Migration from Legacy Systems to SOA Applications: A Survey and an Evaluation

Sukanya Suwisuthikasem and M.H. Samadzadeh

1 Introduction

As computer software grows in power, users demand ever more powerful and reliable programs, resulting in ever larger and seemingly more complex software systems. Thus a major challenge in large-scale software development is managing the complexity encountered during the construction of new applications that is partly attributable to frequent customizations due to requirement changes.

This complexity can potentially be reduced if applications could be implemented based upon an open standard-based interface and communication protocol. From this, all applications can be accessed more efficiently and easily, thus enabling businesses to leverage their existing software systems. In addition, since business requirements are typically subject to frequent changes, the demand on existing systems to evolve is inevitable. Consequently, there is a need to have an efficient method to support the software evolution process. Service Oriented Architecture (SOA), a new approach to developing software systems, has been eventually invented to fulfill this demand.

SOA has gained significant popularity for achieving business goals and implementing business processes in a flexible manner. SOA is becoming a mainstream approach for software development. Abrams and Schulte [1] indicated that during 2007, more than 50 % of large, newly-developed systems and business processes were designed and developed based on the Service Oriented Architecture paradigm. Lublinsky [2] suggested there are three primary reasons that businesses are interested in SOA. First, by adopting SOA they can achieve better alignment between business and Information Technology (IT). Second, SOA enables them to construct more flexible and responsive IT infrastructure. And last, that SOA can simplify the implementation of data integration among a business' applications. Based on this argument, in order to sustain their competitiveness to be leaders in the market, businesses need to transform their legacy systems to SOA applications toward providing more efficient services to their customers.

Generally, migration from legacy systems to SOA applications is carried out manually by domain experts with subject-matter knowledge with some training in software development or with the assistance of software developers [3, 4]. The problem of migrating (i.e., transforming/adapting/retargeting) large-scale legacy software systems to a modern environment, to new hardware platforms, and to new run-time support is a major issue facing the software industry. The focus of this work was to investigate the existing migration approaches and capture the significant features of each approach so as to give the guideline for businesses to choose a tailor-made migration approach suiting them.

2 General Migration Processes

2.1 Legacy System Assessment

Sommerville [6] presented four strategic options for software evolution as illustrated below.

As shown in Fig. 1, there are four clusters for legacy systems as described below.

- Low Quality, Low Business Value: These systems should be discarded since it is costly and unproductive for businesses to keep them.
- Low Quality, High Business Value: These systems are still productive but costly to maintain. Therefore they should be transformed to a system with a new SOA architectural style.
- High Quality, Low Business Value: Although they are inexpensive to maintain, these systems should be discarded as they are unproductive.

S. Suwisuthikasem (✉) • M.H. Samadzadeh
Computer Science Department, Oklahoma State University, Stillwater, USA
e-mail: suwisut@cs.okstate.edu

H. Selvaraj et al. (eds.), *Progress in Systems Engineering: Proceedings of the Twenty-Third International Conference on Systems Engineering*, Advances in Intelligent Systems and Computing 1089,
DOI 10.1007/978-3-319-08422-0_86,

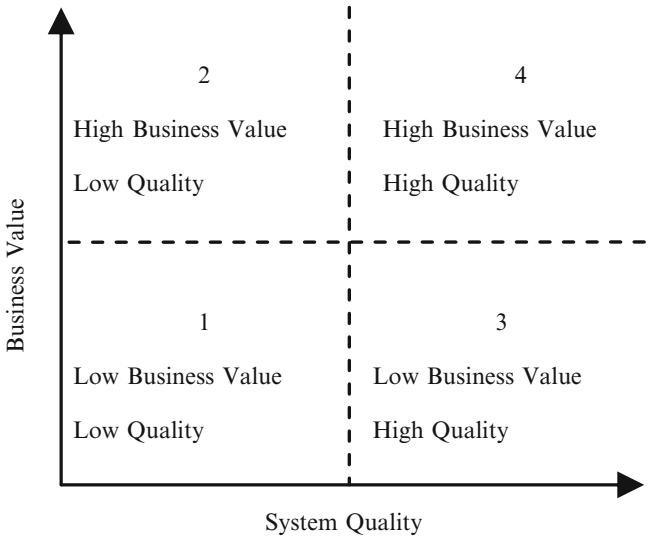

Fig. 1 Legacy System Assessment (Source: Software Engineering, 9th edition, p.253)

- High Quality, High Business Value: It is cost-effective to maintain these systems. So these systems should be kept without making any changes the system.

