Case-Method: A Methodology for Building Large-Scale Case-Based Systems

Hiroaki Kitano
Case Systems Laboratory
NEC Corporation
2-11-5 Shibaura, Minato
Tokyo 108 Japan
kitano@ccs.mt.nec.co.jp

Hideo Shimazu Akihiro Shibata
C&C Information Technology Research Laboratories
NEC Corporation
4-1-1 Miyazaki, Miyamae
Kawasaki 216 Japan
shimazu,shibata@joke.cl.nec.co.jp

Abstract

Developing large-scale systems are major efforts which require careful planning and solid methodological foundations. This paper describes CASE-METHOD, the methodology for building large-scale case-based systems. CASE-METHOD defines the procedure which managers, engineers, and domain experts should follow in developing case-based systems, and provides a set of supporting tools. An empirical study shows that the use of CASE-METHOD attains significant workload reduction in system development and maintenance (more than 1/12) as well as qualitative change in corporate activities.

1 Introduction

This paper describes CASE-METHOD, a methodology for building and maintaining case-based reasoning (CBR:[Riesbeck and Schank, 1989]) systems. CASE-METHOD has been inductively defined through a corporate-wide case-based system actually deployed at NEC corporation. It is now being applied to several case-based systems, ranging from division-wide systems to corporate-wide and nation-wide case-based systems.

Despite increasing expectations for using case-based reasoning as a practical approach to building cost-effective problem-solvers and corporate information access systems, no research has been made on how to develop and maintain case-based systems. In the software engineering community, particularly among practitioners, methodology development or selection is regarded as one of the most important development decisions to make.

In general, software development methodologies define how to organize a project, how each development procedures should be carried out, and how to describe interface between development processes. Often, the methodology provides automated tools, which support some of the development processes involved [Downs et. al., 1988, Wasserman et. al., 1983]. A number of methodologies have been formulated by mainframe manufactures and by consulting firms. Some of these are AD/Cycle by IBM, Method/1 by Andersen consulting, SUMMIT by Coopers and Librant, and NAVIGATOR by Ernst and Young.

If CBR systems are to be integrated into mainstream information systems, a solid methodological support is essential. Unfortunately, however, no methodological support has been provided from CBR community ([Acorn and Walden, 1992] is the possible exception, but only a part of the entire process has been defined). Although there are a few methodologies for building expert systems, such as HSTDEK by NASA[Freeman, 1987], KEMRAS by Alvey project, KADS by ESPRIT, and EX/METHOD by NEC, these methodologies are not applicable for CBR system development, because underlying principles are so different between expert systems and CBR.

Thus, the authors had to develop their own methodology, optimized for case-based systems. CASE-METHOD was designed to be consistent with methodologies for non-CBR systems, so that the corporate information systems division would be able to use the methodology without major trouble. Also, CASE-METHOD was inductively defined based on several CBR projects actually carried out. Thus, CASE-METHOD is already a field-tested methodology. While it is not possible to describe all the details for a full set CASE-METHOD due to space limitation, this paper describes basic components of CASE-METHOD — vision, process definition, and tools.

2 Experience Sharing Architecture

Although CASE-METHOD can be applied to various ranges of projects, from task-specific problem-solvers to nation-wide case-based systems, the main target for CASE-METHOD is corporate-wide information systems. It is acknowledged that the mainstream corporate information system has been designed based on the idea of *Strategic Information System (SIS)*[Wiseman, 1988]. However, how SIS should be designed and how the system should be operated has only been vaguely defined. In addition, SIS confined itself within a traditional information processing paradigm of corporate behavior [Simmon, 1976]. Thus, how knowledge can be transferred and reformulated in the organization has not been the issue of the traditional systems. CBR community, on the other hand, viewed CBR mainly as a new problem-solving mechanism, and have not discussed how the CBR idea impacts corporate information systems.

