Overview Of Automated Reverse Engineering Of Legacy Database System

Tanu Arora
Department of Computer Science
Hi-Tech Institute of Engineering and Technology, Ghaziabad, U.P

Abstract

Legacy system software generally comprises a database (sometimes in the form of a set of files) and a collection of application programs in strong interaction with the former. They constitute critical assets in most enterprises, since they support business activities in all production and management domains. Legacy systems: they typically are one or more decade old, they are very large, heterogeneous and highly complex. In other some instances, new systems are developed to replace older ones that have become too complex or outdated and which resist further modification and evolution. The research work presented here is to focus on the difficult, real-world form of leverage, namely starting with legacy system assets and evolving them. This work is aimed to investigate the supporting tool for automated database reverse engineering of legacy system evolution. It will also cover the model developed for migration and their optimality for transforming a legacy system programs into modern database.

Keywords: Legacy System, reverse engineering, data migration

1. Introduction

Today's system asset is contributing as a greater part of enterprise's total asset, because in order to keep system asset updated enterprises are regularly focusing on various processes and technologies under build and maintenance schedule. Increasingly, software is also being viewed as an asset that represents an investment that consistently grows up in value rather than a liability whose value depreciates over time.

In early stages when systems are very small and their touch points to an organization's activities and functionality are very few, it is possible to restructured or even replace the existing system or subsystem which is not able to satisfy the need of enterprise. But as the time passes through, the system grows up and spread its wings to integrate with various functionalities and activities of the enterprise and become a substantial investment for it whose replacement is more difficult. So, a major issue in today's scenario is how to build software assets that will provide leverage in building future software asset.

The widespread use of computer technology over several decades has resulted in some large, complex system that have evolved to a state where they significantly resists further modification and evolution. Such rigidness to adapt for future changes poses considerable problems (like inflexibility, isolation, non-extensibility, lack of openness etc.) in front of information system manager and organizations as well, if any of the system stops working the business may pulverize to a halt. So today's major issue is how to build database asset that will provide leverage in building future database asset. One of the reasons that the situation is changing so rapidly is the emergence of integrating infrastructures. With improved integration we have seen the World Wide Web (the Web) and electronic commerce flourish. Where one information systems were isolated and difficult to access, they can now be accessed using the Web and interfacing software.

A legacy information system can be defined as "any information system that significantly resists modification and evolution" [1]. A legacy has all or some of the following properties:

- It consist of millions of lines of code
- It is maintained by a large team of engineers from several generations
- It contains the code that originated several years ago and
- It is expected to function for many more years to come.

Existent examples of such type of systems can easily be found within many domains such as automation, automotive, and telecom industries. In such systems, a large effort must be spent on keeping complexity at acceptable levels. If the complexity is allowed to increase without bound, the life expectancy of these systems will be reduced or/and costs will dramatically increase. A legacy information system controls the information flow within the entire organization and is the main medium for presenting the consolidated view of accumulated information about its business.

The rest of the paper is organized as follows: Problem Definition is discussed in Section 2. Section 3 is an overview of Related work done in the area. Section 4 throws light on the proposed work to be done in future and the methodologies th3at will be used to convert legacy database is given in Section 5. Section 6 is the conclusions.

2. Problem Definition

Maintaining the legacy real time software is a multifaceted problem, in order to keep a log life expectancy it is required that the software is carefully engineered to improve long term software quality and reduce the need for reengineering. These information systems are numerous problems to their host organizations. Some of these problems are:

- These systems perform their activities on obsolete hardware platform which is slow and very expensive to maintain.
- Maintaining of such legacy system software is most critical when no documentation is available about it.
- Identifying the faults is very time consuming job because of lack of understanding of the internal workings of the system.
- Integration efforts are greatly hampered by absence of clean interface.
- Working with legacy system is very difficult if not possible to expand.

Considering all such problematic issues is necessary because demand of short time to market limits the time budget available for careful engineering, and the complexity and lack of documentation/models of the software make efficient engineering difficult. Many products require a highly versatile software that can serve many purposes (e.g., product line architectures or an industrial robot with multiple hardware configurations and operating environments), which increases complexity. Recovering the required knowledge and control of poorly documented software components is the main goal of software reverse engineering [2].

