Relations between Customer Requirements, Performance Measures, and General Case Properties for Case Base Maintenance

Ioannis Iglezakis and Thomas Reinartz

DaimlerChrysler AG, Research & Technology, RIC/AM,
P.O. Box 2360, 89013 Ulm, Germany
ioannis.iglezakis@daimlerchrysler.com
thomas.reinartz@daimlerchrysler.com

Abstract. The ultimate goal of CBR applications is to satisfy customers using this technology in their daily business. As one of the crucial issues in CBR for practical applications, maintenance is important to cope with demands changing over time. Review and restore are the two steps in CBR that deal with tasks of maintenance. In order to perform these tasks, we suggested case and case base properties, quality criteria, and restore operators in earlier publications. In this paper, we specify concrete performance measures that correspond to general customer requirements, and analyze the relations between these performance criteria, case properties, and restore operators. We present initial results on theoretical analyzes on these relations, and report on examples of experimental studies that indicate that the suggested case properties and the respective restore operators help to identify maintenance strategies in order to optimize performance of CBR systems over time.

1 Introduction

In earlier publications, we established review and restore as two additional steps that enhance the 4–RE CBR process consisting of retrieve, reuse, revise, and retain [1]. The review step checks the CBR system quality and tests if this quality still meets desired quality criteria, whereas the restore step uses different operators to modify the system to get back to a desired level of quality.

The review step consists of three different tasks. First, the *assess* task evaluates the quality of one of the knowledge containers of a CBR system. Second, the *monitor* task displays or visualizes the results of the assess task and, for example, compares the assessment outcome with thresholds. At the end of the review step, it is the goal of the *notify* task to initiate actions on the results of the monitor task.

The restore step again consists of three tasks. First, the *suggest* task computes potential restore operations that are able to modify the affected knowledge container in order to get back to the desired level of quality. Second, the *select* task ranks the suggested restore operations according to some pre–defined preference criterion. Finally, the goal of the *modify* task is the execution of the selected restore operations.

In previous papers, we defined case and case base properties to detect conflicts between cases and quality measures to assess the quality of the case base, and specified

several restore operations for the case base along with initial heuristics for their application [4,5]. In this paper, we follow up these ideas for case base maintenance and enhance the previous concepts in several ways.

First, we define performance measures for using a CBR system that reflect customer requirements in practical applications. Second, we generalize the case properties to similarity–based properties that no longer rely on specific similarity measures but allow any similarity measure. We will then show how customer requirements, performance criteria, and case properties relate to each other in the context of case base maintenance. Therefore, we present some theoretical results as well as examples for experimental studies that indicate the correctness of the theoretical analyzes.

2 Customer Requirements

Assume typical customers of a help–desk for IT support. They call help–desk agents when they come along a problem with their computer system which they are not able to solve by themselves.[1] Then, the customer describes the problem, and the help–desk agent normally asks additional questions to get a clearer picture of the problem. As soon as the problem is concrete enough, the help–desk agent aims at supporting the customer with a solution, often by remembering similar situations with similar problems.

In terms of case–based reasoning, such a scenario is a typical diagnosis application. The customer problem forms the problem component of a query to the CBR engine. The problem description together with answers to additional questions construct symptoms of the problem definition, often in terms of an attribute–value representation. And finally, the overall solution of the problem corresponds to the solution component of a new case, or matches the solution component of an existing case in the case base (or an adaptation thereof).

In those and similar scenarios, customer requirements are relatively clear:

1. *Customers want an answer to their problem at all.*
   If customers ask a query to some support or a CBR system, either directly or indirectly, they expect an answer. If customers do not get an answer to their query, they get an impression of incompetence, either of the support or of the CBR system.
2. *Customers want correct answers to their problem.*
   Obviously, customers are additionally interested in getting a correct answer. It is possibly not necessary to provide the best correct answer, but it is essential to offer a correct answer.
3. *Customers want fast answers to their problems.*
   Customers with problems are impatient; they want solutions fast. Although, for example, in case of help–desks, there usually exists a complex management for priorities of problems, customers perceive their own problem as the most important and urgent problem.
4. *Customers want confident answers to their problems.*
   In addition to correct and fast answers, customers often expect that the support has a specific confidence. They do not want to try out many solutions until they really

[1] For the moment, we ignore that it is possible to contact modern help–desk organizations in multiple ways beyond telephone calls.

get the right answer to their original problem, but insist on that the first solution is already the correct one. Alternatively, if the support is not able to provide a correct solution, customers want to be confident that the support is in no doubt of its decisions.

Unfortunately, these general customer requirements refer to criteria that we are only able to measure if we really use the CBR system in practice. It is hardly possible to estimate the quality of the CBR system according to these requirements by simply looking at the different knowledge containers. Hence, we are looking for a way firstly to measure some criteria that reflect these customer requirements, and then secondly to estimate the values of these criteria before really using the system in practice. For the first purpose, we now specify concrete measurable performance criteria; and for the second need, we identify relations between performance and case properties.

3 Case Representation and Similarity Measure

Before we start with the definition of performance measures, we specify the case representation as well as the similarity measure which we use subsequently. These definitions follow up the representation which we have defined in previous publications. For a detailed discussion, we refer to [5].

