2022 21st International Symposium on Distributed Computing and Applications for Business Engineering and Science (DCABES)

The complexity attachment in modernization journey

Harveena Rambarassah
School of Computing Science and Mathematics
Kingston University
Kingston Upon Thames, KT1 2EE, UK
K0839075@kingston.ac.uk

Souheil Khaddaj
School of Computing Science and Mathematics
Kingston University
Kingston Upon Thames, KT1 2EE, UK
s.khaddaj@kingston.ac.uk

Abstract – |Often referred to as heritage systems, Legacy systems have proven their efficacy and durability over the decades but struggling to prove their viability in the digital market. Legacy systems are being modernized and migrated with a complexity that increases the cost and risk in those journeys. Their underlying characteristics require an in-depth assessment of the factors contributing to their complexity. Therefore, this paper proposes a systematic review to trace and analyze the emergence of characteristics leading to complexity attachment and associated factors and illustrate different techniques to assess factors.

Keywords - Legacy systems, Legacy characteristics, factors and Legacy assessment

I. INTRODUCTION

The modernization of legacy software has always been complex due to the process's risk, cost and time constraints. Despite the many qualities of those systems, software transformation and modernization have a high cost and risk. Their constitution is substantial because they have been deployed since the 1960's using the procedural programming paradigm. They have grown into millions of lines of codes with many years in development and deployment and poor documentation. Moreover, they suffer from many issues, including maintainability, supportability, and scalability. They cannot adapt to the latest development in hardware and software, such as multicore architecture and microservices, nor the latest technologies, such as edge computing, AI and data analytics.

The risk and success rate of modernization is crucial as business survival depends on those transitions by migrating their core IT applications and business rules. Hence, mitigating the current barriers of the modernization journey should be revisited. Those barriers are factors preventing a swift transition to a modern platform. We believe their underlying properties should be re-assessed to better understand their structural composition and dependencies before embarking on a migration journey.

Thus, the paper aims to investigate the underlying characteristics of legacy software and will demonstrate the different factors contributing to the complexity of those systems to improve and reduce the risk associated with modernization. Along with assessing the influencers of the modernization process, factors such as complexity, cost and effort should also be evaluated before embarking on a migration journey. We aim to use existing competencies from the current system strategically and systematically to reduce defects and failures. Therefore, different techniques and cost drivers have been addressed to mitigate the impact of factors such as costs, efforts, and risks.

This paper is structured in the following manner: Section II explains why legacy modernization has been adopted in the industry; Section III discusses different underlying characteristics that the legacy system is currently supporting; Section IV carries out a legacy assessment to underpin different factors that are impacted by complexity such as cost, effort, and risk and finally section V concludes the paper with some suggestions.

II. LEGACY MODERNISATION

The modernization of legacy systems (LS) contradicts the Lehman law of system evolution [19]. The adaptability issue with modern interfaces and technology is forcing the evolution of legacy systems [2]. The transfer to newer technologies or platforms has many benefits like competitive edge, happier clients, future-ready business, unlocking big data opportunities, better performance and reliability improvement [23]. The digital market adopted those systems by a considerable percentage, forcing sectors such as banking, insurance and life sciences to re-strategize their core business structure [16]. However, there are many challenges when dealing with legacy systems. Primarily, its cumbersome nature from underlying characteristics such as size and inflexibility of code increases the risk of migration. Along with a complexity issue, there is also a very high maintenance cost to support those many lines of codes, with around 85%-of the IT budget spent on the maintenance of those systems [3]. Modernization needs the evolution of modern paradigms and obsolete technologies, businesses cannot cater for digitalized market and offer a competitive edge. Platforms cannot be versatile, agile, or flexible as they are not built with modern technologies, resulting in loss of business. Organization have not been able to cater for the innovation budget due to the increasing cost of maintaining legacy systems [3]. The unavailability of technical documentation has increased both the cost and staffing effort in understanding the system. Moreover, its complexity has also progressed towards expensive maintenance costs. LS have a retiring workforce, and obsolete language needs such as COBOL, natural and mid-range systems [7]. The evolution of newer platforms has made businesses more innovative and profitable.

In summary, the complex nature of legacy has a high effort and high-cost association during the modernization process leading to migration risks. The legacy systems' characteristics should be evaluated before starting a modernization and migration process. They are business-

2473-3636/22/$31.00 ©2022 IEEE
DOI 10.1109/DCABES57229.2022.00078

critical systems and are still being utilized despite their pitfalls. However, to reduce the influence of high cost, effort and risk in those journeys, the root cause should be evaluated. Thus, by beginning the process by assessing legacy characteristics, contributors can be understood, and their impacts can be mitigated.

