Case Base Properties: A First Step*

Craig MacDonald, Rosina Weber, Michael Richter
Drexel iSchool
University of Kaiserslautern
{cmm353,rw37}@drexel.edu, richter@informatik.uni-kl.de

Abstract. In the context of case-based reasoning (CBR) methodology, better understanding of the conditions under which CBR systems are used has the potential to increase our understanding of how uncertainty affects the overall quality of these systems. Our ultimate goal is to determine what properties exist and how these properties may be used to improve the utility of a CBR system, as well as methods of maintenance and evaluation. With this purpose, we start by investigating how properties of case bases can be demonstrated and applied.

1 Introduction

All case-based reasoning (CBR) systems should be good for something. In fact, the success of a CBR system is ultimately determined by its ability to satisfy the user; specifically, in order for a CBR system to be useful it must be able to solve the problem it is intended to solve. Since CBR systems are designed to function under certain conditions, each condition has the potential to add uncertainty to the overall system and, ultimately, affect the ability of the system to solve the given problem. As a result, uncertainty plays a critical role in determining the overall quality of a CBR system. We hypothesize that there are certain properties of CBR systems that are always true under a given set of conditions. Understanding these properties will lead to a better understanding of how uncertainty impacts the overall quality of CBR systems and will help CBR researchers build more effective and more useful CBR systems.

The CBR methodology proposes solutions to new problems by reasoning through a series of steps described in the CBR cycle [1]. In general, a CBR system is designed such that the CBR methodology is used to help users solve a particular problem. A CBR system is composed of two distinct contexts: the user context and the system context. The user context is defined by the problem the CBR system is designed to address, and includes considerations about the domain and its users. The system context is defined by the actual CBR system itself, which can be discussed from the perspective of CBR knowledge bases, i.e., knowledge containers [2]. A CBR system has four knowledge containers, namely, the case base, the vocabulary, the similarity measure, and the solution transformation. This paper focuses on investigating properties of the case base container.

*Craig MacDonald, Rosina Weber, Michael Richter (2008). Case Base Properties: A First Step. Workshop Proceedings from the Ninth European Conference on Case-Based Reasoning Workshop Program, ECCBR 2008, Sep 1st-4th, Trier, Germany(http://www.wi2.uni-trier.de/eccbr08). Workshop on Uncertainty and Knowledge Discovery (http://www.mathematik.uni-marburg.de/~kebi/ws-eccbr-08/) Unpublished.

When evaluating a CBR system we often use the notion of utility. The utility of a CBR system can be defined by its ability to minimize the error of a classification or maximize the usefulness of a diagnosis [3]. When it comes to uncertainty, there is an obvious relationship between the user context and the system context of a CBR system. Specifically, uncertainty can be added to either the user context (from the query) or it can be added to the system context (from the answer). When uncertainty is added on the user side (e.g. if the user does not care or does not know what he or she needs), certainty can be added on the system side (e.g., by reducing the number of cases). When uncertainty is added to the system side (e.g. if there are too many or not enough cases), then it becomes more difficult to satisfy the user.

This notion of uncertainty in the system context is particularly important for maintenance and evaluation. In terms of evaluation, different methods of CBR system evaluation have been discussed. Some methods attempt to measure the ability of a CBR system to provide solutions with a certain confidence threshold [4] while others depend on analyzing whether and to what degree the system can support a number of predefined characteristics [5]. Knowledge about the properties of CBR systems will help designers and researchers develop a better understanding of how to evaluate a CBR system with varying levels of uncertainty embedded within it. There is also a concern that the contexts of a CBR system are usually unknown, are often vaguely defined, and are constantly changing. In order to handle these dynamic contexts, a case base must be maintained. Some researchers focus on maintaining the individual cases ([6], [7]), while others concentrate on maintaining the case base as a whole ([8], [9]). Some research has also shown that examining the competence contributions of individual cases during the initial case base creation stage can maximize problem coverage while minimizing the number of cases, thus reducing the amount of maintenance needed [10]. Further research is needed to determine how properties of CBR systems can be used to determine how much and which types of maintenance may be required to maintain or improve the problem-solving ability of a CBR system.

