Case-Based Retrieval Interface Adapted to Customer-Initiated Dialogues in Help Desk Operations

Hideo Shimazu and **Akihiro Shibata** and **Katsumi Nihei**
Information Technology Research Laboratories, NEC Corporation
4-1-1 Miyazaki Miyamae-ku, Kawasaki, Japan, 216
{shimazu, akihiro, nihei}@joke.cl.nec.co.jp

Abstract

Help desk systems have become increasingly important in the efforts of corporations to maintain customer satisfaction, and Case-Based Reasoning (CBR) provides promising techniques for use in the improvement of such systems. This paper describes multiple interface modes by which customer service operators can respond rapidly to customer-initiated inquiries, retrieving/storing case data from/into a case-base. The proposed interface addresses a major situation assessment problem, the difficulty of attempting to match what may be completely different descriptions of an item to be retrieved, i.e. descriptions resulting from vastly differing points of view. The proposed interface and the similarity assessment algorithm are implemented in the CARET case-based retrieval tool operating on commercial Relational Database Management Systems (RDBMS).

Introduction

This paper describes a novel interface to a case-based retrieval system. It is to be used in a customer support help desk system by which customer service operators can respond rapidly to customer-initiated inquiries, retrieving/storing case data from/into a large-scale case-base. We have implemented the interface design on an enhanced version of CARET, a previously reported case-based retrieval tool that operates on commercial relational database management systems (RDBMS) (Shimazu, Kitano and Shibata 1993).

Users of high-tech products today regard technical support as being nearly as important as the performance of the products themselves. Because help desk operators receive repeated requests, a case-based retrieval system which can show operators those similar previous customer inquiries and the operator replies to them would naturally be a useful tool.

In order for help desk operators to be able to respond to inquiries in a reasonably short time, however, such a case-based retrieval system must provide an easy and rapid interface for similar-case retrievals, but the fact that customers tend to explain their problems in widely varying ways is a serious obstacle to achieving rapid retrievals.

Interface Design Decisions

Typical categories of customer approaches to explaining their inquiries include:

Step-by-step action-based explanation:
The user explains his/her inquiry procedurally and chronologically, for example, "When I am sending email with the menu, can I cancel the email after I have selected the SEND item from the menu?"

Diagram-description-based explanation:
The user envisions the internal model (which may or may not be accurate) of the product and explains his/her inquiry using that model, for example, "Can I cancel some email after I've already put it in my *mail box* for sending?".

Physical-appearance-based explanation:
The user explains his/her inquiry in terms of the physical appearance of the product, for example, "On the right-hand side of the machine, the red lamp just below the RESET switch is blinking. What does this mean?".

A case-based retrieval system interface must be capable of handling all these types of explanations and more. For this reason, we have designed our interface to include a separate mode for each explanation category. When answering a customer call, the operator first determines which of the three categories is most appropriate to the way the customer is describing the inquiry. Then the operator selects a corresponding interface mode, and inputs the customer's inquiry via that mode.

An analysis of customer problems has taught us that the majority of problems arise when a customer acts almost correctly, but makes some minor mistake, and/or simply misunderstands a situation to a slight degree, and on that basis we have developed a new case indexing method based on the three typical categories of customer inquiry descriptions previously listed.

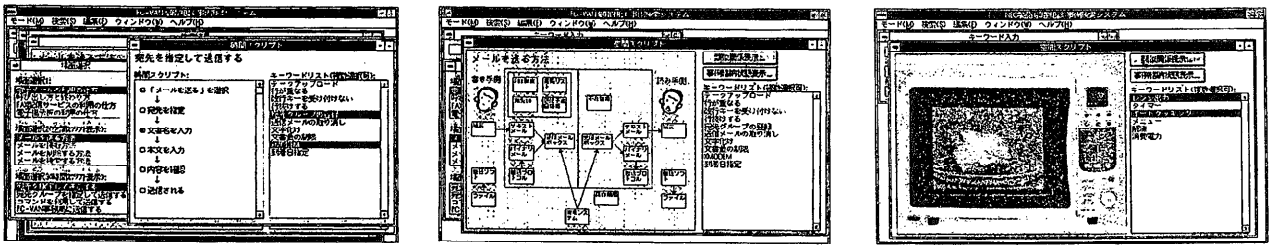

Figure 1: Step-by-step schema (left), Diagram schema (center), and Physical-appearance schema (right)