III. UNDERLYING CHARACTERISTIC IN LEGACY

Overview-Systems should limit development features that lead them to a complex ecosystem. Characteristics such as long-time scale, no- technical documentation and code inflexibility are indicators of the need for modernization. They prevent the system from evolving, thus becoming too expensive to maintain [8]. They prevent businesses from scaling, gaining profit, market value or becoming digital. However, despite many years of deployment, if a system runs within an organization's forecasted budget is not considered to be supported by legacy characteristics. Hindrances to modernization are:

Long-time scale – It represents the lineage from when a system was built until its deployment and utilization. Some legacy systems were developed in the 1960's and are still efficient for daily business needs. However, the lineage has accumulated entropy in those systems, making them very complex. During deployment, there has been a surplus of lines of codes, redundant codes and unexplained dependency inheritance. The internal complexity has become complex and costly to maintain and has a higher probability of defects [6].

Documentation - Systems need artefacts depicting their growth. However, LS are built without any technical documentation and are unsuccessful in updating the maintenance status of the system. This situation leads to the disruption of the system's nature as codes are added without any knowledge. Generally, for system review or for maintenance activity, technical documentation and test cases are checked by programmers. In the LS environment, not having a guide would give programmers the liberty to add to system functionalities and increase duplicity. Moreover, organizations have failed to practice quality in their organizational culture.

Inflexible-codes –This characteristic influences the code and the system infrastructure. The inflexible code results in an accumulation of irrelevant codes in the system. Many new functionalities are being added without considering other interfaces and dependencies. Conversely, the accumulation of lines of codes impacts the system size. This is also why a LS is referred to as cumbersome. The system size is directly related to the number of defects and the underlying complexity. The complexity is high when the system has many components and associated dependencies [2]. Outdated dependencies affect the system performance and require more time and effort. The relative complexity metric has been previously used to enhance quality in project management for the branching out of bugs [1]. Those mentioned characteristics are becoming a barrier to progress towards a newer generation of technologies and principles.

Summary – LS ecosystem creates confusion between co-dependency and dependency, eventually increasing the size of the system over the years. Therefore, inflexibility of codes and complexity results in inaccurate or poor tractability of the system lineage. Also, the purpose of the system may not be determined. Hence, there is a need to scrutinize those systems to extract their business logic [8] [7]. The logic can be transferred to a modern platform but without transferring the underlying characteristic. The complexity assessment of the legacy is essential for business survival before embarking on a migration process. By doing so, complexity and dependency would be identified before performing rule extraction, and risk can be controlled.

IV. LEGACY ASSESSMENT

The business survival demands modernization, functional expansion and the integration of digital transformation projects that LS could not cater for. It is because of its underlying bulky nature and cost. The current legacy migration is the transfer of the current system to a target system or desirable platform. Nevertheless, there are factors contributing to complexity and are perceived as hindrances to the modernization journey [5]. Therefore, evaluating the legacy system is inevitable to understand the impact of the current underlying characteristic of legacy systems. Presumably, assessing underlying characteristics would limit architectural disruption and reduce modernization cost/effort and risk. The primary stage of the modernization process should be scrutinizing the system to demonstrate which factors contribute to complexity. Those factors are as illustrated below:

A. Complexity factor

The inherited complexity and unmanageable control structures are perceived as complex characteristics affecting the system performance [9]. Moreover, graph-based metric measures complexity at the architectural level. Thus, communication, dependencies and functionalities could be comprehended. Consequently, dependencies between the component of the system could be improved. In such a model, unknown variables can be found, enabling us to reduce any potential risk or any potential maintenance cost [11],[15],[10]. The study of complexity is on the estimation of different dependencies and co-dependencies of the system size. However, the vital components from the entropic nature of the system should be known. Hence, the root of variables should be traced to prevent the impact on systems. A complexity assessment can reveal a complexity index since cyclomatic complexity measures the decision logic found in one software module [2]. The complexity involvement for legacy should be assessed to evaluate the degree of importance or the degree of dependency of modules. Moreover, the essence of such a metric is to assess and manage complexity and is also seen as a maintenance activity [1], [12], [9]. It shows to what extent the system is complex. We believe accurate estimation is imperative in modernization to limit risk. The McCabe complexity range assesses a program's linear independent path and categorizes its complexity level [2]. Complexity evaluation should be appraised against the complexity range [4]. The ideal system would be a system falling in the first range of low complexity and low cohesion [13], [2]. The lowest complexity range structure is ideal for modernization and migration due to its low risk and low maintenance cost from dependencies. The system's modularity facilitates the anticipation of

cascading risk and provides a better maintenance service. By decomposing the system, parameters or variables for the component become more visible [11]. The contributors of complexity have been associated with costing, effort and risk. Therefore, assessing complexity efficiently measures discrepancy along with the dependency accumulated in legacy.

