# Validating the Performance of a Case-Based Reasoning System

**Takao Terano**
Graduate School of Systems Management, The University of Tsukuba
3-29-1 Otsuka, Bunkyo-ku, Tokyo 112, Japan
Tel: +81-3-3942-6855, Fax: +81-3-3942-6829,
Email: terano@gssm.otsuka.tsukuba.ac.jp

## Abstract

When we use Case-Based Reasoning (CBR) for practical applications, it is often the case to implement two kinds of problem solvers: case-based one and knowledge-based/conventional one. The complex integration of such plural reasoners often causes serious VV&T problems: performance of problem solving, quality of solutions, and so on. In this paper, we address the performance validation problems of a CBR system. This paper presents a black-box validation approach to quantitatively analyze the performance and proposes three kinds of performance measures. The proposed measures are then applied to validating a CBR system: IRS-CBR (Intelligent Information Retriever with a Case-Based Reasoner). From the experimental results, we conclude that (1) IRS-CBR has succeeded in both speed-up and memory-based learning, and (2) the proposed measures are useful for validating a CBR system.

## 1 Introduction

A Case-Based Reasoning (CBR) system is expected to be one of the techniques to break the bottleneck of knowledge system development [Riesbeck 1989], [Kobayashi 1990-a]. However, when we use CBR for practical applications, we usually face problems caused by the fact that stored past cases cannot cover all the problem spaces. Therefore, in practice, we must implement two kinds of problem solvers: case-based one and knowledge-based/conventional one. The complex integration of a case-based reasoner and conventional problem solver also often cause bad performance of knowledge acquisition processes, problem solving processes, and learning processes. These are critical VV&T problems of a practical CBR system.

The validation of computer systems are, in general, carried out from two points of view ([Terano 1993]). The one focuses on the qualities of solutions. The other concentrates on the performance of problem solving process. Several researches on CBR systems in the literature have reported evaluation results from the former aspect so far (e.g., [Bareiss 1989], [Becker 1991], and [Golding 1991]). In this paper, on the other hand, we will quantitatively analyze the performance of a CBR system: IRS-CBR (Intelligent Information Retriever with a Case-Based Reasoner).

IRS-CBR adapts the CBR method to the task of information retrieval for financial statistical databases. IRS-CBR has two components: a knowledge-based reasoner for general problem solving and a CBR for using past cases. IRS-CBR is also able to store new cases automatically when using it. IRS-CBR is used as a front-end system implemented on a personal computer to communicate with a main frame computer on which financial statistical databases are stored. The system development and validation is difficult because the task contains an open problem and the domain knowledge is incomplete.

To validate IRS-CBR, we have defined three kinds of system-independent quantitative measures to reveal the dynamic features of the performance of a CBR system. We then report some experimental results of a black-box validation approaches.

This paper is organized as follows: In section 2, we discuss a general architecture of a CBR system, of which we will validate the performance. In section 3, we propose three kinds of performance validation measures for CBR systems. In section 4, we describe the features of IRS-CBR. In section 5, we illustrate experimental results to validate the performance of IRS-CBR using the proposed measures. In section 6, we give conclusions and future works.

## 2 On the Architecture of a CBR System

As are found in e.g., [Riesbeck 1989], it is said that a general CBR system has the nine components: (1) *Case Bases* for past cases with indices, problem definitions and solutions, (2) *Knowledge-Based Systems* with Knowledge Bases for specific domains, (3) *Index Assigner* to give indices to a given new problem, (4) *Retriever* to search for relevant cases in the case-base, (5) *Adaptator* to apply the cases to the given problem, (6) *Tester* to judge the success or failure of the adaptation, (7) *Storer* to store the new problem into the case-base when Adaptator succeeds in the application of the cases, (8) *Explainer* to give explanations of "what-is-wrong" when Adaptator fails,

and (9) *Repairer* to repair the causes of the failures in past cases.

