A Form Driven Object-Oriented Reverse Engineering Methodology: A Tool and Case

Cheonsoo Yoo [1] and Heeseok Lee [2]

[1] 4th Systems Development Center, Agency for Defense Development, P.O. Box 132, Songpa, Seoul 138-600. Korea, E-mail: csyoo@chollian.net, Tel: +82-2-3400-2586, Fax: +82-2-403-3512
[2] Graduate School of Management, Korea Advanced Institute of Science and Technology, 207-43 Cheongryang, Dongdaemoon, Seoul 130-012, Korea, E-Mail: hlee@unitel.co.kr, Tel: +82-2-958-3615, Fax: +82-2-958-3604

ABSTRACT

This paper introduces form-driven object-oriented reverse engineering (FORE) methodology, a supporting tool and case. Two primary objectives of the FORE are to capture the form knowledge, and to derive a conceptual object-oriented model by using the knowledge of form semantics. Most of the knowledge about business applications can be compiled from business forms and the users' interaction with the legacy system. A reverse engineer (or designer) produces a conceptual model by using the form knowledge during the overall phases of FORE methodology. The recovered conceptual model consists of an object model and scenario. The object structure is expressed via an object model based on the CRC (Class, Responsibilities, and Collaborators) cards, and a scenario diagram is introduced to describe a sequence of operations. These results correspond to the class diagram and the sequence diagram in UML (Unified Modeling Language), respectively. A prototype system for capturing form knowledge and supporting reverse engineering phases in the FORE is developed to aid reverse engineers. To demonstrate their practical applicability, the methodology and systems are applied to a real-life military application. The methodology is found to be effective for recovering and modeling a conceptual information from legacy applications.

KEYWORDS

Reverse Engineering, Object-Oriented Modeling, Screen Forms, ECRC-Tool, CRC Card, UML

1. Introduction

The solution to the legacy systems seems to improve them so that they may be more maintainable or transformed with the use of a new technology [13][10]. To resolve some problems of legacy systems, a reverse engineering technology is introduced and is tried on real projects or applications. Reverse engineering encompasses wide areas of tasks related to understanding and managing legacy systems [3][14]. One of the major tasks is to identify the semantics of an existing software system. Then it is another important task to derive high-level abstractions of the recovered semantics from the existing system. The term "reverse engineering" is borrowed from hardware development area [6], where it is typically applied to the process of discovering how other companies' products work and are made. In software engineering, the term is used to describe the process of discovering how our own systems work.

The major objectives of this paper are summarized as follows: (i) To recover conceptual semantics by the use of forms from the legacy application, which is unstructured and lacks of document (ii) to propose Form Driven Object-Oriented Reverse Engineering (FORE) methodology [8][9] based on forms which are closer to end users (iii) to develop a prototype system to help a reverse engineer capture form knowledge and design a conceptual model during the reverse engineering phases in the case of developing a new system, and (iv) to apply our methodology to a real case in order to validate it.

In this paper, to overcome the shortcomings described, we propose a reverse engineering methodology not only to capture form knowledge automatically, but also to derive a conceptual object-oriented model [2][12]. Our methodology is a reverse engineering method based on forms. It attempts to a get complete conceptual model made of both data and process view of an application. Furthermore, we adopt the view of object encapsulating data and process. The methodology consists of five phases - form usage analysis, form object slicing, object structure modeling, scenario design, and model integration. First, in the form usage analysis phase, active structure of forms and user interaction information are acquired by using an automated support program during the period of executing a user application, and stored into a form knowledge store. The next phase, form object slicing, analyzes form knowledge in store and then slices form knowledge into unit objects by analyzing group fields, block fields, and operations in the form knowledge store. Third phase, the object structure modeling, finds objects, assigns basic attributes to them, and makes initial object diagram. Especially, structural and collaboration relationships among objects are identified and established. In this phase, ECRC (Extended Class Responsibilities Collaborators) card [7][15] and RDD (Responsibilities Driven Design) [5] mechanisms are used to represent object structure model. The fourth phase, scenario design, derives a flow of events by using operations in the form knowledge store and describes them sequentially as a scenario. In fifth phase, model integration, the recovered semantics from legacy application are further refined. For example, generalization and aggregation hierarchies are added to the class structure diagram, and scenarios are further structured.