In order to bridge these gaps, the authors propose the *Experience Sharing Architecture (ESA)* concept. ESA

facilitates sharing of experiences corporate-wide, thereby promoting organizational knowledge creation, and improves core skills in the corporation. This will be attained through the use of case-based systems integrated with mainstream information systems, such as existing SIS. While the authors agree the importance of SIS and effectiveness of the information processing paradigm of corporate behavior, the authors argue that a new dimension should be added, in order to further enhance the power of corporate information systems.

Since people and organization learn from experiences (see [Badaracco, 1991, Ishikura, 1992, Meen and Keough, 1992, Nonaka, 1991, Nonaka, 1990, Senge, 1990] for discussion on corporate knowledge creation and organizational learning), collecting, sharing, and mining experiences collected in form of *case* is the best approach to improving knowledge level and skills of the organization. In [Kitano, et. al., 1992], effectiveness of case-based systems to support organizational knowledge creation and learning, particularly for Nonaka's theory, has been discussed.

CASE-METHOD provides how to build and maintain the system to support the organizational knowledge creation. In addition, CASE-METHOD defines how organizational knowledge creation can be carried out, in the light of modern information technology. It is a methodology to develop case-based systems, as well as a methodology to implement the knowledge creation cycle.

3 Case-Method Cycle

The methodology employs an iterative approach, which allows the system to evolve as process iterates. Figure 1 shows the system evolution cycle in CASE-METHOD. CASE-METHOD defines the system development process, the case-base development process, the system operation process, the database mining process, the management process, and the knowledge transfer process.

System Development Process This process employs a standard software engineering approach, such as waterfall model or flower model [Humphrey, 1989]. As a development methodology, the goal is to design and develop a CBR system, which can store and retrieve a case-base created in the case-base development process.

Case-Base Development Process The goal of this process is to develop and maintain a large-scale case-base. Details will be described in the next section.

System Operation Process This process defines installation, deployment, and user supports for the CBR system. This follows standard software engineering and RDB management procedures.

Data-Base Mining Process Data-base mining will be carried out using the case-base. Statistical analysis, rule-extraction and other appropriate techniques will be applied. The current model defines how to analyze case-base, using standard statistical procedures, and rule extraction, using decision tree [Quinlan, 1992]. This process is a subject for further research.

Management Process This process defines how the project task force should be formed, what kind of organizational support should be provided to the project,

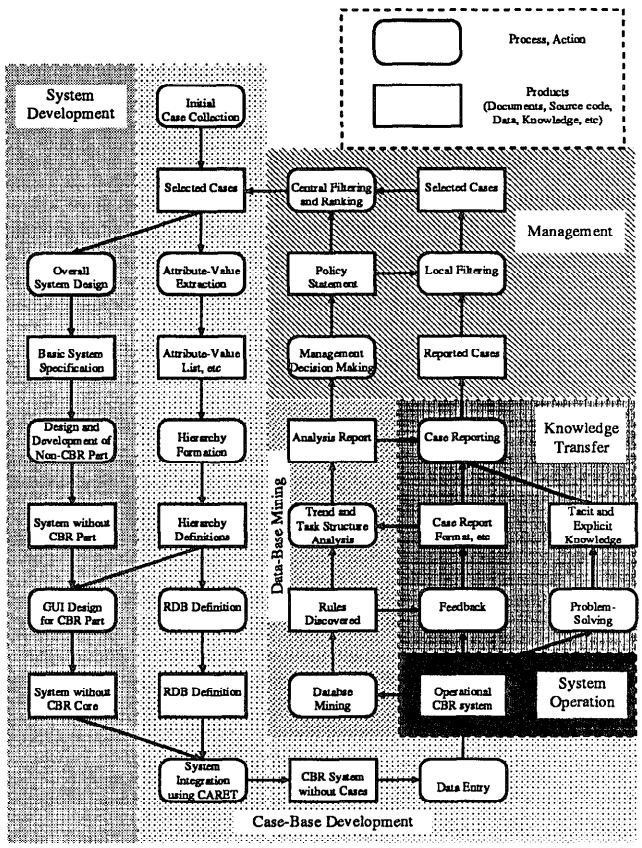

Figure 1: Case-Method Cycle

and what kind of management control should be carried out to obtain a constant flow of high quality cases. The authors have defined a mixed scheme involving bottom-up/top-down control, incentive systems, central/local controls, and case filtering committee. This process should be rearranged for each organization.