In the next section we will discuss the difference between properties of CBR systems and properties of problems that CBR is intended to solve. Here we make a distinction between the user context and the system context, and discuss the role of uncertainty in determining the overall quality of a case base. Section 3 will detail an experiment designed to demonstrate a case base property by investigating how the level of uncertainty in a case base is impacted by the relationship between three simple parameters of the case base. In Section 4 we discuss the implications of our work along with some ideas for future work.

2 CBR Properties

In mathematics, a property is often associated with a set of conditions; the property always applies under the stated conditions. For example, the commutative property of

mathematics says that, under the conditions of addition and multiplication, the order of the elements has no effect on the outcome. Similarly, we define a CBR property as any property that always applies under a given set of conditions, where the conditions are defined by the environment in which the CBR system is intended to be used. These properties describe how those conditions affect the ability of a CBR system to find a solution to a target problem. Some conditions can be further broken down into parameters which together comprise a condition under which a property will always be true. While this definition may not be as precise as it could be, we feel it is adequate for our current purpose.

The CBR methodology applies to a universe that combines the system and the user context [1]. Therefore, the study of properties must consider this entire universe. The user context cannot be modified since it is defined by the problem situation for which the CBR system is designed; hence, the system context must be modified to fit the user context if needed. As a result, we break down our analysis into these two segments. The major distinction we make is in terms of CBR systems and the problems that CBR systems are designed to solve (see Figure 1). An important step in this process is determining which conditions are useful in describing the quality and utility of a CBR system. In this paper we are only concerned with descriptive properties of CBR systems and not with inferential properties.

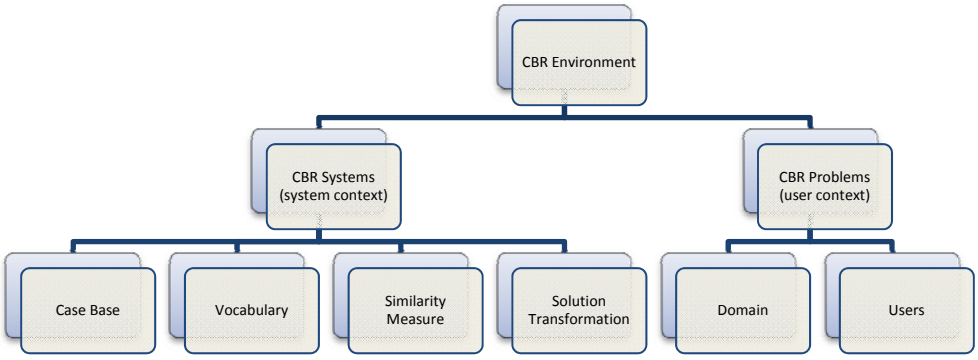

Figure 1. Taxonomy of CBR

2.1 Properties of CBR Problems

To investigate CBR properties, we start by looking at the conditions of a CBR system that are defined by the problems they are designed to address. By investigating these conditions we can get a better understanding of how the overall quality of a CBR system changes under certain conditions and whether we can make general inferences about a CBR system based on the conditions under which it is used. We are not interested in

specific problems and their specific solutions but in very general properties of problems that are commonly solved using CBR. We are also not interested in the knowledge needed for problem solving or in the knowledge contained in the case base (see Section 2.2); rather, we address the general properties of problems that impact the overall quality of a CBR system. Because CBR is a problem-solving method, we will investigate which properties enable a CBR system to solve a certain type of problem; that is, whether there is a set of problems that define the conditions under which certain properties are always true.

