MATEC Web of Conferences **77**, 04005 (2016) DOI: 10.1051/matecconf/20167704005
ICMMR 2016

Comparison and analysis of methods for migrating legacy systems to SOA

LIU Dong[1], ZHANG Caihuan[2]

[1]*Dept. of Information Technology, Luoyang Normal University, 471022 Luoyang Henan, China*
[2]*Dept. of Mathematics, Luoyang Normal University, 471022 Luoyang Henan, China*

Abstract. Migration of legacy systems to SOA has caught lots of attention in both academic and industry. People have proposed many methods, such as wrapping, reengineering etc. However, various characteristics of legacy systems can make the migration complicated, and different enterprises can afford different level migration cost. In this paper, the existing methods and options for evolving legacy systems are compared, and their advantages and disadvantages are identified respectively. Different adaptive situations are listed to assist in making migration decision.

1 Introduction

The generic term legacy systems has been coined to refer to existing IT assets that have been deployed in the past, and in many cases, may have come into the enterprise as a result of a merger or acquisition. No standard definition exists. There are a set of common features: large scale(often millions of lines of monolithic code); old age; obsolete languages such as assembler; lack of consistent (or any) documentation; poor management of data; degraded structure following years of modification; and finally, a support team which is totally reliant on the expertise and knowledge of expert individuals. These factors together cause the cost of maintenance support to be very high, and the applications backlog to be unacceptable [1]. However, it is an undeniable fact that legacy systems generally consist of invaluable assets with embedded business logic representing many years of coding, developments, enhancements, and modifications. In addition, legacy systems is generating unusually high profit margins in many cases and thus, are responsible for a large amount of a company's operating profit. Consequently, many business and IT benefits are realized from reusing legacy assets, evolving them to integrate with or migrate to modern platforms and architectures like Service Oriented Architecture (SOA).

SOA has become an increasingly popular paradigm at present for achieving interoperability between systems because it provides a standard based conceptual framework. At the core of an SOA is a service. A service is a coarse-grained, discoverable, and self-contained software entity that interacts with applications and other services through a loosely coupled, often asynchronous, message-based communication model [2]. Proponents suggest a long list of advantages [3]:

- Simple standards that define the available interfaces and structure of data that is conveyed across those interfaces.
- Platform and language-independent interfaces based on these standards, which allow application to invoke services operating on any device supporting the SOA regardless of the hardware platform, operating system, or implementation language.
- Clear separation of service interface from implementation, thus allowing many service updates to occur without impact on service users.
- • Message-based communication allowing distribution across a wide area.
- Loose coupling between services, minimize interdependencies and facilitating reuse.
- Mechanisms for discovery of services available and for establishing connections with services.

Clearly, one of the most attractive features of SOA to many organizations is that SOA offers the promise of enabling legacy systems to expose their functionality as services without making significant changes to the legacy systems themselves.

Over the past few years, migration of legacy systems to SOA has caught lots of attention in both academic and industry. People have proposed many methods, such as wrapping, reengineering etc. However, various characteristics of legacy systems, such as age, language, and architecture, as well as the target SOA environment, can make the migration complicated. In addition to this, different enterprises can afford different level migration cost. So it may not be immediately obvious whether to migrate or not and which method be suitable for a specific organization. In this paper, we compare the existing methods and options for legacy systems, especially legacy systems with many applications, and identify their advantages and disadvantages respectively. Different adaptive situations are listed to assist in making migration decision.

The remainder of the paper is organized as follows. In Section 2, a few existing methods of migration to SOA are listed. Their advantages and disadvantages are compared and analysed. Accordingly, their adaptive situations are identified respectively. Section 3, illustrates the related works and highlights the main contributions, and Section 4 concludes the paper.

2 Comparison and analysis of the existing methods

2.1 requirements and background

Generally, the primary motivation for migration to SOA is based on one or more of the following requirements: reduce the high cost of ownership; accelerate time to market; monolithic architecture with little or no modularity together with redundant code. closed and obsolete technology; decreasing developers skilled in existing systems; lack of vendor support; lack of consistent documentation.

In the following subsection, we will base our comparison and analysis on the aforementioned requirements. Before presenting our findings, let's consider a situation: an enterprise has many applications. Recently, the enterprise has decided to adopt SOA and developed a SOA infrastructure. A question to be addressed is which application should be migrated to SOA, which method be suitable for it and how. The answer will be various for different application and organization. Our findings provide baseline for making decision through comparison and analysis of different methods.