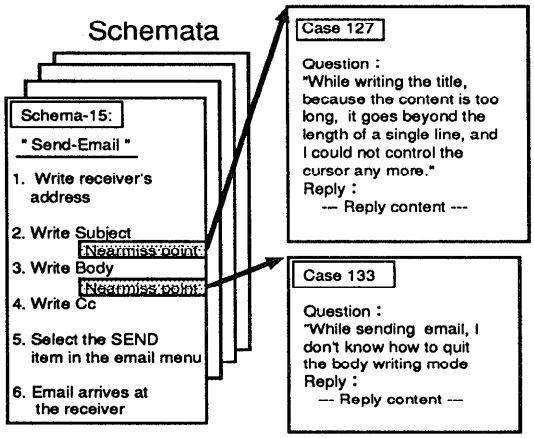

Figure 2: Case Indexing with a Schema and Nearmiss points

For each of these categories there is prepared a set of predefined **schemata**, each of which describes a situation likely to arise as an inquiry (Figure 1).

Step-by-step Schema: For the step-by-step action-based category, the schemata consist of individual lists of the complex combinations of operations required to accomplish the most typical customer goals. This schema structure is similar to script (Schank and Abelson 1977).

Diagram Schema: For the diagram-description-based category, the schemata consist of descriptions of the most typical internal models envisioned by customers.

Physical-appearance Schema: For the physical-appearance-based category, the schemata consist of individual physical views of the product as typically seen by customers.

If a customer's problem description in a case is that he/she acts almost correctly, but makes some minor mistake, and/or simply misunderstands a situation to a slight degree, the case can be simply indexed with a set of a predefined schema and one or a few slight disparities (**nearmiss points**) between the schema and the case.

Figure 2 shows "send-email" step-by-step schema, Schema-15, which represents typical and correct sequences of actions taken when a customer sends email. Case-127 holds a question whose goal and actions are similar to those for Schema-15, but has a nearmiss point which causes this question. A pair of Schema-15 and the nearmiss point becomes the index for Case-127.

When an operator retrieves cases while listening to a customer's inquiry, the operator selects a corresponding schema. The schema is displayed on the terminal. When he/she points at one or a few problem points in the schema, if there are cases whose nearmiss points are equal or similar to those specified problem points, the cases are displayed on the terminal.

When an operator stores a new case, he/she describes the case content, selects an appropriate schema, indicates one or a few problem points in the schema as the nearmiss points, and stores the case into the case-base.

Bridging of Schemata

Similarity values among steps in step-by-step schemata, components in diagram schemata, and pertinent points in physical-appearance schemata are defined by domain experts. Similar-case retrieval over a single schema is carried out using the similarity definitions.

Bridging representations in different schemata is

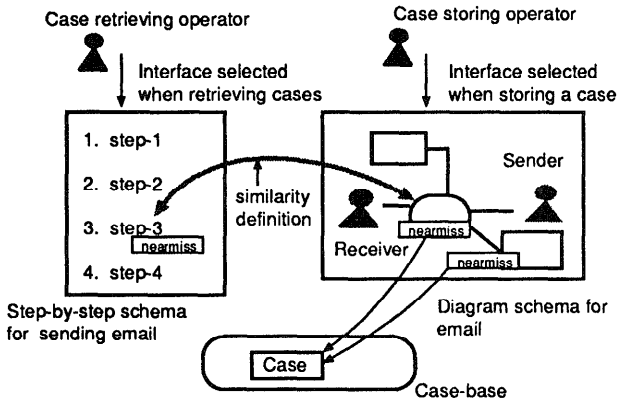

Figure 3: Bridging Different Schemata

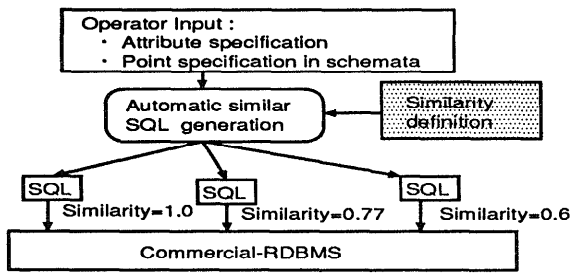

Figure 4: Automatic SQL generation

necessary because it assimilates differences in schema selections by operators. Often, a schema invoked when storing a case is different from a schema invoked when retrieving an identical case. Figure 3 shows a case which is indexed from a diagram schema when stored, because the customer explained his/her problem with spatial expressions. Assume that another customer asks another operator a very similar question with step-by-step expressions. The operator invokes a step-by-step schema and specifies a problem point in the schema. However, since there is no index from the step-by-step schema to the case, the case can not be retrieved. If similarity was predefined between steps in the step-by-step schema and components in the diagram schema, the case is retrieved, although no direct index exists from the step-by-step schema to the case.