These components (3)-(9) are further divided into the three categories: Classifier, Problem Solver, and Learner. Classifier makes the characteristic of a new problem clear, and matches past cases. Classifier corresponds to Index Assigner and Retriever. Problem Solver with Modifier is used to solve a new problem. Modifier mainly solves a new problem by modifying past problem solving results. Problem Solver corresponds to Adaptator, Tester, Explainer, and Repairer. Learner is used to store the solved problem nto the case-base with adequate indices. Learner corresponds to Storer.

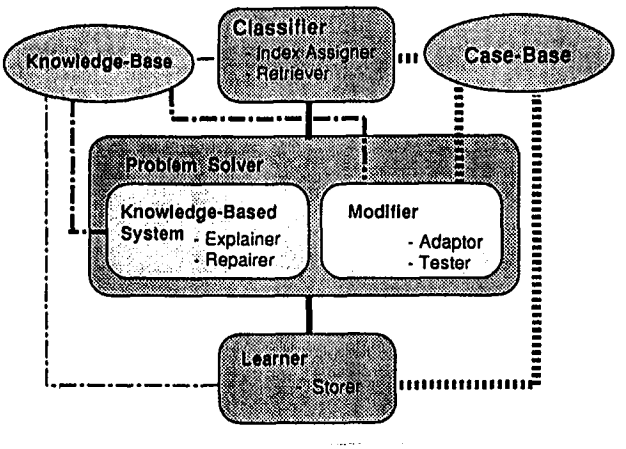


**Figure 1. General Architecture of a Case-Based Reasoner**

To develop the components (5), (6), (8), and (9), the role of domain knowledge is critical. The functions of Problem Solver are attained by either case-based techniques or other knowledge-based ones. It is also difficult to implement the components (3), (4), and (7) without domain knowledge. Thus, practical CBR systems necessarily have both a case base and an implicit/explicit knowledge base. The knowledge base relys on empirical or model-based knowledge to solve a new problem.

Therefore, a general architecture of a CBR system should have the five components shown in Figure 1. In the succeeding sections, we assume that a CBR system has the architecture in Figure 1.

## 3 Performance Validation Measures for a CBR System

The best way to validate the performance of a CBR system is probably to analyze the CBR system component-wise, to statically measure the size of case-bases, source codes, etc., and to dynamically evaluate the execution traces. This is a white-box approach for the validation. However, such an approach is CBR-system-dependent and very time-consuming.

On the other hand, there is a black-box approach without analyzing the contents of the CBR system. For example, the relation of the number of cases and the execution time shows the trade-off between the number of cases versus retrieving the cases. However, such simple data do not give the performance of each component of a CBR system shown in Fig. 1.

Therefore, we will propose the system-independent quantitative measures to reveal the dynamic features of the performance of CBR systems. These are Space Partition Ratio (SPR), Analogical Adaptation Ratio (AAR), and Reusable Case Ratio (RCR). To apply the measures, we assume that the CBR system can solve every given problem either using equipped problem solvers or assisted by the users, and that the characteristics of a given new problem can be determined by the requirement specification. A practical application system must require the former assumption and the latter is required to experimentally evaluate the system performance.

- **Space Partition Ratio**

  SPR is defined as the ratio of the maximum number of cases with the same kind of indices to the total number of cases in a case-base. SPR shows the utility of indices of stored cases and the performance of Learner. If SPR is large, the learning mechanism or the way to give proper indices is not adequate, thus, most given problems must be solved by the same adaptation methods. In such a case, if new problems are always stored in a case-base, the performance of case retrieval becomes worse.

- **Analogical Adaptation Ratio**

  AAR is defined as the ratio of the time required to adapt some old cases to a given problem by analogy to the time required to solve the given problem without CBR. a CBR system requires various adaptation knowledge to apply the cases to new problems. AAR shows the performance of the modification function of stored cases to a new problem. When the CBR system has no Learner, nor automatic case storing functions, AAR also shows the performance of the case-based reasoner. If AAR is large, the adaptation mechanisms/knowledge should be improved.