2.2 Freeze

The term "freeze" means that the organization decides that no further work on the legacy application should be performed. Although freezing seems to sound absurd, it may make sense to deal with some specific application. In fact, people always justify a project at first according to the value and the state of the art of an application as well as cost-benefit analysis. Manager would prefer to have an application maintain its current state in many cases.

The freezing approach has four advantages:

- No additional cost input.
- No time consuming.
- No risk of failure.
- Continue to keep profit generated by the legacy application.

Of course, the disadvantages of this approach are also evident. All requirements mentioned above can't be met. The main pain points organization faced with cannot be solved. However, There are some cases where the freezing approach may be a good option, if:

- The size of the application is small.
- The cost of ownership is low.
- Critical business is not embedded in the application.
- The application is generating low profit margins for organization.
- The application changes infrequently.
- The application has low integration needs with other ones.

2.3 Rewrite

Rewriting means to retire the application and outsource or re-develop from ground-up. Actually, Rewriting is not one of the methods advocated by SOA proponents. But, it may make sense because of the following advantages:

- It can deliver a complete customized solution that can be built exactly to meet the organization's need.
- It is a "big-bang" approach to solve the main problem.
- New and standards-compliant technologies are introduced.
- The organization can gain the control of software source code base.
- New added values are brought by leveraging SOA.

There are a few shortcomings that make many consider the approach incompliant with the idea of SOA:

- Legacy assets cannot be reused so that investment in the past is lost.
- May be risky and long time consuming.
- May be high cost.

Although these shortcoming evidents, in the situations below, this may be an appropriate method.

- It is difficult to understand the business rules embedded in the application.
- The size of the application is small.
- The application involves obsolete or difficult to maintain technologies.
- The application needs to be customized.
- Organization needs the control of software source code base.
- Reliable outsource partner or people of development and maintenance skilled in SOA exist.

2.4 Replace

Replacement sometimes is easy to be confused with rewriting. Replacement here refers to replace an application with an Commercial off-the-shelf (COTS) package. Some common advantages and disadvantages exist between replacement and rewriting methods, for example, both belong to "big-bang" approach, both bring new added value by leveraging SOA, both introduce new and standards-compliant technologies, however ,both throw all the legacy assets away and lose the investment in the existing application. A particular advantage is less risky and time consuming. Some disadvantages differentiating from rewriting method are listed below.

- Easy to be lock in specific vendor.
- Maintenance and modification may be difficult and expensive.
- Organizations lose control of the source code base.
- Consistent functionality with the original cannot be guaranteed.
- We identify the appropriate situation for the method with the following characteristics:
- The business rules included in the application are clear, common and well understood in the organization.
- No core business is involved in the application.
- Small size application and acceptable purchase cost.
- Must be delivered in a short term.
- Infrequently changes after deployment.
- Available COTS package to meet the organization's needs.

2.5 Wrap

Wrapping method adds a new SOA interface to existing application, making it easily accessible by other software components or services. This is a black-box technology since it just surrounds application with a new interface while hiding the complexity of its internals. Wrapping gives legacy application the following benefits in a quick and a simple manner:

- It leverages the investment in existing application by taking advantage of its capabilities.
- It minimizes the risks and costs in migration.
- It provides rapid return on investment.
- It improves the interoperability by interacting with new SOA-enabled applications.
- It can be used as a gradually transition phrase to SOA.

However, wrapping method does have a few pitfalls. Firstly, wrapping method does not change the fundamental characteristics of the legacy application. Secondly, most of the technology pain points mentioned above cannot be solved through wrapping the application, such as problems in maintenance, upgrading and agility etc. Therefore, wrapping method can just be fit for the following situations:

- The size of the application is relatively small and reusable.
- A fast, cost-effective solution is needed.
- The application has a high business value and good quality code.
- Infrequently changes after deployment.
- The application has high integration needs with other ones.

2.6 Reengineer

As defined by Chikofsky and Cross [4], reengineering is the examination and alteration of a subject system to reconstitute it in a new form and the subsequent implementation of the new form. Reengineering generally includes some form of reverse engineering (to achieve a more abstract description) followed by some more form of forward engineering or restructuring.

