Gupta K .M.(2001), Case-Based Reasoning Research and development, Proceedings of the fourth ICCBR ,D. W. Aha & I. Watson (Eds.), Springer, Berlin, Germany, pp. 219-233.

Taxonomic Conversational Case-Based Reasoning

Kalyan Moy Gupta[1,2]

[1]ITT Industries, AES Division, Alexandria, VA 22303
[2]Navy Center for Applied Research in Artificial Intelligence,
Naval Research Laboratory, Washington, DC 20375
gupta@aic.nrl.navy.mil

Abstract. Conversational Case-Based Reasoning (CCBR) systems engage a user in a series of questions and answers to retrieve cases that solve his/her current problem. Help-desk and interactive troubleshooting systems are among the most popular implementations of the CCBR methodology. As in traditional CBR systems, features in a CCBR system can be expressed at varying levels of abstraction. In this paper, we identify the sources of abstraction and argue that they are uncontrollable in applications typically targeted by CCBR systems. We contend that ignoring abstraction in CCBR can cause representational inconsistencies, adversely affect retrieval and conversation performance, and lead to case indexing and maintenance problems. We propose an integrated methodology called *Taxonomic CCBR* that uses feature taxonomies for handling abstraction to correct these problems. We describe the benefits and limitations of our approach and examine issues for future research.

1 Introduction

Case-Based Reasoning (CBR) systems support problem-solving by recalling and applying those experiences or cases that are similar to the problem at hand (Kolodner, 1993). A wide range of applications has been developed using CBR methodology. By far, the most frequently developed real-world applications of CBR include help-desk, interactive troubleshooting, and equipment maintenance systems. These systems engage a user in a series of questions and answers to retrieve cases that solve his/her problem. They have been referred to in the literature as *Conversational CBR* (CCBR) systems (Aha *et al.*, 2001).

A pervasive issue in CCBR case bases, and case bases in general, is that the case contents and their features may be expressed at different levels of abstraction (Baudin & Waterman, 1998; Kolodner, 1993; Shimazu, 1998). For example, a feature in a printer troubleshooting application could be expressed generally as "My printer is showing an error message" or more specifically as "My printer is showing a paper out error." Ignoring abstraction can create unique problems for end users, case base developers, and the CCBR system.

In this paper, we note that the existing CCBR approaches only partially address abstraction. We propose that CCBR systems should be designed with an integrated approach for supporting abstraction. The remainder of the paper is organized as follows. In Section 2, we identify the problems that result when abstraction is ignored in CCBR. In Section 3, we develop an integrated methodology called *Taxonomic CCBR* to overcome these problems. Section 4 summarizes the benefits and limitations

of our approach. Section 5 presents related work on CCBR systems. Section 6 concludes the paper.

2 Abstraction in CCBR Systems

Features of a CCBR system may be expressed at different levels of abstraction. For example, a problem in the domain where weather is a factor in decision-making (e.g., a Noncombatant Evacuation Operation Planning (DoD, 1997)) may record weather conditions at the following levels of abstraction:

(1.) The weather was bad
  (1.1) The weather was stormy
    (1.1.1) The wind speed was very high
      (1.1.1.1) The wind speed was over 90 mi./hr.

Abstraction affects interactive decision support systems in two ways: (1) The communication between a human user and the system becomes problematic (Furnas *et al*., 1987), and (2) the design, development, and maintenance of systems for optimal performance becomes a problem (Pedrycz & Vukovich, 2000). CBR systems are prone to the same problems.

In general, CBR systems have dealt with abstraction in a limited way (Kolodner, 1993; See for example Alterman (1986)). For traditional CBR systems that are highly structured and do not engage in a conversation, the communication between human user and the system is usually not a problem. In these systems, the approach has been to design abstraction into them. For example, Bergmann and Wilke (1996) and Branting and Aha (1995) have used cases and features at predefined levels of abstraction to reduce the representational complexity and to improve the retrieval efficiency of case bases. Likewise, Drastal and Czako (1989) compute an abstract representational feature space for inductive learning to improve the learning and classification accuracy of their system. These systems are restricted to well-defined application domains such as hierarchical planning and do not require conversation. The assumption that system developers can define feature spaces at suitable levels of abstraction for optimal system performance is unrealistic for a highly dynamic CCBR application. In CCBR, not only can case features occur at multiple levels of abstraction, but neither the feature set nor their levels of abstraction can be determined with certainty in advance and over the life-cycle of the case base. The problem is that the domain is often both ill structured and dynamic (i.e., features and cases continue to be added over the life cycle of the case base). In addition, end users have different backgrounds and degrees of expertise than case authors. In the following subsections, we examine the sources of abstraction and the ways in which it affects CCBR systems.

2.1 Sources of Abstraction

We identify the following sources of abstraction:

1. *Variations in the level of domain expertise between users and developers*: The level of domain expertise is the dominant factor in a user's ability to describe and formulate problems (i.e., specify features and their values). Experts can be very

precise, complete, or very abstract in their description, whereas non-experts' description are usually imprecise, incomplete, or ambiguous (Arocha & Patel, 1995). Consequently, there can be significant differences in the levels of abstraction between users and developers.