A central goal of a CBR system is to make clear when it is successful, i.e. the utility is achieved. This means, in our view, when the problem the user is interested in is solved to a satisfactory degree. This has user-independent and user-dependent aspects. The fact that a CBR system solves a problem correctly is user-independent. This does not, however, guarantee the success of a CBR system because it neglects the role of the user. This is described for example, by its ease of use, its treatment of explanation, and the presence of solutions. For example, if the user base consists of technical experts, the CBR system may have a product-oriented vocabulary that utilizes technical terms (such as "processing speed" and "hard disk capacity"). Conversely, if the user base is made up of novices, the system may need a functionality-oriented vocabulary and include explanations (such as "fast enough to play computer games" or "able to store thousands of pictures"). Naturally, these conditions can also be defined by cognitive and affective characteristics of individual users, but for now we only address abstract user conditions. We refer to these conditions as user-dependent conditions (see Figure 2).

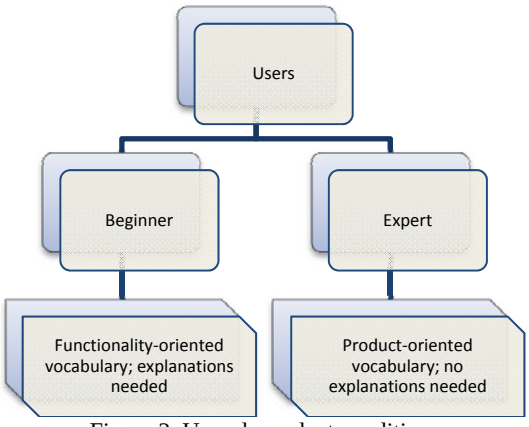

Figure 2. User-dependent conditions

Each user type gives rise to certain demands on the CBR system but they do not affect the system's ability to solve a problem correctly. Instead, we refer to user-independent conditions as the set of parameters that determines the ability of a CBR system to provide a correct and accurate solution to a given problem. We present a

preliminary taxonomy to demonstrate the hierarchy of these conditions under which a CBR property can be demonstrated (see Figure 3). We emphasize that this taxonomy is not meant to be exhaustive; further investigation will undoubtedly lead to revisions and elaborations. Each condition can potentially be broken into a set of parameters which, together, comprise a specific condition under which it may be possible to define a property that will always be true.

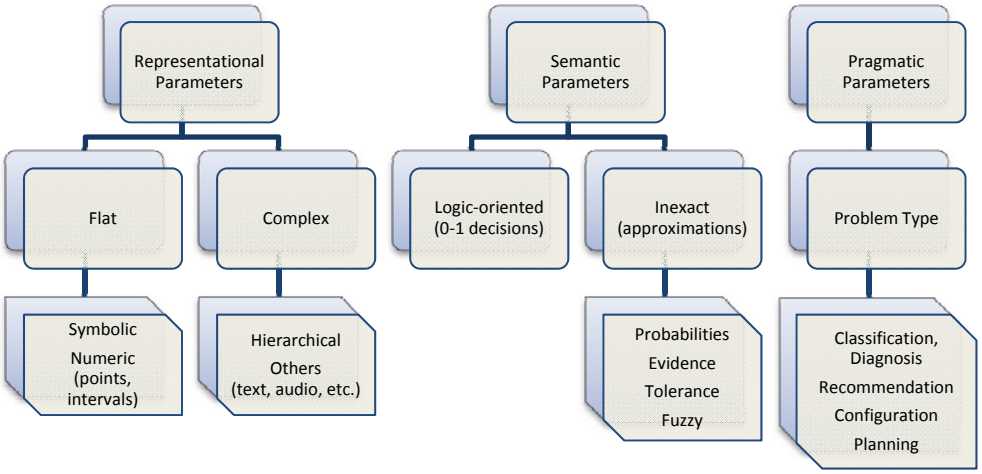

Figure 3. User-independent conditions

In principle, we observe an increasing amount of uncertainty going from left to right in Figure 3. We have a pragmatic parameter with a certain semantic parameter with a certain representational parameter for a certain type of user. In other words, properties of CBR problems can be defined by conditions that are hierarchic quadruples of pragmatics, semantics, representation, and user type; any CBR system needs to reflect that. For example, since a property cannot be true under uncertain conditions, it is not sufficient to describe a CBR property that is always true for a "classification task." Rather, the conditions must be described as a "classification task with logic-oriented parameters with a flat, symbolic representation designed for expert users" in order to be meaningful.