B. Cost and Effort factor

Overview: Project failures result from inaccurate estimation during software production. There are many practices of project management that could be beneficial to limiting failures during software development, such as software cost estimation [21]. Metrics can be used to calculate the project cost from the estimated project size. Moreover, the size parameter in the complexity review can eventually be used for cost estimation along with the duration of the effort, as in COCOMO [22]. We shall focus on cost drivers leading to a high maintenance cost in legacy, followed by the effort factor.

Cost factor- The development life cycle of software is broken down into different parts of project management and envisages the project duration, time and cost of each phase in the development. Unlike system development costs, legacy costs imply maintenance and migration costs. The maintenance cost is derived from the number of lines of codes the system needs to support. In legacy, cost estimation uses the bottom-to-top approach as part of the reverse engineering process since they have existed and operations have been performed for some time [14]. Cost drivers are 1) Technical Debt is a metaphor for accumulating unresolved issues in a software project. They come from the legacy systems' characteristics and result in high maintenance costs and slowing development speed. To overcome technical debts, refactoring has been applied to control costs. The concept of refactoring has been used as a quality technique to manage costs from cascading defects [20]. 2) Six-sigma: The effectiveness of organization quality assurance activities rating (EQAR) has been applied to organizations to eliminate or minimize residual defects. The effectiveness can be measured using the defect density of the delivered product, also called the six-sigma value. The principle of six sigma underlines the density of defects over opportunities/lines of codes. The optimum rating for 6 sigma is 3 defects per 1 million opportunities [14].

Effort factor- The effort estimation illustrates 1) how big is the legacy system, 2) how long will it take to modernize the system 3) how many people will be required on the project and at what rate per hour. Effort evaluates the human involvement in the software development activities. Moreover, the number of lines of code of the system determines the size parameter and, eventually, the effort required. Human involvement is addressed as staff months and may also be used to address new defects in projects [22],[18]. The productivity ratio approach uses KSLOC per staff month to convert the estimated size into the effort. It was used for a retail supply system (RSS) to estimate the duration using the estimated effort from the approach. Additionally, if an increment is estimated to need 477 staff months of effort, the duration is about 22.5 months or about 21 staff members. The cost estimation model illustrates the estimated effort and the estimated duration of the project. On the hand, estimated effort considers 1) phase distribution of effort and 2) phase distribution of labour, whereas estimated duration is about phase distribution of duration [19]. Therefore, performing a cost assessment on supporting functions can denote the cost of support and its maintenance cost [11], [15].

Summary –A complex function with high cost and effort can be risky to migrate due to its core business functions. Therefore, before embarking on a modernization journey, the system needs to be evaluated to understand the factors contributing to its complexity. To do so, cost estimation can be used along with understanding the modernization project's effort and duration. Though, by evaluating different cost drivers, they can be used to control the maintenance cost. Therefore, a less complex system can be migrated to the target system by mitigating risk in modernization. Estimation techniques can also assess the importance of business logic in a complex function. Therefore, evaluating cost and effort along with the dependency of components can provide a better direction to which components should be migrated to the target platform. By doing so, potential risk can also be visible and mitigated to reduce failures [4]. In the risk factor section, risk approaches have been discussed.

C. Risk Factor

In legacy systems, risks are inherent in the software project. All development projects of this scale carry a certain amount of risk. Despite the best practices of software development to mitigate them, defect still exist in modernization [6]. Therefore, to tackle risks, we should investigate the root cause of factors. Due to the numerous unmanageable lines of code, managing risk and controlling factors such as cost and effort for modernization is difficult. Assessing these systems is essential to trace the unknown sources of variables. The modernization journey needs effective risk management and mitigating modernization plan rather than just a migration approach for transition. Moreover, when a system's complexity and risk are high, it is perceived as the right candidate for modernization. Hence, assessing the current system and a modernization plan is necessary to mitigate risk. Additionally, the risk approach should be part of the modernization process to mitigate failure during a legacy transition; for example, the risk-managed modernization approach and Case base reasoning.

Risk managed modernization (RMM) approach is a step-by-step approach for managing risk as part of the modernization process [19]. The key steps are 1) Portfolio analysis: It is an assessment of the most appropriate candidate for modernization by measuring the technical quality of the system against the business value of the system. 2) Reverse engineering: aims to understand the purpose of the legacy system. Doing so reduces the modernization effort and mitigates the risk of potential failures. The system is scrutinized in terms of its architecture, technology and components. 3) Modernization strategy: There are different modernization strategies along with the individual migration condition of the system. Regardless of the modernization strategy, the modernization effort must cater for changing requirements, technology change or compatibility of technologies. Strategies are re-development, incremental migration,

partial migration and wrapping. 5) Estimation of resources: The final stage of RMM is to perform estimation for executing the migration strategy.