Knowledge Transfer Process This process defines methods to transfer knowledge (cases and extracted rules) to related divisions. Network-based system deployment, incentive systems, management control, and newsletter publications have been defined. In addition, how to create a case report format, which is one of the major feed back means, is defined.

These processes form one cycle in the knowledge creation and system evolution. Because of case structure reformulation in the case base development process and rule extraction in the data-base mining process, the quality of knowledge to be stored and transfered improves as the cycle iterates. When the system rearched the maturity stage, the system should hold a case-base consisting of appropriately represented and indexed high quality cases and a set of extracted rules specific to the application domain.

4 Major Process Definitions

Case Collection The first phase in the development requires collecting seed cases. The seed cases provide an initial concept regarding the application domain landscape. In case of SQUAD system[Kitano, et. al., 1992], the authors started with 100 seed cases, to define a crude case format and data structure. In case of the nationwide case retrieval system, the authors are working on several hundred cases from the beginning. As a start up phase in the project, cases are generally collected in unstructured and ununiformed style, such as full-text and other domain-specific forms.

From the second cycle, this phase involves (1) collection of cases which are consistent with the pre-defined case report format, and (2) filtering of cases so that only cases with minimum acceptable quality will be sent to the next phase. Cases are reported in structured style, using pre-defined case report form and full-text with specified writing style.

Products of this phase are (1) a set of case report forms, and (2) a set of case reports in full-text.

Attribute-Value Extraction The goal of the attribute-value extraction phase is to extract all possible elements in case representation and indexing. In the initial cycle, this phase consists of three processes; (1) keyword listing, (2) attribute identification, and (3) value grouping.

The process can be semi-automatic, but a certain amount of human monitoring would be necessary, as new keywords and compound nouns need to be identified by human experts. Each attribute and value is examined, to determine whether or not it is independent from other attributes and values. Ideally, a set of attributes is expected to be a linearly independent set. However, in reality, this is not always possible. Thus, some dependency would be allowed. However, an excessive degree of dependency makes case representation and indexing less transparent.

Products of this phase are (1) a list of attributes, (2) a list of possible values for each attribute, (3) a thesaurus of keywords to be the value of each attribute, and (4) a set of normalized units for problem description and evaluation.

Hierarchy Formation The hierarchy formation phase defines relationships among keywords and attributes. For each attribute identified in the previous phase, a set of keywords has already been grouped. In this phase, relationships between keywords will be defined, mostly by using IS-A relation. The process of defining the relationship will be carried out in both bottom-up and top-down manner. Generally, it starts as a bottom-up process involving sub-grouping a set of keywords, and creates a super-class of one or more keywords. Then, the IS-A relation will be defined between the created super-class and keywords. One or more superclass will be grouped and a superclass for them will be defined. Then, the IS-A relation will be defined between them. This iterative process builds up an IS-A hierarchy in a bottom-up manner. This bottom-up process creates a minimally sufficient hierarchy to cover values for a set of existing cases. However, it does not guarantee whether the defined hierarchy can cope with unknown, but possible, cases. Thus, the top-down process will be carried out to incorporate a set of class and values to cover possible unknown cases. In the top-down process, the domain expert checks to determine whether or not all possible subclass are assigned for each class. If possible subclasses are missing, the missing class will be added.