2. *Variations in information availability and the cost of its acquisition*: Lack of information or the expense of acquiring it at the desired level of abstraction (i.e., detail) can limit a user's ability to provide it. For example, in the description of a weather condition described above, the wind speed information may not be available or it may be too expensive to acquire because it may require a setup of measuring instruments. This is particularly true of customer support and troubleshooting applications where information may be unavailable or may only become available in the future. Variations in information availability and the cost of acquisition affect the level of abstraction.
3. *Variations in decision-making needs*: The most appropriate level of abstraction depends on user's decision-making and problem solving needs (Rosch, 1978). Often the information is available at a higher level of precision (i.e., lower level of abstraction), however, information at a lower level of precision (i.e., higher level of abstraction) may be sufficient. Variations and differences in decision-making needs affect the appropriate level of abstraction.

These sources are to a large extent uncontrollable and CCBR systems must be capable of tolerating their effects. Ignoring them can result in problems discussed in the following section.

2.2 Problems Due to Ignoring Abstraction

Abstraction has not been addressed adequately in CCBR systems. However, the issue of vocabulary differences between end users and case authors has been recognized (Shimazu, 1999). Such differences adversely impact the user system communication and limit the effective use of a CCBR system. Still, the connection between abstraction as a source of vocabulary differences and its adverse impact on CCBR performance has not been established. In a CCBR system, abstract features are frequently used to classify a problem into a case (Trott & Leng, 1997). However, a common over simplification is mixing the abstract features with the more specific case features into a single level (e.g., a list of question answer pairs, (Aha *et al.*, 1998)). This leads to the following problems:

1. *Unwanted correlation among features*: Placing abstract and specific features in the same level introduces significant unwanted correlation among them. This can be problematic for nearest-neighbor matching functions that assume independent uncorrelated features. Often, artful weighting of questions is required to address such feature correlation (Trott & Leng, 1997).
2. *Limited ability to assess similarity*: Ignoring abstract relations between features is in effect ignoring their similarity. In systems that ignore abstraction, two cases describing the same problem using different features related by abstraction cannot be assessed as similar. This leads to redundancy and inconsistency among cases (Everett & Bobrow, 2000; Racine &Yang, 2001).
3. *Redundant questions are generated during conversation*: Depending on the user's approach to problem description, redundant and irrelevant questions are presented

by the conversation algorithm. Often, this is a result of ignoring the abstract relations between questions. Background knowledge in the form of rules and models can be used to answer other questions and overcome the problem (Aha *et al.*, 1998). Question selection algorithms using frequency or information theoretic measures are likely to be misled by abstraction (e.g., Aha et al., 1998, Yang & Wu, 2001), as are those that use probabilistic or belief net strategies (e.g., Montazemi & Gupta, 1996; McSherry, 2001).

4. *Loss of decisional information due to feature generalization*: When a single level of feature abstraction is encoded, often abstract features are retained over specific features to improve the applicability of a case to new problems. This can be problematic because discarding specific features during indexing can cause loss of vital information needed for discriminating cases (Kolodner, 1993).
5. *Difficulty of assigning indices*: Not unlike authors of scientific publications who assign keywords to their paper, case authors face a difficult problem of indexing, without the benefit of a predefined question list. A case author often tends to index a case using multiple features that express the same semantic feature at different levels of abstraction. This is done to improve the recall and to manage conversation (Trott & Leng, 1997).
6. *Inconsistencies develop in case representation when new features are added:* Introducing new generalized features without the ability to accommodate abstraction relationships among them can yield situations where related cases are expressed and represented with a combination of features that are conflicting or at inconsistent levels of abstraction.

In the following section, we propose an integrated methodology to handle abstraction in CCBR systems. To this end, we recognize abstract relations between features and explicitly structure them into taxonomies. We exploit these taxonomies to develop a new case representation scheme. We call a CCBR system that incorporates this methodology as a *Taxonomic CCBR* system.

3 Proposed Taxonomic CCBR System

3.1 Case Representation with Feature Taxonomies

CCBR tools that support problem solving typically use a case representation structure that includes a problem description and a solution. The problem is described by a set of question-answer (i.e., feature and a value) pair, which is the basis of case retrieval (Aha *et al.* 1998; Gupta, 1998). For example, a printer troubleshooting CCBR application may have the following question answer pair: "Do you have a print quality problem?=*Yes.*"

We assume that the Taxonomic CCBR includes the following:

1. A *set of questions* $\boldsymbol{Q}$ that are used for indexing cases**.** The i^{th} member of $\boldsymbol{Q}$ is denoted by q_i ($0 < i ? n$, where n = number of questions in the case base) and has a set of answers $\boldsymbol{a}_i$ applicable to it. The j^{th} member of $\boldsymbol{a}_i$ is denoted by $a_{i,j}$ ($0 < j ? m_i$, where m_i = number of possible answers for q_i). We denote a specific question answer pair (q_i, $a_{i,j}$) by $qa_{i,j}$. We denote the set of all possible question answer pairs by $\boldsymbol{QA}$. For example, in a printer troubleshooting CCBR application, a question in

the set ***Q*** regarding the nature of print quality problems could be "What does the print quality look like?" with "*Black Streaks"*, "*Faded*", and "*No Problem*" as potential answers. For simplicity, we consider only binary and nominal valued questions. Our experience shows that, in CCBR applications, a disproportionately large number of questions are binary and nominal valued (Gupta, 1998).