We should note that there are other conditions that depend on the domain in which the CBR system is used. For example, there may be some domains in which some characteristics are more appropriate or more common (e.g., textual cases in the legal domain). There may also be some domains in which fuzzy representations are adequate, while others require more precise measurements. We also consider other aspects of a CBR system to be domain-dependent; for example, it is only useful to know the creator of a CBR system if we have knowledge about this creator that can somehow inform the overall problem-solving ability of a particular CBR system. Similarly, it is only useful to know

when a case base was created if the domain has time-sensitive or time-dependent information, (such as environmental information). We refer to these conditions as domain-dependent and we hope to address them in future work.

2.2 Properties of Case Bases

The four knowledge containers define the system context of a CBR system. As a result, each container can also be defined by its conditions, which are composed of parameters. In this paper, we are concerned mostly with the properties of the case base knowledge container.

Case Bases. Since case bases are knowledge containers, there are two facets that can impact their performance: the design decisions that influence how they function and the knowledge that is embedded in them from a given domain. We begin by looking at the simplest and most clearly defined parameters of a case base: the number of cases, the number of features per case, and the number of values per feature. It must be emphasized that these parameters, when taken individually, yield no knowledge about the case base or its overall quality and are not sufficient to define a condition under which a property is true. For example, we cannot infer anything about the overall quality of a CBR system by knowing that the case base contains 30 cases; we must also know the number features per case and the number of values per feature in order to make any inference about whether or not a CBR system may be effective. In other words, it is the relationship between these parameters that impacts the ability of a case base to find an accurate solution. Thus, case base properties can only be defined by conditions that are hierarchic triples of the number of cases, the number of features per case, and the number of values per feature.

Demonstrating Properties. In order to demonstrate a property of case bases, we first consider the simplest scenario whereby each of the remaining three knowledge containers of the CBR system has minimum or a negligible amount of knowledge. So we define a case base that requires no transformation of solutions, has the simplest vocabulary (discrete, binary features) and a simple similarity measure (feature counting). Additionally, a case base property should ultimately be evaluated by a metric that the users are concerned about; namely, the problem-solving ability of the CBR system. In other words, we are looking to demonstrate a property that has an impact on the overall ability of a case base to retrieve a solution. Of course, the overall utility depends on the pragmatics. For example, in diagnostics there are often situations where the precise diagnosis is not of interest because always the same therapy applies. For this reason, we concentrate on classification or diagnosis problems.

We define the *minimum similarity threshold* as the lowest possible similarity score between a potential target problem and the most similar case in a case base. We can find this threshold by computing the similarity between every possible target problem with every existing case in a case base. We then take the overall minimum similarity of the

most similar existing case to each possible target case and call this the *minimum similarity threshold*. In mathematical terms, for a given *n* number of cases in a case base, *m* number of target problems in a domain, and *p* values per case:

$$MinSimThr = \min_{i \le m}\left(\max_{j \le n}[Sim(i,j)]\right), \quad (1)$$

where $Sim(i,j)$ is determined by feature counting.

To demonstrate this idea let us first take a trivial example. Let us assume that we have two binary features and the following four cases.

$$\begin{bmatrix}1\\1\end{bmatrix}, \begin{bmatrix}1\\0\end{bmatrix}, \begin{bmatrix}0\\1\end{bmatrix}, \begin{bmatrix}0\\0\end{bmatrix}$$

In this example, any new case will have 100% similarity with one of these four existing cases because this case base consists of every possible target problem in the domain. We can say that the *minimum similarity threshold* for this case base is 100%; hence, there is no uncertainty in this case base. Uncertainty can only be added by changing the values of the parameters, i.e. the number of cases, the number of features per case, or the number of values per feature. We hope to demonstrate that these three parameters are directly related to the amount of uncertainty in a case base.

3 Preliminary Study

Demonstrating a case base property requires some empirical testing to investigate how the values of different parameters affect the overall quality of the case base. Again, to simplify the experiment, we begin with minimal or negligible knowledge in the three remaining knowledge containers: no solution transformation, a simple similarity measure (feature counting) and the simplest vocabulary (discrete, binary features). We will demonstrate how the overall quality of the case base changes as the number of cases decreases and the number of features per case increases.