After a set of hierarchies is defined, the relative importance of each attribute and the distance between individual values will be assigned. Ideally, this weight and distance assignment process needs to be carried out, using a sound statistical and empirical basis. However, in many cases, obtaining such statistical data would be unfeasible. In fact, none of the in-house projects is capable of obtaining such statistics. This is mainly due to the nature of domains and constraints on development and deployment schedules. Plus, in some systems, assigning pre-fixed weights works against achieving the system. Particularly when the user's goals for using the system differ greatly, pre-fixed attribute weights undesirably bias the search space in an unintended fashion. Thus, decisions on how to assign weights and how to value distance measures must reflect characteristics of the domain and an actual deployment plan.

A product of this phase is a set of concept hierarchies created for each attribute. The hierarchies are assigned with similarities between values.

Database Definition and Data Entry Then, database definition will be created using the set of hierarchies just defined. There are several methods to map the hierarchy into relational data-base (RDB). Which method is to be used is up to the system designer. However, CARET case-base retrieval shell supports flat record style database definition, as opposed to structured indexing [Hammond, 1986, Kolodner, 1984], due to the maintenance ease [Kitano, et. al., 1992]. Using RDBMS is an important factor in bringing CBR into the mainstream information system environment. At the end of these processes, a defined RDB contains a set of cases. The system should be operational after this phase.

Feedback The goal of the feedback phase is to provide explicit knowledge to case reporters, so that the quality of cases to be reported could be improved. In addition, it is expected that, by providing the explicit knowledge after extensive knowledge engineering, tacit and explicit knowledge regarding each case reporter may be reformulated in a more consistent manner. This is an important phase in the proposed methodology.

One way of providing a feedback is to create a case report format. The case report format should be created from the hierarchy used to index the case. There are three major benefits for distributing the case report format. First, by looking at items in the case report format, case reporters may be able to understand an overall picture of the domain in which they are involved. In the corporate-wide system, the level of expertise the case reporter has may vary significantly, and some reporter are not aware of the correct classification applicable to problems and counter measures. Distribution of the case report format is expected to improve the quality of reported cases. In fact, improvement in the quality has been confirmed in the SQUAD system applied for the software quality control (SWQC:[Mizuno, 1990]) domain. Second, using the case report format reduces data entry cost. Since all attribute-values are covered in the case report format, a simple bulk data entry strategy can be applied to register reported cases. This leads to substantial cost saving, as will be reported later. Third, by allowing free-form description for items, which can not be represented using the predefined attribute-values, new attributes and values can be identified easily and efficiently. These new attributes and values are added to the indexing hierarchies, and the case report format in the next cycle will include new attribute-values.

The products of this phase is a new case report format to be used for reporting cases in the next cycle.

5 Supporting Tools

A set of tools to support the process has been developed. Some of them are: CARET RDB-based CBR Shell [Shimazu, et. al., 1993], Canae/Yuzu GUI Construction Kit, Hierarchy Editor to help develop concept hierarchies using graphical interface, CAPIT Case-Based Natural Language Interface [Shimazu, et. al., 1992] which generate seed SQL specifications to be used by CARET, and Database Mining Tools to accomplish trend and statistical analysis. Figure 2 shows a list of tools for each part of the system development. The key supporting tool is CARET, the RDB-based CBR shell.

CARET is a CBR shell which operates on commercial relational database management systems (RDBMS), such as ORACLE. When the user specifies attributes and values representing the problem, CARET produces a set of SQL (Standard Query Language) specifications, which will be dispatched to RDBMS. Since SQL cannnot be used to achieve similarity-based retrieval, CARET produces several SQL specifications, ranked by similarity measure calculated using indexing hierarchies. For example, let us assume `Computer` is defined to have subclass `Parallel` and `Serial`, `Parallel` has instances `CM-2` and `PARAGON`, and `Serial` has an instances `SparcStation`. If the user specified `CM-2` as a value for one of the attribute such as `Run-Time-Machine`, a SQL specification with highest similarity should contain `Run-Time-Machine = CM-2`. However, other SQLs with slightly lower similarity value may contain `Run-Time-Machine = PARAGON`. Even lower similarity SQL specifications may contain `Run-Time-Machine = SparcStation`. Such a relaxation strategy has been incorporated in CARET so that nearest neighbour similarity retrieval can be carried out using commercial RDBMS. One question which may be raised is whether or not such an approach can attain reasonable response time in large-scale case-based systems. As will be described later in Fig. 3, a practically acceptable response time has been obtained, using real data.

