I J C T A, 9(20), 2016, pp. 445-451
© International Science Press

Multistage Legacy Software Crisis Detection Matrix

Amit Mishra*, Pradeep Tomar and Anurag Singh Baghel*****

ABSTRACT

Enterprises which are still relying on legacy applications for their core functions face issues in maintaining these solutions at viable cost. This situation compels to think for replacing the legacy applications with suitable modernization. While defining a legacy transformation program and its roadmap, often the core issue comes to identify the state when one can say that now it's the time to start the legacy transformation. At this stage the maintenance of the software becomes very costly. Increased number of regression bugs, non-robust software code raise fear in developers mind to touch the code. In general LegacySoftware usually suffer from lack of packaging, redundancy in software components leading discrepancies, out of support or warranty, issues in collaboration with partners who are already advanced, scarcity of skilled resources on old technologies, software distribution and web enabled interface, increasing maintenance and support cost, high number of incidents in production environment. This situation starts eating most of Information and Communication Technologies (ICT) budget and resources in legacy maintenance and leaving no room for innovation. In nutshell, this situation is called "LEGACY CRISIS". This work is targeted to develop, "Multistage Legacy Crisis Detection Matrix". This is a new model developed for legacy crisis detection, no other model exists today.In the first stage it gives symptom based detection for the management to initiate the transformation process by using the fact based operational matrices. The next level of detection goes more at architectural and code level to do the assessment of the legacy software crisis stage in accordance to the leading design principles.

Keywords: Architectural Analysis; Design Principles; Legacy Crisis; Legacy Transformation; Software Reengineering

1. INTRODUCTION

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. Many enterprises started to use the software systems initially with the objective to ease some of the activities but later they were converted to semi-automated systems. Finally with time they evolved as a back-bone of the organization. Such a set of software are called LEGACY today. With this situation in hand, big organizations are often facing a challenge to transform their systems to modern packaged solutions. Technology evolution is reshaping the way to do information technology (IT), offering a wide variety of development technologies/ languages, packaged solutions, cloud solutions, infrastructure options, hosting options with a high speed of implementation at acceptable costs. These evolutions are also resulting in a high speed of redundancy of existing IT solutions of enterprises, support and availability of experienced staff tends to dwindle, forcing them to evolve and keep abreast with the evolution. Organizations intending to replace their legacy systems to the new sustainable solutions generally apply best-in-class solution approaches and architectural designs but still they face lot of challenges to take the decision when to start this journey.

* Solutions Specialist, Department of ICT ST-Microelectronics India Pvt. Ltd. Greater Noida, U.P., India, *Email: amitg3@gmail.com*

** Assistant Professor, Department of Information and Communication Technologies Gautam Buddha University, Greater Noida, U.P., India, *Email: parry.tomar@gmail.com*

*** Assistant Professor, Department of Information and Communication Technologies Gautam Buddha University, Greater Noida, U.P., India, *Email: asb@gbu.ac.in*

When enterprises define a legacy transformation and its roadmap, often the core issue comes to identify the state when one can say that now it's the time to start the legacy transformation. At this stage the maintenance of the software becomes very costly. Increased number of regression bugs, even small change gives fear in developers to touch the code. In general Legacy software usually suffer from lack of packaging, redundancy in software, discrepancies, out of support or warranty, issues in collaboration with partners who are already advanced, scarcity of skilled resources on old technologies, software distribution and webinterfaces. These all increase maintenance and support cost leading to "Legacy Crisis". In such a situation most of ICT budget and resources are eaten up by the maintenance of legacy, leaving no room for innovation. Finally the need is to do the legacy transformation.

Legacy transformation decision could be based on two main bases. Either it is strategic decision when enterprise goes for phasing out all old and legacy software and deploy new suit of applications or package. In strategic decisions the maintenance cost and operational issues don't play big role in decision making. Other base for legacy transformation decision could be based on legacy crisis situation when organization is forced to think for legacy transformation to remain viable. "Multistage Legacy Crisis Detection Matrix" is comprehensive mechanism of determining point of legacy crisis. In this mechanism the first stage gives symptom based detection for the management to initiate the transformation assessment. The next level of detection is architectural leveland code level assessment of the legacy software crisis stage. Approach of assessment in two stages gives flexibility and optimization of effort in crisis detection.

In summary, legacy transformation is never a straight forward initiative for many reasons. Beyond the traditional fears of management about operational risks associated with the transformation and the challenge with correct technology and solution selection, it is important to detect if legacy crisis already occurred and now is the time to start transformation.