For some being considered application in our context, Reengineering involves reverse engineering (analysing the existing application, identifying the best candidates for migration to SOA), transformation (restructuring the reusable components to services), and forward engineering (implementing services and services composition by considering the new requirements, goals, etc.) three sub-processes. So reengineering is an expensive and complex process, but it greatly increases the legacy systems control and evolution to SOA to meet future business requirements. Advantages and disadvantages of reengineering method are listed in detail below.

Advantages:

- It leverages the investment in existing application by reusing application components.
- All the requirements for migration to SOA can be fulfilled.
- It improves interoperability and agility.
- Operational costs will be reduced due to SOA adoption in the long term.
- It improves end user experience and increases customer satisfaction.

Disadvantages include: it is higher risk of failure; the costs of reengineering are for many organizations too high to afford; the reengineering process may be a long time consuming.

Accordingly, the method is suitable for the following occasions:

- The size of application is large.
- The application has high business value and carries on core business process for the organization.
- The application needs to be migrated to distributional circumstance and be integrated with many other applications may be on heterogeneous platform.
- The application needs to change frequently to meet increasingly volatile business requirements.

3 Related works

Much research has been done in the domain of legacy system analysis and migration. We divide the literature into two categories: process-based and technology-based research.

For the process-based research, a well-known "SOA-Migration Horseshoe" has been proposed [5]. This approach applies reverse engineering techniques to extract a legacy enterprise model from the legacy code, then applies enterprise modelling technologies to create a consolidated enterprise model from which services are identified using forward engineering technologies.

Finally, legacy code is transformed to services. Following the horseshoe model, a conceptual framework (SOA-MF) is proposed which embraces a holistic illustration of a migration process, and facilitates to characterize and isolate the properties of migration approaches in terms of processes it supports, artefacts included, activities carried out, and types of knowledge exploited. The Software Engineering Institute (SEI) proposes a method (SMART) for determining the feasibility of migrating legacy systems to an SOA environment; it accounts for the importance of business drivers and characteristics of the legacy system. IBM's Service Oriented Modelling and Analysis (SOMA) also embrace migration of legacy systems to SOA environments and has some portions that address legacy reuse [6]. Reijnders et al. proposes a legacy to SOA migration method by using method engineering approach [7]. Sneed presents a legacy application reengineering planning approach and suggests a cost-benefit model to support decision making [8]. Based on Sneed's work, Umar et al. presents a decision model for SOA reengineering projects that combines strategic and technical factors with cost benefit analysis for integration versus migration decision [9]. In [10], a two-phase model (SABA) is presented to assist organization to decide rationally among very different options. All the focus of the researches mentioned above has been on the whole process and strategy of migration and decision while pay no attention to the approaches themselves. The work by Almonaies et al. is the closest to our research because it also discusses the various approaches and methods and highlights their strengths and weakness [11]. However, comparison has been done between technologies in different literature with respect to a single approach rather than comparison between these approaches which is what we place emphasis on. In addition, we also disagree with Almonaies et al. classification of the approaches where rewriting, outsource developing and replacing with COTS are classified as one category, i.e. replacement. In our view, these approaches have their own characteristics and may be suitable for different applications or organizations. Furthermore, we insist that reengineering is different from redevelopment, and is one of methods of migrating legacy applications to SOA environment.

For the technology-based research, most of works focus on wrapping method. Sneed's approach presents a migration process at lower level through focusing on technical details [12, 13]. Legacy software assets can be preserved by means of wrapping technology. A black-box, wrapper-based migration process, making the interactive functionality of legacy systems accessible as web services, is described in [14]. In [15], different from black-box wrapping method, a reengineering approach is proposed which applies an improved agglomerative hierarchical clustering algorithm to restructure legacy code and to facilitate legacy code extraction for web service construction. Cetin et al. propose a mashup-based approach for migration of legacy systems to service-oriented computing environment [16]. Furthermore, some other works also focus on reshaping the existing legacy elements to the service-based elements, as in [17-20].

4 Conclusions

It is hard to say which method is the best In practice, the information systems of an organization may consist of many applications, and a mixture of methods here is used to form a strategy based on the specific needs of the given organization. The main contribution of this paper is the comparison and analysis of existing methods about their advantages and disadvantages. We also identify the most appropriate situation for each method in order to provide managers with a baseline for making decision of migrating legacy application to SOA. This is a start point of our further work. We will develop an decision model supported by intelligent decision support system for migration to SOA, and conduct case study to evaluate it.

