Proc. of the Second Intl. Conf. on Advances in Electronics, Electrical and Computer Engineering -- EEC 2013

ISBN: 978-981-07-6935-2

Trade Offs In Migrating From Legacy System To Service Oriented Architecture

Hunney Salhotra Balraj Singh Amandeep Kaur

***Abstract* -- Service Oriented Software Engineering is the most promising engineering paradigm in industry now-a-days. It focuses on development of software systems based on reusable services that can distributed in nature. Migrating legacy system developed in any language to Service Oriented Paradigm is in trend now. Enterprises have more powerful software with lesser cost. There has always been a difference between the methods of migration proposed by academia and as used in industry. Academia suggests that while migration, legacy system should be reverse engineered. But in practice, legacy systems are forward engineered. The paper tries to fill the agreement gap between Industry and Academia upon the migration of legacy system to SOA system. In this research, Memory and CPU utilization of various events of legacy system, Forward Engineered SOA system, and Backward Engineered SOA system were recorded. Recommendation is given that migration technique should be chosen by keeping these trade-offs in consideration. This research paper focuses on bridging the gap between Industrial Practices and Academia Theory.**

Keywords –service; service oriented software engineering; migration; legacy to SOA;

I. Introduction

Service-oriented architecture is a paradigm for creating, realization and maintenance of business processes

Hunney Salhotra
Department of Computer Science and Engineering,
Lovely Professional University
India

Balraj Singh
Department of Computer Science and Engineering,
Lovely Professional University
India

Amandeep Kaur
Department of Computer Science and Engineering,
Lovely Professional University
India

entities called services try to implement distributed and heterogeneous software systems. With acceptance of service-oriented architecture as a new approach in software engineering as a model for the production of software systems, and in order to practical use of SOA –

distributing in big heterogeneous systems. Service-Oriented Architecture (SOA) as one of the most prominent architectures introduced in the past decade, by creating layered architecture and the introduction of basic similar to other existing methods – some SOA specific methodologies are required for system development with emphasis on service-oriented approach [1].

A. *Service Oriented Architecture*

Service-oriented architecture provides a layered architecture trying to create architecture with different abstraction levels. It separates various concepts in each system software development such as issues related to user interfaces, synchronization, services and resources.
SOA resource layer consists of all information sources. Sources may include the previous programs, databases, systems management and even people and knowledge in the organization. The higher layer creates an integrated and extensible infrastructure for interaction between the services by building independent services with well-defined interfaces. Due to the importance of synchronization in distributed systems and the separation of collaboration from computational problems, a separate layer for synchronization is considered in SOA.
Web services are a set of stateless, autonomous, coarse grained, platform-independent and loosely-coupled software components, which are implemented under a specific namespace. Although some characteristics of the SOA paradigm, such as support of loosely coupled services and interoperability, make the service enabling of legacy systems look to be straightforward, it constitutes a key challenge of service design. One of the key features of the service oriented paradigm is to facilitate reuse of business functions provided by legacy systems.

Proc. of the Second Intl. Conf. on Advances in Electronics, Electrical and Computer Engineering -- EEC 2013

ISBN: 978-981-07-6935-2

The SOA migration framework addresses the question of "what does the migration of legacy systems to SOA entail". SOA migration is defined as a modernization technique that moves the system to a new platform while retaining the original system data and functionality. The main motivation behind the modernization of legacy systems to SOA is to achieve the advantages offered by SOA and still reuse the embedded functionalities in the legacy systems [2].

B. *Reengineering from Legacy system to SOA*

Any type of reengineering can be comprised of three basic reengineering processes [3]:

- Analysis of an existing system,
- Logical transformation, and
- Development of a new system.

C. *Service Extraction Process*

- *Consolidated legacy-to-SOA migration approach*: A consolidated legacy-to-SOA migration approach, referred as "ServiciFi method", is used. The serviciFi method is developed using method engineering and concept slicing. The serviciFi method combines the migration feasibility and technology support required for the legacy-to-SOA migration.
- *Candidate service identification***:** The initial two steps of the SEP: patterns identification and service identification represent the candidate service identification phase of the research. Candidate service identification is a challenging task, so a step-wise identification approach is designed.
- *Service Extraction***:** Concept slicing technique is exploited to extract the complete code representing the identified functionality. The concept slicing technique has been successfully used to extract source code for software maintenance.