To support our methodology, a prototype system for capturing form knowledge automatically and supporting the remaining four phases of FORE methodology, is developed and also applied to an example of a legacy application system. Although we focus on the reverse engineering, our work can be extended to software analysis and design, knowledge representation, and many other applications in which the concept of object orientation is used.

We present a real case to show what are the most important to be prepared to solve the real world legacy problem. MND (Ministry of National Defense) in Korea, a government organization, has many large scaled legacy systems to be maintained. Most of these systems were written by COBOL or FORTRAN and various 4^{th} generations programming languages, and have suffered from typical legacy problems. Aiken et al. [1] summarize problems of legacy applications in DoD as follows. These problems are similar to those of MND.

(i) There is no document or lack of consistency of documents;
(ii) High cost is needed to maintain existing legacy applications;
(ii) They have developed with the use of old technologies.

MND's current problems do not seem to be unique to MND and many other business companies have similar ones. Most organizations that have legacy systems may experience all or parts of the problems. Under these situations, we try to search for possible solutions.

2. A Reverse Engineering Support System

A reverse engineering support system is developed on the basis of FORE methodology. As shown in Figure 1, the system consists of two subsystems. One is Agent Program for Acquiring Form Knowledge (APFK). This subsystem is to capture form structure and information about user interaction with legacy system during the execution of a legacy system. The captured information is stored in the form knowledge store. The other subsystem is ECRC-Tool for the support of the modeling based on ECRC technique. Four phases in FORE methodology are supported by ECRC-Tool. This subsystem uses information in the form knowledge store as input source. All outputs generated during the four phases of FORE methodology are stored in object model store.

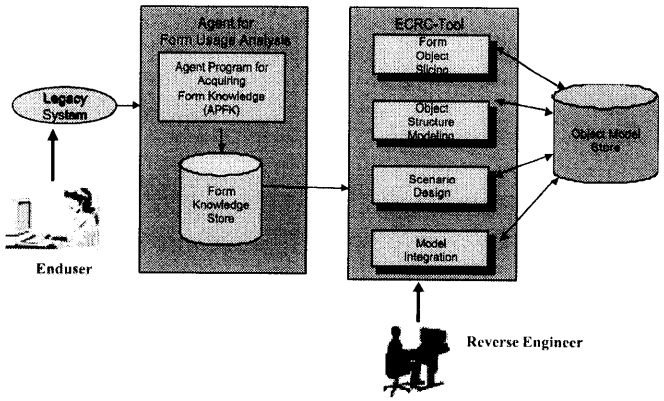


Figure 1 : A Support Tool.

2.1. An Agent Program for Acquiring Form Knowledge

All the information about the form are collected in the Form Usage Analysis phase,. The objective of this phase is to collect as much knowledge about form structures and form operations as possible. Major output in this phase is the form knowledge to be collected by an agent program. Form knowledge consists of two parts, a set of information about the form structure, and the other set of information about form operation and the database access information.

APFK is the support tool which captures the form knowledge during the execution of legacy application by user. It acquires the form structure information, the user interaction, and the database access related information according to the types of events during the execution of an application. Such acquired information is accumulated in the form knowledge store and used in later phases. Layout of form knowledge store is illustrated in Table 1. The meaning of each column is as follows. In this table, form name and field name means the name of form and form field, respectively. Parent name stands for super type of the object identified. Actor is an active object and expressed by Boolean value. Attributes for form field are expressed by F_type and there are four type of attributes such as undefined, atomic, block, group, and computation. SEQ is the order of form fields presented in the form. Result is the value assigned to form field while the application is running.

