights to attributes. Earlier, we saw an example dealing with car repairs where the similarity measure was naively chosen but successful. It ranked the cases and we selected the most similar one because similarity tends to be an adequate proxy for utility.

2.9.2.3 The Case Base Container

The case base container contains experiences as cases. These experiences may be available from the past or may be constructed from variations of existing cases, or be completely artificial. The description of the case base as a knowledge container is straightforward as the case base is typically the main source of knowledge in CBR systems. The implications of the case base as a container of knowledge are discussed in multiple chapters. Representation formalisms are discussed in Chap. 5, Case Representation; quality and maintenance are discussed in Chap. 11, Development and Maintenance.

2.9.2.4 The Adaptation Container

The knowledge in the adaptation container will be used to adapt cases to solve new problems. The most common formalisms adopted for adaptation are rule bases; nevertheless, case bases can be used, and even existing cases from the case base have been used at runtime to extract adaptation knowledge. As previously described in Sect. 2.9.2.4, the knowledge in the adaptation container can be used to transform an existing solution or generate a new solution based on a strategy from a previous solution.

In the adaptation container one finds information on how to modify a solution. In the adaptation container rules are stored for adapting a retrieved solution to a new situation. Such rules are intended to perform a solution transformation that has to take care of the fact that the solutions obtained from the case base using the nearest neighbour principle may still be insufficient (either because of a not very well defined similarity measure or simply because the case base does not contain a better solution). In this situation the solution is adapted. Adaptation knowledge can drastically reduce the number of cases needed in the case base. More is shown in Chap. 9, Adaptation.

2.10 Tools

Tools can speed up design and assessment of an application. We list some tools that are currently available. However, given the dynamics of tools, we recommend that the reader rely on a more agile source, like the Cbrwiki (2011).

In addition, we mention some general-purpose tools, i.e., tools that can be used for building a general CBR system and using it for many applications. Using such systems one can avoid a lot of work, not least because of a graphical user interface with useful visualisation.

Some major examples follow.

1) CBRWorks (http://cbr-works.net) and Orenge (Schumacher 2002). CBRWorks is developed for e-commerce applications but can be used for other purposes also. It contains elements from all knowledge containers and can perform the full CBR cycle. Orenge is a further development and has a more powerful retrieval engine.
2) myCBR. It is open source, developed under the GPL license. It can be viewed as a successor of CBR Works and contains many useful features. See myCBR (http://mycbr-project.net), from where it can be downloaded; this also contains a tutorial.
3) jColibri. It is a general framework that supports many features like graphical interfaces, description logics and ontologies, textual CBR, evaluation, and so on (http://gaia.fdi.ucm.es/projects/jcolibri/). jColibri 2 (Recio-García et al. 2013) has added a number of features and is becoming more and more a reference tool for teaching and research purposes.
4) CBR in Microsoft® Excel. For users familiar with macros in Excel, a simple case retrieval system can be developed in it. A worksheet should be reserved for cases, with their attributes laid out vertically. A different worksheet is used for retrieval, where the new problem is compared to all

cases in the case base. Note that the retrieval worksheet will require one column of computation of similarity for each case. Weights can be listed in a separate sheet and called from the retrieval worksheet. Solutions can be presented in a separate sheet. Such implementation can be extended to include a validation method.

2.11 Chapter Summary

The chapter presents the basic notions used in CBR and necessary for understanding the remainder of this book.

Case-based reasoning is a reasoning methodology for problem solving. It mainly relies on experiences in which problems were solved in the past. CBR reuses previous experiences to solve current, new problems. Problem solving experiences include problems and solutions. Problems and solutions should be explicitly stated in order for the experiences to be successfully reused. CBR can be used to perform multiple reasoning tasks, such as classification, planning, and design. The way to develop a reliable CBR system is by limiting its scope to one single reasoning task. Such a system would be populated by cases that describe experiences of performing the single chosen reasoning task in a given target domain.

The simplest method to represent cases is to use attribute-value representations. With a limited and previously defined set of attributes, each case is populated with individual values for each attribute. This representation allows a case comparison at the level of attributes.

Cases are compared to search for a similar case. Problems are submitted to a CBR system through what we call query problems. Once a new query problem is formulated through the set of attributes defined for case representation, similar cases can be retrieved.

Case retrieval utilizes a similarity measure to search for similar cases whose solutions may be reused to solve the new query problem. How to assess similarity between cases is a core method in CBR.

The problem in the retrieved case is typically very similar to, but not exactly the same as the query problem. This may cause the solution in the retrieved case not to be perfectly suitable for solving the new query problem. Adaptation is the step that modifies the solution in the retrieved case in order to make it perfectly suitable for solving the query problem.

There are two models of CBR. The CBR process model incorporates formulating the problem, retrieving solutions, reusing them, revising and repairing them, and storing them as new experiences.

The CBR knowledge model describes the containers where knowledge is stored. There are four knowledge containers: Vocabulary, Case Base, Similarity, and Adaptation.