D. *SOAMIG*

The SOAMIG-Project[1] aims at providing a general transformation-based migration process model with an emphasis on code and architecture migration. Its phases are described below [4]:

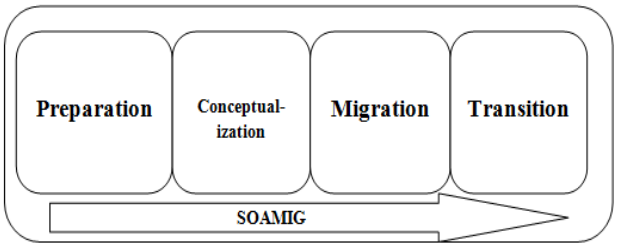

Fig 1: The SOAMIG Phases

- *Preparation*: Starting point of every migration project is legacy code which has to be prepared and standardized in the Pre-Renovation discipline by various reengineering activities to alleviate conversion activities. The project infrastructure including defining project goals and work packages or managing resources is set up in the Project Setup discipline
- *Conceptualization*: A broad automatization seems possible by eligible migration factories. A central activity in migration projects is assessing feasibility of migration and applicability of provided tool sets during Technical Feasibility.
- *Migration*: Migrating the entire system is applied after setting up a general migration strategy and tool support. In the Migration phase, all SOAMIG core disciplines are performed iteratively in different intensities, resulting in a migrated system in production.
- *Transition*: Code migration usually leads to hardly maintainable code, which requires additional reengineering.

With the increasing demand of SOA based application, industry and academia are in continuous debate on the most efficient way of migrating any legacy system to SOA. Academia suggests Reverse Engineering; whereas Industry implements Forward Engineering. There is a gap between the agreements on migration in both. So, the thesis will try to find the best way of migrating from legacy system to SOA and thus finding various trade-offs in resource utilization by SOA system.

II. Implementation

A system, "Book Store Management System" was created in C#.Net and SQL Server at its back end. Book Store Management System is an Inventory Control Module of any book shop. For the research, three similar Book Store Management Systems were created but by following three

Proc. of the Second Intl. Conf. on Advances in Electronics, Electrical and Computer Engineering -- EEC 2013

ISBN: 978-981-07-6935-2

different paradigms. All the logics and programming variable were kept constant for the consistency in the result.

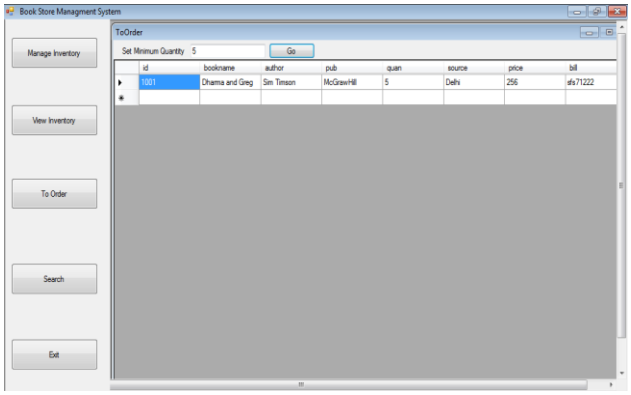

Fig 2: Book Store Management System

Legacy Book Store Management System was created using traditional approach without any services. Reverse Engineered Book Store Management System was created using SOA based approach. Forward Engineered Book Store Management System was created using SOA based approach upon legacy system. Not all functionalities are implemented using services.

Using Windows Task Manager, it's CPU and Memory utilizations for various events were recorded.

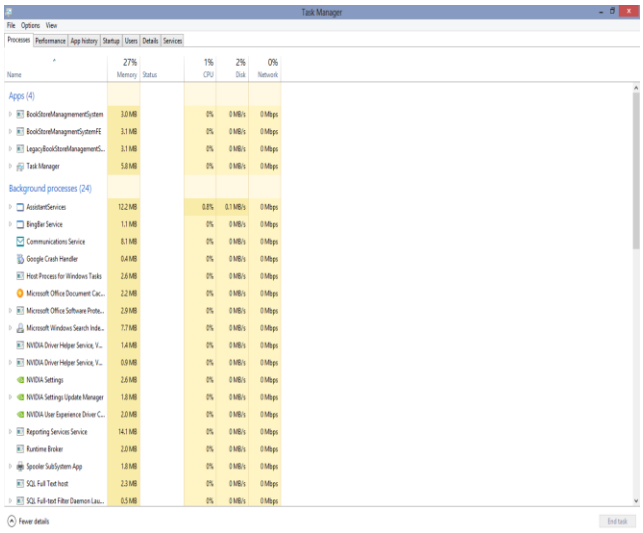

Fig 3: Task Manager for recoding Parameter

Various events in Book Store Management System are:

- Application Start-up (Considered only in memory recording)
- Form load Manage Inventory
- Manage Inventory Save
- Manage Inventory Update
- Manage Inventory Delete
- Manage Inventory Search
- Manage Inventory Close
- Form load View Inventory
- View Inventory Close
- Form Load To Order
- To Order Go
- To Order Close
- Form load Search
- Search Go
- Search Close

Once the data for legacy Book store management were recorded, the system was reverse engineered and a SOA based Reverse Engineered Book Store Management System was developed whose core functionality, i.e., inventory management of book store was based on a service published and used using Visual Studio. All the search functionality and other database features were also based on Service architecture. It is to be noted that all variables and program logic were kept constant. Just the core part was implemented using service, rest all logic remains same.

Finally, the legacy system was re-engineered, but this time forward engineering was done on it. In this SOA Based Forward Engineered Book Store Management System, only the core part of application, i.e., the inventory management module that adds, deletes, updates books from database is based on service architecture. Rest all other searching functionality, To Order functions, Data Viewing functions are based on traditional non service architecture. This is because it is seen that when an application is forward engineered, some of its features tend to have the traditional behaviour while the core part is enhanced. To this property of forward engineering is simulated by not completely implementing service architecture in SOA Based Forward Engineered Book Store Management System.

Proc. of the Second Intl. Conf. on Advances in Electronics, Electrical and Computer Engineering -- EEC 2013

ISBN: 978-981-07-6935-2

III. Result and Discussions

A. *Collective CPU Utilization analysis for three applications.*

Table 1: Collective data for CPU Utilization for Legacy, Forward Engineered, Backward Engineered Book Store Management System

Sr. No	Application Events	Legacy System	Forward Engineered System	Reverse Engineered System
1	Form load Manage Inventory	1.3	1.8	0.9
2	Manage Inventory Save	0.2	1.1	0.9
3	Manage Inventory Update	0.2	0.5	0.7
4	Manage Inventory Delete	0	0.2	0.4
5	Manage Inventory Search	0.3	0.2	0.2
6	Manage Inventory Close	0.3	0.5	0.5
7	Form load View Inventory	1.4	1.3	1.5
8	View Inventory Close	0.2	0.2	0.2
9	Form Load To Order	0.6	0.2	0.5
10	To Order Go	0.7	0.2	0.8
11	To Order Close	0.2	0.3	0.3
12	Form load Search	0.3	0.8	0.8
13	Search Go	0.6	0.5	0.4
14	Search	0.5	0.3	0.3
	Close			

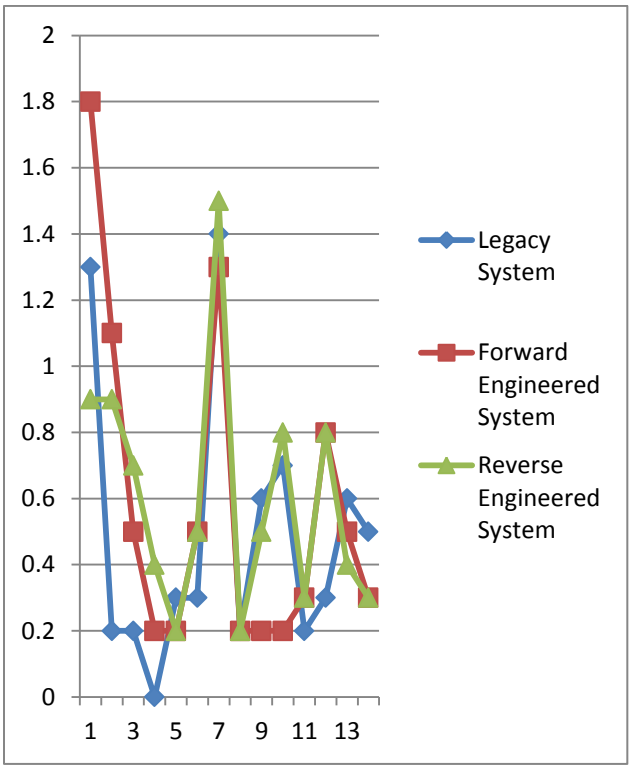

Fig 4: Collective data for CPU Utilization for Legacy, Forward Engineered, Backward Engineered Book Store Management System