Definition 1 (Cases and Case Base).

a) 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 $A := \{a_1, \ldots, a_j, \ldots, a_N\}$.
b) A problem is a set $p_i := \{p_{i1}, \ldots, p_{ij'}, \ldots, p_{iN_i}\}$ with $\forall j' \in [1; N_i]\ \exists a_j \in A$ and $\exists v_{jk} \in V_j : p_{ij'} = v_{jk}$, and $\forall j \in [1; N] : \ |(p_i \cap V_j)| \ \leq 1$. We denote the set of problems as $P := \{p_1, \ldots, p_i, \ldots, p_M\}$.
c) A solution s_i is any item.
d) A case is a tuple $c_i := (p_i, s_i)$ with a problem p_i and a solution s_i. A case base is a set of cases $C := \{c_1, \ldots, c_i, \ldots, c_M\}$.
e) We further assume a separation of C into a training set T and a test set (or query set) Q with $C = T \cup Q$ and $T \cap Q = \emptyset$.

The auxiliary functions in definition 2 count coincidence and difference between values of two problem definitions, e.g., of a case and a query. $S_{\leftrightarrow}$ is the number of values for the same attribute with local similarity 1. For example, for symbolic attributes, this number equals the number of identical values; for numeric attributes, this is the number of values with only small differences, e.g., in comparison to a pre–defined threshold ϵ. $S_{\nleftrightarrow}$ is the contrary function and corresponds to the number of unequal or more discriminant values. $S_{\leftarrow}$ and $S_{\rightarrow}$ count the number of values for attributes that occur in the case but not in the query, and vice versa, respectively. Finally, S_{-} depicts the amount of information that is missing in both problem definitions, i.e., it counts the number of attributes without values, neither in the case nor in the query.

Definition 2 (Auxiliary Functions). *Assume a local similarity measure $sim_j : V_j \times V_j \mapsto [0; 1]$.*

a) $S_{\leftrightarrow} : P \times P \mapsto \{1..N\}$,
$S_{\leftrightarrow}(p_i, p_{i'}) := \left|\{j \in \{1..N\} : |p_i \cap V_j| = |p_{i'} \cap V_j| = 1 \ \wedge \ sim_j(p_{ij}, p_{i'j}) = 1\}\right|$

b) $S_{\leftrightsquigarrow} : P \times P \mapsto \{1..N\}$,
$S_{\leftrightsquigarrow}(p_i, p_{i'}) := \left|\{j \in \{1..N\} : |p_i \cap V_j| = |p_{i'} \cap V_j| = 1 \ \wedge \ sim_j(p_{ij}, p_{i'j}) \neq 1\}\right|$

c) $S_{\leftarrow} : P \times P \mapsto \{1..N\}$,
$S_{\leftarrow}(p_i, p_{i'}) := \left|\{j \in \{1..N\} : |p_i \cap V_j| > |p_{i'} \cap V_j|\}\right|$

d) $S_{\rightarrow} : P \times P \mapsto \{1..N\}$,
$S_{\rightarrow}(p_i, p_{i'}) := \left|\{j \in \{1..N\} : |p_i \cap V_j| < |p_{i'} \cap V_j|\}\right|$

e) $S_{-} : P \times P \mapsto \{1..N\}$,
$S_{-}(p_i, p_{i'}) := \left|\{j \in \{1..N\} : |p_i \cap V_j| = |p_{i'} \cap V_j| = 0\}\right|$

We use the auxiliary functions for two different purposes. First, we define the overall similarity measure by a weighted cumulation of these auxiliary values, and second we use them to specify the general case properties later.

The overall similarity in definition 3 is the normalized weighted sum of above auxiliary values. We definitely consider values that coincide for the same attribute ($S_{\leftrightarrow}(p_i, p_{i'})$) as positive. Different values ($S_{\leftrightsquigarrow}(p_i, p_{i'})$) instead do not contribute positive local similarity values and is therefore not considered. For all of the other values ($S_{\leftarrow}(p_i, p_{i'}), S_{\rightarrow}(p_i, p_{i'})$, and $S_{-}(p_i, p_{i'})$), weights $w_{\leftarrow}$, $w_{\rightarrow}$, and w_{-} decide whether we consider their relations as positive ($w = 1$) or negative ($w = 0$).

Definition 3 (Similarity Measure). *Assume* $w_{\leftarrow}, w_{\rightarrow}, w_{-} \in \{0, 1\}$.
$sim : P \times P \mapsto [0; 1]$,

$$sim(p_i, p_{i'}) := N^{-1} \cdot \Big(S_{\leftrightarrow}(p_i, p_{i'}) + w_{\leftarrow} \cdot S_{\leftarrow}(p_i, p_{i'}) + w_{\rightarrow} \cdot S_{\rightarrow}(p_i, p_{i'}) + w_{-} \cdot S_{-}(p_i, p_{i'}) \Big).$$

For example, if we suppose $w_{\leftarrow} = 0$ and $w_{\rightarrow} = 1$, we implement the following strategy for unknown values. Assume p_i is a problem component of a case whereas $p_{i'}$ is a problem component of a query. If a case specifies a value which is not part of the query, we assume that the known problem is more specific than the query. We either have a more general new problem or we have not yet tested the respective missing symptom for the query. In both cases, we do not increment the overall similarity. In contrast, if a case does not specify a value which is part of the query, we assume that the known problem is more general and hence covers all problems which are more specific. In these situations, we increment the overall similarity. If both values are unknown, w_{-} decides whether we count this coincidence as a positive or negative aspect of similarity.

4 Performance Measures

The previously described customer requirements reflect expectations of customers using a CBR system in any practical setting. For the following more precise considerations, we focus on diagnosis as the broad range of applications. We define four different performance measures that correspond to the customer requirements as we will discuss thereafter.