2. LITERATURE SURVEY

Legacy systems are defined and attempts to cope with them are described in earlier works. In early works Bennett(1995) [1] defined legacy systems as vital software for the organization but no one knows how to manage it and address associated issues. Such systems are supported mostly by outdated technologies and platforms but still these are core for the organization.Legacy systems are mostly with monolithic old architectural design, use old programming languages and hence lack, good interfaces (Bisbal et al., 1999) [2]. Sneed (2006) [3] also shared the view that legacy systems are outdated from technology perspective but still very useful for organization. Depending on their dependency with the external environment they are hard to maintain and evolve. In 2008, CY Lin[4] mentioned issues related to legacy databases and talked about getting rid of legacy through different stages to finally the transformation stage.

Legacy transformation studies from industrial perspective, going through the research papers from various journals and white papers from software companies to have good blend of theory and practical approaches on field. The 'Business Value of Legacy Modernization (Microsoft)' was a white paper from Microsoft-supported programs in 2007[5]. Another practical approach was presented in 2008 [6] for rescuing a troubled software projects by team transformation. This study highlighted that not only technical reasons are making legacy software unmanageable but also the softer issues play bigger role. This is also validated by Khadka et al., 2013 [11] that technological aspect are perceived most crucial in software development but organizational aspects and softer issues have a more significant impact on the actual outcome.In 2012, TCS also published a white paper [8] to highlight the different approaches for modernization, where they have broadly classified it into six main approaches. These 6 approaches were- Business rules externalization, Migration, Package implementation, Product rationalization, Reengineering and Restructuring. The modernization approach for an organization depends on its current state of business alignment, technology used and architecture fitment. A research work done, which attempts to explore the perception of industry

professionals about the legacy systems and their transformation [10]. Also this work analyzes the relationship between industry and academia. Elaborates on the discrepancy in detail using an explorative study to investigate how the practitioners perceive legacy systems.

Based on the study already done, it helps to determine the issues in legacy and coping with them in best possible manner either to maintain legacy or to transform. But none of them help minimize the dilemma of maintaining or transforming legacy. Current research work reveals scientific approach to detect the legacy crisis point and help organizations to define legacy transformation strategy and roadmap with increased confidence. This is done in multiple stages. First stage of detection is operational Key Process Indicators (KPIs) of legacy systems [12] and next stage of detection is based on architectural study [13]. Both the stages together determine the legacy crisis point for the legacy software under question.

3. MULTI-STAGE: LEGACY CRISIS DETECTION METHODOLOGY

Multistage legacy crisis detection methodology gives the crisis detection result in 2 stages. Stage-I is focusing the economic viability attributes via operational matrices. This is primarily at the management level on the recommendation of enterprise architects. While the other stage-II considers the software architecture, design and coding level attributes.

In stage-I, the economic viability attributes, cover the operational issues, incidents reported every month, average effort required closing one incident, average support time required to answer one user query,size and effort ratio for developing a Change Request (CR) and regression defects at the early days of new software version deployment. These matrices are usually operational in nature and are periodically being monitored by management in Operations Reviews (ORs) every month. When the organization's technological roadmap is superimposed with operational reports it gives the Legacy Crisis Symptom Score(LCSS).

If LCSS crosses a threshold limit, detection enters into stage-II., which uses more the existing technical architecture, programming languages, platform integrations and software source code etc. At this stage the tool will again give a heuristic value to indicate the degree of crisis in legacy software. This heuristic value of architectural evaluation is named as Soft Architectural Distance (SAD) [13]. Degree of SADness is key parameter of stage-II in legacy crisis detection matrix. The model developed in this study is a new model no other model exists today on legacy crisis detection.

4. STAGE-I: LEGACY CRISIS DETECTION VIA OPERATIONAL METRICES

Stage-I of legacy crisis detection uses operational matrices as parameters and outcome of this exercise is the LCSS score. It works on the weighted factor method for different attributes contributing to legacy crisis. Determination of these attributes and their weights has been a rigorous exercise. These are determined first via interviewing software engineers, architects, service managers, solution managers and solution directors followed by a survey organized inside the organization. Even if these attributes are common with legacy in every organization still they need to be revalidated in context to specific organization. LCSS computation uses a benchmarking technique to compare the matrices of legacy system vs. average matrices of three most recent (non-legacy) applications in the organization. While comparing with benchmarked applications a normalization process is applied with respect to the user base (number of users using the application) and size of application in terms of Function Points (FPs).