Form	Field	Parent	F_TYPE	ACTOR	O_TYPE	SEQ	RESULT
Name	Name	Name					
String(30)	String(30)	String(30)	String(20)	Boolean	String(20)	Integer	String(50)
.	.	.	UnDecided	T/F	UnDecided	1 . N	.
			Atomic		UserKeying		
			Block		SysProvided		
			Group		DBAccess		
			Computation		Computation		

Table 1 : Layout of Form Knowledge Store.

Figure 2 shows a configuration of APFK. In Figure 2, a legacy system is assumed to be centralized system that consists of host computer, local DBMS, and dummy terminal or PC based terminal. To get form structure and user interaction information, we install client wrapper on the front end of VDT (Video Display Terminal). The monitor agent captures such information whenever a certain event is triggered, and the captured information is transferred to MCI (Monitor Client Interface). Also, in order to acquire database access information, we use a log file. Whenever the log file is updated, the monitor agent in front of DBMS checks the status of the log file, and transfers such information to MCI. Then, MCI stores the returned information into form knowledge store. In Figure 2, DI stands for database interface and CI is client interface. WI and UI is wrapper interface and user interface, respectively.

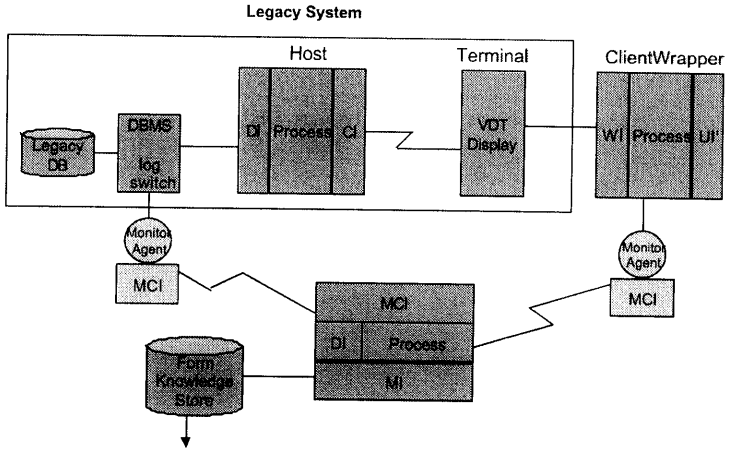


Figure 2 : A Framework of An Agent Program for Acquiring Form Knowledge.

2.2. A Reverse Engineering Support Tool based on ECRC Card

ECRC-Tool helps a reverse engineer derive an object model by the use of form knowledge. It is made up of four modules. First, in the form object slicing module, information about the form structure is read and initial form object model is made. This module flattens the form object by interacting with reverse engineer. If the reverse engineer decides that only flattened objects exist, flattened objects are stored in object model store.
Next, the object structure module helps the reverse engineer identify relationships among objects. Structural relationships are found by analyzing the association of container object and contained objects. Also, collaboration relationships are discovered by analyzing computation field. Collaboration relationships are established among objects containing the required fields to produce the results of computation. The reverse engineer decides whether these relationships are reasonable or not according to the information produced by the module supporting the phase of the object structure modeling. If all relationships are found and they are related to objects, a completed object structure model is stored in the object model store.
Third, in the scenario design module, information about operations in the form knowledge store is read to describe a scenario. The reverse engineer describes a sequence of operation according to the order of operations to be occurred. This sequence of operation is modeled using object process action scenario and is stored in the object model store. Currently, this module is left unfinished.
Finally, the model integration module integrates the resulting models derived from each form. During the integration, a reverse engineer checks homonyms and synonyms of the object names. If homonyms or synonyms are found, the reverse engineer handles them by the method of naming conflict resolution [11]. The final result generated by the support of ECRC-Tool is accumulated in the object model store. follows. Following Figure 3 is an example of form knowledge for work order form [4]. Figure 4 shows complex form object made by the selection of form knowledge.