Several criteria need to be defined and quantified so as to measure the quality of an SOA system. This can be performed by interviewing some domain experts or using a questionnaire. Fortunately, there are a lot of researchers working in this area by defining a number of criteria to assess legacy systems. Four basic attributes of a legacy system, namely Business Value, Decomposability, Obsolescence, and Deterioration, and guidelines to measure each attribute were introduced by Cimitile et al. [25]. Ransom et al. [26] presented a method to assess legacy systems namely technical, business, and organizational aspects. This method could help businesses to assign a value to each assessment characteristic and also to interpret these values. They indicated that their method can be tailored to specific organizations.

2.2 Feasibility Analysis

Khadka et al. [12] proposed the seviciFi method using method engineering to determine the economical and technical feasibility of the migration based on the legacy system's charactersitics and the requirements of the target SOA application. Erradi et al. [13] proposed a decision framework to help organizations to select the optimal combination of legacy modernization options. Aly and Amir [14] presented a decision making tool using decision theory and the weighted sum methodology to generate the most optimal strategy to be used in modernizing legacy systems. They also proposed an automated decision making process for choosing a migration strategy using a combination of the approaches proposed in [23] and [13]. Aversano and Tortorella [27] proposed a strategy to help businesses define the evolution requirement to SOA based on characteristics of their legacy systems. They specified nine steps in software evolution: analyze the organization, reconstruct processes, identify and analyze the processes, identify technology, formulate evolution requirements, assess the legacy software system, define software system evolution requirements, reengineer the processes, and perform the system evolution. The authors reported the first stage of testing their method with two different systems: a bank system and a public administration system. According to their result from the first stage, this method is certainly applicable however it needs additional information and it also needs to be supported by a specific integrated and Web-based software environment.

2.3 Migration

Migration refers to any approach that can be applied to a legacy system in its entirety in the process of transforming it to SOA architecture. This section discussed how legacy code components are identified, decomposed, and extracted using several techniques. The User interfaces of legacy systems are reengineered to be SOA-based system compatible. Migration strategies incorporate both top-down and bottom-up approaches and aim to produce a system with an improved SOA-compatible design.

Aly and Amir [14] proposed a method which automatically generates a modular software structure from a given source code by using spectral clustering. First, undirected graphs are generated based on the existing dependencies among the components and then spectral clustering is performed to generate the component structure of the target system. To evaluate this method, they applied it to CoCoME, a small and well-documented software system. They compared the resulting software structure to the one created by an expert. They reported that the result were similar. However, the component structure resulting from their approach cannot be represented by any architectural style. In their case study, the legacy system is a three-tier architecture but the result is a single-layer architecture.

Chen et al. [8] and Millham [11] presented a service oriented reengineering approach using feature analysis to extract candidate services from legacy systems. Feature analysis addresses the understanding of features in software systems and defines mechanisms for carrying a feature from the problem domain into the solution domain. In [8], the specified feature analysis activities are: identifying system features, constructing a feature model to organize the defined features, and identifying their implementation in the legacy system through feature-location techniques. Based on a feature model, certain service identification and packaging processes are performed that result in service delegation.

