On Quality Measures for Case Base Maintenance

Thomas Reinartz[1], Ioannis Iglezakis[1], and Thomas Roth–Berghofer[2]

[1] DaimlerChrysler AG, Research & Technology, FT3/AD,
P.O. Box 2360, 89013 Ulm, Germany
thomas.reinartz@daimlerchrysler.com
ioannis.iglezakis@daimlerchrysler.com

[2] tec:inno GmbH,
Sauerwiesen 2, 67661 Kaiserslautern, Germany
thomas.roth-berghofer@tecinno.com

Abstract. Case base maintenance is one of the most important issues for current research in Case–Based Reasoning (CBR). In this paper, we outline two novel steps as part of the maintenance phase of the CBR process. The review step covers assessment and monitoring of the knowledge containers whereas the restore step actually modifies the contents of the containers according to recommendations resulting from the review step. Here, we focus our attention on the review step for the case base. For this purpose, we define several quality measures based on different case and case base properties that describe specific characteristics of the case base such as correctness, consistency, uniqueness, minimality, and incoherence. These measures allow an initial implementation of the review step for the case base container. We conclude the paper with an outline of future work to extend these aspects of maintenance in CBR.

1 Introduction

During the last decade, Case–Based Reasoning (CBR) evolved from initial ideas originating in cognitive science to a well established intelligent technology suitable to support various applications. One of the consequences is that the focus of current CBR research has moved from issues of case base modeling and acquisition, retrieval and indexing tasks, and similar basic challenges more and more towards application and practice oriented goals. In particular, the early research focus on creating an initial case base is now of less interest but what to do with the contents of a case base over time has become one of the crucial questions in CBR today.

First papers on these issues called efforts along these lines Case Base Maintenance (CBM) (e.g., see [4]). From our point of view, the overall concept of CBM includes all tasks that occur during the lifetime of a CBR system. However, a general framework that describes all the tasks in CBM in a unique way is not yet available.

In this paper, we focus our attention on CBM activities that affect the case base rather than the vocabulary, similarity, or adaptation knowledge containers [8]. For simplicity, we consider any case base as a set of cases — currently, we do not think about indexing schemes, specific representational issues, or similar additional aspects related to the case base.

In practice, a case base as a set of cases changes over time. New cases are added, old or invalid cases are deleted, similar cases are combined to more general cases, conflicting cases are corrected and so on. All of these changes only happen if some kind of indicator invokes the respective mechanisms to change the case base. Therefore, we have to define means of quality control that enable specific realizations of such indicators. In this paper, we argue that quality measures based on basic properties of single cases and sets of cases are able to implement strategies for quality control in CBR.

This paper is organized as follows. In the next section, we report on some prelimenary considerations related to quality control in CBR and briefly describe two novel steps as part of the maintenance phase of the CBR process — these steps are the review and the restore step, and here we focus our attention on the review step. Then, we define the notations necessary to specify quality measures. Thereafter, we describe and define various properties of single cases and sets of cases. These properties are then utilized to define quality measures in section 5. We close the paper with concluding remarks and issues for future work.

2 Quality Control for Case–Based Reasoning

Since early definitions of the CBR process which emphasized the application cycle of CBR, it is now more and more accepted that there is an additional maintenance cycle in CBR needed to cope with all issues that arise when CBR is used in practical applications and especially when the case base contents change over time [3]. We view the first three REs in the current state–of–the–art standard CBR process [1] — Retrieve, Reuse, Revise — as the application cycle whereas the fourth RE — Retain — is the first step in the maintenance phase of CBR.

Initially, the difference between application and maintenance originates from the separation of tasks for using the case base (i.e., find appropriate existing experiences and reuse them for novel situations) and additional tasks to keep the case base in a usable state. However, current research in the area of CBM does not sufficiently define what this usable state looks like and which criteria help to detect situations with less usable case bases.

In software engineering, such usability aspects of a program or system during the stages of its development as well as later when the program or system is used in practice are considered by quality control. This is the initial motivation to consider quality control in CBR, too. Obviously, there is an additional aspect beyond quality control that enables actions to improve the quality of a program or system when the control mechanisms indicate that the quality is not appropriate.

