Soft Architectural Distance (SAD): How Far is the Legacy Architecture from SOA Architecture Principle?

Amit Mishra[1], Pradeep Tomar[2] and Anurag Singh Baghel[2]
[1](Research Scholar, Gautam Buddha University, Greater Noida)
[2] (Assistant Professor, Gautam Buddha University, Greater Noida, UP (INDIA))

ABSTRACT: *It is a well-accepted fact that technology is changing rapidly, hence the need to keep upgraded with latest trends is the key to success. This is especially true for big enterprises where the challenge is not only to keep the enterprise running but also to collaborate with other enterprises doing similar business. So, there is a definite need to transform such legacy applications running in organization. This study presented here helps to analyze the current state of LEGACY software architecture and develops a methodology to compare it with Service Oriented Architecture (SOA). The notion of Soft Architectural Distance (SAD) is developed in this work. Degree of SADness will indicate the level of gaps in current architecture w.r.t. Service Oriented Architecture (SOA). Higher the SAD score greater is the GAP and likely to enter in legacy crisis. This SAD score is normalized to value 1 (highest). Then threshold can be defined / decided suitably for any organization. If the evaluation score goes beyond this threshold value it is recommended to improve the architecture or move next steps in legacy transformation strategy. Work is aimed to benefit industry in decision making process for legacy transformation projects. It is also useful in architectural analysis of existing systems.*

Keywords: *Legacy transformation, Service Oriented Architecture, SOA, Soft Architectural Distance, Design Principles*

I. INTRODUCTION

A legacy system is an operational system that has been designed, implemented and installed in a radically different environment than imposed by the current ICT strategy [1]. With this situation in hand, big organizations are often facing a challenge to transform their systems to modern packaged / standard solutions. As observed by [2] upward migration from procedural and structured programs to object-based technologies is very difficult and it is often impossible to predict how the system is going to evolve during the process of development. Most important factor affecting the legacy application and its transformation strategy is it's current state of architecture. Most likely if LEGACY is in bad shape it will be far away the standard design principles. Reason behind this gap in architecture is incremental changes in the software for a long period of duration. When enterprises define a legacy transformation and its roadmap, evaluation of current architectural state is essential. This evaluation has to be with some benchmark strategies and industry standards. Nearness or farness of the current architecture and design principles from the standards indicates which strategy to follow. SOA is one of such principles. This study focuses on the evaluation methodology to determine the nearness or farness of current state of architecture w.r.t. SOA architecture principles. Study presented here is limited to the LEGACY applications in Visual Basic for graphical user interface, Tuxedo as middleware, C, Pro*C, PL/SQL and shell scripts as back end layer. But the same approach can be extended to other technologies and packages.

II. RELATED WORK

In recent past years, a lot of studies were done to define SOA methodology and principles. A kind of fashion trend in industry and academia was created. Many studies have focused on migrating LEGACY to SOA. They enlisted the approaches to SOA migration and steps to follow. But none found specific on evaluating architectural deviations in LEGACY vs. SOA to define the degree of nearness or farness.

SOA is recognized as a way to improve business flexibility, adaptability and better align IT through the breakdown of technology-driven barriers among internal and external organizations. The promise is powerful but SOA raises unique challenges and can be derailed unless an effective governance framework is established to clearly identify roles and responsibilities. According to a Gartner report, "SOA governance is not an option, it is an imperative". Gartner estimates that the failure to implement working SOA governance mechanisms will be the most common demise of SOA projects. Malinverno, Vice President of Research at Gartner further suggested that "Reuse is not a benefit of SOA but a hurdle that