4.1 Coverage

The first performance measure is coverage. A set of cases covers a (query) case if and only if there exists a case within the set of cases that is at least as similar to the (query) case as a pre–defined similarity threshold τ. This similarity threshold τ corresponds to the minimum required similarity that CBR systems use to decide whether to suggest the solution of the most similar case as the solution of a query or not. For example, using a similarity threshold of 0.5 with a simple similarity measure basically counting matching values, this definition of coverage requires that at least half of the values of a case match the values of a query until we accept that this case covers this query.

Definition 4 (Coverage). *Assume* $q = (q_p, q_s) \in Q$, $t = (t_p, t_s) \in T$, $T' \subseteq T$, *and* $\tau > 0$.[2]

a) T' covers $q :\Longleftrightarrow \exists t \in T' : sim(t_p, q_p) \geq \tau$.
b) T' correctly covers $q :\Longleftrightarrow \exists t \in T' : sim(t_p, q_p) \geq \tau \ \wedge \ t_s = q_s$.[3]
c) The coverage set *of* T' *is* $V(T') := \{q \in Q : T' \text{ covers } q\}$.
d) The correct (or positive) coverage set *of* T' *is* $V^+(T') := \{q \in Q : T' \text{ correctly covers } q\}$.
e) The coverage *of* T' *is* $P_V(T') := |Q|^{-1} \cdot |V(T')|$.
f) The correct (or positive) coverage *of* T' *is* $P_V^+(T') := |Q|^{-1} \cdot |V^+(T')|$.

Pure coverage only requires a case with sufficient similarity; it does not necessarily expect that this case is also able to correctly classify the respective query. For the latter situation, we also define a notion of correct (or positive) coverage (see definition 4). For further discussions and specifications, we also define the coverage set of a set of cases and its correct (or positive) pendant. The (correct or positive) coverage as a performance measure is then the relative number of (correctly) covered cases in Q.

4.2 Accuracy

Accuracy is probably the most prominent performance measure that many researchers use to evaluate their approaches. Accuracy (or classification accuracy) counts the number of correct solutions of a CBR system using a case base to solve a set of queries. In classification domains, this number of solutions is compatible to the number of correct classifications.

First, we define when a case (correctly) classifies a query, namely, when this case is the most similar case in the case base in comparison to the query (and the solution components coincide). Furthermore, we specify the (correct or positive) classification set of a case (see definition 5). The accuracy is then the relative number of correctly classified cases in Q.

[2] q_p and t_p are the problem components of q and t, and q_s and t_s are the solution components of q and t, respectively. If it is clear that we mean q_p or t_p rather than q_s or t_s, we also use q and t instead of the more detailed notation q_p and t_p.

[3] Note, in real applications we do not know whether $t_s = q_s$ in advance. However, for experimental purposes, we assume a separated original case base into training and test cases (see above) such that we know the solution of queries beforehand.

Definition 5 (Accuracy). *Assume* $q = (q_p, q_s) \in Q$, $t = (t_p, t_s), t' = (t'_p, t'_s) \in T$, *and* $T' \subseteq T$.

a) $t \in T$ classifies $q \iff \nexists t' \in T, t \neq t' : sim(t'_p, q_p) > sim(t_p, q_p)$.[4]
b) $t \in T$ correctly classifies $q \iff t$ *classifies* $q \wedge t_s = q_s$.
c) T' *(correctly) classifies* $q \iff \exists t \in T' : t$ *(correctly) classifies* q.
d) The classification set *of* T' *is* $A(T') := \{q \in Q : \exists t' \in T' : t' \text{ classifies } q\}$.
e) The correct (or positive) classification set *of* T' *is* $A^+(T') := \{q \in Q : \exists t' \in T' : t' \text{ correctly classifies } q\}$.
f) The accuracy of T' *is* $P_A^+(T') := |Q|^{-1} \cdot |A^+(T')|$.

Note, the crucial difference between coverage and accuracy is the desired minimum similarity τ and the relation between solutions of cases and queries. For pure coverage, we demand a minimum similarity τ between cases and queries but do not state any constraints on solutions. For accuracy, we count any correct solution of the most similar case in comparison to a query regardless the exact value of similarity. Hence, coverage is stronger in a sense that it requires a minimum similarity but accuracy is stronger in a sense that it expects correct solutions.

4.3 Retrieval Time and Storage Space

The retrieval time and exact storage space that is needed to cope with the case base depends on the machine used for retrieval and the number of cases in the case base. Since we are not able to change machine characteristics by maintenance, we identify retrieval time with storage space which in turn corresponds to the number of cases for simplicity. Consequently, the respective performance measure P_T only counts the number of cases in the case base.

Definition 6 (Retrieval Time and Storage Space).
The retrieval time *and* storage space *(indicator) of* T *is* $P_T(T) := |T|$.

4.4 Confidence

Finally, the confidence of a CBR system in its decisions is a fourth performance measure. Thereby, we presume that observed similarities are appropriate as an indication of confidence. The higher the similarity of a case that classifies a query in comparison to the query is, the more confident we assume the CBR system is in its decision.

Definition 7 (Confidence). *Assume* $T' \subseteq T$, $t^*_q \in T'$ *is* the *case that classifies* $q \in Q$, *and* $sim^+(t^*_q, q) := sim(t^*_q, q)$ *if* t^*_q *correctly classifies* q *as well as* $sim^+(t^*_q, q) := 0$ *if* t^*_q *does not correctly classify* q.

a) The (average) confidence *of* T' *(on* Q*) is* $P_C(T) := |Q|^{-1} \cdot \sum_{q \in Q} sim(t^*_q, q)$.[5]

[4] In case of ties, i.e., two different cases have the same highest similarity in comparison to the query, we assume some order on T such that *the* case which classifies $q \in Q$ is always unique.
[5] We assume that T always classifies any $q \in Q$.

b) The (average) correct (or positive) confidence *of T' (on Q) is* $P_C^+(T) := |Q|^{-1} \cdot \sum_{q \in Q} sim^+(t_q^*, q)$.