2. *Feature Taxonomies*: We define a feature taxonomy ***T*** to be an acyclic directed graph comprising nodes t_j ($0 < j$? l, where l = number of nodes in the taxonomy). A node t_j in a taxonomy includes a question-answer pair drawn from the set ***QA***. It is related to a set of parent nodes ? $_j$. The relationships between node t_j and its parents are either of type *is-a-type-of* or *is-a-part-of*. If the set ? $_j$ is empty t_j is the *root node* of the taxonomy. The node t_j is also related to a set of child nodes denoted by ? $_i$. If the set ? $_i$ is empty, t_j is *a leaf node* in ***T.*** The relationships between nodes are transitive. If a node t_i is a ancestor of t_j we denote this relationship by t_i = ? (t_j). Also, an ancestor node is said to *subsume* all its descendent nodes. Figure 1 shows a subset of the taxonomy from a CCBR application for printer troubleshooting. In this figure, the node t_2 representing question-answer pair "Print quality problem?=*Yes*" has one parent node t_1 and two child nodes t_5 and t_6 .

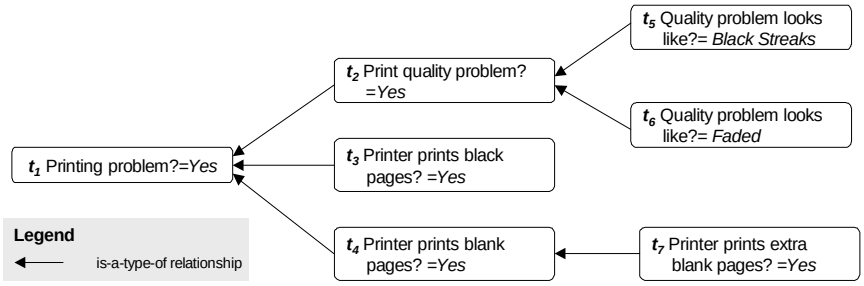

Figure 1. Subset of a feature taxonomy from a printer troubleshooting application

A Taxonomic CCBR includes a set of taxonomies each representing a family of question-answer pairs interrelated by abstract relations.

3. *A set of cases* ***C***: We define a case C_k ($0 < k$? r, where r = number of cases in the case base) to include the following:
   a. *Problem State*: This is a set of question-answer pairs $\boldsymbol{P}_k$**.** The members of $\boldsymbol{P}_k$ are drawn from the set of question-answer pairs ***QA*** in the case base and are denoted by $p_{k,i}$ ($0 < i$? r_k, where r_k = number of question answer pairs in $\boldsymbol{P}_k$). We apply the following representational rules to the question-answer pairs in $p_{k,i}$:
      i. **Only one question-answer pair from a taxonomy can be included in a case**: No two question-answer pairs in a case refer to the same question and no two pairs are related by an abstract relation. This representational rule eliminates redundant indexing and correlation among features in a case. For example, referring to the Figure 1, a case cannot simultaneously include question answer pairs t_2 and t_6, but may include either t_2 or t_6 .
      ii. **The most specific available and applicable question-answer pair is used to represent the case**: This representational rule is based on the assumption that specific question-answer pairs are more likely to include the necessary information required to discriminate cases. For example,

referring to Figure 1, if t_6 applies to the case and is known at the time of indexing the case, t_6 should be included in it.

b. *Solution*: We denote a solution in case C_k by S_k. The solution includes a sequence of actions that corrects the problem described by the problem state.

Table 1 compares the Taxonomic CCBR case representation with a current approach (e.g., Aha *et al.*, 1998). Taxonomic CCBR eliminates redundant indexing and reduces the number of features used for representing the case. Further, the textual problem description in cases is eliminated by our search technique presented in Section 3.2.

Table 1. Comparison of the Taxonomic CCBR case representation with a typical CCBR case representation for a printer troubleshooting application

Case Representation	Example Text	Taxonomic CCBR	CCBR
Case Title	Ink cartridge is damaged causing black streaks.	Included	Included
Text Description	Vertical black streaks or smears appear on successive pages	***Excluded***	Included
qa pairs	Do you have a print quality problem? = *Yes*	***Excluded***	Included
	What does the print quality look like? = *Black streaks*	Included	Included
	Does cleaning the printer with cleaning paper remove problem? = *No*	Included	Included
Solution	Check toner cartridge and replace if it is low in toner or damaged. For toner level, check the indicator on the left side of the cartridge.	Included	Included

For simplicity, we ignore feature weighting that is typically included in CCBR systems.

3.2 Taxonomic CCBR Processes

A problem solving session with a CCBR system proceeds as follows (Aha *et al.*, 1998; Gupta, 1999) (See Figure 2). The user *describes* a problem with a short textual description. Next, the system *retrieves* cases by *search*ing, *match*ing, and *rank*ing (Gupta & Montazemi, 1997). The user and the system engage in a *conversation* where the system selects, rank orders, and presents questions to the user and the user refines his/her problem description by selecting and answering questions from those presented by the system. The conversation and retrieval iterate until the user finds a case that solves his/her problem or determines that no existing case solves his/her problem. Depending on the situation the user may select a case and apply its solution or trigger new case acquisition.

Consider a user query description *Q*. It includes the following:

1. *Textual problem description (QT)*: Search for potentially relevant cases is initiated by the textual problem description.
2. *Problem description represented by a set of question answer pairs* ***QP***. The members of ***QP*** are drawn from the set of question answer pairs ***QA*** in the case base and are denoted by qp_i $(0 < i ? w_k$, where w_k = number of question answer pairs in ***QP***).

Using this notation, we present our methodologies for case retrieval, conversation, and acquisition.