In response of all the above dictated problems various solutions have been proposed. These solutions can be classified into following three categories: redevelopment, wrapping and migration. Redevelopment involves rewriting existing application. Wrapping involves developing a software component called wrapper that allows an existing software component to be accessed by other components who need not be aware of its implementation. Legacy Information System Migration allows legacy systems to be moved to new environments that allow information systems to be easily maintained and adapted to new business requirements, while retaining functionality and data of the original legacy systems without having to completely redevelop them. This leads to the following main problems studied in this resea3rch work:

1. Can the validity and accuracy of extracted models be quantized?

2. Are the overheads of model extraction acceptable?

3. Can the method of model extraction be evaluated?

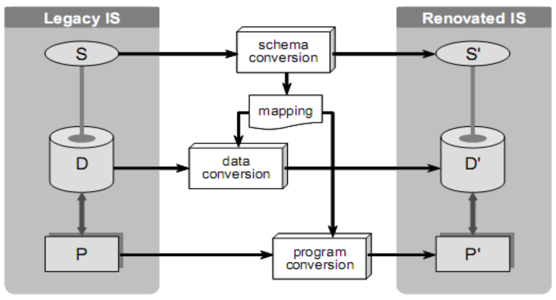
3. Related works

Legacy information system migration is a major research and business issue; few comprehensive approaches to migration have been developed.
For example, Tilley and Smith (1995) [3] discuss current issues and trends in legacy system reengineering from several perspectives (engineering, system, software, managerial, evolutionary, and maintenance). They propose a framework to place reengineering in the context of evolutionary systems. The butterfly methodology proposed by Wu et al. (1997)[4] provides a migration methodology and a generic toolkit to aid engineers in the process of migrating legacy systems. This methodology, which does not rely on an incremental strategy, eliminates the need of interoperability between the legacy and target systems.

There is more than one way to migrate a legacy software system. Some approaches are quite straightforward and inexpensive, but lead to poorly structured results that are difficult to maintain. Others, on the contrary, produce good quality data structures and code, but at the expense of substantial intelligent (and therefore difficult to automate) code restructuring. We have built a reference model based on two dimensions, namely data and programs. Each of them defines a series of change strategies, ranging from the simplest to the most sophisticated.

Following figure is defining the first database system migration process. [4]

Figure 1: Overall view of the database first system migration process



System migration consists in deriving a new database from a legacy database and in further adapting the software components accordingly [5].Considering that a database is made up of two main components, namely its schema(s) and its contents (the data), the migration comprises three main steps: (1) schema conversion, (2) data conversion and (3) program conversion. Figure 1 depicts the organization of the database-first migration process, which is made up of sub processes that implement these three steps. Schema conversion produces a formal description of the mapping between the objects of the legacy (S) and renovated (S') schemas. This mapping is then used to convert the data and the programs.

Alternatively, **Young (1970)** proposes a procedural data structure mapping technique whereas **Sibley and Taylor (1970)** suggest a similar technique, but propose using a nonprocedural approach. Another important contribution was a PhD dissertation by **Smith (1971)** who began to address generalized issues of data translation. A common feature of the 1970s research is a focus on the definition of common languages for the purpose of defining data, storage, and mapping processes.

Youn and Ku (1992) provide a concise but rather insightful overview of many of the main issues of data migration along with some helpful examples. The article contains many of the primary issues that should be considered during migration. Consequently, Youn and Ku discuss extraction and loading, followed by transformation and data integration. As part of initial planning process, they emphasize the necessity of developing a conceptual model of the source system which can then be used to develop a model of the target system. The importance of schema integration is also addressed by **Elmasri, Navathe, and Larson (1984, 1986)**.[6]

Hudicka (1999) also provides a good overview of the phases for data migration. His breakdown may be slightly different than Youn and Ku's, but both articles provide useful starting points. Hudicka points out that in the case of migrating from legacy systems which are based on hierarchical databases, the migration process needs to be planned especially carefully, since many of these systems do not enforce referential integrity, while two cornerstones of this older structure – de-normalization and redundancy – are in precise contradiction to more modern relational theory.

Brodie and Stonebraker propose an approach called the "Chicken Little Methodology" (1995) – an eleven-step strategy for migration, employing a series of gateways. With this approach, the legacy and target systems are operated in parallel during the operation. The target system is small at the outset, but grows during the migration process until it replaces the legacy system. For example, a "forward gateway" is created which enables the legacy application access to the new system. At the same time, there is also a "reverse gateway" for the target application to have access to the legacy system. The authors recognize the overall complexity of this system, though, and submit that this complexity still presents a technical problem and ongoing research challenge

4. Proposed Work

The proposed work is to:

1. To examine the tools methods and technologies of automated programmed analysis to recover structure and constraints of a database.
2. To study about the migration methodologies and their optimality for transforming a legacy system programs into modern database.
3. To study about adaptation for migration to modern database by legacy system. Discussion about the adoption of legacy database migration techniques.
4. To perform the consistency check to ensure the reliability between evolving database schema and associated queries.