Since we are barely interested in confidence of single decisions, we average confidence over all classifying cases and queries. For regular confidence, we count all similarities between classifying cases and the respective queries, whereas correct (or positive) confidence only takes into account similarities between cases and queries with correct classifications. Hence, we are able to distinguish between the average confidence of a CBR system in general, and the average confidence in its correct decisions. The related average wrong (or negative) confidence is simply the difference between average confidence and average correct (or positive) confidence.

4.5 Related Performance Measures

The most closely related work on similar performance measures is the research by Smyth and colleagues [7]. They define the local competence contribution of individual cases by two sets, the coverage set and the reachability set. The coverage set of a (training) case is the set of (query) cases that this case is able to solve, whereas the reachability set of a (query) case corresponds to the set of cases which are able to solve it. For both sets, Smyth and colleagues use a notion of 'solves'. If we assume that 'solves', for example, means in terms of our terminology $s_i = s_{i'}$ for a case $c_i = (p_i, s_i)$ and a (test) query $c_{i'} = (p_{i'}, s_{i'})$, and we additionally require that case and query have at least a similarity to each other of τ, then their coverage set definition is the same as $V^+(\{c_i\})$ in terms of our concepts.

The correct (or positive) classification set here is also comparable to the coverage set by Smyth et al. However, a classification set only considers queries that were really classified by a case rather than cumulating all cases that were potentially able to correctly classify the query. At the moment, we do not have any corresponding concept to Smyth et al.'s reachability set but it is easily possible to extend our definitions along these ideas and specify comparable sets as soon as we encounter a concrete need for them.

If we compare the definitions here and those of Smyth and colleagues, we observe that their notion of 'solves' is more general than the definitions for coverage and classification sets here, but on the other hand coverage, accuracy, and, especially, confidence cover more general aspects of performance than Symth et al.'s concepts do. Furthermore, their resulting maintenance strategies directly refer to their competence models whereas we only use these concepts to indirectly measure performance but rely on case and case base properties for maintenance purposes.

For more discussion of related work, we refer to earlier publications on case and case base properties (e.g., see [4,5]) and discussions in directly related papers (e.g., see discussions of related work in [6] and [7]).

5 General Case Properties

In previous work, we defined several case properties and used them to specify different quality measures [4]. These properties and measures were based on a simple similarity

Table 1. Examples for Pairs of Cases with Conflicts with respect to General Case Properties

	p_i	s_i	$p_{i'}$	$s_{i'}$	conflict	$\mathcal{S}_{\leftrightarrow}$	$\mathcal{S}_{\nleftrightarrow}$	$\mathcal{S}_{\leftarrow}$	$\mathcal{S}_{\rightarrow}$	$\mathcal{S}_{-}$	Δ
1	$v_{11}\ v_{21}\ v_{31}$	s_1	$v_{11}\ v_{21}$	s_2	$\neg$ sim–consistent	2	0	1	0	2	–
2	$v_{11}\ v_{21}\ v_{31}$	s_1	$v_{11}\ v_{21}\ v_{31}$	s_1	$\neg$ sim–unique	3	0	0	0	2	–
3	$v_{11}\ v_{21}\ v_{31}$	s_1	$v_{11}\ v_{21}$	s_1	$\neg$ sim–minimal	2	0	1	0	2	–
4	$v_{11}\ v_{21}\ v_{31}\ v_{41}$	s_1	$v_{11}\ v_{21}\ \ v_{42}\ v_{51}$	s_1	$\neg$ sim–incoherent$_2$	2	1	1	1	0	2

measure that only compares coincidence between values, i.e., local similarities yield 1 if two values of two cases (or a case and a query) for the same attribute are identical and 0 if they are not the same. The set–oriented notation for problems and solutions, which comprise cases, enabled definitions of properties and measures that mainly used set operations to compute the necessitated conditions and relations.

By now, we generalized these concepts and generated new definitions of general case properties and resulting quality measures that only use the general concept of a similarity measure rather than assuming a specific instantiation of such a measure. In this section, we cite these more general definitions which in their nature are comparable to the previous specifications. The original simple properties and measures are specializations of the general ones presented here. For detailed explanations of case properties and quality measures, we refer to [5].

Definition 8 (General Case Properties). *Assume* $G \subseteq C$, $c_i \in G$, *and* $1 \leq \Delta \in \mathbb{N}$.

a) c_i *sim–consistent within* $G \iff \nexists c_{i'} \in G : s_i \neq s_{i'} \wedge \mathcal{S}_{\leftrightarrow}(p_i, p_{i'}) + \mathcal{S}_{\leftarrow}(p_i, p_{i'}) = N_i \geq N_{i'} \wedge \mathcal{S}_{\leftrightarrow}(p_i, p_{i'}) > 0 \wedge \mathcal{S}_{\leftarrow}(p_i, p_{i'}) \geq 0 \wedge \mathcal{S}_{\rightarrow}(p_i, p_{i'}) = 0$.

b) c_i *sim–unique within* $G \iff \nexists c_{i'} \in G, c_{i'} \neq c_i : s_i = s_{i'} \wedge \mathcal{S}_{\leftrightarrow}(p_i, p_{i'}) = N_i = N_{i'} \wedge \mathcal{S}_{\leftrightarrow}(p_i, p_{i'}) > 0$.