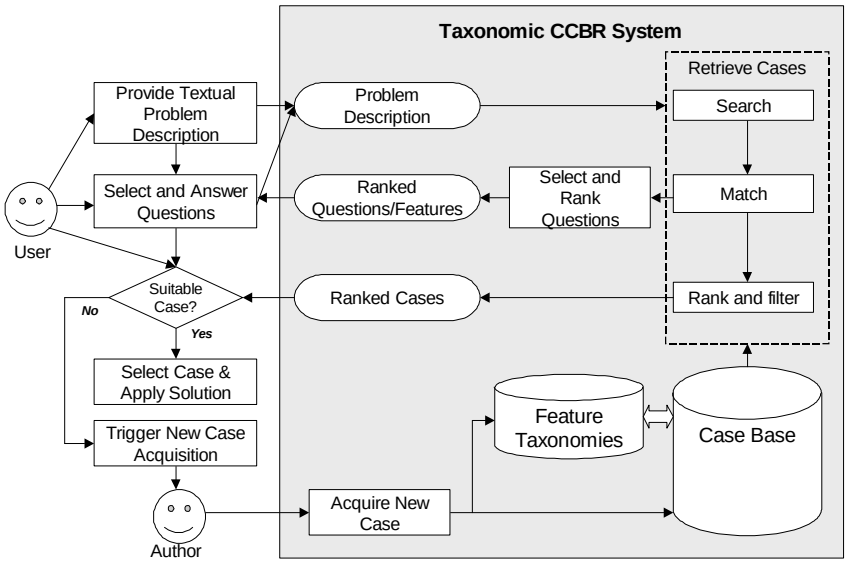

Figure 2. User interactions with a Taxonomic CCBR system

Case Retrieval. Case retrieval includes searching, matching, ranking and selection steps.

Step I-Search

Search for potentially relevant cases is initiated by *QT*. The search includes the following steps:

1. *Question-answer pair identification*: The system matches the textual problem description *QT* with question answer pairs in $\boldsymbol{QA}$ to identify the question answer pair that is most similar to the users textual description. One of the many available string and text matching algorithms can be used for this purpose (e.g., using n-grams). We denote this initially identified question-answer pair as qp_1. This step is only performed once in a problem solving session and it eliminates the need for a text problem description in a case.
2. *Search scope expansion by taxonomy traversal*: For each member qp_i in $\boldsymbol{QP}$ traverse its corresponding taxonomy $\boldsymbol{T_i}$ to identify all its descendants. This step expands the search scope.
3. *Candidate cases selection*: For all the descendant nodes identify their associated cases. An inverted index could be used for this purpose or the cases can be indexed directly on the taxonomies. We denote the candidate set of cases for a query *Q* by $\boldsymbol{C_Q}$.

Step II-Match

Matching a user query *Q* with each candidate case in $\boldsymbol{C_Q}$ establishes its rank. It involves assessing similarity between the set of question-answer pairs in $\boldsymbol{QP}$ and $\boldsymbol{P_k}$ (i.e., the question-answer pairs of the k^{th} candidate case). For each candidate case, the matching takes place in two steps:

1. *Question answer pair similarity assessment*: This involves considering the user's query level by reference to taxonomies. If the user's question-answer pair is an ancestor of a question-answer pair in the case, it matches with a score of 1**.** This is based on the assumption that there is no information loss when moving from general to a more specific question-answer pair. However, when the user's question-answer pair is a descendant of a question-answer pair in the case, the similarity is less than 1. This is based on the assumption that moving from specific question-answer pair to a general one causes information loss. Formally, we denote the similarity of a question-answer pair qp_i in ***QP*** and question-answer pair $p_{k,i}$ in candidate case C_k by $sim_k(qp_i, p_{k,j})$. We compute similarity between two nodes in a taxonomy as follows:

if $qp_i = ?(p_{k,j})$, i.e., qp_i is an ancestor of $p_{k,j}$

$sim_k(qp_i, p_{k,j}) = 1$; (1)

else if $p_{kj} = ?(qp_i)$, i.e., qp_i is a descendant of $p_{k,j}$

$sim_k(qp_i, p_{k,j}) = (n+1-m)/(n+1+m)$; (2)

where, n = the number of links between qp_i and the root of the taxonomy

m = is the number of links between $p_{k,j}$ and qp_i.

Otherwise $sim_k(qp_i, p_{k,j}) = 0$; (3)

Note that the similarity metric is asymmetric because of its reference to the user's query level. Furthermore, the metric in equation 2 considers the depth and the density of the taxonomy to establish a notion of semantic distance by including a normalizing factor in the denominator. For the same m, the similarity of nodes at deeper levels of the taxonomy is higher. Table 2 shows an example similarity computation by referring to the taxonomy shown in Figure 1.

Table 2. Example of question-answer pair similarity assessment

$(qp_i, p_{k,i})$	**Applicable Condition**	sim_k
t_2,t_6	qp_i is an ancestor of $p_{k,j}$	1.0
t_6,t_2	qp_i is a descendant of $p_{k,j}$, n=2, m=1	0.5
t_6,t_1	qp_i is a descendant of $p_{k,j}$, n=2, m=2	0.2
t_4,t_6	*Otherwise*	0.0

2. *Aggregate similarity score assessment using a similarity metric*: We compute the overall similarity of a user query ***QP*** with a case problem description $\boldsymbol{P_k}$ denoted by $Sim_k(\boldsymbol{QP},\boldsymbol{P_k})$ by adapting the Rogers and Tanimoto (1960) similarity coefficient to include graded similarity as follows:

$$Sim_k(\boldsymbol{QP}, \boldsymbol{P_k}) = \frac{\sum_{i \in QP,\, j \in P_k} sim_k(qp_i, p_{k,j})}{T} \quad (3)$$

Where T is the number of taxonomies common to the question-answer pairs of ***QP*** and P_k.