The CARET System

The proposed interface and the similarity assessment algorithm were implemented in the CARET case-based retrieval tool operating on commercial RDBMS. This section describes the algorithm in detail.

Nearest Neighbor Retrieval

CARET uses nearest neighbor retrieval. Typically, a similarity between a query (Q) and a case (C) in the

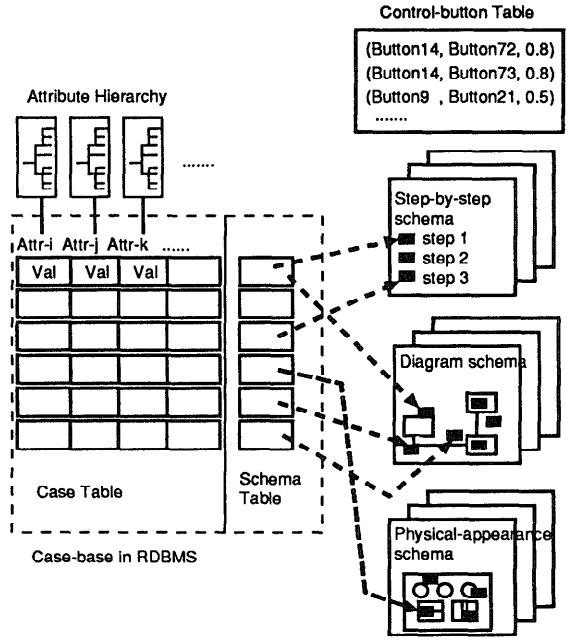

Figure 5: Case and Schema Representation

case-base ($S(Q,C)$) is the weighted sum of similarities for individual attributes:

$$S(Q,C) = \frac{\sum_{i=1}^{n} W_i \times s(Q_i, C_i)}{\sum_{i=1}^{n} W_i} \quad (1)$$

where W_i is the i-th attribute weight, $s(Q_i, C_i)$ is similarity between the i-th attribute value for a query (Q) and that for a case (C) in the RDB. Traditional implementations would compute the similarity values for all records, and sort records based on their similarity.

CARET uses a commercial RDBMS for its case-base manager. The use of RDBMS, however, automatically forces CARET to generate SQL specifications (Chamberlin *et al.* 1976) to carry out any case-base retrievals. Since SQL does not entail any similarity-based retrieval features, CARET has a method to carry out similarity-based retrievals using SQL. CARET generates SQL specifications in varying degrees of similarity, and the generated SQL specifications are dispatched to RDBMS to retrieve cases from RDB (Figure 4).

Case and Schema Representation

Cases are represented as a flat record of n-ary relations, as shown in Figure 5. Each record has a corresponding record which holds schema indices. The similarities between values in individual attributes for a case record are defined using an attribute hierarchy, as shown in Figure 6. These attribute hierarchies are defined by domain experts.

Schemata are also stored in RDBMS. Figure 7 shows schema representations. A schema has two layer struc-