needs to be overcome in order to improve business agility and lower software maintenance [3]. Khadka et al. [4] provided very good insight on existing literature around SOA. They systematically reviewed 121 primary studies on legacy-to-SOA evolution using software reengineering reference framework. They provided holistic approach of study and overview, focusing on methods and techniques used in a legacy-to-SOA evolution. They conducted a systematic literature review to consolidate different approaches on SOA evolution, reported in past one decade. Based on this study they discussed current research approaches, methods, and techniques used and an identification of research directions for future researches. Managing business issues in transformation of LEGACY to SOA is key. Business IT alignment (BITA) is essential aspect of it. From a business perspective, a legacy to SOA migration is generally triggered by new business needs and goals. One of the challenges of legacy to SOA migration is to define an appropriate scope for "business-IT alignment" [5]. Key to overcome this challenge is to deconstruct, analyze and identify business components contributing towards business goals of the enterprise [6]. From a technical perspective, achieving and maintaining the non-functional characteristics of the legacy applications is a key challenge while developing a target architecture. Target architecture should consider a well-defined SOA objective. The Software Engineering Institute (SEI) has developed a SOA research Agenda [7] within which the migration of legacy applications to SOA is also reviewed. The agenda outlines research areas, each identified with its rationale, overview of current research, and highlights research challenges and gaps. It also provides details on specific research challenges related to the maintenance and evolution of service-oriented systems.

Razavian and Lago [8] highlighted potential gap between the theory and practice of legacy-to-SOA migration. They conducted a systematic survey in industry and based on the results they contradicted the existing academic research proving it to be theoretical [9], [10]. They discussed the differences and identified future research directions. Almonaies et al. [11] survey approaches that deal with moving legacy software to SOA. They compare approaches and highlight strengths and weaknesses, thus providing a model for assisting decision support for migration projects. [14] Reveals the organization's approach for SOA adoption. When it comes to define and evaluate architectural distance of existing application vs. SOA principles, studying the attributes of SOA design principles is required. They are collected and consolidated from different studies and reference documents from internet. [15] Presents a six-phase structured process that combines migration planning and migration execution aspects of a legacy to SOA migration.

III. SOA ATTRIBUTES

Notion of Soft Architectural Distance (SAD) is giving a rationale and quantification in gaps of architecture principles of SOA vs. legacy. This is based on SOA attributes. Following main attributes of design and architecture are considered in this study. It is supported at high level in [13] with more focus on enterprise architecture premises.

Abstraction: Service interface hides technical implementation details of services. In legacy applications it is common to find, mixed up in a kind of "architectural spaghetti", code fragments concerned with database access, business logic, presentation aspects and exception handling, among others. It is basically decoupling business logic with presentation logic [12**].Loose coupling**: Reducing dependencies between system components and making all remaining dependencies explicit. Loose coupling reduces the impact that a change to one component has on other components, thereby improving a system's maintainability. Coarse Grained Nature of Services: One Logical unit of work to be appropriately defined in order To Form Services of Desired Granularity [12].Compliance: Services Follow All Standards and Norms, Which Helps in Reusability and Easy Consumption by Variety of Services and Domains or Technologies. Interoperability: Interacting One Service to Others. Facilitates, Ways To Call Services From Each Other Helping Also To Define Appropriate Level Of Granularity. Searchablity: Easy To Find Suitable Services And Parameters To Consume, It Could Be A Declared Publishing Of Services With Signatures And Consumption Protocols. Replaceability: Easy Ways to Replace One Service by a Similar Service. Encapsulation: Internal Implementation Details Should Not Be Required By The Consumer. It Is A Way To Expose A Discrete System Capability As An Autonomous Unit That Can Be Used By Any Consumer Application Requiring This Feature. Encapsulation Hides The Inner Workings Of The Capability. Law Of Composition: A Service Should Be Designed Such That It Can Be Combined With Other Services To Accomplish A Larger Process. Service Autonomy: Services Not Dedicated To Any Particular Application Or Business Domain. The List Of Above Attributes Can Be Further Classified As Primary or Fundamental SOA Attributes and Others Are Derived SOA Attributes. Classification Is Mentioned Below.