Step III-Rank and Select

A set of retrieved cases $\boldsymbol{C_R}$ that is a subset of candidate cases $\boldsymbol{C_Q}$ is presented to the user ranked in descending order by the similarity score.

Conversation. Conversation involves presenting the user with a rank ordered set of questions derived from $\boldsymbol{C_R}$. We denote such a set presented in response to user query description Q by $\boldsymbol{Q}^Q$. We select and rank order the members of $\boldsymbol{Q}^Q$ in the following steps:

1. *Taxonomy selection*: By reference to $\boldsymbol{C_R}$, select the applicable taxonomies based on their question-answer pairs.
2. *Question scoring and selection:* To select an ordered list of questions, by reference to applicable taxonomies selected in step 1, we first score each question-answer pair in the taxonomies and then score the corresponding questions as follows:
   a. Question-answer pair and taxonomy scoring: The score at a node t_i in a taxonomy $\boldsymbol{T}$ is denoted by $s(t_i)$. It is the sum of scores of all its child nodes and the aggregate similarity scores of those retrieved cases that are indexed by it. We perform a backward pass on the taxonomy to compute all its node scores. Consider the example shown in Figure 3. It shows an extension of the printing problem taxonomy with example case similarities and node scores. We assume that the user's problem description contains a question-answer pair t_1 based on which the cases 1,2,3,7, and 11 were retrieved with scores as shown. For example, we compute the $s(t_4)$ as follows:

$$(s(t_4) = 0.6) = (Sim_{11} = 0.2) + (s(t_7) = 0.4)$$

   We denote the score of the root node as the taxonomy score $s(\boldsymbol{T})$. In the example, $s(\boldsymbol{T}) = 2.0$.

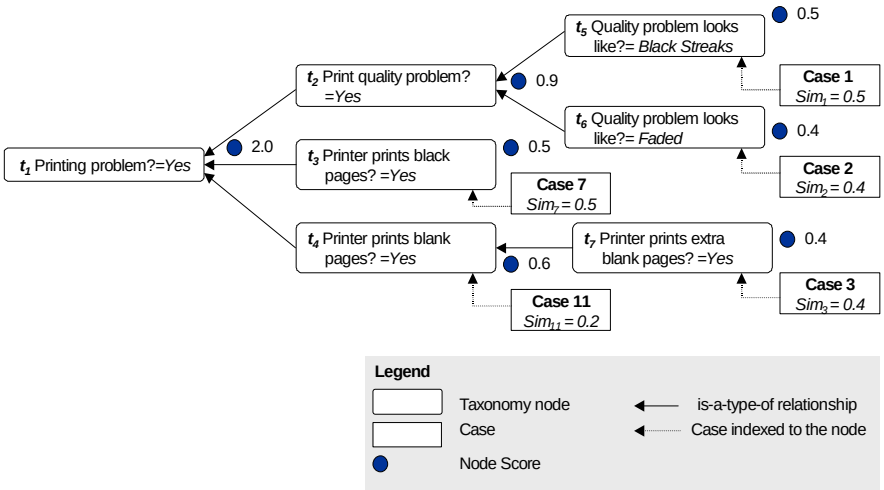

Figure 3. An example of question-answer pair scoring using the backward pass algorithm**.**

   b. Question selection and scoring: If the user problem description already contains a question-answer pair from the taxonomy, then we select its children. By reference to Figure 3 in our previous example, given that the user had selected t_1, we select nodes t_2, t_3, and t_4 for presenting questions. If the user problem description does not contain any question-answer pair from the taxonomy, then we select the most specific node that subsumes the set of retrieved cases $\boldsymbol{C_R}$. Consider the following example scenario by reference to the taxonomy shown in Figure 3. We assume that $\boldsymbol{C_R}$ consists of only case 1

(Sim_1 =0.5) and case 2 (Sim_2 =0.5). We also assume that the cases 3,7, and 11 are not retrieved because their scores were below the specified threshold (e.g., ? 0). In this scenario, the most specific node subsuming $\boldsymbol{C_R}$ is t_2 with a score $s(t_2) = 0.9$.

3. *Question ranking:* The score of a question q_i, denoted by $s(q_i)$ is a tuple '< >' with two values: taxonomy score $s(\boldsymbol{T})$, and sum of its corresponding question-answer pair scores. For example, the score for the question "print quality problem" is $s(q)$ = <2.0, 0.9>. The members of $\boldsymbol{Q}^Q$ are sorted in descending order on their taxonomy score followed by the question-answer score.

During a problem solving session, if the user selects more than one question-answer pair from the same taxonomy, the most specific question-answer pair is retained in ***QP.*** For the example taxonomy shown in Figure 1, if the user selects t_1 (Printing Problem?=*Yes*) and t_2 (Print Quality Problem=*Yes*), t_2 is retained and t_1 is discarded. This rule is the same as the one that applies to the case representation to enforce representational consistency.