5. Methodology to be used

Three different paradigms can be followed to perform the research in the field of computer science and application (Peter Wegner, 1976). These paradigms are as follows:

5.1 The mathematical method: Here, abstraction of phenomena (e.g. computers, programming languages, algorithms), and reasoning about this abstraction are used to obtain information about the phenomena. For example, in deductive reasoning, a set of assumed premises are logically (i.e., mathematically) proven to lead to a conclusion. One could say that deductive reasoning is a proof of the fact that a cause leads to an effect. The conclusion (or the effect) is then shown to be true if and only if the premises (or the cause) are valid.

5.2 The empirical method: Here, knowledge projected on a hypothesis is derived from experiments and other methods of data collection. For example, experiments can show that a set of independent variables are controlling a set of dependent variables. The experiments should be controlled, i.e., they should be constructed such that there are no confounding variables (i.e., variables that should be in the set of independent variables, but are not included there). Also, it should be clear that the dependent variables are valid measures of the phenomenon targeted by the hypothesis. Finally, external validity, that several experiments indicate the same result, should be established. This form of triangulation is made to increase the confidence in the result.

5.3 The engineering method: Here, the efficient fulfillment of a specific set of specifications and requirements are in focus. This is achieved by the conception of, e.g., a device, an algorithm, or a method. Essential in the engineering method is to by experiment or formal proof provide evidence of the fulfillment of the requirements.

In this work, we will be using all of the above methods to solve our set of problems:

Engineering Method: For automatic modeling and implementation of a tool suite for proper evaluation.

Mathematical Method: for checking time automata and to formulate a comparison measure for quantifying the difference between two entities, which can be systems or models.

Empirical method: the tool suite has been evaluated in a state of practice industrial system, as well as in a controlled experimental study.

6.Conclusion and Future work:

In this paper we described an overview of Legacy Database system, problems associated in its migration and different automated models used for Reverse Engineering of Legacy Database System. As proposed in future we will be examining the legacy systems programs in maintaining database reverse engineering and related evolutions. To study about the migration methodologies and their optimality for transforming a legacy system programs into modern database.

References:

[1] Brodie, M. and M. Stonebraker (1995), Migrating Legacy Systems, Morgan Kaufman, San Francisco, CA

[2] Chikofsky, E. J., Cross, J. H., 1990. Reverse Software 7 engineering and design recovery: taxonomy. IEEE (1), 13–17.

[3] Bergey, J. K., Northrop, L. M. & Smith, D. B. (1997). Enterprise Framework for the Disciplined Evolution of Legacy Systems. Technical Report CMU/SEI-97-TR007,Carnegie Mellon University/Software Engineering Institute.

[4] Bisbal, J.; Lawless, D.; Bing Wu; Grimson, J., "Legacy Information systems: issues and directions," *Software, IEEE* , vol.16, no.5, pp.103,111, Sep/Oct 1999

[5] Brodie, M. & Stonebraker, M. (1995). Migrating Legacy Systems: Gateways, Interfaces, and the Incremental Approach. Morgan Kaufmann.

[6] Elmasri, R & Navathe, S. B. (1984). Object Integration in Database Design. Proceedings of IEEE Conference on Data Engineering. Los Angeles.

[7] D. Aebi, ’Data Re-engineering - A Case Study’, Proceedings 1st East-European Symposium on Advances in Databases and Information Systems (ADBIS’97), September 1997, Springer-Verlag electronic Workshops in Computing, Ed.: C.J.van Rijsbergen.

[8] H. M. Sneed, 'Encapsulating Legacy Software for Use in Client/Server Systems', Proceedings 3 Working Conference on Reverse Engineering, November 1996, pp. 104-119

[9] Meir Manny Lehman, Programs, life cycles and the laws of software evolution. Proceedings of the IEEE, 68(9):1060–1076, September 1980.

[10]Meir Manny Lehman, Laws of software evolution revisited. In Proceedings of the 5th European Workshop on Software Process Technology, pages 108–124, October 1996.

[11]Peter Wegner. Research paradigms in computer science. In Proceedings of the 2nd International Conference on Software Engineering, pages 322–330. IEEE Computer Society Press, October 1976.