c) c_i *sim–minimal within* $G \iff \nexists c_{i'} \in G : s_i = s_{i'} \wedge \mathcal{S}_{\leftrightarrow}(p_i, p_{i'}) + \mathcal{S}_{\leftarrow}(p_i, p_{i'}) = N_i > N_{i'} \wedge \mathcal{S}_{\leftrightarrow}(p_i, p_{i'}) > 0 \wedge \mathcal{S}_{\leftarrow}(p_i, p_{i'}) > 0 \wedge \mathcal{S}_{\rightarrow}(p_i, p_{i'}) = 0$.

d) c_i *sim–incoherent*$_\Delta$ *within* $G \iff \nexists c_{i'} \in G : s_i = s_{i'} \wedge \mathcal{S}_{\leftrightarrow}(p_i, p_{i'}) + \mathcal{S}_{\nleftrightarrow}(p_i, p_{i'}) + \mathcal{S}_{\leftarrow}(p_i, p_{i'}) = N_i = N_{i'} \wedge \mathcal{S}_{\leftrightarrow}(p_i, p_{i'}) > 0 \wedge \mathcal{S}_{\nleftrightarrow}(p_i, p_{i'}) \geq 0 \wedge \mathcal{S}_{\leftarrow}(p_i, p_{i'}) \geq 0 \wedge \mathcal{S}_{\rightarrow}(p_i, p_{i'}) \geq 0 \wedge \mathcal{S}_{\leftarrow}(p_i, p_{i'}) = \mathcal{S}_{\rightarrow}(p_i, p_{i'}) \wedge \mathcal{S}_{\nleftrightarrow}(p_i, p_{i'}) + \mathcal{S}_{\leftarrow}(p_i, p_{i'}) = \Delta$.

Table 1 shows examples of pairs of cases, their conflict to each other, and the resulting values for the auxiliary functions in definition 2.

6 On Relations between Customer Requirements, Performance Measures, and General Case Properties

The ultimate goal in any CBR application is to fulfill customer expectations. In section 2, we briefly characterized the probably most important customer requirements. However, these requirements and the corresponding performance measures rely on true

performance of the system, and an ad–hoc estimation of the degree of fulfillment of the requirements by a CBR system is not possible without using the CBR system in practice.

In this section, we first show how customer requirements, performance measures, and case properties relate to each other, and then argue that we are able to use case properties as early indicators for later performance, and hence for the degree of fulfillment of customer requirements by a CBR system.

6.1 Customer Requirements and Performance Measures

If we consider customer requirements, we observe the following correspondence to performance measures:

a) The first customer requirement corresponds to coverage. Customers want an answer to their query, and if a CBR system covers a query it is able to provide an answer, even if we demand a minimum similarity between classifying cases and queries as many CBR systems do.
b) The second customer requirement corresponds to correct coverage or accuracy. Obviously, customers want correct answers; if a CBR system correctly covers a query, it provides a solution that is correct, if the case base does not contain a more similar case in comparison to the query that results in an incorrect solution. Therefore, it is important to additionally measure accuracy. The higher the accuracy of the CBR system is, the more correct solutions it really provides, no matter if the similarity between cases and queries is high or low.
c) The third customer requirement corresponds to retrieval time, and hence to storage space. Customers want fast answers, and if retrieval time is short, answers are fast. There is always a trade–off between coverage, correctness, and speed. A customer is probably willing to wait longer for a correct answer rather than having a wrong solution fast.
d) The fourth customer requirement corresponds to confidence. Customers are interested in confident answers. For example, in a help–desk setting customers want to know how likely the provided solution is going to be the correct solution before they invest time and possibly money in realizing the suggested solution.

According to these relations, we are able to measure customer requirements and the degree of fulfillment for any specific CBR system by analyzing the respective performance measures. However, these performance measures rely on characteristics of retrieval and an a–priori knowledge on expected queries. Retrieval is costly, and a–priori knowledge on expected queries is usually not available in real-world applications.

Therefore, we aim at other criteria that we are able to measure without using the CBR system and testing its retrieval results, and without knowing anything about expected queries in advance. In the following, we argue that it is possible to approximate some expectations on relative performance (with respect to the previously defined performance measures) by analyzing the case properties within a case base. In addition, we also aim at rules that indicate how the performance measures change over time when we apply restore operations to modify cases in the case base for maintenance purposes.

6.2 Performance Measures and Case Properties

Ideally, we want theoretical relations between performance measures and case properties. For example, if we take into account coverage as the performance measure, uniqueness as the case property, and assume a 1–nearest–neighbor algorithm for retrieval, we theoretically know that conflicts that violate uniqueness do not influence coverage. If there exists a case that covers a query, it does not matter if this case exists more than once, and similarly, if there does not exist any case that covers a query, the situation does not change if we have cases multiple times. Likewise, it does not make any difference in terms of coverage if we utilize restore operations to remove conflicts that contradict uniqueness.

However, as extensive theoretical analyzes that we conducted show, relations between performance measures and case properties are not always that clear, and moreover these relations are usually not independent of other assumptions, for example, on similarities between cases and queries. In order to exemplify the relatively complex theoretical relation between performance measures and case properties, we again consider coverage.