We illustrate the advantages of the question selection and ranking technique by the following two examples. We assume that users interact with a Taxonomic CCBR system to troubleshoot printers, parts of which have been presented earlier.

Example 1, Abstract problem description: Consider a scenario where the user starts with a textual problem description *QT* such as "I am having printing problems." The CCBR application determines using search step 1 that the corresponding question-answer pair is "Printing Problem?=*Yes*." Based on this question-answer pair, the search expands the query to include all its descendents using the taxonomy presented in Figure 3 and retrieves cases 1,2,3,7, and 11. Let us assume that the overall similarity scores were the same as presented earlier. The user is presented with the following questions in order (for simplicity, we exclude other questions that might be derived from the candidate cases):

Do you have a print quality problem?
Is the printer printing blank pages?
Is the printer printing black pages?

We assume that the user answers the question "Print quality problem?" with a *Yes*. Consequently, the question answer pair "Print quality problem?=*Yes*" is used to replace the earlier general question "Printing problem?=*Yes.*" The traversal on taxonomy from node t_2 (i.e., search step 2) implies that only cases 1 and 2 are retrieved with scores 0.5 and 0.4. In this iteration, only the nodes t_1, t_2, t_5, and t_6 have scores greater than 0. The nodes that are subordinate to t_2 are t_5 and t_6. Therefore, the question presented is "What does the quality problem look like?" This example illustrates how the conversation guides a user in refining his/her problem description when the query is expressed at an abstract level.

***Example 2, Specific problem description*:** Consider a scenario where the user starts with a textual problem description *QT* such as "I have black streaks on my printout." Taxonomic CCBR application determines using search step 1 that the corresponding question-answer pair is "Printing quality problem?=*Black Streaks*." Since this

question-answer pair does not have any descendents, the search only retrieves case 1 and there is no further conversation.

Comparing examples 1 and 2, we note that the retrieval and conversation appropriately respond to the abstraction level of user's problem description. On the one hand, when a user expresses his/her query at an abstract level, s/he is guided in progressively refining his/her query. On the other hand, when the user expresses a query at a concrete or specific level s/he is spared irrelevant general questions and presented with a set of cases with high precision.

Case Acquisition. During a problem solving session, if no suitable cases are presented to the user, s/he can trigger new case acquisition. Typically, in such a scenario, the unsolved problem session comprising the user's problem description is complemented with a solution by a case author. The case author then adds the new case to the case base by following these steps:

1. Add new questions to the case base. Depending on the availability of information in a new case, the case author has the flexibility to retain and create new question-answer pairs at appropriate levels of abstraction. Our proposed methodology does not force an author to generalize or specialize a new question to one that already exists as would happen in CCBR system where abstraction is ignored.
2. Add the corresponding question-answer pairs to existing taxonomies or create new taxonomies.
3. Index the new case with appropriate question-answer pairs.

4 Advantages of Taxonomic CCBR

The following are the benefits of Taxonomic CCBR:

1. *Consistent and efficient representation*: Taxonomic CCBR methodology ensures representational *consistency* in user queries and stored cases. That is, a case or query cannot be indexed by more than one question-answer pair that belong to the same taxonomy. It also makes the case representation *efficient* since it is indexed by *fewer* and only the most specific question-answer pairs available at the time of indexing. There is no need for redundant indexing with a combination of general and specific features because the taxonomic search appropriately expands the search scope.
2. *Accurate and responsive retrieval:* Representational consistency eliminates any unwanted correlation among features that could result from inherent abstract relationships between question-answer pairs. This simplifies and improves matching. The retrieval is responsive to the abstraction level in the user's query since it appropriately expands or contracts search scope using taxonomic traversal.
3. *Responsive conversation with reduced information load*: The conversation is responsive to the level of abstraction in a user's query. Furthermore, the information load during the conversation is reduced since only the questions from the most appropriate level of abstraction are selected.
4. *Simplified and flexible case maintenance*: Since the representational consistency can be enforced by feature taxonomies, error prone and redundant indexing of cases is eliminated. Case maintenance is simplified as new features are introduced because only the taxonomies need to be maintained. In addition, the taxonomic

CCBR methodology affords *indexing flexibility* by allowing case authors to create new question-answer pairs at appropriate levels of abstraction thereby preserving information at the level of abstraction at which it was originally expressed.

These benefits, however, come at an expense (See Table 2). First, the computational complexity is increased by taxonomic traversal during retrieval and conversation. A worst-case scenario analysis shows that this increase is linear with the maximum number of nodes in any taxonomy (i.e., m). However, the gains in representational efficiency ($\hat{q} \leq q$) can partially offset this increase. Second, the space requirement in Taxonomic CCBR increases linearly with the number of question-answer pairs in the case base $O(Q.A)$. Third, additional knowledge engineering is needed to develop and maintain feature taxonomies, which is not discussed here.

Table 2. Computation and Space Complexity Comparison

Complexity	CCBR	Taxonomic CCBR
Computation	$O(Cq^2)$	$O(C\hat{q}^2 m)$
Space	$O(CQ \leq QA)$ Cases + Question answers	$= O(CQ \leq QA)$ Cases + Taxonomies

C= number of cases in the case base
Q= total number of questions in the case base
A= Maximum number of possible answers per question
q = Maximum number of questions per case in regular CCBR
$\hat{q}$ = Maximum number of questions per case in Taxonomic CCBR ($\hat{q} \leq q$)
m = Maximum number of nodes in a taxonomy

Nonetheless, we conjecture that our claimed benefits are likely to outweigh the above expenses. As presented in Section 6, we intend to empirically investigate this conjecture in our future research. We plan to compare the existing CCBR performance with Taxonomic CCBR in terms of representational efficiency, retrieval accuracy, conversational efficiency, and development and maintenance effort (e.g., taxonomies).