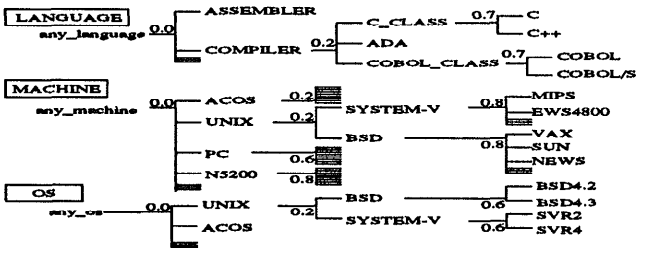

Figure 6: Attribute Hierarchy Example

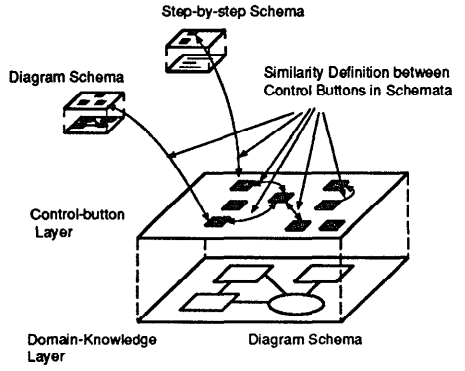

Figure 7: Schema Representation

tures; the domain-knowledge layer and the control-button layer. Steps in a step-by-step schema, diagrams for a diagram schema, and a physical appearance image of a physical-appearance schema are drawn in the domain-knowledge layer as background image data. The control-button layer holds control buttons, which overlap on the background image and are pointed-and-clicked when retrieving and storing cases. Control buttons should be allocated at positions where users may point as problem points. Similarity values between control buttons in schemata are stored in the control-button table. This representation is similar to Hypercard (Goodman 1988), except that control buttons can be defined their similarities values.

Similarity Assessment Algorithm

This section describes the CARET enhanced similarity assessment algorithm.

Step1: Creating NVSs and NBSs A user specifies attributes and values representing the problem, invokes a specific schema on the display, and points-and-clicks one or a few control buttons on the displayed schema.

(1) Neighbor Value Sets For each user specified attribute, CARET refers to the abstraction hierarchies, as shown in Figure 6, to generate a set of values neighboring the user specified value. For example, assume that the user specified `C++` as a value for the `Language` attribute, `C++` is an element in *the first-order neighbor value set (1st-NVS).* `C` is an element in *the second-order neighbor value set (2nd-NVS).* `ADA`, `COBOL` and `COBOL/S` are elements in *the third-order neighbor value set (3rd-NVS).*

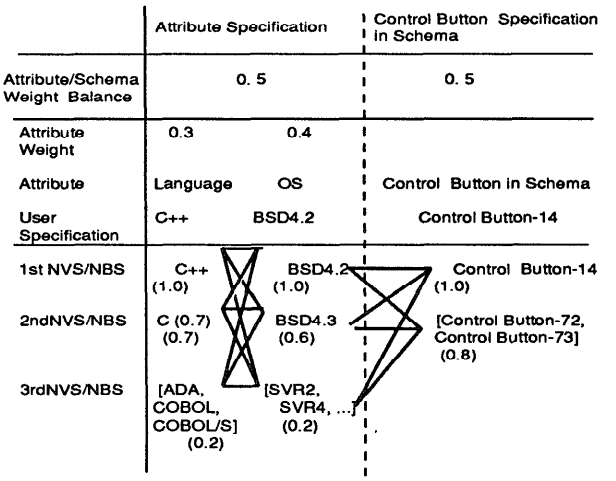

Figure 8: An Example showing Possible NVS/NBS Combinations

(2) Neighbor Control Button Sets For each user specified control button in a schema, CARET refers to the control-button table, as shown in Figure 5, to generate a set of control buttons neighboring the user specified control button. For example, assume that the user specified `Control Button14`, `Control Button14` is an element in *the first-order neighbor control button set (1st-NBS).* `Control Button72` and `Control Button73` are elements in *the second-order neighbor control button set (2nd-NBS).*

Step2: Enumerating the Combinations Next, all possible neighbor value/control button combinations are created from the n-th order neighbor value/control button sets. Figure 8 illustrates how such combinations are created. This example assumes that the user specified values for attribute `Language` and `OS`, and `Control Button14` in a schema. All value combinations under attribute `Language` and `OS`, and `Control Button14` will be created. In this example, 18 combinations ($3 \times 3 \times 2$) are generated. Each combination of NVS/NBSs becomes the seed for SQL specifications.

Step3: Calculating Similarity Value for Each Combination For each combination, a similarity value is calculated using similarity between the user specified values/control buttons and those in each combination created in the previous stage. The calculation is similar to that for the weighted nearest neighbor, except that not all attributes are involved. The CARET

algorithm does not compute any attributes not specified by the user. Whether the user specified the attribute or not is shown in a mask matrix (M), which is a one-dimension matrix whose size is equivalent to the case-base degree. The matrix element M_i will be 1, if the user specified the value for the attribute i. Otherwise, M_i will be 0. $S(Q,F)$, the similarity between a query(Q) and a NVS/NBS combination (F) is the weighted sum of the attribute part similarity, $S_a(Q,F)$ and the schema part similarity, $S_s(Q,F)$.