It is important to emphasize that we are making the following assumptions for this experiment:

1. that there are no differences in case quality,
2. that there are no redundant cases,
3. that there are no incomplete cases,
4. that no explanations are required, and
5. that all features have the same weights.

We realize that achieving these assumptions is a difficult problem in itself. Again, we concentrate on classification problems.

Our experiment begins with the simple case base as described in Section 2.2, where we have four cases with two binary features. Assume that each of these four cases also has one implied solution feature that denotes the class to which the case belongs. We

demonstrated that this hypothetical case base has a minimum similarity threshold of 100% because it contains a case with 100% similarity to every possible target problem. But what happens if one of these four cases is taken away? There is now a possible target problem that has no existing solution with 100% similarity and there are two existing cases with equal similarity of 50%, as shown in Table 1.

Table 1. Similarity between target problems and each case in the case base

Target Problems	Case Base $\begin{bmatrix}1\\0\end{bmatrix}$	$\begin{bmatrix}0\\1\end{bmatrix}$	$\begin{bmatrix}0\\0\end{bmatrix}$	Maximum Similarity
$\begin{bmatrix}1\\1\end{bmatrix}$	0.50	0.50	0.00	0.50
$\begin{bmatrix}0\\1\end{bmatrix}$	0.00	1.00	0.50	1.00
$\begin{bmatrix}0\\0\end{bmatrix}$	0.50	0.50	1.00	1.00
$\begin{bmatrix}1\\0\end{bmatrix}$	1.00	0.00	0.50	1.00
Minimum Similarity Threshold:				**0.50**

Removing one case from the original case base drops the minimum similarity threshold from 100% to 50%; this implies that there exists at least one problem in the target domain that has maximum similarity of just 50% with an existing case. As a result, uncertainty has been added to the case base. With this case base as a starting point, we will demonstrate the relationship between the number of cases, the number of features per case, and the ability of a CBR system to return a solution to a problem.

Methodology. We began by creating six hypothetical case bases that consist of the maximum number of non-redundant cases given the number of binary features (ranging from 2 features to 7 features). We will refer to these case bases as *fundamental case bases*, since they consist of every possible target problem in the domain (see Table 2).

Table 2. Fundamental Case Bases

Number of Features	2	3	4	5	6	7
Number of Cases	4	8	16	32	64	128

Then, starting with the fundamental case base, we created a set of test case bases by randomly removing one case at a time and comparing the test case base with the fundamental case base by calculating the minimum similarity threshold at each step until there was only one case remaining. For example, with two binary features per case we created three test cases bases with three cases, two cases, and one case, respectively. At each stage, i.e. after each case is removed, every case from the fundamental case base was considered as a target problem. We should note that the minimum similarity threshold of a fundamental case base is 100%, since it contains a case with 100% similarity to every possible target problem. So, we hope to demonstrate that the minimum similarity

threshold will increase as the number of cases increases, and the similarity threshold will decrease as the number of features per case increases. We note again that in our computations we make use of the assumption that all features have the same importance. It is clear that this is often not the case but we consider it as only a technical aspect.

Results. We first demonstrate the relationship between the number of cases and the minimum similarity threshold of a case base. As shown by Table 3, the minimum number of cases needed to achieve a given threshold increases as minimum similarity threshold increases. So a property of a case base is that, *for a given number of binary features, more cases are needed in order to achieve a higher minimum similarity threshold*.

Table 3. Minimum number of cases needed to achieve the given threshold with the given number of features

Number of Features	Minimum Similarity Threshold						
	0%	≥25%	≥50%	≥60%	≥70%	≥80%	≥90%
2	1	2	2	-	-	-	4
3	1	2	5	5	-	-	8
4	1	2	4	8	8	-	16
5	1	8	8	8	16	16	32
6	1	4	8	16	32	32	64
7	1	4	16	32	32	64	128

For example, if a CBR problem requires four binary features, a designer can be confident that only eight cases are needed in the case base to ensure a minimum similarity threshold of at least 70%. Specifically, the designer can be confident that adding more cases to the case base will have no effect on the minimum similarity threshold. We should note that although the number of target problems that reach the minimum threshold can be reduced by adding more non-redundant cases to the case base, the threshold itself remains unchanged. The notion of completeness is an important aspect to consider when evaluating a CBR system but it is beyond the scope of this paper.