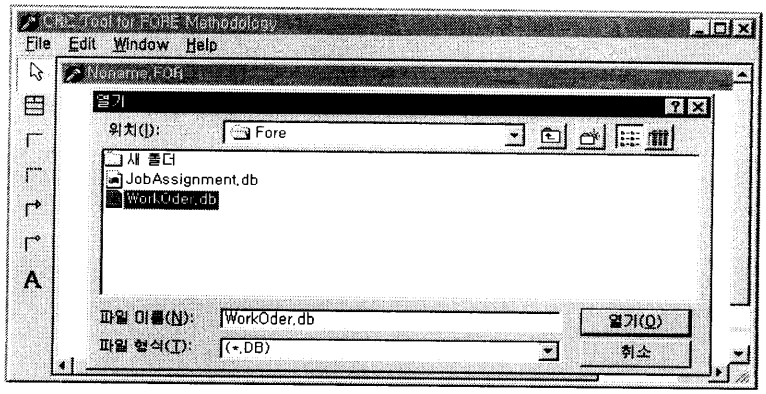


Figure 3 : Form Knowledge Input.

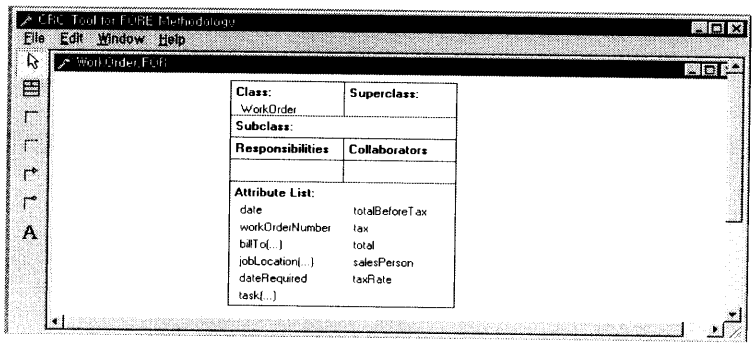


Figure 4 : Form Object for WorkOrder Form by ECRC-Tool.

3. A Case Study

3.1. Case Description

This chapter introduces a case of typical legacy application of MND(Ministry of National Defense) in Seoul, Korea. MND is a government organization for managing national defense affairs. MND has been using MND Budget System for several years. MND Budget System consists of six major subsystems and eleven budget areas as shown in Figure 5. In order to produce annual MND budget plan, all of the subordinate organizations of MND require their own annual budget to MND through the use of MND Budget System. Budget Managerial Department of MND produces total budget plan by using budget data from the subordinate organizations.
Current budget system was developed by a simple migration of the old system, which was a single-user PC version. A relational database system and on-line menu-driven interface using screen forms was applied under the multiple user environment. But the old MND Budget System is hard to maintain because it does not have documents and its source code is not readable without applying a systematic methodology. Most of tables in a relational model are not normalized. That is a legacy application made up of spaghetti code and tables in abnormal form.

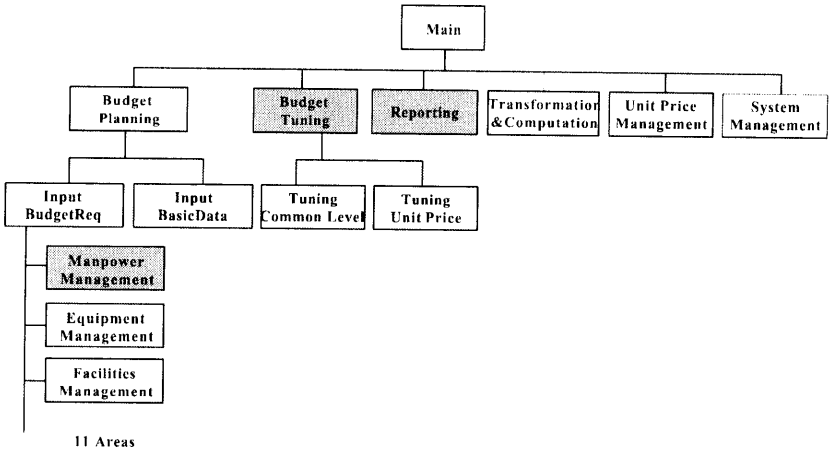


Figure 5 : Overall Structure of MND Budget System.