Performance Measures and Similarity First of all, if we take coverage, when does coverage change? The coverage of a set T' of cases changes if the number of queries that T' covers changes. This number again changes if either T' after modification by maintenance operators does not cover a query q anymore, or if T' then additionally covers an extra query q' which it did not cover before. In terms of the definition of coverage, we reveal the following conditions:

(i) $P_V(T) > P_V(T')$
$\iff \exists q \in Q : \exists t \in T : sim(t_p, q_p) \geq \tau \ \wedge \ \nexists t' \in T' : sim(t'_p, q_p) \geq \tau$
$\iff \exists q \in Q : \exists t \in T : sim(t_p, q_p) \geq \tau \ \wedge \ \forall t' \in T' : sim(t'_p, q_p) < \tau.$

(ii) $P_V(T) < P_V(T')$
$\iff \exists q \in Q : \nexists t \in T : sim(t_p, q_p) \geq \tau \ \wedge \ \exists t' \in T' : sim(t'_p, q_p) \geq \tau$
$\iff \exists q \in Q : \forall t \in T : sim(t_p, q_p) < \tau \ \wedge \ \exists t' \in T' : sim(t'_p, q_p) \geq \tau.$

Both conditions mean if we modify a case by any restore operator and this modification in turn changes the coverage of the case base, then the modified case *before* its modification is the only case which covers q for the first condition (if it is not, there exists still another case which covers q), and the modified case *after* its modification is the only case which covers q for the second condition, respectively. We further assume that one of these conditions is true subsequently. [6]

If the modified case is the only case that is responsible for changes of coverage, this change is only possible if modification changes the similarity between the case and the respective query; for the first condition, this modification must decrease this similarity, for the second condition, it must increase the similarity.

[6] Otherwise, relations have to consider more cases and the relation of similarities between those cases, the modified case, and queries.

Performance Measures and Restore Operators If we now assume that we are only interested in positive effects of maintenance in terms of the performance measures, we are able to restrict further analyzes to the second condition. When does modification increase similarity between a case and a query? For modification, we consider restore operators `remove`, `specialize`, `generalize`, `cross`, and `join` [5]. We neither take into account `adjust` and `alter` since both operators are only concatenations of `specialize` and `generalize`, nor `abstract` and `combine` since these two operators state additional assumptions on representation and retrieval which we can not expect in general diagnosis applications. The following definition 9 recapitulates the restore operators which we consider here from earlier publications for completeness.

Definition 9 (Restore Operators). *Assume $C^{\subseteq}$ is the set of all subsets of C, $G \subseteq C$, and $c_i = (p_i, s_i), c_{i'} = (p_{i'}, s_{i'}) \in G$.*

a) `remove`*: $C^{\subseteq} \times C \mapsto C^{\subseteq}$,* $\texttt{remove}(G, c_i) := G \setminus \{c_i\}$
b) Assume $p_i \cap V_j = \emptyset$ and $v_{jk} \in V_j$.
`specialize`*: $C \times V \mapsto C$,* $\texttt{specialize}(c_i, v_{jk}) := (p_i \cup \{v_{jk}\}, s_i)$
c) Assume $p_i \cap V_j = \{v_{jk}\}$.
`generalize`*: $C \times V \mapsto C$,* $\texttt{generalize}(c_i, v_{jk}) := (p_i \setminus \{v_{jk}\}, s_i)$
d) Assume $1 \leq \Delta \in \mathbb{N}$, $|p_i \cap p_{i'}| + \Delta = N_i = N_{i'}$ or $p_i \subsetneq p_{i'}$ or $p_{i'} \subsetneq p_i$, and $s_i = s_{i'}$.
`cross`*: $C \times C \mapsto C$,* $\texttt{cross}(c_i, c_{i'}) := (p_i \cap p_{i'}, s_i)$
e) Assume $1 \leq \Delta \in \mathbb{N}$, $|p_i \cap p_{i'}| + \Delta = N_i = N_{i'}$ and $\forall a_j \in A : |[(p_i \cup p_{i'}) \setminus (p_i \cap p_{i'})] \cap V_j| \leq 1$ or $p_i \subsetneq p_{i'}$ or $p_{i'} \subsetneq p_i$, and $s_i = s_{i'}$.
`join`*: $C \times C \mapsto C$,* $\texttt{join}(c_i, c_{i'}) := (p_i \cup p_{i'}, s_i)$

The `remove` operator is certainly the strongest operation that maintenance allows. However, `remove` does not directly influence coverage in terms of changing similarities between the modified case and queries, since the modified case no longer exists after its removal. Nonetheless, simple analyzes show that removal of a case always results in the same or lower coverage but it does not positively influence coverage at all.

Now, assume two cases $t, t' \in T$, a query $q \in Q$, and $w_{\leftarrow} = 0$, $w_{\rightarrow} = 1$, and $w_{-} = 1$ for the definition of similarity measure sim (see definition 3). If we consider `specialize`, `generalize`, `cross`, and `join`, we observe the following characteristics:

- $sim(\texttt{specialize}(t, v_{jk}), q) = sim(t, q)$ if $V_j \cap q = v_{jk}$,[7]
- $sim(\texttt{specialize}(t, v_{jk}), q) = sim(t, q) - N^{-1}$ if $V_j \cap q = v_{jk'} \neq v_{jk}$, and
- $sim(\texttt{specialize}(t, v_{jk}), q) = sim(t, q) - N^{-1}$ if $V_j \cap q = \emptyset$.

  All in all, $sim(\texttt{specialize}(t, v_{jk}), q) \leq sim(t, q)$.
- $sim(\texttt{generalize}(t, v_{jk}), q) = sim(t, q)$ if $V_j \cap q = v_{jk}$,
- $sim(\texttt{generalize}(t, v_{jk}), q) = sim(t, q) + N^{-1}$ if $V_j \cap q = v_{jk'} \neq v_{jk}$, and
- $sim(\texttt{generalize}(t, v_{jk}), q) = sim(t, q) + N^{-1}$ if $V_j \cap q = \emptyset$.