Table I Classification of SOA attributes as Fundamental and Derived

Primary/Fundamental SOA attributes	**Derived SOA attributes**
ABSTRACTION, LOOSE COUPLING, COARSE GRAINED NATURE of services	COMPLIANCE, INTEROPERABILITY, SEARCHABLITY, REPLACEABILITY, ENCAPSULATION, LAW OF COMPOSITION SERVICE AUTONOMY

IV. SOFT ARCHITECTURAL DISTANCE (SAD)-METHODOLOGY

Primary or Fundamental attributes will have higher weightage in evaluating any existing application's architecture, while the derived attributes play minor role in evaluation. Above 10 identified attributes is weighted in total of 100, the fundamental attributes occupy 60% weight while rest seven derived attributes contribute on 40%. This weightage is taken on the empirical experience of execution. While in general the weights could change but approach discussed in this paper would remain applicable. The method of analysis is to be done via asking answers to the twenty five primitive questions around the landscape and architecture of concerned software. Again these questions are recommendations of experts and based on experience. Against each answer as "YES", cell values will be counted from the matrix (table) to derive observed closeness score to each SOA attribute. If answer is not "YES" the values are not to be counted from matrix while determining the closeness score. In the best case each attribute will attain it's maximum value and will reach to closeness score equals to weight factor (W) for the corresponding attribute. Below is the snapshot of this table to compute the observations after the analysis and answers to each question. Weights to each attribute are denoted by (W_i). After the answers are completed for each question the scores under matrix to be added at the bottom. This sum will give the observed scores. Observed scores are denoted by (O_i).

Table II SOA analysis matrix of LEGACY systems, to derive closeness score for SOA attributes

Q. Sr. No.	AS-IS architecture analysis Questions	Answer : YES/ NO ?	ABSTRACTION (W=20)	COARSE GRAINED (W=20)	LOOSE COUPLING (W=20)	COMPLIANCE (W=5)	INTEROPERABILITY (W=5)	SEARCHABLITY (W=5)	REPLACEABILITY (W=5)	ENCAPSULATION (W=10)	LAW OF COMPOSITION (W=5)	SERVICE AUTONOMY (W=5)
1	Packaged ?	YES	0.8	0.5	0.5	0.2	0.2	0	0	0.5	0.1	0.1
2	Home grown?	YES	0.5	0	0	0	0	0	0	0	0	0
3	Frame work used ?	YES	0.5	0.2	0.2	0.2	0.2	0.2	0.2	0.5	0.4	0.1

Table III Questionnaire set for SOA analysis matrix

Q. Sr. No.	AS-IS architecture analysis Questions
1	Packaged solution?
2	Home grown applications?
3	Development is done using framework?
4	Batch Programs in 3GL (3rd generation languages)?
5	3-Tier architecture to keep presentation and biz layer separated?
6	Monolithic structure?
7	Ecosystem is heterogeneous (e.g. C, JAVA, SAP, DB scripts etc.)?

8	Nature of services is database centric?
9	Nature of services is functionality/logic centric?
10	Age of LEGACY in range of 20 years?
11	Age of LEGACY in range of 10 years not older?
12	Age of LEGACY in range of 5 years not older?
13	Services used by SELF/mother application only?
14	Services published and used by self and others?
15	Application/service design is in the form of functions for one logical set of actions?
16	Services / functions follow international level standards?
17	Services do not follow international standards but follow at organization level standards?
18	Functions and Services: signatures clearly defined?
19	Functions do not use global or static variable?
20	Functions/Services signatures are published at common place?
21	User Requirements Document (URD) exists at product level?
22	User Requirements Document (FGD) exists at product level?
23	User Requirements Document (DD) exists at product level?
24	User Requirements Document (TGD) exists at product level?
25	Service names represent the purpose correctly (brief description compliments it)?

Once observations about closeness for each attribute are available, SAD value for software/application under question can be derived. SAD will be proportional to the difference between corresponding weight and observation score (SAD α (Wi-Oi)). At the same time SAD will also have to consider the weights of each attribute. Since the methodology has to normalize the SAD on the scale of "1", following formula can be used to determine the notional value of SAD. This is called degree of SAD ness.