From the above diagram, The CPU Utilization:

- Is *least* in Legacy Book Store Management System because all the operations are locally present and no overhead CPU utilization is done in calling, connection building, message passing to services
- Is *more* in SOA Based Forward Engineered Book Store Management System since some service architecture is used and some overhead CPU utilization is done in calling, connection building, message passing to services
- Is *most* in SOA Based Reverse Engineered Book Store Management System since whole application is based service architecture is used and mostly overhead CPU utilization is done in calling, connection building, message passing to services

Proc. of the Second Intl. Conf. on Advances in Electronics, Electrical and Computer Engineering -- EEC 2013

ISBN: 978-981-07-6935-2

B. *Collective Memory utilization for three applications.*

Table 2: Collective data for Memory Utilization for Legacy, Forward Engineered, Backward Engineered Book Store Management System

Sr. No	Application Events	Legacy System	Forward Engineered System	Reverse Engineered System
1	Application Start-up	3.1	4	3
2	Form load Manage Inventory	3.4	4.6	3.3
3	Manage Inventory Save	4.7	5.2	4.5
4	Manage Inventory Update	4.8	4.7	4.6
5	Manage Inventory Delete	4.8	4.7	4.6
6	Manage Inventory Search	4.6	4.5	4.5
7	Manage Inventory Close	4.8	4.9	4.8
8	Form load View Inventory	6.4	5.2	5.1
9	View Inventory Close	4.9	4.9	4.9
10	Form Load To Order	6.5	6.8	5
11	To Order Go	6.9	6.9	6.5
12	To Order Close	5	4.9	4.8
13	Form load Search	6.5	6.4	6.3
14	Search Go	7	6.7	6
15	Search Close	4.9	4.8	4.8

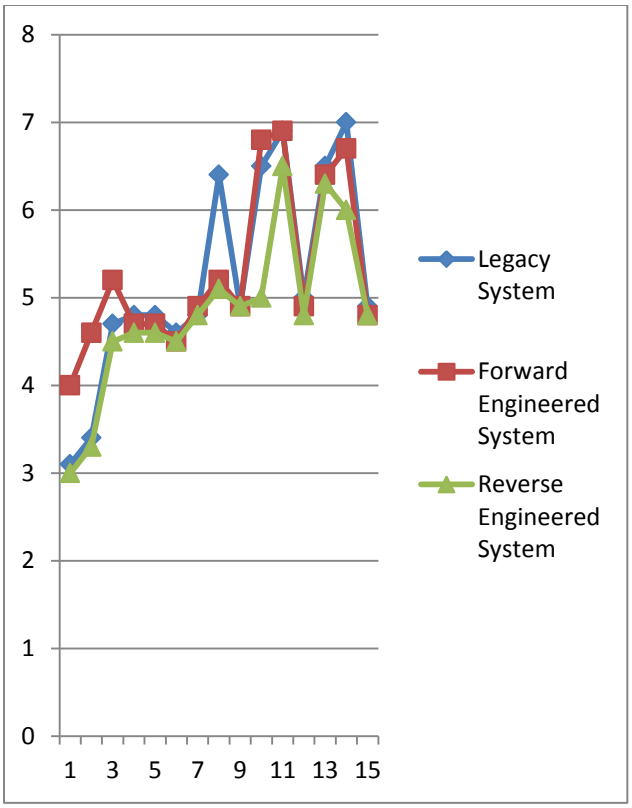

Fig 5: Collective data for Memory Utilization for Legacy, Forward Engineered, Backward Engineered Book Store Management System

From the above diagram, The Memory Utilization:

- Is *most* in Legacy Book Store Management System because whole application is loaded into the memory once it is executed
- Is *more* in SOA Based Forward Engineered Book Store Management System because only those parts that are not service oriented are loaded into the memory and rest of the part is loaded when the service is called or required
- Is *least* in SOA Based Reverse Engineered Book Store Management System because very less part of the application gets loaded into the memory as most of the part of the application is based on service architecture. So, services are loaded to and executed from memory only when required or called.

