A Dual-Spiral Reengineering Model for Legacy System

Xiaohu Yang, Lu Chen, Xinyu Wang
College of Computer Science and Technology
Zhejiang University
Hangzhou, China
{yangxh, bluelucy, wangxinyu}@zju.edu.cn

Jerry Cristoforo
Enterprise Information Services
State Street Corporation
Boston, USA
jacristoforo@statestreet.com

***Abstract*—Computer Technology has been in existence for many years; as time goes on there is an increasing number of outdated systems. Legacy System Reengineering has become a critical effort in many technology organizations. Several reengineering models have been developed, such as the Byrne Model and the Evolutional Model. These models have addressed some of the issues in the reengineering process, but other issues remain. This paper presents a dual-spiral reengineering model, which performs as a cyclic approach, and addresses issues that those models do not, making the reengineering process more successful. This model has been successfully applied to a legacy financial software system reengineering effort.**

Index Terms—Dual-Spiral Model, Legacy System, Reverse Engineering, Software Reengineering

I. INTRODUCTION

Legacy system is a term first introduced in the 1990s. It tends to be a system which has been in use by an organization for more than ten years with technology that has been outdated. Any organization with an existing legacy system will eventually need to evaluate various options. One option is to search for a vendor based system with equivalent functionality. Another approach is to re-write the system utilizing a new platform. In this paper we discuss an evolutionary approach for legacy system migration.

There are two options to migrate legacy systems to new platforms or frameworks [1]. One approach is the complete redevelopment of the legacy system with new functional specifications and use cases. The other is an evolutionary migration of the current legacy system, which is called reengineering. The disadvantage of the first approach is high risk, high cost, with long development and conversion schedules. Furthermore, understanding the business logic encoded in the software and the corresponding documentation is a complex effort. While an evolutionary migration may take longer, the ability to minimize risk is a crucial benefit.

Much research has been done to make the reengineering process more effective and efficient. Popular reengineering models include the Byrne Model [2], and Evolutional Model [3], which give approaches that have benefit to the reengineering process but have short-comings as well. For example, it is very difficult to add new requirements in the Byrne Model.

This paper will present a new model – the Dual-Spiral Reengineering Model. It is based on a spiral model of software development and enhancement [4], and tries to resolve issues found in the Byrne and Evolutional models. We also present a case study of applying the Dual-Spiral Model to a legacy financial system reengineering.

II. RELATED WORK

As leading research, the Byrne Model [2] has been active for a long time and is still utilized today. The general idea is described in Fig. 1

Using this model, reengineering starts with the source code of an existing legacy system, and follows with reverse engineering on the legacy system to produce the system abstraction. After that, forward engineering, which includes four levels: conceptual; requirements; design and implementation, concludes the entire reengineering process.

This model established that a reengineering process should involve the entire software system. For this reason, the target system (reengineered system) must be frozen until the process has been completed; in other words, no changes are possible during its execution [5]. This is rarely an acceptable approach in most business environments.

Based on the characteristics of different projects, there are also other different reengineering models, such as RENAISSANCE [6], Hybrid Reengineering [7], Pattern-based Software Reengineering [8], and Iterative Reengineering of Legacy System [9].

One reengineering model that deserves mention is the Evolutional Model [3]. It has addressed many issues in the Byrne Model. This evolutionary-based model makes full use of the spiral model of software development and enhancement provided by Boehm [4], where a system evolves in incremental patterns until it is completely reengineered [3].

The basic principles of the Evolutional Model include dividing modules, then beginning spiral procedures for every module and terminating after all modules have been altered.

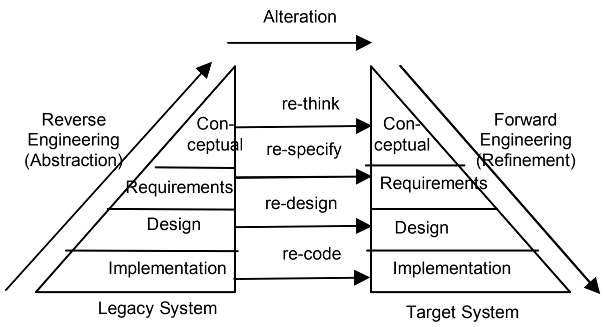

Figure 1. General Model for Software Reengineering

While the Evolutional Model has the advantage of introducing incremental changes throughout the process, it has its own short-comings as well:

1) It is not suitable for the reengineering process when the migration is between two totally different technologies, which are unable to work together. For example, if the migration is from C to C++, the Evolutional Model works fine; but if the migration is from Power Builder to Java/J2EE, it can not be applied;

2) It may cause some architectural conflicts in the target (reengineered) system. During the reengineering process, the requirement that the reengineered modules must work with the other legacy system modules is very important to the reengineering process. In order to support downward compatibility, two sets of interfaces will remain as well as the corresponding legacy structures and protocols. When the target system is complete, the entire structure will include the new technology, but the interfaces between modules may retain the legacy structure.