Soft Architectural Distance (SAD) degree =

$$\frac{\sum_{i=1}^{n} Wi(Wi-Oi)}{\sum_{i=1}^{n} Wi^2}$$

Table IV Illustration of computation of degree of SAD ness

	Weight Score (W_i)	Observed Analysys Score (Oi)	(W_i -Oi)	wi(wi-Oi)	Wi*Wi
ABSTRACTION	20	4	16	320	400
COARSE GRAINED NATURE of services	20	5	15	300	400
LOOSE COUPLING	20	2	18	360	400
COMPLIANCE	5	2	3	15	25
INTEROPERABILITY	5	1	4	20	25
SEARCHABLITY	5	1	4	20	25
REPLACEABILITY	5	1	4	20	25
ENCAPSULATION	10	3	7	70	100
LAW OF COMPOSITION	5	2	3	15	25
SERVICE AUTONOMY	5	2	3	15	25
	100		Sum Wi*(W_i -Oi)	1155	1450
Soft Architectural Distance (SAD) score = **Sum (Wi*(Wi -Oi))/Sum (Wi)²**				0.797	

V. RESULTS AND DISCUSSION

This study describes a scientific approach to analyze the state of architecture. To validate the notion of SAD, real live examples are used to support the study. The exercise of SAD computation was applied on 10 applications and then compared the number of live defects recorded in remedy tool. SAD score high follows the trend of high defect counts with in 2 quarters of new release on these applications. This indicates poor architectural structure. This proves that the computation of degree of SAD ness corresponds to the notion of the concept. Higher is the score poorer the architecture current state. The validation can be further done with respect to the productivity of evolution change requests as well.

Similar trend can be derived from the productivity data as well. Normally if the architecture is not well done the change request implementation will also be costly resulting into low productivity. This is evident as there will be less reusability, if business logic is not separated it is prone to have regression errors. Similarly other attributes of SOA, like absence lose coupling will also complicate the new enhancements to existing software. It is observed in study that if the degree of SAD ness is below 0.25 the architecture is quiet safe and more it goes high more is the poor condition and it's repercussions. In context to LEGACY transformation, high degree of SADness indicates higher efforts in transformation.