- **Reusable Case Ratio**

  RCR is defined as the ratio of the time to solve a new problem with the same indices of stored cases against the time required to solve the given problem without CBR. RCR shows direct usability of stored cases to a new problem. This can be used to evaluate the performance of Classifier: the mechanisms of retrieving cases and case-base organization. If RCR is large, the case-base organization should be improved.

# 4 Intelligent Information Retriever with a Case-Based Reasoner

## 4.1 Background

To use financial statistical databases, users must handle multiple databases to get *proper* information. When we use multiple financial statistical databases, we must represent our requirements in *proper* technical terms; select *proper* databases which contain required information; and use specific retrieval commands in each database.

IRS-CBR is used as a front-end system implemented on a personal computer to communicate with a main frame computer on which financial statistical databases are stored. The tasks of IRS-CBR are interpretation and design [Kobayashi 1990-b] in the sense that IRS-CBR first analyzes the user's shallow requirements; then translates them into the concepts of information retrieval; and generates concrete information retrieval commands for a specific database implicitly described in the requirements.

The pre-requirements on the performance of IRS-CBR is to generate proper commands within one or two minutes on a personal computer, because it takes about three to six minutes to get information from databases on a main frame computer [Terano 1992].

## 4.2 System Configuration and Functions

IRS-CBR has two components: a knowledge-based reasoner for general problem solving and a CBR for using past cases. IRS-CBR is also able to store new cases automatically. The system configuration of IRS-CBR is shown in Figure 2.

The current version of IRS-CBR is implemented in Prolog. The case-base of IRS-CBR is simply organized. Each case is stored in the form of the predicate of Prolog language. Variables in the predicate correspond to the indices of the case. To retrieve the case-base, to generalize a new problem, and to translate the problem into the commands, IRS-CBR has five kind of knowledge-bases: Synonym Dictionary, Concept Frames, Approximation Heuristics, Database Selection Heuristics, and Database Frames. These knowledge-bases can be interactively modified if necessary, while using IRS-CBR.

In order to solve a new problem requested by the user, IRS-CBR utilizes past cases three times. If the adaptation of the cases fails, IRS-CBR tries to apply a knowledge-based system to solve the problem. Because the knowledge is incomplete, the system may also fail. In such a case, if the user has enough knowledge, the user can directly input the proper commands to retrieve the databases. In each step, when successful, information retrieval commands are generated and transferred to the main frame. The successfully solved cases are automatically stored into the case-base with generalized indices.

The steps are as follows:

(1) For a given requirement, IRS-CBR retrieves the case-base to directly apply past cases.

(2) If unsuccessful in (1), the requirements are generalized using Synonym Dictionary in the knowledge-bases in order to match the past cases.

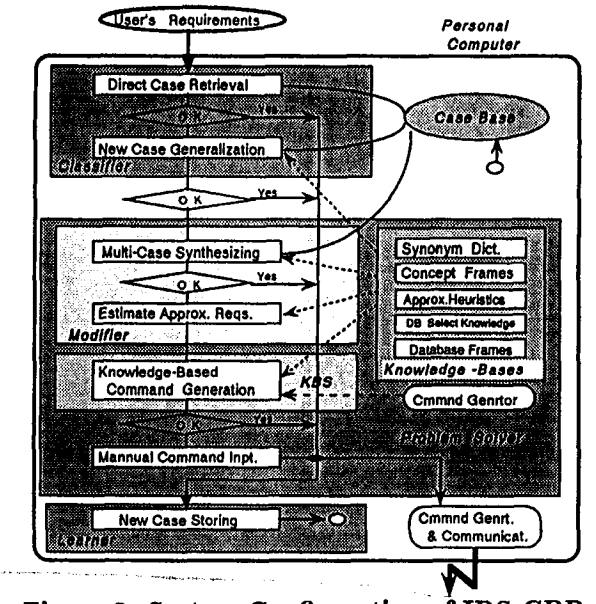


**Figure 2. System Configuration of IRS-CBR**

(3) If unsuccessful in (2), IRS-CBR tries to adapt multiple cases to the given problem, in which both Synonym Dictionary and Concept Frames in the knowledge-bases are used.