The objective of this paper is to address the issues described above, and in general, make the reengineering process more effective and efficient.

III. Dual-Spiral Reengineering Model

The Dual-Spiral Reengineering Model is described in Fig. 2.

The main workflow in Dual-Spiral Reengineering Model requires that the two systems (legacy one and target one) work together, and move the functionality (not the modules) from the legacy system to the target system step by step, as in the spiral model. During the entire process, the active functionality in the legacy system is in a decremental pattern, and the active functionality in the new target system is in an incremental pattern, as Fig. 3. This process facilitates the addition of new features while transitioning the existing functionality, which will be discussed later. A communication proxy is added between the two systems to facilitate them working as a complete system from the overall user perspective. The implementation of the proxy can vary based on the specifics of the architectures involved, e.g. scripts, database triggers, etc.

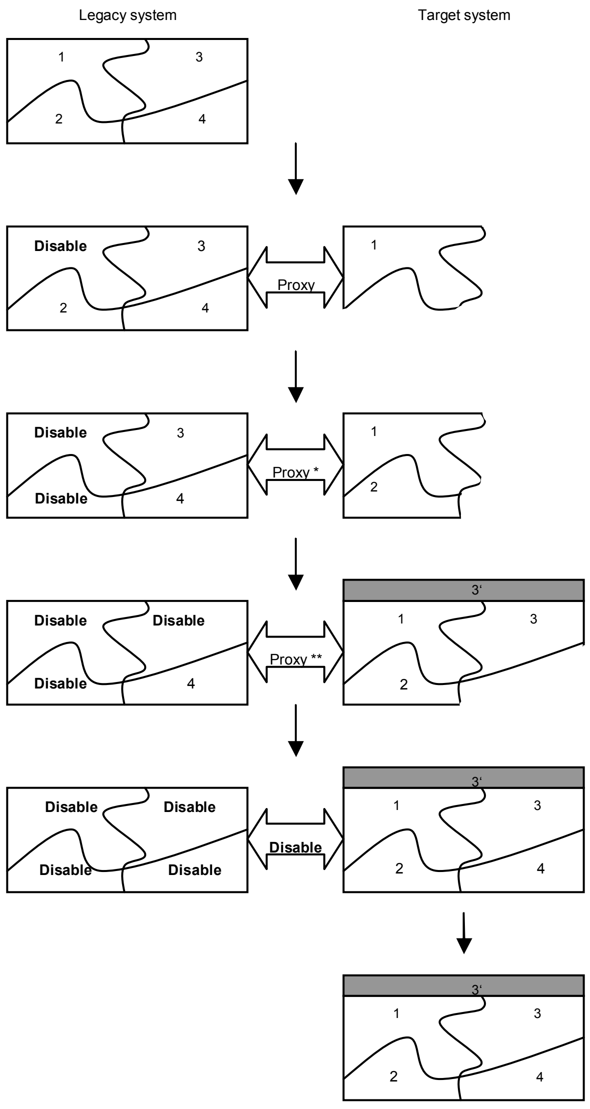

Figure 2. Dual-Spiral Reengineering Model

The reengineering process in Dual-Spiral Model is divided into three main steps: divide the functionality, start the spiral procedure, and terminate the spiral procedure. The following notations are used in describing the Dual-Spiral Model:

S_1 : Legacy System

S_2 : Target System

F_i : Functionality in Legacy System;

f_j : Old Functionality in Target System which perform the same as in Legacy System;