2.1 Maintenance Steps in CBR

These comparisons and ideas lead to the definition of two novel steps as part of the maintenance phase of the CBR process. Figure 1 shows the review and the restore steps in CBM along with the control flow between them. The two steps start with a case base as their input. This case base is either the result of an initial knowledge acquisition step

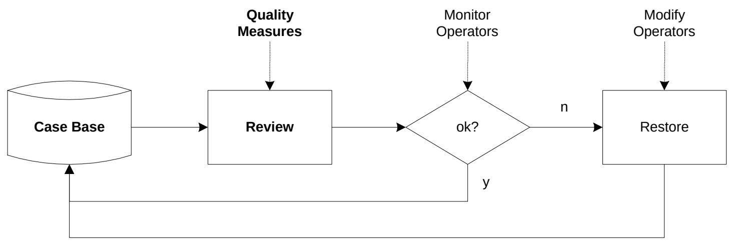

Fig. 1. Maintenance steps in CBR

at the beginning of a CBR project or the outcome of modifications to the case base at a certain stage after the case base is already in use.

The review step considers the current state of the case base and assesses its quality. For this purpose, we have to define quality measures that allow the computation of values that indicate the current quality of the case base. These quality values are then monitored and specific indicators lead to the initiation of the restore step.

The restore step utilizes specific modify operators to change or to adapt the contents of the case base. In an ideal setting, the review step already suggests specific changes to return to the level of quality desired. If there is no need to go to the restore step since the quality values are still in good shape, we simply return to the case base, keep it unchanged, and start with the next iteration of the maintenance cycle.

All in all, we propose a new 6RE CBR process which consists of six steps: Retrieve, Reuse, Revise, Retain, Review, and Restore. The first three steps form the application (or usage) cycle whereas the last three build the maintenance phase in CBR.

The two maintenance steps in figure 1 completely abstract from other issues in maintenance of CBR systems. We are aware of the fact that additional maintenance tasks are important if we work on the maintenance task in a more complete way. For example, we also have to consider the retrieval behavior and the appropriateness of the similarity measures over time. However, we argue that it is important to start with the case base as the central knowledge container in CBR, and that it is also important to start with reasonable chunks of tractable subtasks in maintenance.

Moreover, we focus our attention on the review step in this paper and consider the restore step as a topic for future work and further considerations.

2.2 Related Work

If we consider existing work towards quality measures for CBR that also consider the case base, we are aware of the following references.

For example, Aha and Breslow [2] use measures suggested by CBR vendors to optimize conversational CBR systems.

In the area of unstructured case bases, Racine and Yang [7] give a definition for inconsistency and redundancy. Inconsistency can be detected through background knowledge or sections which contradict each other and redundancy is observed through subsumption. These measures are limited to the underlying CBR approach.

More general measures are defined by Leake and Wilson [5]. They use measures (regularities) to define two kinds of relationships. First, they define relations between the similarity of problems and the similarity of solutions. Second, they define relations between prior problems and new problems.

Racine and Yang [6] suggest different measures (criteria) to evaluate case bases. However, their focus of interest is consistency — they define measures for intra–case inconsistencies and inter–case inconsistencies for consistency management and the optimization of validation methods. Intra–case inconsistency is defined as a case property which deals with constraint violations of attributes within a case. Inter–case inconsistency is defined as a case base property which can be found across two or more cases.

Smyth and Keane [9] optimize the contents of the case base to preserve the competence of the case base through case deletion. They define the two measures coverage and reachability to achieve this. Coverage of a case is the set of all problems in the problem area which can be solved by this case through adaptation. Reachability of a case is the set of all cases which are used to solve this case through adaptation. Coverage and reachability cannot be calculated because the possible set of problems is in general too vast. Thus the assumption is made that the problem distribution in the case base is representative and a heuristic is used.

Further work of Smyth and McKenna [10] brings up the measures of case density and group density for modeling competent CBR systems. Another strategy of competence preserving is given by the definition of the measure of coverage from Zhu and Yang [11]. Both strategies are case addition rather than case deletion policies.