Whole MND Budget System includes one hundred sixty five input forms, eighty nine forms, and one hundred twenty five relational tables. It is too complex to handle as a single case. Therefore, our case is restricted to two subsystems, which are budget tuning subsystem and reporting subsystem, and single budget area, which is manpower management. In total, this case includes twenty forms. Even though the case is limited in scope, it covers most of major forms and functions. Grayed parts in Figure 5, budget tuning subsystem, reporting subsystem, and manpower management area, are included in this case.

3.2. Results

Twenty five objects and thirty one objects are derived from budget tuning subsystem and reporting subsystem, respectively. Sixteen objects are derived from manpower management area. However, because most of objects are redundant, the number of objects is reduced. All the objects from reporting subsystem are found in objects

from budget tuning subsystem. Therefore, they are merged in a single object model. In the process of object slicing, container objects are identified. They are used to link the objects by the use of structural relationship. Collaborative objects can be found by analyzing object operations in each form process.
ECRC-Tool can show objects and their relationships graphically. Figure 6 is the object structure model produced through ECRC-Tool. Structural relationship and collaborative relationship only are shown in this figure. Object names are expressed in an abbreviated form and attributes are omitted for the simplicity of presentation.

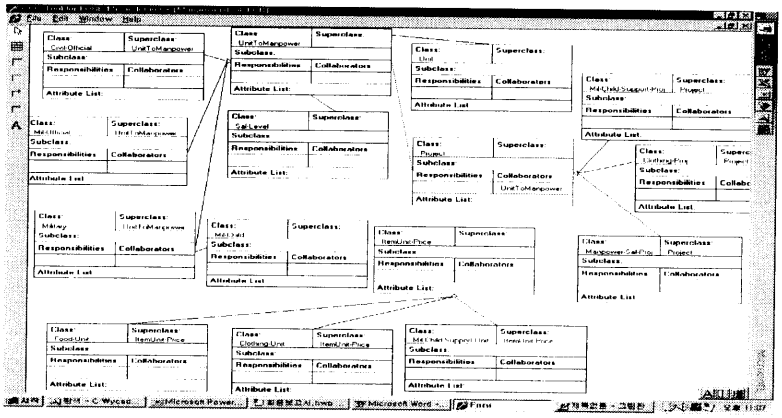
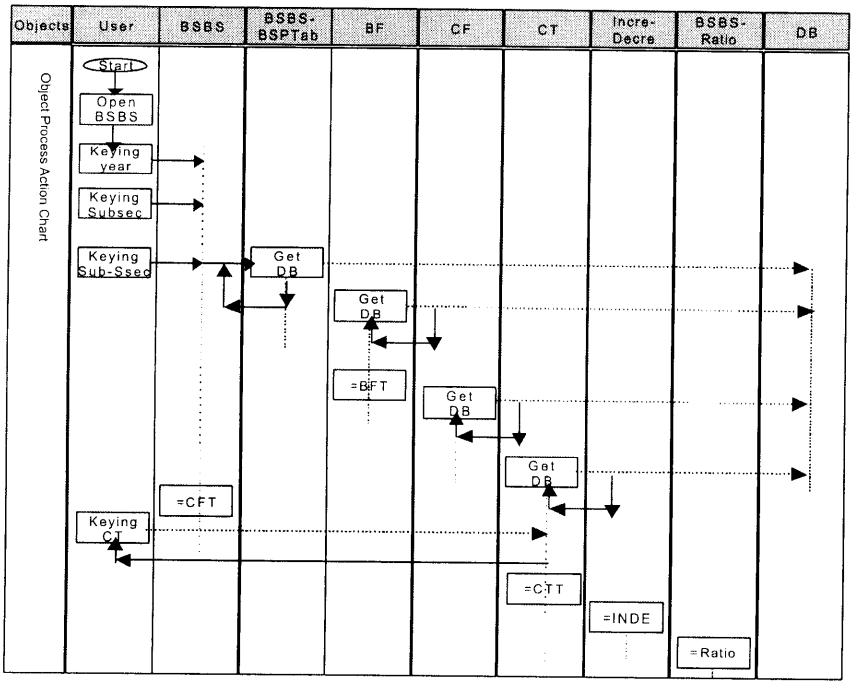