f_k' : New Functionality in Target System

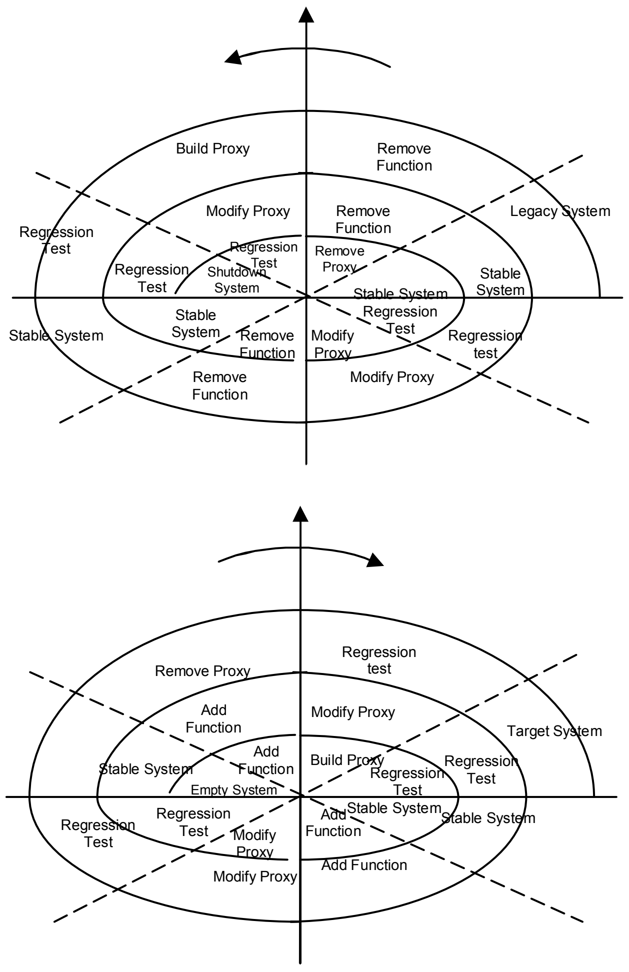

Figure 3. Decremental Legacy system Pattern and Incremental Target system Pattern in Dual-Spiral Model

A. *Divide the Functionality*

One of the biggest differences between Evolutional Model and Dual-Spiral Model is that Dual-Spiral Model is based on **functionality** division, rather than module division. At the beginning, the legacy system will be analyzed and divided into a list of functionalities. The result will be:

$$S_1 = \sum_{i=1}^{n} F_i \ ; \ S_2 = 0$$

B. *Start the Spiral Procedure*

In fact, the sub-reengineering procedure in the Dual-Spiral Model is very suitable to using the Byrne Model, like reverse engineering and then forward reengineering. Every spiral procedure will almost follow the procedures as:

```
while ($S_1 \neq 0$) {
    Select one or more functionalities $F_i$ in $S_1$ ;
    Shut down functionalities $F_i$ in $S_1$ :
        $S_1 = S_1 - \begin{cases} F_i, & \text{if } f_j = F_i \\ \sum_{i=n_1}^{m_1} F_i, & \text{if } f_j = \sum_{i=n_1}^{m_1} F_i \\ F_i, & \text{if } \sum_{j=n_2}^{m_2} f_j = F_i \end{cases}$
    Add same functionalities $f_j$ in $S_2$ :
        $S_2 = S_2 + \begin{cases} f_j, & \text{if } f_j = F_i \\ f_j, & \text{if } f_j = \sum_{i=n_1}^{m_1} F_i \\ \sum_{j=n_2}^{m_2} f_j, & \text{if } \sum_{j=n_2}^{m_2} f_j = F_i \end{cases}$
    if (need to introduce new functionalities){
        $S_2 = S_2 + \sum_{k=n_3}^{m_3} f_k'$
    }
}
```

In terms of functionality transition, we consider three situations during the reengineering process in every spiral procedure.

First is **one-to-one**, which means we implement one functionality in the Target System with one functionality in Legacy system, as

$$S_1 = S_1 - F_i, \ S_2 = S_2 + f_j; \ \text{if } f_j = F_i$$

Second is **one-to-many**. This means that we implement one functionality in Target System, which is equivalent to more than one functionality in Legacy System as:

$$S_1 = S_1 - \sum_{i=n_1}^{m_1} F_i, \ S_2 = S_2 + f_j; \ \text{if } f_j = \sum_{i=n_1}^{m_1} F_i$$

The last situation is **many-to-one**. It happens when we need to implement more than one functionality to be equivalent with only one functionality in Legacy System. This could be described as:

$$S_1 = S_1 - F_i, \ S_2 = S_2 + \sum_{j=n_2}^{m_2} f_j; \ \text{if } \sum_{j=n_2}^{m_2} f_j = F_i$$

After the transition of functionality from the Legacy System to the Target System, the Dual-Spiral Model allows us to add new functionality into Target System freely and easily. The part marked 3' in Fig. 2 has implied this feature. One simple presentation for this piece of work could be:

$$S_2 = S_2 + \sum_{k=n_3}^{m_3} f_k'$$

Except in the work above, an effort should be made to build a proxy between the legacy system and the new target system. In every sub spiral procedure, the proxy needs to be modified.

After the development in every sub procedure, the legacy system (which has shut down part of reengineered functionality), and the new system (which has implemented new functionality) should work together, and regression testing should be performed. In addition thorough user acceptance testing should be performed to validate business functionality. Only after the end users sign-off on the business functionality should we continue with the next sub procedure

C. Terminate the Spiral Procedure

When all legacy system functionality has been moved into the new target system and been approved by the users, the entire spiral procedure could be terminated. At this point, the legacy system should be shut down and the communication proxy should be removed. Finally, the target system will be running as a standalone system. The final result will be:

$$S_2 = \sum_{j=1}^{n'} f_j + \sum_{k=1}^{m'} f_k' \; ; \; S_1 = 0$$

D. Advantages of Dual-Spiral Reengineering Model

The main advantage of the Dual-Spiral Model is that the step by step spiral procedure facilitates the addition of new functions into the new target system. In addition, the model performs a cyclic approach for incrementally growing a system's degree of definition and implementation while decreasing its degree of risk [10]

The Dual-Spiral Model avoids the problems which were introduced by the Evolutional Model, and is suitable for many different types of technology reengineering. Whether the conversion is from C to C++ or from Power Builder to JAVA/J2EE, the Dual-Spiral Model is an appropriate and useful approach. Moreover, the communication proxy between the legacy system and the new target system avoids the situation legacy structure persists in the newly reengineered system.