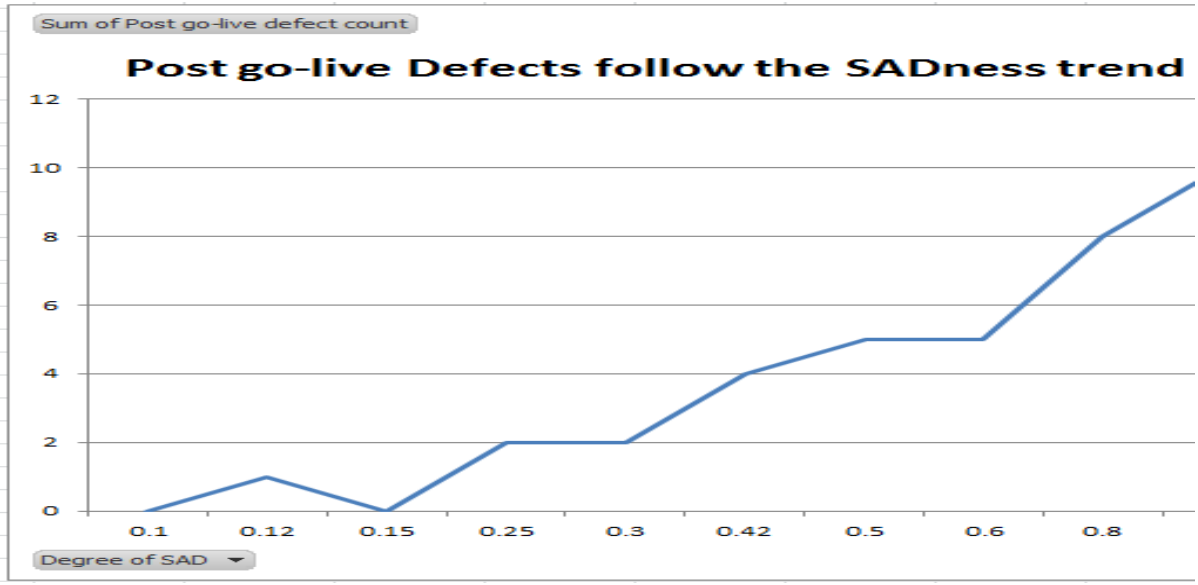

Fig1: Defect trend w.r.t. degree of SAD

VI. CONCLUSION

The evaluations of the architecture are never been a simple exercise. Even if they are studied mostly they are in the form of subjective analysis. With the subjective results of study the decision making is always tough. This paper presents an approach to study and rate a legacy application vs. SOA architecture principles. This paper contributes to the objective approach and determination. The soft architectural distance is normalized to the degree of SAD ness with maximum value=1, and recommendation to treat an architecture good or bad on SOA benchmark has been made very easy and methodical. It is expected to help software industry for architectural evaluations, especially for the companies which are running on LEGACY applications and thinking to go for transformation. This study can further be extended to automate the evaluation by parsing the software code and deriving similar answers as manually presented here. Moreover the approach can be reused to other architecture standards and not only limiting this to SOA. Methodology discussed in this paper is purely manual. Further to extend the work there is a need to automate the evaluation for few attributes using same methodology. So, that computation of degree of SAD ness is made easy and fast.

REFERENCES

[1] Du Bois, B., "Towards an ontology of factors influencing reverse engineering. In STEP '05:", Proceedings of the 13th IEEE International Workshop on Software Technology and Engineering Practice, pages 74–80, Washington, DC, USA. IEEE Computer Society. (2005).
[2] H. Tromp and G. Hoffman. "Evolution of legacy systems, strategic and technological issues", based on a case. Paper also accepted to the workshop on Evolution of Large-Scale Industrial Software Applications (ELISA), 23 September 2003, ICSM 2003.
[3]Saran, Cliff, "SOA will fail without governance warns Gartner," http://ComputerWeekly.com, September 7, 2006

[4] R. Khadka, A. Saeidi, A. Idu, J. Hage, and S. Jansen, "Legacy to SOA Evolution: A Systematic Literature Review, Migrating Legacy Applications: Challenges in Service Oriented Architecture and Cloud Computing Envioronment", IGI, 2013
[5] S. Murer, B. Bonati, and F. J. Furrer, "Managed Evolution." Springer, 2011
[6] L. Cherbakov, G. Galambos, R. Harishankar, S. Kalyana, and G. Rackham, "Impact of service orientation at the business level," IBM Sys.J.,vol. 44, no. 4, pp. 653–668, 2005.

[7] G.A. Lewis, D.B. Smith, and K. Kontogiannis, "A Research Agenda for Service-Oriented Architecture (SOA): Maintenance and Evolution of Service-Oriented Systems," technical report, Software Eng. Inst., 2010.
[8] M. Razavian and P. Lago, "A Survey of SOA Migration in Industry," Proc. Ninth Int'l Conf. Service-Oriented Computing, pp. 618-626, 2011
[9] M. Razavian and P. Lago, "A Frame of Reference for SOA Migration," Towards a Service-Based Internet, pp. 150-162, Springer, 2010
[10] M. Razavian and P. Lago, "Towards a Conceptual Framework for Legacy to SOA Migration," Proc. Int'l Conf. Service-Oriented Computing, pp. 445-455, 2009.
[11] A.A. Almonaies, J.R. Cordy, and T.R. Dean, "Legacy System Evolution Towards Service-Oriented Architecture," Proc. Int'l Workshop SOA Migration and Evolution, 2010.
[12] Carlos Matos, Reiko Heckel, "Migrating Legacy Systems to Service-Oriented Architectures", Proceedings of the Doctoral Symposium at the International Conference on Graph Transformation (ICGT 2008)
[13] A S Alwadain and Fayez Hussain, "A Model of the Factors Influencing Enterprise Architecture Evolution in Organizations", IEEE 12th International Conference on Digital Object Identifier: 10.1109/ITNG.2015.126, Publication Year: 2015 , Page(s): 740 – 743
[14] Marc N Hanes and William Haseman, "Service-Oriented Architecture Adoption Patterns", 42nd Hawaii International Conference on Digital Object Identifier: 10.1109/HICSS.2009.388 Publication Year: 2009 , Page(s): 1 – 9
[15] Ravi Khadka, Amir Saeidi, Slinger Jansen, Jurriaan Hage, "A Structured Legacy to SOA Migration Process and its Evaluation in Practice", IEEE International Symposium on the Digital Object Identifier: 10.1109/MESOCA.2013.6632729, Publication Year: 2013 , Page(s): 2 - 11