Figure 6 : Object Structure Model for Manpower Management Area

Information about form process operation is captured from user interaction with BudgetSummarybyStructute form process in a case. In general, keying and database access operations are major information. A sort of first cut scenario is generated by the use of the information. Figure 7 shows object process action scenario diagram.



* Acronyms: BF (BudgetBeforeFix), CF (BudgetCurrentFix), CT (BudgetCurrentTuning), BFT (BeforeFix-Total), CFT (CurrentFix-Total), CTT (CurrentTuning-Total), Subsec (Subsection), Sub-Ssec(Sub-Subsection), BSBS (BudgetSummarybyStructure)

Figure 7 : Object Process Action Scenario Diagram for BudgetSummarybyStructure Form Process

3.3. Discussion

The application of FORE methodology to the MND Budget System mainly focuses on deriving its object model. The result implies that the FORE methodology can contribute to the documentation of the legacy applications effectively.

The form-driven approach and object-oriented modeling technique adopted in FORE are new attempts in reverse engineering. The proposed methodology does not replace all of the existing reverse engineering methodologies. Instead, it can complement them, because the results derived through FORE methodology can be used to maintain legacy application or to migrate old system into a new system. In addition, the methodology should be applied to real life projects differently. Process model such as object sequence diagram, scenario, and use case diagram is weak, because program source codes are not considered as input source. Other input sources would be needed to model the process. FORE works well with legacy applications consisting mostly of screen based information handling systems. Budget, stock management, and order entry system are such applications. In many organizations, these constitute the core applications with heavy transaction volume and non-trivial maintenance efforts. On the other hand, for applications that contain the functions with complex algorithms, FORE methodology may have constraints.

As seen in the MND Budget case, the derived models can be used as system documents. In this case, the derived results may be useful as a starting point for further revision.

Further research is required as follows. The form knowledge captured automatically through APFK is insufficient to recover more complete object model. Thus, reverse engineer or user intervention is required to recover object operation. Otherwise, other input sources have to be found. On the other hand, since MND Budget System is too large, all the forms in that system are not applied consistently. Fortunately, there are many similar forms with the same fields in the case. In general, we found that most forms may be used repeatedly. Therefore, our case with two major subsystems and one area show that it can cover most of the system. For example, all the objects derived from reporting subsystem are the same as those of budget tuning subsystem. Even though other forms are added to the case, we may come to the conclusion that objects are not increased in a large number. Only few objects might be derived relatively. In spite of these limitations, our methodology is likely to help the responsible person in charge of budget system to understand the system or document it.

4. Conclusions and Future Work

This paper proposes the FORE methodology based on the object-oriented technology. The FORE methodology recovers the object oriented conceptual model by the use of form structure and user interaction with legacy application. Without reentering input information of legacy application, FORE methodology can analyze a real application system. In order to capture working status of legacy application based on forms, an agent program is developed for capturing form knowledge (APFK) in the form usage analysis. Moreover, the proposed methodology and the APFK are applied to a real-life case and our methodology appears to be working successfully. Also, ECRC-Tool is developed to aid four phases of reverse engineering. Our approach can be applicable not only to maintenance problems of legacy application in the organization, but also to migration or reengineering various kinds of legacy applications.