IV. CASE STUDY

Four years ago, we initiated a Trade Order Management System reengineering project. The existing legacy system was built with Power Builder and Sybase, and was constrained by obsolete technology, an outdated design, and limited performance throughput and maintainability. Our purpose was to reengineer the existing legacy system to a state-of-the-art web-based application, which made use of current industry-standard products such as Web Sphere, Java and Oracle.

This Trade Order Management System is a global application that is used in different locations by different groups of users. It was not practical to roll out a new system to all users at the same time, which meant that the Byrne Model was not an option for this reengineering effort. Also, due to the technical transition from Power Builder to JAVA/J2EE, it was difficult to use the Evolutional Model as well. However, the differing functionality in this application had the advantage of being relatively insular from one type of user to another. This meant that, in most cases, transactions entered in one processing site rarely crossed over into another. As a result, we made the decision to roll out the application on a function-by-function basis and in a staged approach. The Dual-Spiral Model was used as the model for this reengineering effort.

Since the system mainly focused on order management, we divided the functionality according to the order life cycle: trade entry, price acquisition, and trade execution. Database triggers were used as a proxy, moving the data between the two databases to facilitate compatibility. At this time, four phases have been performed and 95% of the functionality has been moved into the new J2EE application. In addition, we have added two major new features. In the very near future, we will move all functionality into the new target system and shut down the old legacy system. When the database triggers are removed, the reengineering effort will be complete.

V. CONCLUSION

This paper presents a new model – the Dual-Spiral Reengineering Model. It is based on a spiral model of software development and enhancement, and addresses issues in the Byrne Model and Evolutional Model. With this model, the functionality of the legacy system can be smoothly transitioned to the new system and at the same time, support new features and functional enhancements. Therefore, we have successfully applied this new model to a legacy financial system reengineering effort.

ACKNOWLEDGMENT

This work is part of Trade Order Management System reengineering project. This project is funded by State Street Corporation, USA. It is collaboration between Zhejiang University, China and State Street Corporation, USA.

The content of this paper contains the opinions and conclusions of the authors only and do not necessarily represent the position of State Street Corporation or its subsidiaries, affiliates, officers, directors or employees.

REFERENCES

[1] H.M.Sneed and R.Majnar., "A case study in software wrapping," *International Conference on Software Maintenance*, Bethesda, MD, pp. 86–93, Nov1998.

[2] E. J. Byrne, "A conceptual foundation for software reengineering," *Conference on Software Maintenance*, Orlando, Florida, pp. 226–235, Nov 1992.

[3] Lu Ping, and Xiaohu Yang, "An evolutional model of legacy system reengineering," *9th Joint International Computer Conference*, Zhuihai, China, pp.160-164, Nov 2003

[4] B. Boehm., "A spiral model of software development and enhancement," *IEEE Computer*, Vol.21, No.5, pp. 61-72, May 1988.

[5] Alessandro Bianchi, Danilo Caivano, Vittorio Marengo, and Giuseppe Visaggio, "Iterative reengineering of legacy functions," *17th IEEE International Conference on Software Maintenance,* Florence, Italy, pp. 632-641, Nov 2001.

[6] Marco Battaglia, Giancarlo Savoia, and John Favaro, "RENAISSANCE: a method to migrate from legacy to immortal software systems," *2nd Euromicro Conference on Software Maintenance*

and Reengineering (CSMR'98), Florence, Italy, pp. 197-200, March 1998.

[7] Linda H. Rosenberg and Lawrence E.Hyatt, "Hybrid re-engineering," *Software Technology Conference,* Utah, April 1997.

[8] William C. Chu, Chih-Wei Lu, Chih-Peng Shiu, and Xudong He, "Pattern-based software reengineering: a case study," *Journal of Software Maintenance*, Vol.12, No.2, pp.121-141, March 2000.

[9] Alessandro Bianchi, Danilo Caivano, Vittorio Marengo, and Giuseppe Visaggio, "Iterative reengineering of legacy systems," *IEEE Transactions on Software Engineering*, Vol.29, No.3, pp.225-241, March 2003.

[10] B. Boehm., and W. Hansen, eds., "Understanding the spiral model as a tool for evolutionary acquisition," *CrossTalk*, pp. 4-11, May 2001.