(4) If unsuccessful in (3), IRS-CBR tries to convert the requirement into approximate ones using Approximation Heuristics in the knowledge-bases. If there are multiple candidates, IRS-CBR shows them to the user for selection.

(5) IRS-CBR tries to generate information retrieval commands for the specific databases, which is attained by using Database Frames and Database Selection Heuristics.

(6) If unsuccessful in (5), IRS-CBR requires the user to manually input proper information retrieval commands.

(7) IRS-CBR generates the commands and communicates with the main frame. At the same time, the user's requirement is stored in the case-base as a successful case.

(8) The user gets the retrieval results from the main frame.

Figure 3 shows a typical performance graph of IRS-CBR. We must evaluate whether the performance is good or bad. The performance of IRS-CBR can be affected by various factors of the above steps. These are summarized in the following. The time for the step 1 linearly increases when the number of stored cases increase, because the case-organization mechanism is very simple. Each step 2 to 5 requires the constant time because the size of knowledge-bases and the inference steps are considered to be constant. The time for step 6 depends on the difficulty of the requirements and the expertise of the user. However, if the knowledge acquired in step 6 is accumulated as new cases, step 6 can be skipped. Step

7 and 8 definitely depends on the database retrieval and communication time in a main frame computer. The time should be omitted from the validation.

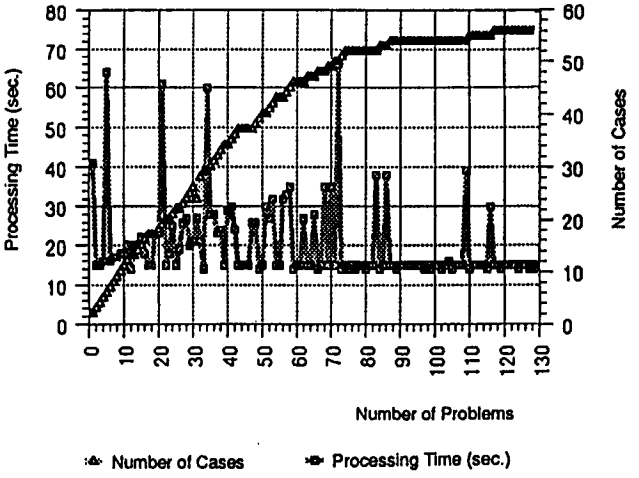


**Figure 3. Typical Performance of IRS-CBR**

# 5 Validating the Performance of IRS-CBR

To validate the effectiveness of the proposed measures and IRS-CBR, we have performed intensive experiments on IRS-CBR from quantitative standpoints.

## 5.1 Experiments on IRS-CBR

The original user interfaces are menu-driven ones and designed for manual input. The interface may cause the disturbance for measuring time. Thus, we have developed special input interfaces, which read user's requirements from files. We have also removed the main frame interface, because it will become hard to estimate the performance if it exists. There remains the user interface for step 6 in the previous section, because there is no way to acquire such kind of knowledge without the interface. We have prepared 60 typical user's requirements. With these preconditions, we have carried out the following experiments on the performance evaluation.

- **Experiment (I): When cases are available or unavailable**

  To evaluate the performance of the case adaptation function, we consider the following situations: One is that there are some initial cases in the case-base which are applicable to user's requirements. The other is that there are also some initial cases in the case-base, however, they cannot be applied to user's requirements. The utility of AAR is estimated from the experiment.

- **Experiment (II): Different requirements**

  This is to evaluate the performance improvement using cases and the performance defect by accumulating cases. In the experiment, the requirements are selected not to be applicable in the following processes so that all are stored in the case-base after processing. The result indicates the utility of SPR.

- **Experiments (III): Requirements without problem solving knowledge**

  The objective is to evaluate the performance of knowledge acquisition functions of IRS-CBR. We have changed the requirements by manually input commands. In such a case, IRS-CBR has no problem solving knowledge in the knowledge-bases, so that we can estimate the knowledge acquisition capability by measuring the processing time when the cases are accumulated. The result also indicates the utility of RCR.