In summary, all of these existing measures are limited to the underlying CBR approaches, and they use or affect other knowledge containers like the similarity container or need heuristics to be calculated.

2.3 Requirements for Quality Measures in CBR

In conclusion, the initial thoughts on quality control in CBR and the review of the relatively few existing papers on quality measures for CBR lead to the following requirements on quality measures.

First, the quality measures and their values have to reflect user requirements that usually ask for a small, comprehensive, and consistent case base. Second, these measures should be simple in the sense that their computation is possible in practice. Third, we also expect the measures to be application independent and as system independent as possible. Although specifics of quality control are possibly context dependent, the general concepts of quality control should be valid across all applications and systems.

3 Notation

In this section, we define notations to represent cases and sets of cases. These definitions are necessary to later specify concrete quality measures for CBR. For all notations, we aim at as general definitions as possible with maximum flexibility to cover cases and case bases in most CBR applications and systems.

We start the notations with the basic components for building cases, namely attributes, values, problems, and solutions (see definitions 1 to 3). We choose an attribute–value representation since it ensures high flexibility and generality. We are able to transform most case representations into an attribute–value representation if necessary.

Definition 1 (Attributes and Values). *An attribute a_j is a name accompanied by a set $V_j := \{v_{j1}, \ldots, v_{jk}, \ldots, v_{jN_j}\}$ of values. We denote the set of attributes as $A := \{a_1, \ldots, a_j, \ldots, a_N\}$, and the set of values as $V := \bigcup_{j=1}^{N} V_j$.*

Each attribute consists of an identifier (the name) and a set of possible values. For simplicity, we assume for all attributes, especially for quantitative attributes, that the set of values contains only ”occurring” values. This means by definition and the dynamic nature of CBR that sets of values are dynamic as well. For example, adding a new alternative answer to a question in conversational CBR means extending the set of values for this specific attribute.

Now, we define problems as sets of attribute values and solutions as any form of information that contributes to the solution of a given problem. Note, we generally assume that the single fault assumption holds, i.e., a single problem description only covers a single problem which in turn requires a single solution. If we encounter a case which covers more than a single problem, we have to transform this case into more than one case until the single fault assumption holds again. However, we do not further elaborate this issue here.

Definition 2 (Problem). *A problem is a set $p_i := \{p_{i1}, \ldots, p_{ij'}, \ldots, p_{iN_i}\}$ with $\forall j' \in [1; N_i]\, \exists a_j \in A\, \exists v_{jk} \in V_j : p_{ij'} = v_{jk}$, and $\forall j \in [1; N] : \mid (p_i \cap V_j) \mid \leq 1$. We denote the set of problems as $P := \{p_1, \ldots, p_i, \ldots, p_M\}$.*

The first condition in definition 2 ensures that for each element in the set of values of a problem there exists a corresponding attribute and a respective value in its set of values. The second condition makes sure that for each attribute a problem does not contain more than a single value. The latter assumption simplifies from situations where it makes sense to allow more than a single value for specific attributes. However, this situation is either emulated by adding extra sets of single values to the set of values or by relaxing the second condition at a later stage of research.

This particular set–oriented definition enables easy implementation of the quality measures defined below. Note, we do not assume that a single problem specifies values for each available attribute. In contrast, a problem only contains those values for attributes which are relevant to describe the specific situation at hand.

Definition 3 (Solution). *A solution is any item s_i. We denote the (multi-) set of solutions as $S := \{s_1, \ldots, s_i, \ldots, s_M\}$.*

For a solution, we do not define any additional requirements in order to avoid restrictions for solutions in any way (see definition 3). For example, a solution possibly contains any text or media information which contains a description how to ”solve” the problem. The more structured the domain is, the more structured information or definition we provide for a solution. For example, in classification domains, a solution is simply yet another attribute with a domain of values containing all occurring class labels. On the restrictive side of this flexibility, we are only able to think about solutions in terms of match or mis–match but not in terms of any other more fine–grained meanings.