We next demonstrate the relationship between the number of features per case and the minimum similarity threshold of a case base. As shown by Table 4, the maximum number of features needed to achieve a certain threshold decreases as the minimum similarity threshold increases. So a property of a case base is that, *for a given number of cases, less binary features are required in order to achieve a higher minimum similarity threshold*. For example, if a CBR problem consists of 15 cases, a designer should have no more than four binary features if he or she wishes to achieve a minimum similarity threshold of at least 70%. Adding more features to this case base will result in a smaller minimum similarity threshold, i.e. more uncertainty is added.

So, a property of a case base is not the number of cases or the number of features; rather, it is the behavior of these parameters, under a given set of conditions, with respect to the ability of a CBR system to find a solution (which is represented by finding a candidate case with a similarity score above some threshold).

Table 4. Maximum number of features needed to achieve the given threshold with the given number of cases[1]

Number of Cases	Minimum Similarity Threshold						
	0%	≥25%	≥50%	≥60%	≥70%	≥80%	≥90%
1	7	-	-	-	-	-	-
2	7	4	2	-	-	-	-
3	7	4	2	-	-	-	-
4	7	7	4	2	2	2	2
5-7	7	7	4	3	-	-	-
8	7	7	6	5	4	3	3
9-15	7	7	6	5	4	-	-
16	7	7	7	6	5	5	4
17-31	7	7	7	6	5	5	-
32	7	7	7	7	7	6	5
33-63	7	7	7	7	7	6	-
64	7	7	7	7	7	7	6
65-127	7	7	7	7	7	7	-
128	7	7	7	7	7	7	7

CBR Properties in the Real World. We will demonstrate the usefulness of these properties by examining a real-world example. Consider the following features of a case base that is designed to diagnose an individual according to the following symptoms:

Table 5. Features from a real world case base

Features	**Values per feature**
Nausea	2: yes/no
Fever	2: yes/no
Listless	2: yes/no
Blood Pressure	3: high, normal, low
Shortness of Breath	2: yes/no
Diagnosis (solution)	6: influenza, migraine, heart problems, stress, vitamin deficiency, hangover

We would like to provide a diagnosis (i.e., a solution) to new patients by comparing their symptoms to those of other patients with known diagnoses. Specifically, we would like to determine how many cases would be required in this case base in order to produce a diagnosis with a minimum similarity threshold of 80%. We have already demonstrated a property that describes the relationship between the number of cases and the ability of the case base to provide an adequate solution. Unfortunately, because one of the problem features is not binary, this case base does not fit the conditions under which this property is true. In this instance, uncertainty has been added to the problem context. So, as we

[1] Since this experiment was only conducted with case bases with up to 7 binary features, the maximum number of features needed to achieve the given threshold is most likely higher than 7 in some instances.

discussed in Section 1, certainty must be added to the system context in order to compensate. This is done by adding knowledge to one of the knowledge containers in the CBR system.

In other words, we currently have a property that is always true under the following conditions:

Table 6. Conditions under which case base properties are always true

Knowledge Container	**Knowledge is:**	**Parameters**
Case base	*One property that holds*	# cases, # features
Vocabulary	negligible	all discrete, binary features
Similarity measure	negligible	feature counting
Solution transformation	minimum	none

We know that knowledge can be shifted from one container to another without harming the problem-solving ability of the system. Similarly, we can add knowledge to the system context in order to account for uncertainty in the problem context. Therefore, in order to utilize the property for this new problem, we can add certainty to the system by adding knowledge to the vocabulary container.