Part of system	Tools	Process Definition
User Interface	Canae/Yuzu GUI, CAPIT NLI,...	
Case-Base Building	Hierarchy Editor, ...	Case-Method (Case-base building process)
Case-Retrieval	CARET RDB-based CBR Shell	Case-Method (CBR system integration)
Adapation Rule-Based Part	Excore Inference engine, ...	EX/Method, Case-Method
Non-AI Part	CASELAND CASE Tools,...	SEA/I, OMT

Figure 2: Tools and Process Definitions in Case-Method

Users may describe the problem by chosing values for each attribute from menus provided by the graphical user interface, or by sending seed SQL specification. The seed SQL specification may be generated by CAPIT natural language user interface[Shimazu, et. al., 1992], so that users may interact with the system in natural language.

6 Empirical Results

Although several on-going projects employ the methodology and tools described in this paper, an empirical result is reported on the effectiveness of the approach using a corporate-wide case-based system applied for software quality control (SWQC). The project was initiated in 1981 as a corporate-wide quality control project, and the case-based system was introduced recently. The authors have accumulated over 25,000 cases as of December 1992, and 3,000 cases are now being reported every year by over 15,000 active participants. The case-based system was called SQUAD and its motivation and the system architecture have been described in [Kitano, et. al., 1992].

System Development The authors have observed a significant reduction in the system development cost. For this kind of system, the expected workload for developing the entire system (but excluding a knowledge-base) is about 10 man-months. However, the system was completed with less than 4 man-months of workload. Since this workload includes successive up-grading of the CARET CBR shell itself, the real workload for the SQUAD itself is estimated to be about 1.5 man-months. Since CARET reached the well-defined state, the authors expect the next system can be built within a 1 man-month workload. There are two major contributing factors for this workload reduction.

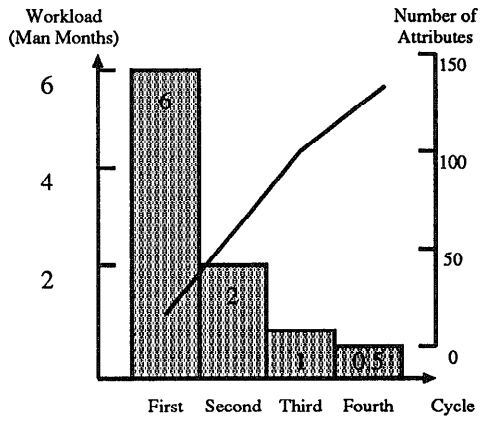

Figure 3: Case-Base Building Workload

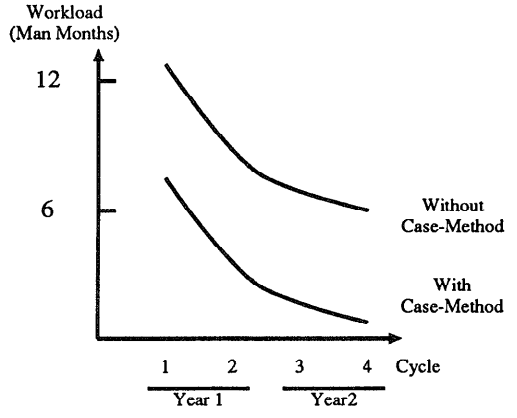

Figure 4: Total Workload

First, use of RDBMS in CARET offered a significant workload saving for building a specific case storage mechanism. All necessary functionalities and performance tuning facilities have been provided by the commercial RDBMS.