## 5.2 Experimental Results

### 5.2.1 Results when cases are available or unavailable

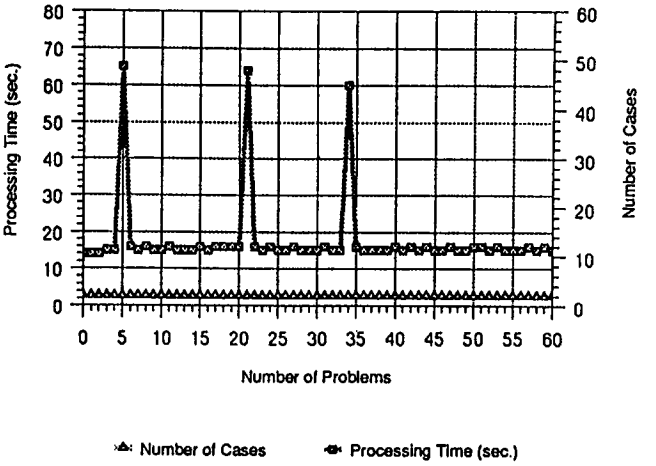


**Figure 4. Performance of IRS-CBR When Cases are Available**

Figure 4 shows the performance results when cases are available. From the figure, it takes about 16 seconds to process a normal new requirement if there exist applicable cases. The normal requirements are processed from step 1 or 2 (Classifier) in the previous section. We have 3 abnormal requirements here, in which the contents of Concept Frames are modified manually.

On the other hand, Figure 5 shows the performance when cases are unavailable. From the figure, it takes about 42 seconds to process a new requirement. It take much more time than the results of Figure 4, because the steps 1 to 5 for knowledge-based problem solving are all executed in this case.

From the two experiments, we find Analogical Adaptation Ratio (AAR) is equal to 0.38. This suggests that CBR in IRS-CBR is effective to improve the problem solving performance.

### 5.2.2 Results for different requirements

Figure 6 shows the performance results for different requirements. In case 1, it takes 41 seconds to process

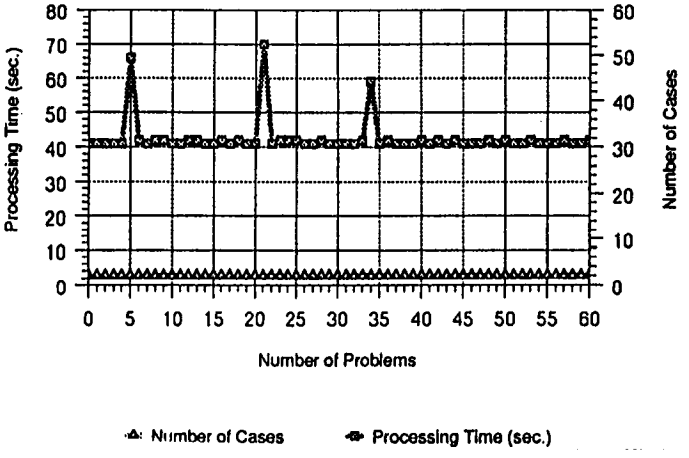


**Figure 5. Performance of IRS-CBR When Cases are Unavailable**

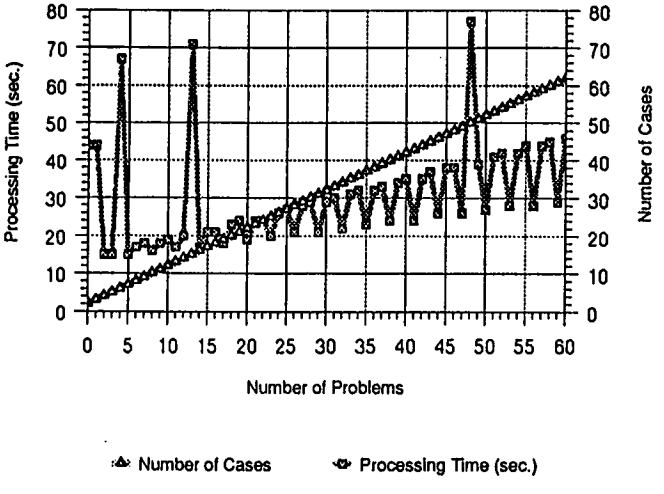


**Figure 6. Performance of IRS-CBR for Different Requirements**

the requirement, because steps 1 to 5 are all executed in this case. Same as the previous results, cases processing taken over 60 seconds are abnormal ones, in which the contents of Concept Frames are modified manually.