From reading this chapter, the reader has a deeper understanding of the CBR process. However, we recommend you do not yet jump into designing your own CBR system, not until after reading Chaps. 3 and 4; the technical details are presented in Part II.

2.12 Background Information

The first substantial publication on case-based reasoning is the 1993 book by Kolodner (1993). It introduces the main problem areas, thoroughly describing case representation, structure, indexing, retrieval, adaptation and learning.

CBR has roots outside of computer science, mainly in cognitive science, psychology, and language understanding. The first CBR systems were built within this context. The use of analogy for reusing previous events is discussed in Carbonell (1983). The many roots of CBR today are discussed in Richter and Aamodt (2005).

One of the most cited foundational articles is the 1994 article by Aamodt and Plaza (1994). The CBR cycle as presented by Aamodt and Plaza is a simple and complete way of visualising the CBR methodology as a whole. It introduces the CBR cycle and names the four R's in the cycle: *retrieve, reuse, revise*, and *retain*. An early cycle for modelling the CBR process, referred to as a CBR flowchart, is given in (Riesbeck and Schank 1989). The CBR cycle was extended in many ways to describe additional activities like maintenance and learning. Some of these extensions are described by Bridge (2005). More historical information is in Chap. 1, Introduction, and in Appendix B, Relations and Comparisons with Other Techniques.

The knowledge containers were introduced by Michael M. Richter; see, for instance, Richter (1998). We return to them in Chaps. 10 and 11 on learning and on development and maintenance. When systems are developed or improved, the contents of the containers are the objects of interest.

The example in Fig. 2.1 is based on a problem discussed on Car Talk from National Public Radio® on 19 February 2011.

2.13 Exercises

<u>Exercise 1</u>

Suppose you are in the automotive domain. Look at the three contexts manufacturing cars, marketing cars and repairing cars. Find for each context typical attributes that would not be used in the other contexts.

<u>Exercise 2</u>

Describe the purpose of shifting knowledge from the case base to

a) the similarity measure,
b) the adaptation container.

What is the influence on the size of the case base?

<u>Exercise 3</u>

Give an example where the retain step of the process model does not improve the performance of the CBR system.

<u>Exercise 4</u> (Intended for readers who understand databases)

Write a process cycle for databases. Can you identify some knowledge containers?

Exercise 5 (Intended for computer scientists)

Name some knowledge containers for other knowledge-based systems such as rule-based reasoning, fuzzy expert systems, and ontologies.

Exercise 6

Find useful sub-containers for adaptation.

Exercise 7

Propose an application domain where CBR can be used to provide solutions to problems. Consider what source of cases you would have.

Exercise 8

Describe characteristics that you would require for a problem to be solved with the CBR methodology.

Exercise 9

Describe an area of expertise that you master, e.g., playing a game. Describe how you would explain to someone what makes cases similar to others so that their solutions can be swapped with minimal adaptation.

Exercise 10

Elicit similarity knowledge from an expert (not you) in any domain in which you are not a master. In other words, elicit for the expert's domain of expertise what makes cases similar to others so that their solutions can be swapped with minimal adaptation.

Exercise 11

Name one AI methodology and an example problem you are familiar with and then list advantages and disadvantages you see when comparing it with CBR.

2.14 References

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

Bridge DG (2005) The virtue of reward: performance, reinforcement and discovery in case-based reasoning. Keynote at ICCBR 2005: 6th international conference on case-based reasoning, Chicago, IL, USA, August 2005. Awards, Honours, Affiliations. http://www.cs.ucc.ie/~dgb/recognition.html Accessed 28 Feb 2013

Carbonell JG (1983) Learning by analogy: formulating and generalizing plans from past experience. In: Michalski R, Carbonell JG, Mitchell T (eds) Machine learning: an artificial intelligence approach. Springer-Heidelberg, p 137–159

Cbrwiki (2011) Case-based reasoning Wiki. http://cbrwiki.fdi.ucm.es/wiki/index.php. Accessed 18 Jul 2011

Kolodner JL (1993) Case-based reasoning. Morgan Kaufmann, San Mateo, CA

Recio-García JA, González-Calero PA, Díaz-Agudo B (2013) jColibri2: A framework for building Case-based reasoning systems. Science of Computer Programming (in press)

Richter MM (1998) Introduction. In: Lenz M, Bartsch-Spörl B, Burkhard H-D et al (eds) Case-based reasoning technology: from foundations to applications. Lecture notes in artificial intelligence, vol 1400. Springer, Berlin, p 1

Richter MM, Aamodt A (2005) Case-based reasoning foundations. The Knowledge Engineering Review 20(3):1–4

Schumacher J (2002) Empolis Orenge — an open platform for knowledge management applications. In: Minor M, Staab S (eds) Experience management: sharing experiences about sharing the experience. Papers from the 1st German workshop on experience management, Berlin, March 7-8, 2002. GI, Bonn, p 61