Second, Canae/Yuzu, a GUI construction environment, dramatically reduced the workload needed for user interface development. Since SQUAD extensively uses menus and tables for user interface, pre-defined parts for the user interface eliminated requirements for coding these parts of the software. The authors assessment indicates that the user interface development workload was reduced to 1/10.

Case-Base Building and Maintenance Application of the methodology resulted in qualitative and quantitative change in case-base building and maintenance.

On the quantitative side, the authors have observed a reduction in workload for building case-based from cases reported from various divisions. Before the methodology was introduced, the case report format was free-form with about 20 items. One domain expert has been working on the case-base building for her full-time job. Yet, it took almost 6 months to add 1,500 cases reported twice a year. There are two activity cycles in a year. Thus, processing over 3,000 reported cases for each year took a whole year. This is almost 6 man-months workload for 1,500 cases, which represents the cases needed to be processed in one cycle. By introducing the methodology described in this paper, the workload began to decrease. After a fourth cycle, the total workload was reduced to 0.5 man-months for 1,500 cases, 1/12 of initial workload. At this cycle, the number of attributes used reached 130. Figure 3 shows the history of workload reduction. Thus, total system development cost and maintenance cost has been reduced dramatically (Figure 4).

There are qualitative effects as well. As the number of attributes and possible values increases, more case have been covered by a set of values which are already on the case report form. Current coverage is over 95%. Thus, the case-base building process became less expertise-demanding. It was turned into a simple data entry task, which can be automated in the next step. However, there are cases which still need special handing. These are cases which cannot be covered by the values and attributes defined in the case reports. A knowledge engineer on the development team analyzes and registers these cases. At the same time, new values or new attributes are added to the case report form, so that the coverage can be increased in the next cycle.

In addition, quick turnaround for data entry enabled the SWQC division to carry out detailed analysis of the new cases. This also enabled the SWQC division to inspect the quality of cases, using extra-time created as a result of workload reduction.

Run Time Performance The CARET performance on commercial RDBMS has attained a practically acceptable speed. Using Oracle RDBMS on SparcStation2, the average response time for a query to a case-base of up to 1,500 cases (with 130 indexing attributes) is about 2.0 seconds. Figure 5 shows response time for various queries on various case-base sizes. Queries -2 and -3 are normal queries, and query-1 is the worst case performance.

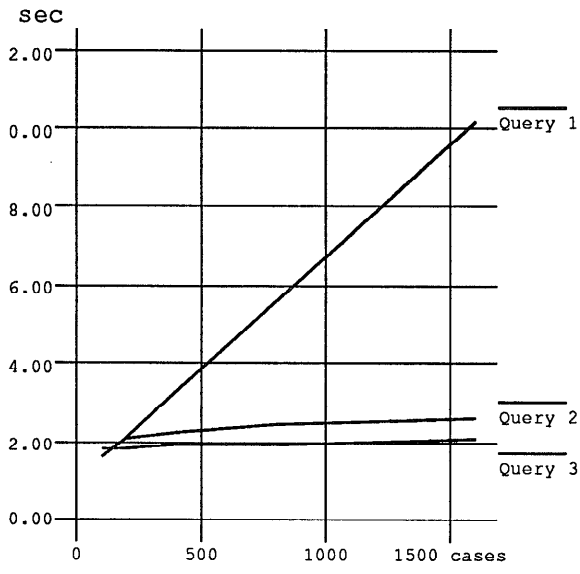

Figure 5: Case-Base Retrieval Time

7 Conclusion

This paper has described CASE-METHOD, a methodology for building large-scale case-based systems. This is the first methodology defined for case-based systems development. The methodology was defined inductively through actual development projects, ranging from task-specific systems to corporate-wide and nation-wide systems. It is the battle-proven methodology.

CASE-METHOD provides a set of process definitions and supporting tools. Empirical study demonstrates that CASE-METHOD effectively reduced system development and maintenance cost, as well as offering qualitative changes in the corporate activities. However, the authors have yet to define an effective means to validate and verify the system behavior due to the same reason pointed out in [Hennessy and Hinkle, 1992]. In order for the proposed methodology to be accepted as a mainstream system development methodology, these issues and consistency with the ISO-9000-3 need to be addressed. However, CASE-METHOD has been sucessfuly deployed, and would be the first step toward a methodology for building large-scale case-based systems.