Acknowledgement

This work is supported by the Key Research Program of Higher Education, Henan Province of China, under the grant No. 15A520024.

References

1. Bennett, K.H., M. Ramage,M. Munro, Decision Model for Legacy Systems. IEE Proceedings Software, June 1999. 146(3): p. 153-159.
2. Brown, A., S. Johnston,K. Kelly,Using Service-Oriented Architecture and Component-Based Development to Build Web Service Applications, Rational Software Corporation,R.S. Corporation2002
3. Lewis, G., E. Morris, O.B. L., S. D.,L. Wrage,SMART: The service-oriented migration and reuse technique, CMU/SEI-2005-TN-029,Software Engineering Institute,CMU,2005
4. E.J., C., C.I. J.H., Reverse engineering and design recovery: A taxonomy. IEEE Software, 1990. 7(1): p. 13-17.
5. Winter, A.,J. Ziemann. Model-Based Migration to Service-Oriented Architectures. in Proceedings of the International Workshop on SOA Maintenance Evolution (SOAM 2007), 11th European Conference on Software Maintenance and Reengineering (CSMR 2007). March 20-23, 2007. Amsterdam, the Netherlands: IEEE Computer Society.
6. Arsanjani, A., S. Ghosh, A. Allam, T. Abdollah, S. Ganapathy,K. Holley, SOMA: A Me-thod for Developing Service-Oriented Solutions. IBM Systems Journal, 2008. 47(3).
7. Reijnders, G., R. Khadka, S. Jansen,J. Hage,Developing a legacy to SOA Migration Method, UU-CS-2011-008,Department of Information and Computing Sciences, Utrecht University,2011
8. Sneed, H., Planning the reengineering of legacy systems. IEEE Software, 1995: p. 24–34.
9. Umar, A., A. Zordan, Reengineering for service oriented architectures: A strategic decision model for integration versus migration. Journal of Systems and Software, 2009. 82(3): p. 448-462.

10. H., B.K., R. M.,M. M., Decision Model for Legacy Systems. IEE Proceedings Software, June 1999. 146(3): p. 153-159.
11. Almonaies, A.A., J.R. Cordy,T.R. Dean, Legacy system evolution towards service-oriented architecture, in SOAME 2010: International Workshop on SOA Migration and Evolution. 2010: Madrid, Spain p. 53-62.
12. Sneed, H.M. Wrapping legacy COBOL programs behind an XML-interface. in WCRE '01:Proceedings of the EighthWorking Conference on Reverse Engineering (WCRE'01) 2001. Stuttgart, Germany
13. Sneed, H.M. Integrating legacy software into a service oriented architecture. in CSMR '06:Proceedings of the Conference on Software Maintenance and Reengineering. 2006. Bari,Italy
14. Canfora, G., A.R. Fasolino, G. Frattolillo,P. Tramontana, A wrapping approach for migrating legacy system interactive functionalities to service oriented architectures. J. Systems and Software, 2008. 81(4): p. 463-480.
15. Zhang, Z., H. Yang. Incubating services in legacy systems for architectural migration. in APSEC '04: Proceedings of the 11th Asia-Pacific Software Engineering Conference. 2004. Busan, Korea
16. Cetin, S., N.I. Altintas, H. Oguztuzun, A.H. Dogru, O. Tufekci,S. Suloglu, A mashup-based strategy for migration to service-oriented computing, in IEEE International Conference on Pervasive Services. 2007. p. 169–172.
17. Li, S.H., D.C. Yen,C.C. Chang, Migrating legacy information systems to web services architecture. Journal of Database Management, 2007. 18(4): p. 1-25.
18. Chen, F., S. Li, H. Yang, C.H. Wang,W. Cheng-Chung Chu, Feature analysis for service-oriented reengineering, in Software Engineering Conference. 2005.
19. Liu, Y., Q. Wang, M. Zhuang,Y. Zhu, Reengineering legacy systems with RESTful web service. Computer Software and Applications, 2008: p. 785-790.
20. Zhang, Z., H. Yang,W. Chu, Extracting reusable object-oriented legacy code segments with combined formal concept analysis and slicing techniques for service integration. Quality Software, 2006: p. 385-392.