  All in all, $sim(\texttt{generalize}(t, v_{jk}), q) \geq sim(t, q)$.
- $sim(\texttt{cross}(t, t'), q) = sim(t, q) = sim(t', q)$ if $t = t'$, and

[7] For simplicity, we identify sets with only a single element with this element, i.e., $\{x\} = x$.

- $sim(\texttt{cross}(t, t'), q) = sim(t, q)$ if $t \subseteq t'$ (similar for t').
  Other general relations are not possible since similarity can increase and decrease depending on the specific relation between values of cases and query.
- $sim(\texttt{join}(t, t'), q) = sim(t, q) = sim(t', q)$ if $t = t'$, and
- $sim(\texttt{join}(t, t'), q) = sim(t', q)$ if $t \subseteq t'$ (similar for t).
  All in all, $sim(\texttt{join}(t, t'), q) \leq sim(t, q)$ and $sim(\texttt{join}(t, t'), q) \leq sim(t', q)$.

In conclusion, if we are interested in positive effects on coverage by maintenance using the specified restore operations, we conclude that the only operator which ensures — without additional assumptions — that the similarity does not decrease is `generalize`. Hence, if we want to positively influence coverage by maintenance, it is wise to test `generalize` as the first modification operator.

6.3 Further Analyzes

For all performance measures beyond coverage which we used as an example here, similar analyzes are possible, and the results of such analyzes are comparable to those presented here. In some situations, it is possible to derive general theoretical relations between case properties, the resulting conflicts between cases, restore operators, and performance measures. In other situations, it is necessary to state additional assumptions, for example, on similarities between cases and queries, to make the analyzes tractable. We are currently elaborating on such investigations as part of our overall research on maintenance.

An extra type of analysis which we have not yet extensively pushed forward considers relations between the specific type of conflict and changes of the performance measures if we conduct maintenance using restore operators to modify affected cases. Again, initial results on these analyzes show that, in some situations, it is possible to derive general heuristics in case of specific conflicts which type of operator is preferable, and which of the affected cases to modify first, in order to get best results in terms of performance. However, in other situations, again additional assumptions are necessary to make these theoretical analyzes feasible.

7 Experimental Evaluation

Up to this point, the theoretical analyzes show that it is hardly possible to infer general relations between case properties and performance measures for any type of situation without additional assumptions. In order to further check if conflicts indicate not only quality problems within the case base but also indirectly demonstrate that performance measures currently do not yield optimal values, and to see if it is possible to positively manipulate performance values by modifying cases with restore operations when we observe violated case properties, we now turn to an experimental evaluation.

7.1 Experimental Set–Up

Figure 1 outlines the experimental procedure. For each of the ten different data sets from the UCI repository, we initially separate the original data set into five folds of 20

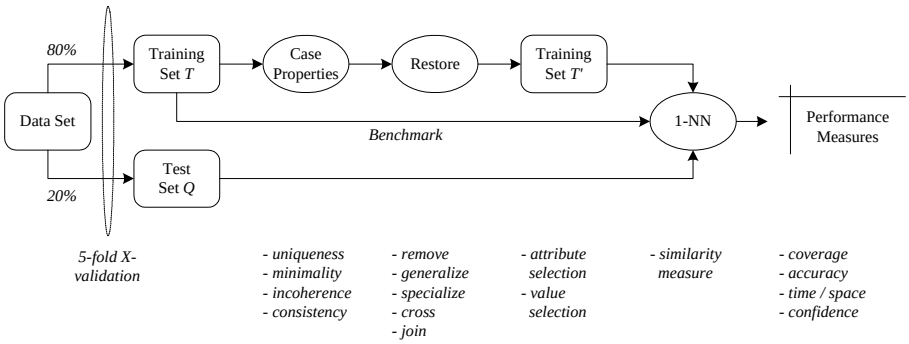

Fig. 1. Experimental Procedure

percent of the entire data each. Thereafter, we split the data into a training set T that consists of four folds and a test set Q that contains the remaining fifth fold. We apply the case properties uniqueness, minimality, incoherence, and consistency to detect conflicts between cases which violate these properties. Then, we utilize the restore operators `remove`, `specialize`, `generalize`, `cross`, and `join` separately to maintain the case base by modifying cases in order to eliminate conflicts. The result of the restore step forms the modified training set T'.

Now, we apply a simple 1–nearest–neighbor algorithm as the CBR mechanism for retrieval and classification with similarity measure sim with $w_{\leftarrow} = 0$, $w_{\rightarrow} = 1$, $w_{-} = 1$, and $\epsilon = 0.05$ for local similarity between numeric attributes. Thereafter, we compute the four performance measures $P_V(T')$ with $\tau = 0.75$, $P_A^+(T')$, $P_T(T')$, and $P_C(T')$ and their positive pendants as the overall results of our experiment. The benchmark for comparing these results is the application of the same 1–nearest–neighbor algorithm and computation of the performance measures using the original training set T. We repeat the entire experimental procedure five times, using each of the five folds as the test set once; hence, we implement a 5–fold cross validation experimental design.

For operators `generalize` and `specialize`, we conduct the following strategy to select attributes and values for manipulation. We assess all attributes with information gain ratio as if we were estimating the quality of an attribute for creating the first split in building a decision tree [3]. We then choose the least important attribute according to this selection criterion as the attribute for which we eliminate the value in case of `generalize`, or identify the most important attribute for which we add a value for `specialize`. For the latter operator, we additionally have to decide which value forms the new value for the selected attribute. Therefore, we assume that the most often occurring value for the respective attribute for cases with the same class is the most promising value for our purposes.