For both sets, the set of problems and the set of solutions, we assume that they contain M elements. Since we are aware of the fact that the same solution possibly solves different problems, we allow the set S of solutions to contain the same element more than once. Hence, S in definition 3 is possibly a multi–set. We further assume that the enumeration of problems and solutions in P and S corresponds to each other, i.e., the solution for a given problem has the same index in S as the problem has in P.

For the moment, we also presume that P and S contain components of ”real” cases, i.e., cases that (currently) exist in the case base and that make sense in terms of the application domain. If the contents of the case base change, both sets P and S change, too. In addition to such real cases, we also have potential cases (i.e., cases which can occur by combining any possible combination of attribute values) and meaningful cases (i.e., potential cases that in fact lead to a case that corresponds to any possible case in the application domain that makes sense). For example, occurring queries to the CBR system provide initial hints for such meaningful cases.

Definition 4 (Case and Case Base). *A case is a tuple $c_i := (p_i, s_i, q_i)$ with a problem p_i, a solution s_i, and additional quality information q_i. A case base is a set of cases $C := \{c_1, \ldots, c_i, \ldots, c_M\}$.*

Definition 4 states that a case is a tuple containing a problem component p_i, a solution component s_i, and an additional information component q_i. This additional information component is comprised of any extra information that is necessary for quality control and that is related to the life cycle of this specific case. For example, time stamps for the time the case is added to the case base and the time of its last access or the relative number of correct usages of the case are typical information items. In this paper, we do not consider this additional information but focus on the problem and solution components of cases.

In comparison to other definitions and representations of cases, we do not distinguish between problem — as an initial short characterization of the problem — and situation which is then a more detailed description of the context of the entire challenge. We omit this distinction to keep the representation as simple as possible. If we utilize this kind of representation in an application domain where there typically is an initial short problem characterization and then a more detailed situation description, we simply assume that the final problem component in definition 4 already contains both, the problem and the situation, possibly acquired during a dialogue or by asking for more detailed symptoms through tests.

Example 1 (Set of Cases). Assume a domain with $A := \{a_1, a_2, a_3, a_4\}$, $V_1 := \{v_{11}, v_{12}, v_{13}\}$, $V_2 := \{v_{21}, v_{22}, v_{23}, v_{24}\}$, $V_3 := \{v_{31}, v_{32}\}$, and $V_4 := \{v_{41}, v_{42}, v_{43}, v_{44}, v_{45}\}$. $G := \{c_1, c_2, c_3, c_4, c_5, c_6\}$ in table 1 is a set of cases in this domain.

4 Case and Case Base Properties

In this section, we present several case and case base properties. Case properties describe characteristics of single cases whereas case base properties provide information

Table 1. Six cases from an example case base

	p_i				s_i	q_i
c_1	v_{12}	v_{23}	v_{31}		s_1	q_1
c_2	v_{13}	v_{23}			s_2	q_2
c_3	v_{13}	v_{23}			s_3	q_3
c_4	v_{13}	v_{23}	v_{32}	v_{41}	s_4	q_4
c_5	v_{13}	v_{23}	v_{32}		s_5	q_5
c_6	v_{12}		v_{31}	v_{45}	s_6	q_6

about sets of cases, i.e., subsets of cases in the case base. We define case properties as the atomic concepts to specify case base properties which in turn form the basis for the definition of quality measures for CBR. In particular, atomic properties are independent of each other, i.e., there is no general relation between any pair of two properties.

For case properties, we further distinguish between isolated and comparative characteristics. Whereas isolated properties only consider characteristics of a single case to define properties of that single case, comparative properties also take into account information on other cases and compare them to the single case considered. Note, comparative case properties still define only characteristics of a single case although they consider more than a single case.

4.1 Isolated Case Properties

The most important isolated case property is correctness. We consider a case as correct if the given solution component really "solves" the problem specified by the problem component (see definition 5). The crucial part of this definition is the notion "solves". At this point, we basically use a place holder for any type of relation "solves" denoted by $\mathcal{S}$. Thereby, the definition is open to all application domains. For example, in traditional classification domains, the solution of a case is a class label and the relation $\mathcal{S}$ between problem and solution holds if this class label corresponds to the true class label of the problem. Note, as soon as we want to compute the correctness of a case, we have to define relation $\mathcal{S}$ precisely.