So, from the above points it is clear that there is a trade-off between Memory and CPU utilization among the three applications. So, it is recommended that, correct type of application should be chosen according to the hardware platform. If we are migrating from Legacy to SOA bases application then we must consider following points:

ISBN: 978-981-07-6935-2

- If system has sufficient amount of memory, and low CPU power, then the decision of migrations should be reconsidered. Because more and more the application will become service oriented, more CPU utilization will be done.
- If system has balanced amount of CPU power and memory, then SOA based Forward Engineering of Legacy System can be considered. Because Forward Engineering of Legacy System doesn't completely make it SOA based and an organization can get more out of it.
- If the system has more CPU power but less memory, then SOA Based Reverse Engineering of Legacy System can be considered. Because Reverse Engineering of Legacy System makes it SOA based completely and not whole application is loaded in the memory at start-up and services are loaded only when they are required. But additional CPU overheard is done due to calling, connection building, and message passing to services.

IV. Conclusion and Future Scope

The thesis tries to fill the agreement gap between Industry and Academia upon the migration of legacy system to SOA system. Academia proposes that legacy system should be reverse engineered whereas Industry Forward Engineer the legacy system while migrating to SOA system. In this research, Memory and CPU utilization of various events of legacy system, Forward Engineered SOA system, and Backward Engineered SOA system were recorded. It was found that CPU and Memory have a trade-off in all three applications. It has been analysed that:

1. CPU utilization is least and Memory utilization is most in legacy system.
2. CPU utilization is more and Memory utilization is more in Forward Engineered SOA system.
3. CPU utilization is most and Memory utilization is least in Backward Engineered SOA system.

Recommendation is given that migration technique should be chosen by keeping these trade-offs in consideration.

The application used as legacy and migrated to SOA had only one module. And thus only one service was taken into consideration. In future, more complex applications with more modules can be migrated involving more services in SOA system and their CPU-Memory trade-off will further enhancement in this research.

V. References

[1]http://en.wikipedia.org/wiki/Service-oriented_software_engineering
[2]Kamari Saeid (2012) "*A Conceptual Overview of Service Oriented Software Systems Development*", Journal of Basic and Applied Scientific Research
[3]Khadka Ravi (2012) *"Service Identification Strategies in Legacy-to-SOA Migration"*, Springer
[4]C. Zillaman, A. Winter (2011) *"The SOAMIG Process Model in Industrial Applications"*, ACM
[5]Balagurusamy, E. (2008) *Programming in C#*, Tata McGraw Hill, New Delhi
[6]Bohra Rashmi, Rathore V.S. (2011) *"An Evaluation of Services Development in e-Commerce Migrating to SOA"*, International Journal of Soft Computing and Engineering(IJSCE)
[7]Chapin Ned (2010) *"Software Characteristics of SOA"*, ACM
[8]E. Di Nitto, D. Meilander, S. Gorlatch, A. Metzger, H. Psaier, S. Dustdar, M. Razavian, D.A. Tamburri, P. Lago (2012), *"Research Challenges on Engineering Service-Oriented Applications"*, IEEE
[9]Elmasari, R.; Somayajulu D.V.L.N.; Navathe S.B.; Gupta S.K. (2008) *Fundamentals to Database Systems*, Pearson Education, Delhi
[10]Jerker Delsing, Fredrik Rosenqvist, Oscar Carlsson, Armando W. Colombo, Thomas Bangemann (2012) *"Migration of Industrial Process Control Systems into Service Oriented Architecture"*, 2012
[11]Karbunen Harri, Jantti Marko, Anne Eerola, (2012) "*Service Oriented Software Engineering (SOSE) Framework*", IEEE.
[12]Oldevik J. , Olsen G.K. (2011) *"Model Driven Migration of Scientific Legacy Systems to Service Oriented Architecture"*, Joint Proceedings of MDSM 2011 and SQM 2011
[13]Razavian Marayam, Lago Patricia (2011) *"A Survey of SOA Migration in Industry"* , Springer – Verlag Berlin Heidelberg 2011
[14]http://www.oracle.com/us/technologies/soa/soa-suite-066466.html
[15]www.microsoft.com/sqlserver/en/us/default.aspx
[16]www.homeandlearn.co.uk/csharp/csharp.html