$$S(Q,F) = \frac{W_a \times S_a(Q,F) + W_s \times S_s(Q,F)}{W_a + W_s} \quad (2)$$

$$S_a(Q,F) = \frac{\sum_{i=1}^{n} M_i \times W_i \times s(Q_i,F_i)}{\sum_{i=1}^{n} M_i \times W_i} \quad (3)$$

$$S_s(Q,F) = \frac{\sum_{j=1}^{m} s(Q_j,F_j)}{m} \quad (4)$$

Where $s(Q_i,F_i)$ in $S_a(Q,F)$ is similarity between the i-th attribute value for a query (Q) and that for a combination (F), m in $S_s(Q,F)$ is the number of specified control buttons in a schema, and $s(Q_j,F_j)$ is the similarity between the j-th control button for a query (Q) and that for a combination (F). W_i, W_a and W_s are entered manually by domain experts. For example, the similarity of a combination, (["C"],["BSD4.3"], ["Control Button72, Control Button73"] to the user's query specification is 0.72 by the following calculation:

$$S_a(Q,F) = \frac{0.3 \times 0.7 + 0.4 \times 0.6}{0.3 + 0.4} = 0.64 \quad (5)$$

$$S_s(Q,F) = \frac{0.8}{1} = 0.8 \quad (6)$$

$$S_t(Q,F) = \frac{0.5 \times 0.64 + 0.5 \times 0.8}{0.5 + 0.5} = 0.72 \quad (7)$$

Step4: Generating SQL Specifications Each individual combination is translated into a corresponding SQL specification. Since SQL does not involve the similarity measure, the value assigned in the previous process is stored in CARET, and is referred to when the query results are returned. The only SQL expression type used here is the SELECT-FROM-WHERE type. For example, the combination example shown in the previous step is translated into the following SQL specification:

```
SELECT * FROM case_table, schema_table
WHERE language = 'C'  and OS = 'BSD4.3' and
 button in ('Control Button72', 'Control Button73')
 and  case_table.case_id = schema_table.case_id;
```

Empirical Results

Although several on-going projects (Kitano *et al.* 1992) employ the tools described in this paper, an empirical result is reported on the effectiveness of the proposed interface using a help desk support system applied for personal electronic mail service. Figure 9 shows some of the customer inquiries among several hundred cases stored in the case-base, also indicated is the result of a subjective evaluation, to compare if individual inquiries were effectively retrieved by diagram schemata, step-by-step schemata, and conventional keyword search-based retrieval. Cases were obtained from NEC internal publications and were complemented by an engineer from the service support team.

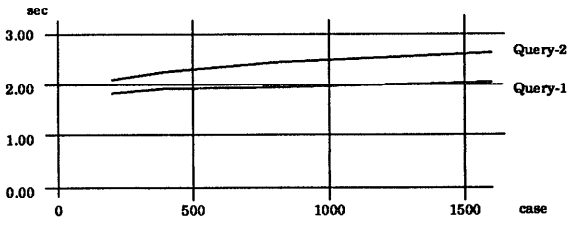

Figure 10: Case-Base Retrieval Time

Goal-plan questions are effectively retrieved using diagram schemata, because customer explanations tend to refer to the internal models of the domain when they cannot find concrete action sequences, although their goals are concrete. Since symptoms-cause-recover questions are straightforward, they are easily handled using step-by-step schemata. However, unfortunately, customer inquiries exist, for which neither step-by-step schemata nor diagram schemata can retrieve appropriate cases effectively. Many of them are context free inquiries and the symptom causes are not related to the present context; for example, for the inquiry, "Suddenly I could not move my cursor", the reason is keyboard cable disconnection.

The CARET performance on commercial RDBMS has attained a practically acceptable speed. The experiments were carried out on SUN Sparc Station 2, using ORACLE version 6 installed on SunOS version 4.1.2. Figure 10 shows the response times measured for typical user queries. The average response time for a query is about 2.0 seconds.

Related Work

The interface proposed in this paper addresses a major situation assessment problem (Kolodner 1993). A number of other CBR researches have addressed the problem. CYRUS (Kolodner 1984) was the first system to address this problem. CASEY (Koton 1988) bridges inferences before attempting retrieval and using evidence heuristics to bridge further differences in descriptions during matching. ANON (Owens 1988) introduced proverbs for matching abstract characterizations of situations. CASCADE (Simoudis 1992) uses model-based validation. All the systems presented above utilize significant domain-specific knowledge for similarity assessment. On the other hand, CARET