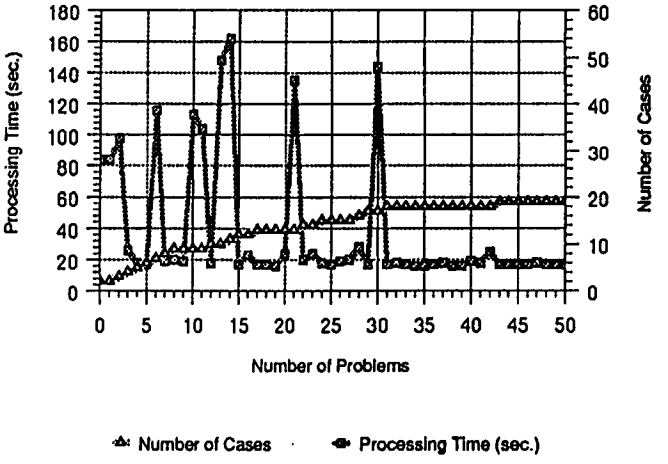


**Figure 7. Performance of IRS-CBR without Problem Solving Knowledge**

As the organization method of the case-base is simple, the processing time linearly increases when the number of cases are accumulated. When the number of cases reaches 60, the processing time becomes the same as in knowledge-based solver without the case-base.

From the Figure 7, we can observe that there are two directions of processing time. This indicates that the requirements are classified into two groups with different kinds of indices. The ratio is 2:1. Thus, Space Partition Ratio (SPR) is equal to 0.67. The result suggests that the case organization mechanism (Learner) should be modified to adapt the features of input problems.

### 5.2.3 Results for the requirements without problem solving knowledge

Figure 7 shows the performance results for the requirements without problem solving knowledge. In the experiment, we have interpreted the requirements defining the new concepts. For example, the following is one of specific knowledge which can be hardly represented in conventional knowledge-based approaches: "The amount of Germany means the sum of the amounts of West-Germany and East-Germany."

It usually takes much more time to process the requirements than the previous experiments, because the experiment requires many user interactions. Especially, in the case taken over 100 seconds to process, we have manually given the corresponding retrieval commands. However, after case 30, the performance becomes stable. The processing time also becomes almost the same as shown in Figure 8. In the stable stages, therefore, memory-based learning or implicit empirical knowledge

acquisition are attained. This is observed from the value of RCR of the later stage of the experiment (RCR is equal to 0.14).

### 5.3 Discussion

The discussion on the experimental results must have two directions. The one is the performance of IRS-CBR. The other is the evaluation of the proposed measures.

Experiments (I) and (II) have been carried out in the artificial conditions. Although the experiment (I) has shown principal benefits of CBR in IRS-CBR, the experiment (II) has suggested the possibility of the worse performance of the CBR component. However, the experiments (III), which have been in the practical conditions, have shown that we can succeeded in speed-up learning for the problems usually required by the user, and that memory-based learning for the problems in which the domain knowledge is hardly represented in symbolic manner. In this sense, using CBR techniques in the domain of IRS-CBR, the performance of the system have been clearly improved. Therefore, we can conclude that it is worth implementing complex CBR components in IRS-CBR.

These results may be obtained by usual white-box approaches. However, using the proposed measures, the results has become clearer. That is, the value of SPR in (II) has suggested the necessity of modification of the case organization, the values of AAR in (I) and (III) have indicated the performance of the modification processes of CBR, and RCR in (III) has shown the performance of memory-based learning To use the measures, however, we must identify the characteristics of input new problems: what kinds of components of a CBR system mainly solve them. If not, we could not separate the effects of AAR and RCR. We could not determine what a CBR system learns.