7.2 Experimental Results

Table 2 shows experimental results for operator `generalize` in order to eliminate conflicts that violate one of the case properties uniqueness, minimality, incoherence,

Table 2. Experimental Results

Data Set	P_V	P_V^+	P_A^+	P_C	P_C^+
australian	94.6(94.5)	92.0(91.9)	89.4(88.6)	85.9(85.5)	76.7(75.6)
breast-cancer	94.8(94.4)	88.8(88.5)	81.8(80.8)	86.4(85.8)	70.4(69.1)
bridges-version1-ms	81.3(80.3)	60.5(59.5)	66.8(66.8)	83.6(83.5)	55.5(55.4)
bupa	76.5(72.5)	70.1(65.2)	83.5(83.8)	83.9(81.2)	69.0(67.1)
hayes-roth	68.1(68.1)	65.0(55.9)	93.9(84.8)	73.5(73.5)	69.2(61.9)
hepatitis	98.1(98.1)	87.1(87.1)	75.5(75.5)	88.6(88.6)	67.0(67.0)
ionosphere	38.7(38.7)	38.7(38.7)	92.9(92.9)	70.7(70.7)	66.9(66.9)
pima-indians-diabetes	78.9(76.2)	69.9(65.7)	81.9(81.5)	76.7(75.4)	62.5(61.1)
processed-hungarian	94.2(93.5)	89.1(88.1)	86.4(86.1)	83.4(83.2)	72.0(71.6)
voting-records	100.0(100.0)	99.5(99.5)	91.0(91.2)	98.5(98.5)	89.6(89.9)

and consistency in their general similarity–based definition. The table lists five different performance criteria: Coverage P_V, correct coverage P_V^+, accuracy P_A^+, confidence P_C, and correct confidence P_C^+. Left–hand of each column, we see results for the modified case base T' after maintenance, whereas right–hand in brackets we observe the respective benchmark results for the entire training set T before maintenance. Note, we do not report on P_T in this table, since we only list results for operator `generalize` which does not modify the case base size at all.[8]

For coverage, we perceive that the experimental results exactly fit our expectations resulting from the theoretical analysis. Coverage as well as correct coverage remain constant or increase if we apply `generalize` as the maintenance operator. This is coherent with the theoretical analysis that similarities between generalized cases and queries do not decrease.

For accuracy, maintenance has a positive effect on five data sets, a neutral relation holds for three data sets, and we see slightly worse results on the remaining two data sets. Hence, we also infer that maintenance using `generalize` for cases which violate the defined case properties yields promising results in terms of the second performance criterion.

Finally, for confidence, we notice that values again remain constant or increase as for coverage except for data set voting records and correct confidence. It is interesting to see that in most cases it looks like accuracy and confidence are highly correlated, i.e., both performance measures vary in comparable orders of magnitude. Hence, we tend to infer that confidence increases when accuracy does; the more correct classifications a CBR system provides, the more confident it is in its decisions.

All in all, we conclude that the theoretical analyzes give correct hints which operator to test first for maintenance and optimization of performance, and that the suggested general case properties and restore operators are a promising instrument to maintain CBR systems in practical applications when we aim at optimization of performance criteria that correspond to customer requirements.

[8] We refer to [2] for some results on P_T and P_A^+ using operator `remove`.

8 Conclusions

In this paper, we presented extensions of our research on case base maintenance. We discussed several customer requirements and defined different performance criteria that are equivalent computable measures for these requirements. Since we are not able to estimate specific values of the performance criteria for any CBR system in advance, and we are also not able to predict changes in performance if we maintain the case base using restore operations, we argued that it is possible to estimate values and changes of performance by considering case properties and their changes after maintenance operations on the case base.

Theoretical analyzes showed that it is hardly possible to derive general rules for the relations between customer requirements, performance criteria, and case properties without making additional assumptions — for example, on the specific similarity measure used for retrieval — to keep the analyzes feasible. However, initial experimental results show that application of restore operators indeed positively influences the performance measures, and that the theoretical analyzes provide first hints which operator is likely to perform best in terms of increasing performance values.

Future research aims at extensions of both, the theoretical analyzes and the experimental studies. The whole research currently offers many parameters to vary, and additional calibrations for all methods seem to have the potential for further examinations. For example, we plan to vary the used similarity measure and the various thresholds such as τ and ϵ, as well as to conduct further experiments with combinations of different restore operators. We strongly believe that we are able to derive more heuristics for appropriate applications of the right restore operators depending on the type of conflicts and the desired effect in performance.

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. Ioannis Iglezakis. The conflict graph for maintaining case–based reasoning systems. In *Proceedings of the 4th International Conference on Case–Based Reasoning*, pages 263–275. Springer–Verlag, 2001.
3. J. Ross Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.
4. Thomas Reinartz, Ioannis Iglezakis, and Thomas Roth–Berghofer. On quality measures for case base maintenance. In *Proceedings of the 5th European Workshop on Case–Based Reasoning*, pages 247–259. Springer–Verlag, 2000.
5. Thomas Reinartz, Ioannis Iglezakis, and Thomas Roth–Berghofer. Review and restore for case–base maintenance. *Computational Intelligence: special issue on maintaining CBR systems*, 17(2):214–234, 2001.
6. Barry Smyth and Elizabeth McKenna. Competence guided incremental footprint-based retrieval. *Knowledge–Based Systems*, 14(3):155–161, 2001.
7. Barry Smyth and Elizabeth McKenna. Competence models and the maintenance problem. *Computational Intelligence: special issue on maintaining CBR systems*, 17(2):235–249, 2001.