Main steps to compute Legacy Crisis Symptom Score (LCSS)-

Step-I: Determination of attributes and their weights contributing to legacy crisis – operationally

Step-II: Chose 3 non-legacy most recent applications in organization to be used for comparison to determine LCSS

Step-III: Calculate normalization factor (NF) for legacy application vs. 3 Benchmarked Applications (BAP) in context to user base and size in FPs

$$NFU = \frac{\text{Number of active users of BAP1} + \text{Number of active users of BAP2} + \text{Number of active users of BAP3}}{3*(\text{Number of active users of legacy system under question})} \quad (1)$$

$$NFS = \frac{(\text{Number of FPs for BAP1} + \text{Number of FPs for BAP2} + \text{Number of FPs for BAP3})}{3*(\text{Number of active users of legacy system under question})} \quad (2)$$

Finally, the normalization factor for legacy system

$$(NF) = (NFU+NFS)/2 \quad (3)$$

Step-IV: Matrix computation with weight factor based on 10 attributes finalized

Table 1
Legacy Crisis Symptom Score Matrix

Attribute	*Weight Factor % (W)*	*LEGACY score (L)*	*BAP1 Score (M)*	*BAP2 Score (N)*	*BAP3 Score (O)*	*BAP Avg. (Y)*	*Legacy Factor (Z=L/Y)*	*Weighted Factor (W*Z*NF)*
Number of Interrupts to Production (ITPs) in a quarter	5	L1	M1	N1	O1	(M1+N1+O1)/3	L1/Y1	W1*Z1*NF
Number of incidents reported (per month)	15	L2	M2	N2	O2	(M2+N2+O2)/3	L2/Y2	W2*Z2*NF
Number of questions asked by users on application behavior (per month)	10	L3	M3	N3	O3	(M3+N3+O3)/3	L3/Y3	W3*Z3*NF
Number of incident ticket violated Service Level Agreement(SLA)	15	L4	M4	N4	O4	(M4+N4+O4)/3	L4/Y4	W4*Z4*NF
Number of problem tickets created (per month)	10	L5	M5	N5	O5	(M5+N5+O5)/3	L5/Y5	W5*Z5*NF
Number of average man days required in Keep Lights On (KLO) support (per month)	17	L6	M6	N6	O6	(M6+N6+O6)/3	L6/Y6	W6*Z6*NF
Effort vs. CR size	10	L7	M7	N7	O7	(M7+N7+O7)/3	L7/Y7	W7*Z7*NF
Number of defects in the 1st month of new version deployment	8	L8	M8	N8	O8	(M8+N8+O8)/3	L8/Y8	W8*Z8*NF
Regression Defects reported in the 1st month after new version	5	L9	M9	N9	O9	(M9+N9+O9)/3	L9/Y9	W9*Z9*NF
Attrition rate(per annum) of engineers working in legacy area	5	L10	M10	N10	O10	(M10+N10+O10)/3	L10/Y10	W10*Z10*NF
Sum (Weights)	100	NF= Normalization factor wrt to FP size and User base				$LCSS = NF \sum_{k=0}^{10} Wk * Zk$		

Step-V: Compute final LCSS score as

$$LCSS = NF \sum_{k=0}^{10} Wk * Zk \quad (4)$$

Step-VI: Decision to enter into second stage of computation after crossing the limit of threshold value of LCSS. This threshold value is organization specific. But it is observed that any score more than 300 is high enough to go for stage-II.

5. STAGE-II: LEGACY CRISIS DETECTION VIA ARCHITECTURAL ANALYSIS

Stage-II of legacy crisis detection is more on the technology front. It's on the aspects of current architecture of legacy. In this study architectural evaluation is done with respect to Service Oriented Architecture (SOA) principles. Depending on the degree of deviation from SOA standard architecture principle is used to detect the legacy crisis point. Notion of SAD [13] represents the state of architecture of legacy w.r.t. SOA. It uses SOA attributes and answer to 25 architecture questions in context to legacy system under question.Weights to each attribute are denoted by (Wi).Observed scores are denoted by (Oi).Degree of SADness is computed based on the formula-

$$\textit{Soft Architectural Distance (SAD) degree} = \frac{\sum_{i=1}^{n} Wi(Wi - Oi)}{\sum_{i=1}^{n} Wi^2} \quad (5)$$

SAD score high represents poor state of architecture. Organization specific threshold for this score is also derived to take the decision of legacy crisis point. Weights (W_i) are defined by the architects community of organization against each Service Oriented Architecture (SOA) attribute. Answers to the 25 questions as discussed in [13] will compute the SAD score. Once observations about closeness for each attribute are available, SAD value for software/application under question can be derived. Methodology normalize the SAD score on the scale of 1. This value closure to 1 represents bad architecture while lower values represent standard industry architecture. Complete approach is discussed in [13], however the illustration of computing SAD is depicted in below table for Sales Order application -

Table 2
Illustration of computation of degree of SADness

	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	

6. RESULTS AND CONCLUSION

Study presented here helped in detecting the legacy crisis point in 2 stages. In first stage the symptom is indicated in operational matrices, which indicates management to decide for coping with legacy crisis. This grants the management buy in automatically. This methodology was applied to detect the legacy crisis symptom indicated in matrices using Remedy tool reports for the Sales and Marketing legacy applications. LCSS score for Sales Order application computed more than 500, which is then submitted to stage-II for SAD score computation. SADness score for Sales Order was approximately 0.8. Both the stages qualify it perfectly for legacy crisis.LCSS score correctly gave the indication that most of the IT budget was eaten up

by KLO activities leaving no room to think for transformation. Results of stage-I were supported with stage-II results and management took decision to include the Sales Order, legacy application in transformation roadmap.Once the LCSS is framed for a legacy application in Excel sheet computation, it is almost free computation. As it is generated monthly, management has a trend to see if the situation is deterring. While it helps in detecting the legacy crisis point of legacy applications, it also indicates even for new application that while carrying over the changes the state of architecture and application robustness is not compromised.