In a research conducted by Millham [11], data and control dependencies among the component files of a legacy system are analyzed and then clustered into groups. Cuadrado et al. [9] described a case study of the evolution of an existing legacy system towards a more maintainable SOA system. To define the specific evolution plan, the architecture of the legacy system was recovered. This approach was applied to a medical imaging system evolving it into an SOA-based application. Matos and Heckel [3] proposed the new methodology for migration based on source code analysis for identifying the contribution of code fragments to architectural elements and a graph transformation approach for architectural migration. Alahmari et al. [10] introduced a framework to identify optimal services from legacy code with the appropriate level of granularity, by focusing on the significance of the classification of service types, to define service properties.

Reddy et al. [15] proposed guidelines for evaluating the suitability of existing assets by identifying the core principles of SOA, namely cohesion, reusability, discoverability, loose coupling, abstraction, formal contract, composability, and statelessness. They argued that organizations can use their guidelines to improve the quality of their migrations by considering their defined metrics and guidelines. Stroulia et al. [16] described the overall process for legacy system migration to a Web-based system using the CelLEST method. This method addresses the issue of migration based on understanding and modeling the users' interaction with the legacy system's interface. There are three main steps in this method. The first step is to model the behavior of the old system using a state transition diagram. The second step is to find the users' tasks as frequently-occurring interaction patterns to recover the specifications of the application's functions. The last step is to construct the new user interface allowing the legacy functions to be accessible over the Web.

Aversano et al. [17] presented a migration project aiming to integrate a COBOL system into a Web-enabled system. The legacy system was dissected into user interfaces and server (application logic and database) components. The user interfaces were migrated into a Web browser shell using Microsoft Active Server Pages (ASP) and VBScript. All server components were wrapped with dynamic load libraries written in Microfocus Object COBOL, loaded into Microsoft Internet Information Server, and accessed via the ASP pages.

Werth et al. [18] introduced Business Service Management as an interdisciplinary approach for business-driven deployment of SOA. The main purpose of this work was to represent a business' characteristics and requirements toward IT as business processes. Bhallamudi and Tilley [19] presented the Evolution Process Framework for SOA (a mechanism for analysis of existing SOA migration projects) to learn about factors such as technology selection, migration approach utilized, legacy system type, and SOA governance that influence the success or failure of each project.

Table 1 Relationship between Design Properties and Quality Attributes [24]

	Effectiveness	Understandability	Feasibility	Reusability
Coupling		↓	↓	↓
Cohesion	↑	↑		↑
Complexity		↓		
Design Size		↓		
Service Granularity	↑	↑	↑	↑
Parameter Granularity	↑	↑	↑	
Consumability		↑		↑

Mohagheghi and Sæther [20] applied the model-driven approach to construct a methodology and a tool for transforming a legacy system into a service oriented application. O'Brien et al. [21] used architecture reconstruction in the process of migration. To accomplish this, dependencies among components in the legacy system were identified. Based on this information, an essential step in making decisions regarding migration of legacy components to services was devised. They also suggested that using architecture reconstruction techniques in conjunction with other analytical methods could provide an essential set of analytical methods for decision making.

Marchetto and Ricca [22] proposed a stepwise approach based on testing to migrate Java application into SOA-based application. This approach, which is a hybrid top-down and bottom-up approach, was applied to four Java applications. Lewis et al. [5], [23] described their Service Oriented Migration and Reuse Technique (SMART), a technique helping businesses to analyze legacy capabilities for use as services in an SOA. Their technique considers the specific interactions that will be required by the target SOA and any changes that must be made to the legacy components. They described a wide range of information about legacy functionalities, the target SOA, and the potential services that were aggregated to produce a service migration strategy.

2.4 Evaluation

Shim et al. [24] suggested that, in order to evaluate the quality of an SOA application, a quality assessment model is needed that defines the desired quality attributes and measures them. From this, design problems can be detected and resolved before the development of the system. The relationship between design properties and quality attributes is described in Table 1. Shim et al. [24] also