Definition 5 (Correctness). *Assume a binary relation $\mathcal{S} \subseteq P \times S$, and a case $c_i \in C$.*
c_i *correct* $:\Longleftrightarrow \; \mathcal{S}(p_i, s_i)$.

For the moment, we do not define any other case properties that only consider characteristics of a single case although we expect that the additional information component q_i of a case naturally leads to more isolated case properties. For example, the information component q_i enables control of time and usage aspects and is therefore able to indicate the necessity of maintenance based on these aspects. In this paper, we consider the behavior of a single case in relation to other cases in the case base as more important, and we will now focus our attention on this situation.

4.2 Comparative Case Properties

In this subsection, we turn to comparative case properties that describe characteristics of single cases in relation to other cases. For all of the following definitions in this subsection, we assume that each case is correct.

Definition 6 (Consistency). *Assume $G \subseteq C$, and $c_i \in G$.*
c_i *consistent within* $G \ :\Longleftrightarrow \ \nexists c_{i'} \in G : p_{i'} \subseteq p_i \ \wedge \ s_i \neq s_{i'}$.

The first comparative case property describes consistency within a set of cases. A single case is consistent within a set of cases if and only if there does not exist any other case which solves the same or a more general problem differently. In general, this definition covers the issue of alternatives. We assume that for each possible problem there exists a ”best” solution. For example, the quality of a solution is characterized by the complexity of its realization — is better to re–boot a computer than to buy a new one.

Consequently, we want the case base to consist only of cases that have ”best” solutions and do not allow any alternative solutions. This also excludes the possibility of an alternative mentioned in the same case or as an extra case which has the same problem component but a different solution component. Although we do not want alternatives, we certainly must specify a concept, namely consistency, which enables the quality control mechanisms to detect alternatives and to suggest changes to the case base to avoid alternatives.

Moreover, definition 6 also covers situations where the subset relation between problem components with different solution components indicates that the more general problem definition possibly lacks some detailed information which is necessary to make the two cases distinctive.

Definition 7 (Uniqueness). *Assume $G \subseteq C$, and $c_i \in G$.*
c_i *unique within* $G \ :\Longleftrightarrow \ \nexists c_{i'} \in G, c_{i'} \neq c_i : p_{i'} = p_i \ \wedge \ s_i = s_{i'}$.

Definition 7 declares a case as unique if and only if there does not exist another case within the considered set of cases which solves exactly the same problem in exactly the same way.

A more general relation which covers a similar situation is subsumption (see definition 8). A case subsumes another case if its problem component is a true subset of the problem component of the subsumed one and the solution component remains the same. We call cases without any different case that subsumes it minimal.

If we assume that the elements in the problem component define the problem as a conjunction of all elements, then the subsumes relation corresponds to solving a more general problem in the same way. The more specific problem characterization then possibly contains too many details unnecessary to specify the core problem. The restore step in CBM is responsible for resolving such situations accordingly.

Definition 8 (Minimality). *Assume $G \subseteq C$, and $c_i \in G$.*
c_i *minimal within* $G \ :\Longleftrightarrow \ \nexists c_{i'} \in G : p_{i'} \subsetneq p_i \ \wedge \ s_i = s_{i'}$.

The most complex comparative case property describes situations with two cases within a set of cases that coincide in most of their components except for a specific number Δ of values (see definition 9). We call a case incoherent within a set of cases if and only if there does not exist any other case which overlaps with the case in more than a pre-determined number of components. We consider incoherent cases as positive since the more cases differ, the broader is the spectrum of problems that they cover.

Definition 9 (Incoherence). *Assume* $G \subseteq C$, $c_i \in G$, *and* $1 \leq \varDelta \in \mathbb{N}$.
c_i *incoherent within* $G :\Longleftrightarrow\ \nexists c_{i'} \in G : |\, p_i \cap p_{i'} \,| + \varDelta = N_i = N_{i'} \ \wedge\ s_i = s_{i'}$.