The major contributions of this research can be summarized as follows. First, our method uses forms as an new input source. Especially, screen forms of legacy application are used. Second, it can capture form knowledge automatically. An get program is used to acquire the form knowledge composed of form structure and user interaction information during the execution of legacy application. Third, it recovers the object model encapsulating both data and process from legacy applications. The object model is also usable as the target conceptual model of reverse engineering research. Fourth, it can map resulting model to formal object-oriented model, UML [12]. In order to use more useful the recovered semantics, it is needed to represent them in a formal model such as UML. Fifth, it can be applied to a real-life case with moderate size. The results show that our method is practical applicability to other reverse engineering projects.

The following are some further research areas on the basis of this paper. First, various input sources are required to derive more complete model. It is important to search for other information except the existing form knowledge from legacy applications while is run. Second, to derive the object method with process logic and object sequence diagrams is weak. Third, support tool has to be extended. Agent program for acquiring form knowledge(APFK) as well as ECRC-Tool is needed to make up for the weak points in the current system. More phases need to be automated in order to reduce human intervention.

REFERENCES

1. Aiken, P., Muntz, A. and Richards, R. (May 21 – 23, 1993), "A Framework for Reverse Engineering DoD Legacy Information Systems," *Proceedings Working Conference on Reverse Engineering*, Baltimore, Maryland, IEEE Computer Society Press , 180-191.

2. Booch, G. and Rumbaugh, J. (1995), *Unified Method for Object-oriented Development Document Set Version 0.8*, Rational Software Corporation.

3. Chikofsky, E. J. and Cross II, J.H., "Reverse Engineering and Design Recovery : A Taxonomy," *IEEE Software*, 7(1), Jan. 1990, 13-17.

4. Choobineh, J., Mannino, M. and Tseng, V. (Feb. 1992), "A Form-Based Approach for Database Analysis and Design," *Communications of the ACM,* 35(2), 108-120.

5. Hutt, A.T.F.(1994), *Object Analysis and Design - Description of Methods*, A Wiley-QED Publication, John Wiley & Sons.

6. Kim, Y-G. (Winter 1997), "Improving Legacy Systems Maintainability," *Information Systems Management*, 14(1), 7-11.

7. Lee, H., Lee, C. and Yoo (1999), C., "A Scenario-Based Object-Oriented Hypermedia Design Methodology," *Information & Management*, 36(3), 121-138.

8. Lee, H. and Yoo, C. (4-7 July 1999), "A Form-Driven Object-Oriented Reverse Engineering Methodology," *5th International Conference of the Decision Science Institute*, Athens, Greece.

9. Lee, H. and Yoo, C., "A Form Driven Object-Oriented Reverse Engineering Methodology," *INFORMATION SYSTEMS*, Forthcoming (Accepted 12 December, 1999).

10. Markosian, L., Newcomb, P., Brand, R., Burson, S. and Kitzmiller, T., "Using an Enabling Technology to Reengieer Legacy Systems," *Communications of the ACM*, 37(5), 58-70.

11. Navathe, S.B. and Awong, A.M. (1987), "Abstracting Relational and Hierarchical Data with A Semantic Data Model," *Proceedings 6th Int. Conf. on the Entity-Relationship Approach,* North-Holland, Amsterdam, 305-336.

12. Quatrani, T. (1998), *Visual Modeling with Rational Rose and UML*, Addison-Wesley.

13. Umar, A. (1997), *Application (RE) Engineering – Building Web-Based Applications and Dealing with Legacies*, Prentice Hall.

14. Waters, R.C. and Chikofsky, E. (May 1994), "Reverse Engineering: Progress Along Many Dimensions," *Communications of The ACM*, 37(5), 23-24.

15. Wilkinson, N.M.(1995), *Using CRC Cards – An Informal Approach to Object-Oriented Development*, SIGs Books.