Pergamon

Information Systems Vol. 25, No. 3, pp. 235–259, 2000

Printed in Great Britain
0306-4379/00 $20.00

PII: S0306-4379(00)00017-X

A FORM DRIVEN OBJECT-ORIENTED REVERSE ENGINEERING METHODOLOGY†

HEESEOK LEE[1] and CHEONSOO YOO[2]

[1]Corporate Information Systems Laboratory, Graduate School of Management, Korea Advanced Institute of Science and Technology, 207-43 Cheongryang, Dongdaemoon, Seoul 130-012, Korea

[2]2nd Technology Research Center, Agency for Defense Development, P.O. BOX 132, Songpa, Seoul 138-600, Korea

(Received 25 November 1998; in final revised form 1 March 2000)

Abstract — Legacy applications are valuable assets that should be integrated into business systems in succeeding generations. In order to take advantage of these applications, progressive companies seek to improve current operations by reverse engineering. This paper proposes the form driven object-oriented reverse engineering (FORE) methodology by using forms to recover semantics of legacy applications. Forms are exceptionally easy to understand because of the user-oriented nature of the contents of business. This form driven object-oriented reverse engineering methodology consists of five different phases: form usage analysis, form object slicing, object structure modeling, scenario design, and model integration. Knowledge of the form structure and the user's interaction between legacy application is compiled to extract the design semantics. This application demonstrates the practical usability of the FORE methodology by transforming the resulting object models into well-known UML-based models.

Key words: Reverse Engineering, Object-Oriented Modeling, Screen Forms, Legacy Applications, UML

1. INTRODUCTION

Today's business is rapidly changing, and companies are therefore developing new information architectures and systems. Most business organizations use business applications that support operations successfully, but existing applications, which have stable capability in present operations, are no longer suitable. Typically, these legacy applications are not likely to support new business environments and rules sufficiently, because most of them were developed many years ago by the use of conventional information technologies such as COBOL programming languages and file systems under the host-based platform.

Legacy applications are valuable assets; it is probably not an appropriate measure for most business organizations to give up on the legacy system in the near future [24]. However, legacy systems have been experiencing typical problems [1, 29]. First, the legacy applications utilize system-specific information, but they sometimes work without any documents. As a result, it is difficult to have further developments because of lack of the business knowledge about legacy applications and the understanding of systems. Second, they are complex and expensive to maintain. In particular, their function can not be divided into more manageable components. Third, most legacy systems use technologies developed in the 1970s. They are host-based, and not easily integrated with current needs.

There are several strategies to resolve the problems of the legacy applications, but any strategy that either gives up the applications or develops new ones from scratch is not desirable. Access to the legacy systems and data warehousing is the way to utilize the legacy data in the new systems. Another solution to the legacy problems is to develop and improve the applications so they are easier to maintain. In this case, the migration strategy is crucial to recover the semantic of the legacy applications [43]. To solve the problems, a reverse engineering technology is introduced. One of its major tasks is to understand the semantics of existing software; the other is to abstract the recovered semantics to the abstractions of higher level [15]. The term "reverse engineering" is borrowed from the field of hardware development [44], where it is generally applied in an effort to discover how other companies' products are made. In this paper, we define reverse engineering in

†Recommended by Peri Loucopoulos

terms of acquiring a higher conceptual abstraction from the physical systems. The object-oriented modeling technique is employed, because objects can provide consistent services to both legacy and next generation applications. A main core of the object-oriented modeling is to identify objects. It may be easier to reengineer the current systems by using the objects.

This paper proposes a form driven object-oriented reverse engineering (FORE) methodology. 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. Forms play the role of the user interface for the end-users. Forms are the most widely used official communication objects in many business organizations [42]. The CODASYL End User Facility Committee (EUFC) has recognized the importance of forms, and recommends the forms-based approach as the leading tool for the user interface [16, 17]. Most of the knowledge about business applications can be compiled from business forms and the user's interaction with a legacy system. A reverse engineer (or designer) produces a conceptual model by using the form knowledge during the overall phases. The recovered conceptual model consists of an object model and scenario. The object structure is expressed via the object model based on the CRC (Class, Responsibilities, and Collaborators) cards, and a scenario diagram is introduced to describe a sequence of operations [45, 28]. These correspond to the class diagram and the sequence diagram in UML (Unified Modeling Language), respectively [35].

2. REVERSE ENGINEERING

In this approach, the existing studies regarding reverse engineering are explored. Four criteria are chosen to describe the existing studies. First, the reverse engineering and the forward engineering are considered in terms of the direction of the engineering process. Second, existing studies are surveyed by what the input sources for each methodology are. In other words, the methodologies are classified in accordance with the categories that utilize the data, the source programs, the forms and all the factors in the existing systems as the input sources. Third, existing studies are analyzed from the standpoint of the way in which the methodologies see the system. Final criterion is what conceptual model should be applied in order to transform the recovered information into a semantic model.

Figure 1 depicts the concepts of reverse and forward engineering [6]. Forward engineering is the process of developing or modeling a system. Generally, forward engineering is called a development methodology. Forward engineering produces more physical outputs than those of previous steps in the process of the system development do. Reverse engineering is the backward process used to recover higher level conceptual elements from physical systems.

Table 1 compares existing studies based on the four criteria described earlier. They are categorized into reverse engineering and forward engineering based on the basis of development sequence. A detailed discussion of forward engineering is excluded in this paper because reverse engineering is our primary interest. The studies on the reverse engineering are explored further, according to the orders indicated in Table 1. Input source means that input data are used in the methodology. Additionally, methodologies are classified in terms of view such as data, process, and object. Methodologies are also categorized according to the target model that is used to describe the recovered semantics.

2.1. Reverse Engineering Utilizing Database Schema as Original Input Sources

Studies on the reverse engineering of databases are classified into three groups: transforming the relational schema to an ER model, the network schema to an ER model, and the hierarchical schema to an ER model. First, reverse engineering methodologies for transforming the relational database schemas to ER model are based on functional and inclusion dependency [2, 21]. The conceptual model of the original database is recovered via the catalog information in the RDBMS (Relational Database Management System) and tuples in the legacy database. In Batini et al. and Navathe et al., relations are classified with respect to their primary key [5, 32]. Such a classification can be performed through an interaction with the user. The relational schemas are then interpreted