With parameter $\varDelta$ we trigger the extent of overlapping information within coherent cases. For example, if $\varDelta$ is 1, two cases are coherent if all of their values are the same except for a single value (in each case). Note, this difference in a single component corresponds either to a situation with two different values for the same attribute or with different values for separate attributes.

For each pair of coherent cases, we possibly consider modification operators that generalize the two cases to a single one which represents both original cases. Removing the distinguishing values completely, joining them as a single set of alternatives, or generalizing to the next abstract level in a hierarchy of values are potential generalization operators.

Example 2 (Case Properties). Assume G in table 1, and $\varDelta = 1$.
(i) c_1 is consistent within G;
c_4 is not consistent within G except if $s_2 \equiv s_3 \equiv s_4 \equiv s_5$.
(ii) c_1 is unique within G;
c_2 is not unique within G if $s_2 \equiv s_3$.
(iii) c_1 is minimal within G;
c_5 is not minimal within G if $s_2 \equiv s_5$ or $s_3 \equiv s_5$.
(iv) c_2 is incoherent within G;
c_6 is not incoherent within G if $s_1 \equiv s_6$.

4.3 Case Base Properties

In the previous subsections, we defined several case properties to make decisions about the quality of cases. Now, we build upon these characteristics and specify properties for sets of cases, and hence for an entire case base which is simply a set of cases containing all cases in the case base.

Definition 10 (Case Base Properties). *Assume* $G \subseteq C$.
(i) G *is correct* $:\Longleftrightarrow\ \forall c_i \in G :\ c_i$ *correct.*
(ii) G *is consistent* $:\Longleftrightarrow\ \forall c_i \in G :\ c_i$ *consistent within* G.
(iii) G *is unique* $:\Longleftrightarrow\ \forall c_i \in G :\ c_i$ *unique within* G.
(iv) G *is minimal* $:\Longleftrightarrow\ \forall c_i \in G :\ c_i$ *minimal within* G.
(v) G *is incoherent* $:\Longleftrightarrow\ \forall c_i \in G :\ c_i$ *incoherent within* G.

As case properties started with a notion of correctness for single cases, we also define a notion of correctness for a set of cases (see definition 10). This notion of correctness uses the correctness of single cases — a set of cases is correct if and only if all of its cases are correct.

In a similar vein, we also adopt the definition of consistent, unique, minimal, and incoherent. Definition 10 summarizes the extensions of these definitions for single cases to sets of cases in the same way as for correctness.

5 Initial Quality Measures for Case Base Maintenance

In this section, we use the previously defined case and case base properties to specify initial quality measures for case base maintenance. We start the set of initial quality measures with several degrees of quality. Definition 11 summarizes degrees of correctness, consistency, uniqueness, minimality, and incoherence.

Definition 11 (Degrees of Case Base Properties). *Assume $C^{\subseteq}$ is the set of all subsets of C.*

(i) $\mathcal{D}_{\surd} : C^{\subseteq} \mapsto [0;1],\ \mathcal{D}_{\surd}(G) := \dfrac{\mid \{c_i \in G \ \mid \ c_i\ correct\} \mid}{\mid G \mid}$

(ii) $\mathcal{D}_{\otimes} : C^{\subseteq} \mapsto [0;1],\ \mathcal{D}_{\otimes}(G) := \dfrac{\mid \{c_i \in G \ \mid \ c_i\ consistent\ within\ G\} \mid}{\mid G \mid}$

(iii) $\mathcal{D}_{\neq} : C^{\subseteq} \mapsto [0;1],\ \mathcal{D}_{\neq}(G) := \dfrac{\mid \{c_i \in G \ \mid \ c_i\ unique\ within\ G\} \mid}{\mid G \mid}$

(iv) $\mathcal{D}_{\subsetneq} : C^{\subseteq} \mapsto [0;1],\ \mathcal{D}_{\subsetneq}(G) := \dfrac{\mid \{c_i \in G \ \mid \ c_i\ minimal\ within\ G\} \mid}{\mid G \mid}$

(v) $\mathcal{D}_{\Delta} : C^{\subseteq} \mapsto [0;1],\ \mathcal{D}_{\Delta}(G) := \dfrac{\mid \{c_i \in G \ \mid \ c_i\ incoherent\ within\ G\} \mid}{\mid G \mid}$

Each of these degrees computes the number of ”good” cases in terms of the defined case properties and divides this number by the total number of cases within the considered set of cases. Hence, each degree counts the relative number of ”good” cases within a given set of cases.