Table 7. Vocabulary transformation

Feature	**Values per feature**
Blood Pressure	3: high, normal, low
transforms into	
High Blood Pressure	2: yes/no
Low Blood Pressure	2: yes/no

So we now have a system context that fits the conditions of the property demonstrated in the previous section (see Table 8).

Table 8. Conditions of the new transformed problem

Knowledge Container	**Knowledge in:**	**Parameters**
Case base	*One property that holds*	# cases, # features
Vocabulary	Blood pressure feature with 3 values converted to 2 binary features: High Blood Pressure and Low Blood Pressure	all discrete, binary features
Similarity measure	negligible	feature counting
Solution transformation	minimum	none

We now have a case base that exists under the conditions necessary for the known properties to always be true. So, from Table 3, we need at least 64 cases in this case base in order to achieve a minimum similarity threshold of at least 80%.

4 Discussion and Conclusions

We have demonstrated two properties of a case base that associate the number of cases, the number of features per case, and the number of values per feature to the overall quality of a case base. With this knowledge, we can begin to tailor case bases to the specific needs of the users and the specific contexts of a problem. For example, a weather prediction CBR system may require a similarity threshold of just 60%, while a surgery assistance CBR system may require a similarity threshold of 90%. With the results from the experiment described in this paper, we can then provide custom case bases that have the appropriate number of cases, features per case, and values per feature that will yield the desired similarity threshold. Furthermore, if we consider similarity threshold to be an additional parameter of a case base, we can provide guidance to users on how to build a case base if just three of the four parameters are known.

By exploring the properties of case bases, we hope lay a foundation for future CBR research into the general problem solving ability of CBR. Once there is sufficient knowledge about these properties, researchers can begin to explore the idea of which CBR systems are "better" at solving certain types of problems without requiring any additional knowledge about the specific problem the system is designed to address.

References

[1] Aamodt, A., and Nygaard, M.: Different Roles and Mutual Dependencies of Data, Information, and Knowledge - an AI Perspective on Their Integration. Data and Knowledge Engineering 16:191-222 (1995)

[2] Richter, M.M.: The Knowledge Contained in Similarity Measures. Keynote at the 1st International Conference on Case-Based Reasoning, Sesimbra, Portugal (1995)

[3] Richter, M.: Foundations of Similarity and Utility. In Proceedings of the Twentieth Annual Conference of the International Florida Artificial Intelligence Research Society. Key West, FL: AAAI Press (2007)

[4] Cheetham, W.: Case-Based Reasoning with Confidence. In: Blanzieri, E., Portinale, L. (eds.): Advances in Case-Based Reasoning, LNCS, vol. 1898, pp. 15--25. Springer, Berline (2000)

[5] Gu, M. and A. Aamodt. Evaluating CBR Systems Using Different Data Sources: A Case Study. In: Carbonell, J.G., Seikman, J. (eds.): Advances in Case-Based Reasoning. LNCS (LNAI), vol. 4106, pp. 121--135. Springer, Berlin (2006)

[6] Roth-Berghofer, T.: Knowledge Maintenance of Case-Based Reasoning Systems - The SIAM Methodology. KI - Künstliche Intelligenz (2003)

[7] Ferrario, M. A. and B. Smyth (2001). Distributing Case-Base Maintenance: The Collaborative Maintenance Approach. Computational Intelligence 17(2): 315-330.

[8] Shiu, S.C.K., Wang, X.Z., Yeung, D.S.: Neuro-fuzzy approach for maintaining case bases. In: Pal, S.K., Dillon, T.S., Yeung, D.S. (eds.) Soft Computing in Case Based Reasoning, chapter 11. Springer, London (2001)

[9] Leake, D., Smyth, B., Wilson, D., Yang, Q.: Introduction to the Special Issue on Maintaining Case-Based Reasoning Systems. Computational Intelligence 17(2): 193--195 (2001)

[10] Smyth, B., McKenna, E.: Building Compact Competent Case-Bases. In: Althoff, K.-D., Bergmann, R., Brantin, K. (eds.): Case-Based Reasoning and Development. LNCS (LNAI), vol. 1650, pp. 329—342. Springer, Berlin (1999)