Table 2 Methods/Techniques Used in the Migration Process

Legacy System Assessment			
Method/Technique	Tools	Pros.	Cons.
Standard Decision Framework [25]	–	– Suitable for any kind of legacy system	– Needs a lot of documentation – Needs experts – Can be affected by errors or biases
Assessment Method [26]	–	– Can be tailored for any specific kind of legacy system – Can be iterated to reduce inaccuracy of the assessment – Provides practical advice and guidance to businesses	– Not yet fully tested and evaluated
Feasibility Analysis			
Method/Technique	Tools	Pros.	Cons.
– SMART [23]	– SMIG	– Helps businesses to analyze and determine if a legacy system can be exposed as services	– Needs experts – Need lots of documentation and user feedback
– Method engineering [12]	– serviciFi	– Helps businesses select supporting technology based on migration feasibility	– Processes are performed manually – Not support for large projects
– Decision theory – Weighted sum methodology – Gap analysis [14]	– Decision Making tool	– Can be customized for specific organizations – High degree of automation	–
– Decision framework [13]	–	– Helps businesses to create the most beneficial and cost-effective migration approach meeting a business' requirement	– Low degree of automation – Cannot guarantee the QoS of the target SOA system
Migration			
Method/Technique	Tools	Pros.	Cons.
– Code Analysis – Pattern Matching – Graph Transformation [8]	– ATX (L-CARE) [35] – EMF [36]	– High degree of automation	– Ongoing project – Support a specific language
– Feature Analysis – Slicing Technique [8], [11]	– WSW [29] – Captain Feature [33] – FEAT[34]	– High degree of automation	– Not suitable for large projects – Supports a specific language
– Undirected Graph – Spectral Clustering [7]	– UML model	– High degree of automation – Supports large projects – No need for documentation	– Needs a method to identify meaningful clusters, otherwise the result will not be correct
– Architecture Recovery [5], [9]	– MOOSE [37] – Rigi [38] – QAR [30] – Jude[31] – Omondo UML Studio [32] – Eclipse TpTP [28]	– High degree of automation	– Not supporting large projects – Specific to Java applications
– Domain Analysis – Wrapping [11]	– UML model – TAGDUR [39] – CORBA/IDL parser cc	– High degree of automation	– Specific type of target SOA: Web-Based system

defined three metric groups that can be directly applied to design components. The metrics in the first group, service internal metrics, use service internal elements such as service name, operations provided by the service, and characteristics of the messages defined in the service. The second group of metrics is service external metrics that use information from services they are connected to. Metrics in this group are used to measure the characteristics of the consumer and producer services that are either directly or indirectly connected to a given service. The last group is system metrics. The metrics in this group are used to measure the characteristics of the entire system in general.

In Table 1, an up/down arrow means increase/decrease of the attribute vis-à-vis an increase/decrease of the respective property.

3 Existing Migration Techniques/Tools

From Section 2, we can summarize the techniques and tools that support the migration as follows.

The information summarized and tabulated in Table 2 can be used to choose from among the techniques and tools available for each step of the migration process.

4 Conclusions

In this paper, several approaches for migrating legacy systems to SOA architecture are investigated and the main processes and tools of each approach are captured and analyzed. The different approaches are compared and contrasted based on their key features and their target legacy system types. Using this comprehensive comparative analysis, businesses can create tailor-made approaches that could best fit their needs and satisfy their requirements.