7. SCOPE, LIMITATIONS AND FUTURE DIRECTIONS

Scope of this work was limited only to the detection of legacy crisis point. Current work evaluates the crisis based on exercise of SAD computation on architectural aspects where development team fills the assessment sheet and matrix technical parameters e.g. program structure, modularity, interdependency, encapsulation and service autonomy etc. Future work in this direction is to automate this assessment process with code scan to prepare the inputs for SAD computation. It will really exploit the methodology by making the assessment faster. So, this two stage model can be enhanced to three stage model by introducing this automated technical architecture assessment. So, the first stage will remains with LCSS computation on operational matrices, second stage is automated SAD computation, third and final stage would be refinement of stage two via human intervention to cover the historical aspects like age of legacy, reusability of components and it's extent.

Technical approaches to address the legacy transformation has already been worked by several authors in legacy transformation. However this transformation is not only the technical subject. There are several softer challenges which need to be addressed in legacy modernization journey. Typical softer challenges are: Natural inertia, fears of losing jobs, indispensability syndrome, legacy knowledge only remains in minds not digitized, week business case, not freezing the evolution of legacy while transforming it, not setting right expectations with stakeholders about the results of transformation etc. Solution to these issues are scope for the future work ongoing.

REFERENCES

[1] K. Bennett, "Legacy Systems,Coping with Success" IEEE Software, Volume:12 , Issue-1, ISSN: 0740-7459, Jan. 1995, pp.19–73.

[2] Bisbal, J., Lawless, D., Bing, W., & Grimson, J. "Legacy information systems: issues and directions." IEEE Software, Volume: 16, Issue-5, Pg. 103-111. 1999

[3] Sneed, H.M. "Wrapping Legacy Software for Reuse in a SOA.", Multikonferenz Wirtschaftsinformatik,2, 345-360. 2006

[4] Chang-Yang Lin, "Migrating to Relational Systems: Problems, Methods, and Strategies", Contemporary Management Research, Pages 369-380, Vol. 4, No. 4, December 2008

[5] Microsoft, "The Business Value of Legacy Modernization", http://www.microsoft.com , 2007

[6] Kim Man Lui and Keith C. C. Chan, "Rescuing Troubled Software Projects by Team Transformation: A Case Study With an ERP Project", IEEE Transactions On Engineering Management, Vol. 55, No. 1, 2008

[7] Venkatraghavan N Iyer-Infosys, "Legacy Modernization: Modernize and Scale", http://www.infosys.com , 2008

[8] Anupam Karar and Ramesh Kumar(TCS), "Transform Legacy Systems for Improved Customer Experience", http://feeds2.feedburner.com/tcswhitepapers,2012

[9] Oladipo Onaolapo Francisca1 and Anigbogu Sylvanus Okwudili, "A Software Reverse Engineering Methodology for Legacy Modernization", International Journal of Advances in Engineering & Technology, 2011

[10] Belfrit Victor Batlajery, "Revisiting legacy systems and legacy modernization from the industrial perspective", PHD Thesis University of Utrecht, Utrecht,the Netherlands, 2013

[11] Khadka, R., Saeidi, A., Jansen, S., Hage, J., & Haas, G. P.,"Migrating a Large Scale Legacy Applicationto SOA: Challenges and Lessons Learned.", Proceedings of the 20th WCRE, Koblenz, Germany. IEEE., Pages: 425 - 432,DOI:10.1109/WCRE.2013.6671318, 2013

[12] Amit Mishra, Pradeep Tomar and Anurag Singh Baghel, "TRANSFORMING FROM LEGACY TO PACKAGED/

STANDARD S/W SOLUTIONS : FOR BIG ENTERPRISES", Journal Of Software Engineering Tools and Technology Trends, ISSN: 2394-7292, Volume-3, Issue-1, Jan-Apr 2016, Pg.5-11

[13] Amit Mishra, Pradeep Tomar and Anurag Singh Baghel, "Soft Architectural Distance (SAD): How Far is the Legacy Architecture from SOA Architecture Principle?", AETM'16, Third National Conference on Advances in Engineering Technology & Management, IOSR Journal of Computer Engineering(IOSR-JCE), e-ISSN:2278-0661, p-ISSN:2278-8727, April 2016, Pg. 7-12