In comparison to the case base properties, these degrees enable a more refined consideration of the characteristics of the case base. The previously defined case base properties are essentially special cases for which each of the degrees yields value 1.

The purpose of the different degrees in terms of the maintenance steps in CBR is to get values that provide an indicator for the review step and the decision whether the case base state is still acceptable or not. For example, we specify a threshold and then classify a case base as acceptable if and only if each of the degrees is above this threshold.

Of course, we also think of more complex monitor operators that consider the degree values over time and react on the dynamic changes of the resulting functions. For example, we invoke maintenance in terms of applying modify operators as soon as the difference between two measurements is too large into a decreasing direction rather than only comparing absolute degree values and a pre–specified threshold.

Definition 12 (Quality Measures). *Assume $C^{\subseteq}$ is the set of all subsets of C, and $w_{\surd}$, $w_{\otimes}$, $w_{\neq}$, $w_{\subsetneq}$, $w_{\Delta} \in [0;1]$ with* $\sum\limits_{x \in \{\surd,\otimes,\neq,\subsetneq,\Delta\}} w_x = 1.$

(i) $\mathcal{Q}_{min} : C^{\subseteq} \mapsto [0;1],\ \mathcal{Q}_{min}(G) := \min\limits_{x \in \{\surd,\otimes,\neq,\subsetneq,\Delta\}} \{\mathcal{D}_x(G)\}$

(ii) $\mathcal{Q}_{max} : C^{\subseteq} \mapsto [0;1],\ \mathcal{Q}_{max}(G) := \max\limits_{x \in \{\surd,\otimes,\neq,\subsetneq,\Delta\}} \{\mathcal{D}_x(G)\}$

(iii) $\mathcal{Q}_{w} : C^{\subseteq} \mapsto [0;1],\ \mathcal{Q}_{w}(G) := \sum\limits_{x \in \{\surd,\otimes,\neq,\subsetneq,\Delta\}} w_x \cdot \mathcal{D}_x(G)$

Definition 12 shows three examples for quality measures in CBR that cumulate the various degrees specified above to a single overall value of case base quality. The first variant considers the minimum value of degrees as the crucial number for quality control. If we assume that we compare quality values with a given threshold to trigger the restore step, this means that quality control based on this measure invokes modify operators as soon as a single degree — regardless which one — becomes inappropriate. The second example in definition 12 does exactly the opposite. It starts to recommend changes to the case base only if all degrees depict the same bad quality.

The third measure is a compromise between those two extremes and considers the average degree value if we set all weights to the same value. On the other hand, this measure also allows the user to set preferences on specific aspects. For example, the user does not care about redundant cases and sets $w_{\neq}$ to zero, but does not want any inconsistencies within any subset of cases in the case base and consequently initializes $w_{\otimes}$ to a relatively high value.

Example 3 (Quality Measures). Assume G in table 1, G is correct, $s_1 \equiv s_6$, $s_2 \equiv s_3 \equiv s_5$, $\Delta = 1$, and $w_{\surd} = w_{\otimes} = w_{\neq} = w_{\subsetneq} = w_{\Delta} := 1/5$.

Then, $\mathcal{D}_{\surd}(G) = 1$, $\mathcal{D}_{\otimes}(G) = 5/6$, $\mathcal{D}_{\neq}(G) = 2/3$, $\mathcal{D}_{\subsetneq}(G) = 5/6$, and $\mathcal{D}_{\Delta}(G) = 2/3$. Moreover, $\mathcal{Q}_{min}(G) = 2/3$, $\mathcal{Q}_{max}(G) = 1$, and $\mathcal{Q}_{w}(G) = 4/5$.