References

[Acorn and Walden, 1992] Acorn, T. and Walden, S., "SMART: Support Management Automated Reasoning Technology for Compaq Customer Service," *Innovative Applications of Artificial Intelligence 4,* AAAI Press, 1992.

[Badaracco, 1991] Badaracco, J., *The Knowledge Link,* Harvard Business School Press, 1991.

[Downs et. al., 1988] Downs, E., Clare, P., and Coe, I., *Structured Systems Analysis and Design Method,* Prentice Hall International, 1988.

[Freeman, 1987] Freeman, M., "HSTDEK: Developing A Methodology for Construction of Large-scale, Multi-use Knowledge Bases," *NASA Conference Publication 2492,* NASA/Marshall Space Flight Center, 1987.

[Hammond, 1986] Hammond, C., *Case-Based Planning: An Integrated Theory of Planning, Learning, and Memory,* Ph.D. Thesis, Yale University, 1986.

[Hennessy and Hinkle, 1992] Hennessy, D. and Hinkle, D., "Applying Case-Based Reasoning to Autoclave Loading," *IEEE Expert,* Oct. 1992.

[Humphrey, 1989] Humphrey, W., *Managing the Software Process,* Addison-Wesley, 1989.

[Ishikura, 1992] Ishikura, Y., *Building Core Skills of the Organization,* NTT Publishing, 1992 (in Japanese).

[Kitano, et. al., 1992] Kitano, H., Shibata, A., Shimazu, H., Kajihara, J., and Sato, A., "Building Large-Scale and Corporate-Wide Case-Based Systems," *Proc. of AAAI-92,* San Jose, 1992.

[Kolodner, 1984] Kolodner, J., *Retrieval and Organizational Strategies in Conceptual Memory: A Computer Model,* Lawrence Erlbaum Assoc., 1984.

[Meen and Keough, 1992] Meen, D. and Keough, M., "Creating the learning organization," *The McKinsey Quarterly,* No. 1, 1992.

[Mizuno, 1990] Mizuno, Y., *Total Quality Control for Software,* Nikka-giren, 1990 (in Japanese).

[Nonaka, 1991] Nonaka, I., "The Knowledge Creating Company," *Harvard Business Review,* Nov.-Dec., 1991.

[Nonaka, 1990] Nonaka, I., *A Theory of Organizational Knowledge Creation,* Nikkei, 1990 (in Japanese).

[Quinlan, 1992] Quinlan, R., *C4.5: Programs for Machine Learning,* Morgan-Kaufmann, 1992.

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

[Senge, 1990] Senge, P., *The Fifth Discipline: The Art & Practice of The Learning Organization,* Doubleday, 1990.

[Shimazu, et. al., 1992] Shimazu, H., Arita, S., and Takashima, Y., "Design Tool Combining Keyword Analyzer and Case-Based Parser for Developing Natural Language Database Interfaces," *Proc. of COLING-92,* Nantes, 1992.

[Shimazu, et. al., 1993] Shimazu, H., Kitano, H., and Shibata, A., "Retrieving Cases from Relational Data-Base: Another Stride Towards Corporate-Wide Case-Based Systems," *Proc. of IJCAI-93,* 1993.

[Simmon, 1976] Simmon, H., *Administrative Behavior, 3rd edition,* Free Press, 1976.

[Wasserman et. al., 1983] Wasserman, A., Freeman, P., and Pacella, M., "Characteristics of Software Development Methodologies," (Eds.) Olle, T., Sol, H., and Tully, C., *Information Systems Design Methodologies,* North Holland, 1983.

[Wiseman, 1988] Wiseman, C., *Strategic Information Systems,* Irwin, 1988.