Goal-Plan Questions	Diagram	Step-by-step	Keyword
"How can I transfer data files in email?"	Good		Moderate
"How can I up-load file in my FD?"	Good		Moderate
"Is there a max size of an email body?"	Good	Moderate	Moderate
"Can I output my email to FAX?"	Good		Moderate
"How can I down-load email content?"	Good	Good	Moderate

Symptom-Cause-Recover Questions	Diagram	Step-by-step	Keyword
"While up-loading, CR does not work"	Good	Good	Moderate
"While writing body, can I add receiver?"	Moderate	Good	
"Arrived email includes noise"	Moderate	Good	Moderate
"After up-loading, the cursor is lost"		Good	

Figure 9: Subjective Comparisons for Schema Applicability to Cases

solves this problem for interactive case-based systems by treating it as a user interface problem – providing the user with several ways of describing a new situation and providing bridges between the user's and the system's possibly different points of view.

Several case-based help desk systems have been built and are being used. CLAVIER (Hennessy and Hinkle 1992) helps a user to configure the layout of composite airplane parts for curing in an autoclave. CASCADE (Simoudis 1992) suggests how a user recovers from VMS crashes. These systems are not required rapid case retrieval/store interface. The Compaq SMART system (Acorn and Walden 1992) using CasePoint (Inference 1993) and GE's help desk system using Cognitive System's ReMind (Kriegsman and Barletta 1993) both achieved rapid case retrievals. These systems use natural language for expressing customer-initiated situation descriptions. CARET, instead, provides schemata as a framework for situation assessment under customer-initiated dialogues.

Conclusion

This paper has described a case-based retrieval interface adapted to customer-initiated dialogues in help desk operations. Using the interface, customer service operators can respond rapidly to customer-initiated inquiries, retrieving/storing case data from/into a case-base.

A case is indexed with a schema and one or a few nearmiss points between the schema and the case. The proposed interface provides three schema types, step-by-step schema, diagram schema, and physical-appearance schema. These schemata are used as several ways of describing a new situation and providing bridges between the user's and the system's possibly different points of view.

The proposed interface and the similarity assessment algorithm have been implemented in the CARET case-based retrieval tool operating on commercial RDBMS.

References

ANSI Database Language SQL with Integrity Enhancement. 1989. ANSI X3.135.1-1989.

Chamberlin, D.D., et al., 1976. SEQUEL2: A Unified Approach to Data Definition, Manipulation, and Control. IBM J. Res. Develop.

Acorn, T.L., and Walden, S.H., 1992. SMART: Support Management Automated Reasoning Technology for Compaq Customer Service, In *Proceedings of IAAI-92*

Goodman, D., 1988. HYPERCARD DEVELOPER'S GUIDE, Bantam Books

Hennessy, D.H., and Hinkle, D., 1992. Applying case-based reasoning to autoclave loading, In *IEEE Expert*, October 1992.

Inference Corporation, 1993. CBR Express and CasePoint: Product Introduction, Presentation slides for NDS Customers, Tokyo

Kitano, H., Shibata, A., Shimazu, H., Kajihara J., Sato, A., 1992. Building Large-Scale Corporate-Wide Case-Based Systems: Integration of Organizational and Machine Executable Algorithms. In *Proceedings of AAAI-92*

Kolodner, J., 1984. Retrieval and organizational strategies in conceptual memory: A computer model. Lawrence Erlbaum Associates, Hillsdale, NJ.

Kolodner, J., 1993. Case-Based Reasoning, Morgan Kaufmann, San Mateo, CA.

Koton, P., 1988. Reasoning about evidence in causal explanation, In *Proceedings of AAAI-88*

Kriegsman, M, and Barletta R., 1993. Building a Case-Based Help Desk Application, In *IEEE Expert*, December 1993.

Oracle, 1989. Database Administrator's Guide. and other ORACLE manuals. Oracle Corporation

Owens, C., 1988. Indexing and Retrieving Abstract Cases, In *Proceedings of Case-Based Reasoning Workshop*, AAAI-88

Schank, R., and Abelson, R.P., 1977. Scripts, Plans, Goals, and Understanding, Erlbaum, Hillsdale, N.J.

Shimazu, H., Kitano, H., and Shibata, A., 1993. Retrieving Cases from Relational DataBase: Another Stride Towards Corporate-Wide Case-Based Systems, In *Proceedings of IJCAI-93*

Simoudis, E., 1992. Using Case-Based Retrieval for Customer Technical Support, In *IEEE Expert*, October 1992.