References

1. Charles Abrams, Roy W. Schulte: Service Oriented Architecture Overview and Guide to SOA Research. Gartner Research (2008)
2. Boris Lublinsky: Defining SOA as an Architectural Style, IBM, http://www.ibm.com/developerworks/architecture/library/ar-soastyle/
3. Carlos Matos, Reiko Heckel: Migrating Legacy Systems to Service Oriented Architectures. In: Doctoral Symposium at the International Conference on Graph Transformation (ICGT 2008), Vol. 16, pp.1-15 (2008)
4. Gerardo Canfora, Anna Rita Fasolino, Gianni Frattolillo, Porfirio Tramontana: Migrating Interactive Legacy Systems to Web Services. In: 10th European Conference on Software Maintenance and Reengineering, pp. 27-36. Bari, Italy (2006)
5. Grace Lewis, Edwin Morris, Dennis Smith: Analyzing the Reuse Potential of Migrating Legacy Components to a Service Oriented Architecture. In: 10th European Conference on Software Maintenance and Reengineering, pp. 15-23. Bari, Italy (2006)
6. Ian Sommerville: Software Engineering, 9th Edition. Pearson Education Inc., Essex, England and Addison-Wesley Publishers. Boston, MA (2011)
7. Constanze Deiters, Andreas Rausch, Mirco Schindler: Using Spectral Clustering to Automate Identification and Optimization of Component Structures. In: 2nd International Workshop on Realizing Artificial Intelligence Synergies in Software Engineering (RAISE), pp. 14-20. San Francisco, CA (2013)
8. Feng Chen, Shaoyun Li, Hongji Yang, Ching-Huey Wang, William Cheng-Chung Chu: Feature Analysis for Service Oriented Reengineering. In: 12th Asia-Pacific Software Engineering Conference: APSEC '05, pp. 201-208. Taipei, Taiwan (2005)
9. F. Cuadrado, B. Garcia, J. C. Dueas, H. A. Parada: A Case Study on Software Evolution Towards Service Oriented Architecture. In: 22nd Int. Conf. on Advanced Information Networking and Applications: AINAW 2008, pp. 1399-1404. Okinawa, Japan (2008)
10. Saad Alahmari, Ed Zaluska, David De Roure: A Service Identification Framework for Legacy System Migration into SOA. In: 7th International Conference on Services Computing, pp. 614-617. Miami, FL (2010)
11. Richard Millham: Migration of a Legacy Procedural System to Service Oriented Computing Using Feature Analysis. In: International Conference on Complex, Intelligent, and Software Intensive Systems (CISIS), pp. 538-543. Krakow, Poland (2010)
12. Ravi Khadka, Gijs Reijnders, Amir Saeidi, Slinger Jansen, Jurriaan Hage: A Method Engineering based Legacy to SOA Migration Method. In: 27th IEEE International Conference on Software Maintenance, pp.163-172 (ICSM) (2011)
13. Abdelkarim Erradi, Sriram Anand, Naveen Kulkarni: Evaluation of Strategies for Integrating Legacy Applications as Services in a Service Oriented Architecture. In: IEEE Int. Conf. on Services Computing (SCC'06), pp. 257-260. Chicago, IL (2006)
14. Sherif G. Aly, Rafik Amir: Automated Selection of Legacy Systems SOA Modernization Strategies Using Decision Theory. International Journal of Software Engineering and Its Applications, Vol. 3, No. 4, pp. 65-86 (2009)
15. Vinay Kumar Reddy, Alpana Dubey, Sala Lakshmanan, Srihari Sukumaran, Rajendra Sisodia: Evaluating legacy assets in the context of migration to SOA. In: 10th IEEE International Symposium on High Perfor-mance Distributed Computing, pp.51–63. Springer Science + Business Media, LLC (2008)
16. E. Stroulia, M. El-Ramly, P. G. Sorenson: From Legacy to Web Through Interaction Modeling. In: 18th Int. Conf. on SW Maintenance, pp. 320-329. Montreal, Canada (2002)
17. Lerina Aversano, Gerardo Canfora, Aniello Cimitile, Andrea De Lucia: Migrating Legacy Systems to the Web: An Experience Report. In: 5th European Conference on Software Maintenance and Reengineering, pp. 148-157. Lisbon, Portugal (2001)
18. Dirk Werth, Katrina Leyking, Florian Dreifus, Jörg Ziemann, Andreas Martin: Managing SOA Through Business Services: A Business-Oriented Approach to Service Oriented Architectures. In: 4th International Conference on Service-Oriented Computing: ICSOC 2006, LNCS 4652, pp. 3-13. Chicago, IL (2007)
19. Pushparani Bhallamudi, Scott Tilley: SOA Migration Case Studies and Lessons Learned. In: IEEE Int. Systems Conference (SysCon), pp. 123-128. Montreal, Canada (2011)
20. Parastoo Mohagheghi, Thor Sæther: Software Engineering Challenges for Migration to the Service Cloud Paradigm: Ongoing Work in the REMICS Project. In: IEEE World Congress on Services, pp. 506-514. Washington, DC (2011)
21. Liam O'Brien, Dennis Smith, Grace Lewis: Supporting Migration to Services Using Software Architecture Reconstruction. In: 13th IEEE International Workshop on Software Technology and Engineering Practice, pp. 81-91. Budapest, Hungary (2005)
22. Alessandro Marchetto, Filippo Ricca: From objects to services: toward a stepwise migration approach for Java applications. In: International Journal of Software Tools Technology Transfer. Springer-Verlag (2009)
23. Grace Lewis, Edwin Morris, Dennis Smith: The Service Oriented Migration and Reuse Technique (SMART). In: 13th IEEE International Workshop on Software Technology and Engineering Practice, pp. 222-229. Budapest, Hungary (2005)
24. Bingu Shim, Siho Choue, Suntae Kim, Sooyong Park: A Design Quality Model for SOA. In: 15th Asia-Pacific SE Conference, pp. 304-410. Beijing, China (2008)
25. Aniello Cimitile, Anna Rita Fasolino, Filippo Lanubile: Legacy Systems Assessment to Support Decision Making. In: IEEE Workshop on Empirical Studies of Software Maintenance (WESS '97), pp.145-150. Bari, Italy (1997)
26. Jane Ransom, Ian Sommerville, Ian Warren: A Method for Assessing Legacy Systems for Evolution. In: 2th Euromicro Conference on Software Maintenance and Reengineering, pp. 128–134. Florence, Italy (1998)