We get more alternative quality measures if we allow the weights w_x to be any function of G, or by adding aspects of time, or by taking into account measures of usage information, and so on. At this point, the concept of specifying quality measures is sufficiently general to enable appropriate definitions of respective quality control mechanisms in almost any domain.

6 Concluding Remarks

In summary, we considered case base maintenance as one of the most important issues in current CBR research and suggested two novel steps, the review and the restore steps, within the maintenance phase of the CBR process. We emphasized quality control in the review step and proposed several quality measures based on case and case base properties to monitor the quality of the case base.

Beyond the exact and precise definition of a framework which covers all aspects and tasks in CBM, our suggestions for future work point in several directions. First, we further analyze the specified quality measures, their relation to each other, and how their values actually affect the case base. Second, we plan to implement the quality measures and to test if they are really appropriate to measure the quality of a case base and to indicate when the restore step in maintenance is necessary. This issue implies that we also define different mechanisms to monitor the quality values and to specify concrete moments to invoke the restore step.

The current definition of quality measures considers local similarity only as a matching function — for this aspect, we also plan to extend our definitions to other more fine–grained local similarity measures. Moreover, extensions to other knowledge containers beyond the case base are an important issue for future work.

Another line of future research is the restore step itself. We have to define which modify operators are necessary to change the case base and to improve its current quality. In particular, this task also means that we have to analyze dependencies between quality control and modification of the case base.

Finally, issues of computation play an additional role. For example, if the case base becomes too large to compute the quality measures in a reasonable amount of time, we have to consider aspects of clustering cases in order to compute quality values in local case base areas and then cumulate these local values to an overall quality of the entire case base.

Acknowledgments

We thank Andre Hüttemeister and Mehmet Göker for their initial ideas on quality measures for maintenance in CBR which led to the definitions presented in this paper.

References

1. Agnar Aamodt and Enric Plaza. Case–based reasoning: Foundational issues, methodological variations, and system approaches. *AI Communications*, 7(1):39–59, 1994.
2. David W. Aha and Leonard A. Breslow. Refining conversational case libraries. In *Proceedings of the Second International Conference on Case–Based Reasoning*, pages 267–278, 1997.
3. Mehmet Göker and Thomas Roth-Berghofer. The development and utilization of the case–based help–desk support system HOMER. *Special Issue of the International Journal "Engineering Applications of Artificial Intelligence"*, 12(6), 1999.
4. David B. Leake and David C. Wilson. Categorizing case–base maintenance: Dimensions and directions. In *Proceedings of EWCBR–98, Advances in Case–Based Reasoning*. Springer–Verlag, 1998.
5. David B. Leake and David C. Wilson. When experience is wrong: Examining CBR for changing tasks and environments. In *Proceedings of the Third International Conference on Case–Based Reasoning*, 1999.
6. Kirsti Racine and Qiang Yang. On the consistency management of large case bases: the case for validation. In *Proceedings of the AAAI–96 Workshop on Knowledge Base Validation, American Association for Artificial Intelligence (AAAI)*, 1996.
7. Kirsti Racine and Qiang Yang. Maintaining unstructured case bases. In *Proceedings of the 1997 International Conference on Case Based Reasoning*, pages 553–564, 1997.
8. Michael M. Richter. The knowledge contained in similarity measures. Invited Talk at the International Conference on Case–Based Reasoning, 1995.
9. Barry Smyth and Mark T. Keane. Remembering to forget: A competence–preserving deletion policy for case–based reasoning systems. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, pages 377–382, 1995.
10. Barry Smyth and Elizabeth McKenna. A portrait of case competence: Modelling the competence of case–based reasoning systems. In *Proceedings of the 4th European Workshop on Case–Based Reasoning.*, pages 208–220, 1998.
11. Jun Zhu and Qiang Yang. Remembering to add: Competence–preserving case addition policies for case base maintenance. In *Proceedings of the International Joint Conference in Artificial Intelligence (IJCAI)*, 1999.