5 Related Work

The commercial success of CCBR applications has been particularly noteworthy in troubleshooting and help-desk tasks. However, the performance of these systems can be significantly affected by the extent of their scope, complexity, and application domain dynamics. This issue presents opportunities for formalizing and improving CCBR. For example, Montazemi and Gupta (1996) presented a diagnostic CBR application for troubleshooting AC motors called TRAAC. TRAAC used an adaptive agent to converse with a troubleshooter. The conversation was generated using a belief net to assist the troubleshooter in the identification of potential hypotheses and tests. The user selections were used to retrieve appropriate cases. While this approach was effective, it required the development and the maintenance of belief nets. Furthermore, they did not recognize the abstract relations among features (e.g., "Motor vibration" and "Drive-end motor vibration"). This research recognizes such

abstract relations and exploits it for retrieval and conversation. However, we do not consider causal and evidential relationship among features that might be pertinent to troubleshooting applications.

Aha *et al.* (1998) addressed the problem of redundant questions in conversation by means of their CCBR tool NaCoDAE. They examined rule-based and model-based approaches to automatically answer questions. Their approach led to more efficient conversation and retrieval. In their example application to printer troubleshooting, they include abstraction as *instance-of* and *implies* relations. However, unlike the Taxonomic CCBR, their approach was not explicitly motivated by abstraction. Aha *et al.* (2001) present problems of representational redundancies arising from case design guidelines that promote the use of general and specific features to index cases. They correct these inconsistencies in part by automatically revising case libraries. In contrast, we exploit the abstract relations in the feature taxonomies to create a robust representation framework that eliminates representational inconsistencies and simplifies similarity assessment. Instead of automatically answering questions we exploit the taxonomy to tailor the conversation and retrieval to the level of abstraction in the user's problem description.

McSherry (2001) addressed user interface issues in CCBR application for sequential diagnosis. His research focused on determining an optimal conversation strategy based on an attribute's ability to confirm or disconfirm candidate hypotheses. Our research differs from his in terms of assumptions we make about the domain. We assume heterogeneous case structures where abstraction is an issue combined with the domain dynamics as opposed to homogeneous case structures at a single-level of abstraction. Consequently, our research focuses on feature taxonomies to support the differences in conceptual level of end users. It conducts conversation and retrieval that is responsive to these differences.

Carrick *et al.* (1999) addressed the problem of trivial or repeated questions that CCBR systems prompts a user. They reduce the number of questions asked of the user by accessing other information sources that can be used to answer them. Their question generation strategy considers information quality together with the cost of acquiring information from additional sources to conduct the conversation. Their notion of information quality is very similar to our question-answer score. However, they did not consider feature abstraction in their system nor do they address the representational problems that arise from it.

6 Conclusion

In this paper, we established that sources of variation and differences in levels of abstraction are, in large part, uncontrollable in application domains typically targeted by CCBR systems such as help-desk and troubleshooting. Because CCBR systems are mixed-initiative systems their performance can be adversely affected when abstraction among features is ignored. We argued that ignoring abstraction could cause representational inconsistencies, redundant conversations, poor case retrieval performance, and numerous indexing and case maintenance problems.

We presented an integrated methodology called Taxonomic CCBR that explicitly represents abstract relations in taxonomies. We showed that the methodology eliminates representational inconsistencies, generates non-redundant conversation that

adapts to the abstraction level of a user's problem description, and dramatically simplifies case indexing and case base maintenance. In our future work, we plan to empirically assess the benefits and limitations of Taxonomic CCBR. We recognize that the success of Taxonomic CCBR could be limited by the availability of tools and methodologies for acquisition and maintenance of taxonomies. We will investigate these methodologies in future research.

While we contend that the Taxonomic CCBR methodologies can improve the performance of existing CCBR systems, there can be other kinds of relationships among features that could be pertinent to CCBR systems and were not included in our methodology. For example, troubleshooting and diagnostic domains can include causal and evidential relations among features (e.g., Montazemi & Gupta (1995); McSherry, 2001). We intend to explore this issue further in our research.

Finally, we believe that this research is relevant to textual CBR systems that retrieve short text documents (Ashley & Lenz, 1998). In such systems, document cases do not undergo extensive knowledge engineering and abstraction is a common problem. A problem resulting from abstraction is that of case redundancy and inconsistency (Everett & Bobrow, 2000; Racine &Yang, 2001). In our future work, we will examine the applications of this research to textual CBR.

Acknowledgements

This research was funded by the Design For Safety Program of the NASA Ames Research Center, Naval Research Laboratory, and the Office of Naval Research. Thanks to David Aha, Karl Branting, David McSherry, Len Breslow, and the three anonymous reviewers for their helpful comments on an earlier version of this paper.

References

Aha, D.W., Breslow, L.A., & Munoz-Avila, H., (2001), Conversational Case-Based Reasoning, *Applied Intelligence*, 14(1), pp. 9-31.

Aha, D.W., Maney, T., & Breslow, L.A., (1998), Supporting Dialogue Inferencing in Conversational Case-Based Reasoning, *Advances in Case-Based Reasoning* (B. Smyth and P. Cunningham (Eds.)), 4th European Workshop EWCBR-98, Dublin, Ireland, pp. 262-273.