The limitation of the measures is that they must be used for the performance validation but for the quality of solutions of a CBR system.

## 6 Concluding Remarks

Recently, a number of researches on developing integrated CBR systems with other techniques, [Golding 1991], [Rajamoney 1991], and [Hammond 1989] are remarkable. However, there are few researches concerning the performance validation of CBR systems. Bareiss has discussed the quality of solutions from past cases against the new cases by the classification knowledge of Protos in medical diagnosing [Bareiss 1989]. Becker, et al. have analyzed the indexing problems [Becker 1991]. Veloso, et al. have had experiments on problem solving performance of PRODIGY, and have proposed new indexing techniques [Veloso 1991]. These researches concentrate on quality validations and do not discuss the performance validation measures. In this sense, the measures proposed here are practically important.

In this paper, we have proposed quantitative measures to evaluate the performance of a CBR application system. Then we have described the features of IRS-CBR: a intelligent information retriever with a case-based reasoner for financial statistical databases. To validate the effectiveness of the measures, we have illustrated the results of quantitative experiments on the performance of IRS-CBR.

From the experimental results, we conclude that (1) IRS-CBR has succeeded in both speed-up and memory-based learning in a practical sense, and (2) the proposed measures are useful for the performance evaluation of a CBR application.

Directions for related future work include the application of the proposed measures to various problems, and the exploration of validation measures of solution quality of a CBR system.

## References


[Bareiss 1989] Bareiss, R.: *Exemplar-Based Knowledge Acquisition.* Academic Press, 1989.

[Becker 1991] Becker, L., Guay, T.: Measures for the Evaluation of Case-Based Suggestion. *Proc. Case-Based Reasoning Workshop'91*, pp.170-178, 1991.

[Golding 1991] Golding, A.R., Rosenbloom, P.S.: Improving Rule-Based Systems Through Case-Based Reasoning,*Proc. AAAI-91*, pp. 22-27(1991).

[Hammond 1989] Hammond,K.J.: *Case-Based Planning: Viewing Planning as a Memory Task.* Academic Press, 1989.

[Kobayashi 1990-a] Kobayashi, S., Terano, T., Mizoguchi, R., Motoda, H.: Research Activities of Knowledge Acquisition and Learning in Japan. *Proc. JKAW '90 (1st Japanese Knowledge Acquisition for Knowledge-Based Systems Workshop)*, pp.113-133, 1990.

[Kobayashi 1990-b] Kobayashi S., Terano T.,(eds.): *Knowledge Systems Handbook.* (in Japanese) Ohm Co., Tokyo, 1990.

[Rajamoney 1991] Rajamoney, S. A., Lee, H. -Y.: Prototype- Based Reasoning: An Integrated Approach to Solving Large Novel Problems. *Proc. AAAI-91*, pp. 34-39(1991).

[Riesbeck 1989] Riesbeck,C.K.,Schank,R.C.: *Inside Case-Based Reasoning.* Lawrence Erlbaum, 1989.

[Terano 1992] Terano, T., Nabeta, S.: What a CBR System Learns: Performance Evaluation of a Case-Based Information Retriever. *Proc. 2nd JKAW (Japan Knowledge Acquisition for Knowledge-Based Systems Workshop)*, pp.341-355, Nov., 1992.

[Terano 1993] Terano, T.: A The JIPDEC Checklist-Based Guideline for Expert System Evaluation. forthcoming in O'Leary, D.E. (ed.): *AAAI Workshops on VV&T, 1988-1992*, 1993. (Also available as The University of Tsukuba, GSSM Research Report, No.92-09, 1992.)

[Veloso 1991] Veloso, M.M., Carbonell, J.G.: Variable-Precision Case Retrieval in Analogical Problem Solving.*Proc. Case-Based Reasoning Workshop'91*, pp.93-106, 1991.