27. Lerina Aversano, Maria Tortorella: An assessment strategy for identifying legacy system evolution requirements in eBusiness context. Journal of Software Maintenance and Evolution: Research and Practice, pp. 255-276. John Wiley & Sons, Ltd. (2004)
28. Eclipse TPTP (Test and Performance Tools Project), an Eclipse top-level project, http://www.eclipse.org/tptp
29. H. Guo, C. Guo, F. Chen, H. Yang: Wrapping Client-Server Application to Web Services for Internet Computing. In: 6th International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT'05). Dalian, China (2005)
30. Arciniegas, J.L: Contribution to Quality-driven Evolutionary Software Development Process for Service Oriented Architecture. Ph. D. Thesis, Polytechnic Uni of Madrid (2006)
31. Jude (Java and UML Developer Environment), a Java UML modeling tool, http://jude.change-vision.com
32. Omondo Eclipse UML Studio, an Eclipse plug-in for UML modeling, http://www.omondo.com
33. K. Czarnecki, U. W. Eisenecker: Generative Programming, Addison Wesley (2000)
34. M. P. Robillard, G. C. Murphy: FEAT: A Tool for Locating, Describing, and Analyzing Concerns in Source Code. In: 25th Int. Conf. on SE. Oregon, Portland (2003)
35. The migration specialists, http://www.atxsoftware.com/
36. Eclipse. Eclipse Modeling Framework (EMF), http://www.eclipse.org/emf/
37. Ducasse, S., Lanza, M., Tichelaar, S.: Moose: an Extensible Language-Independent Environment for Reengineering Object-Oriented Systems. In: 2nd International Symposium on Constructing Software Engineering Tools: CoSET'00 (2000)
38. Kazman, R. O'Brien, L., Verhoef, C: Architecture Reconstruction Guidelines, 2nd Edition, CMU/SEI-2002-TR-034 (2002)
39. Richard Millham, Jianjun Pu, Hongji Yang: TAGDUR: A Tool for Producing UML Sequence, Deployment, and Component Diagrams Through Reengineering of Legacy Systems. In: 8th IASTED Int. Conf. on SE and Application: SEA (2004)
40. Chapter 5: Introduction to CORBA IDL, http://documentation.progress.com/output/Iona /orbix/gen3/33/html/orbix33java_pguide/IDL.html. IONA Technologies PLC (2000)