Alterman, R., Griffin, D., (1994), Remembering Episodes of Question Answering, *Proceedings of the Second European Workshop on Case-Based Reasoning*, EWCBR-94, M. Keane, J.P. Haton, M. Manago (Eds.), pp. 235-242.

Arocha, J.F. & Patel, V.L., (1995), Diagnostic Reasoning by Novices: Accounting For Evidence, *Journal of Learning Sciences*, 4(4), pp. 355-384.

Ashley, K.D., & Lenz, M. (Eds.), (1998), Textual Case-Based Reasoning, Papers from the AAAI-98 Workshop, AAAI Tech. Rep. WS-98-12, *AAAI Press*, Melno Park, CA.

Baudin, C., & Waterman, S., (1998), From Text to Cases: Machine Aided Text Categorization for Capturing Business Reengineering Cases, *Proceedings of the AAAI Workshop on Textual Case-Based Reasoning, Technical Report WS-98-12*, pp. 51-57.

Bergmann, R., & Wilke, W., (1996), On the Role of Abstraction in Case-Based Reasoning, (I. Smith and B. Faltings (Eds.)), Advances in Case-Based Reasoning, *EWCBR-96*, pp. 28-43.

Branting, K.L., & Aha, D.W., (1995), Stratified Case-Based Reasoning: Reusing Hierarchical Problem Solving Episodes, *IJCAI* 1995, Vol. 1, pp. 384-390.

Carrick, C., Yang, Q., Abi-Zeid, I., & Lamontagne, L., (1999), Activating CBR systems through autonomous information gathering. *Proceedings of the Third International Conference on Case-Based Reasoning*, Munich, Germany: Springer, pp. 74-86.

DoD (1997). *Joint tactics, techniques and procedures for noncombatant evacuation operations* (Joint Report 3-07.51). Washington, DC: Department of Defense, Joint Chiefs of Staff.

Gupta K .M.(2001), Case-Based Reasoning Research and development, Proceedings of the fourth ICCBR ,D. W. Aha & I. Watson (Eds.), Springer, Berlin, Germany, pp. 219-233.

Drastal, G., & Czako, G., (1989), Induction in an abstraction space: A Form of Constructive Induction," *Proceedings of the Eleventh International Joint Conference on AI*, Vol.1, pp. 708 –712.

Everett, J.O., Bobrow, D.G., (2000), Resolving Redundancy: A Recurring Problem in a Lessons Learned System, *Intelligent Lessons Learned Systems: Papers from the AAAI Workshop (Technical Report AIC-00-005)*, Washington DC: Naval Research Laboratory, Navy Center for Applied Research in Artificial Intelligence, pp. 12-16.

Furnas, G.W., Landauer, T.K., Gomez, L.M., & Dumais, S.T., (1987), The Vocabulary Problem in Human-System Communication, *The Communications of the ACM*, 30(10), pp. 964-971.

Gupta, K.M., & Montazemi, A.R., (1997), Empirical Evaluation of Retrieval in Case-Based Reasoning Systems using Modified Cosine Matching Function, *IEEE Transactions on Systems, Man, and Cybernetics*, 27(5), pp. 601-612.

Gupta, K.M., (1998), Knowledge-Based System For Troubleshooting Complex Equipment, *International Journal of Information and Computing Scienc*e, 1(1), pp. 29-41.

Kolodner, J.L., (1993), Case-Based Reasoning, *Morgan Kaufman*, San Mateo, CA.

McSherry, D., (2001), Interactive Case-Based Reasoning in Sequential Diagnosis, *Applied Intelligence*, 14(1), pp. 65-76.

Montazemi A.R., & Gupta K.M., (1996), An Adaptive Agent for Case Description in Diagnostic CBR Systems, *Computers in Industry*, 29(3), pp. 209-224.

Pedrycz, W., & Vukovich, G., (2000), Granular Worlds: Representation and Communication Problems, *International Journal of Intelligent Systems*, 15(11), pp. 1015-1026,

Racine, C., and Yang. Q., (2001), Redundancy and Inconsistency Detection in Large and Semi-Structured Case Bases, *IEEE Transactions on Knowledge and Data Engineering*, To appear.

Rogers, D.J., & Tanimoto, T.T., (1960), A Computer Program for Classifying Plants, *Science*, Vol. 1332, pp.1115-1118

Rosch, E., (1978), Principals of Categorization, in Cognition and Categorization, E. Rosch and B. Llyod, (Eds.) *Lawrence Earlbaum Associates Publishers*, Hillsdale, New Jersey, NJ, pp. 28-48.

Shimazu, H., (1999), Translation of Tacit Knowledge into Explicit knowledge: Analyses of Recorded Conversations between Customers and Human Agents, *Exploring Synergies of Knowledge Management and Case-Based Reasoning*, Papers from the AAAI Workshop, TR WS-99-10, pp. 81-85.

Trott, J.R., Leng, B., (1997), An Engineering Approach for Troubleshooting Case Bases, *Case-Based Reasoning Research and Development*, (Eds.) D.B. Leake and E. Plaza, ICCBR-97, pp.178-189.

Yang, Q., & Wu, J., (2001), Enhancing the Effectiveness of Interactive Case-Based Reasoning with Clustering and Decision Forests, *Applied Intelligence*, 14(1), pp. 49-64.