CBR uses the modelling approach and mitigating risks of failures of similar software projects system. The model uses past case scenarios' problems and mitigates them into the current case. Features and attributes are predicative characteristics in the for-caste model. Those characteristics have been described as drivers in understanding the complexity of the files generated, used, maintained or the number of reports generated [17]. CBR can be merged with any software cost estimation technique to reduce failures from factors. For example, CBR can be combined with functional point analysis to reduce complexity in terms of the size of the system, effort, duration and the cost of development. The four stages of the R4 model; Retrieval: Retrieves similar cases to target problem, Reusability: Reuses past solutions, Revise: To adapt the previous solution to new cases. Retention: To apply the solution to the target and retain the solution in the case based.

V. CONCLUSION AND FUTURE WORK

Despite the huge maintenance cost and underlying issues from known features, Legacy systems have been functional for day-to-day operations. However, due to the evolution of modern paradigms and digitalization, the perspective of businesses was changed. The risk attached to business rules made business owners reluctant to adopt a modernization journey. Our contribution to this paper is to assess the underlying characteristic of LS and understand the influence of factors on complexity. Thus, factors can be controlled and failures can be limited during modernization. Estimations are believed to be a significant part in project planning at the earlier stage of development, although the probability of inaccuracy could be higher. Business logic, interfaces and dependencies support a legacy system, thus, providing readily available data for evaluation. The efforts and costs of those modules could be measured. The complexity can be avoided by knowing and evaluating the inter-dependency between modules. Therefore, we have used a complexity assessment, cost & effort assessment, and risk management modernization plan as the critical techniques in our work. By doing so, the risk of failure can be mitigated, modernization costs can be reduced, and systems can be modernized with managed complexity. While investing in existing methodologies, there was a need to address a change management approach to reduce the impact of the change from the complexity assessment. Furthermore, the predicative model for modernization and novel risk-modernization approaches for software engineering can be crafted.

REFERENCES

[1] Munson, J.C. and Khoshgoftaar, T.M., 1992. Measuring dynamic program complexity. IEEE software, 9(6), pp.48-55.

[2] Watson, A.H., Wallace, D.R. and McCabe, T.J., 1996. Structured testing: A testing methodology using the cyclomatic complexity metric (Vol. 500, No. 235). US Department of Commerce, Technology Administration, National Institute of Standards and Technology.

[3] ZDNet (2014), “Here’s what your tech budget is being spent on”, Available on:https://www.zdnet.com/article/heres-what-your-tech-budget-is-being-spent-on/ (Accessed: 03 August 2018).

[4] Méndez, M. and Tinetti, F.G., 2014. Integrating Software Metrics for Fortran Legacy into an IDE. In XX Congreso Argentino de Ciencias de la Computación (Buenos Aires, 2014).

[5] BLUEphoenix solutions (2013) Legacy Forensics. Available at: https://drive.google.com/file/d/1IJ0GQANn_CAUi8FeFGKr5Ijwy5hUY-fb/view?usp=sharing (Accessed: 07 July 2020).

[6] C. Birchall, Re-Engineering Legacy Software. Manning Publication Co; New York, pp 10-50, 2016.

[7] R.Khadka, A.Saeidi, A.Idu, J.Hage,“Legacy to SOA Evolution: A Systematic Literature Review”, Available at:http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.736.1357&rep=rep1&type=pdf .(Accessed: 10 December 2018)

[8] M. L. Brodie and M. Stonebraker, Legacy Information Systems Migration: Gateways, Interfaces, and the Incremental Approach. San Francisco,CA: Morgan Kaufmann Publishers Inc, 1995.

[9] Shao, J. and Wang, Y., 2003. A new measure of software complexity based on cognitive weights. Canadian Journal of Electrical and Computer Engineering, 28(2), pp.69-74.

[10] Wu, L., Sahraoui, H. and Valtchev, P., 2005, June. Coping with legacy system migration complexity. In 10th IEEE International Conference on Engineering of Complex Computer Systems (ICECCS'05) (pp. 600-609). IEEE.

[11] Cherkaskyy, M. and Sadek, A.S., 2004, February. The levels of program complexity. In Proceedings of the International Conference Modern Problems of Radio Engineering, Telecommunications and Computer Science, 2004. (pp. 396-397). IEEE.

[12] Dantsin, E., Eiter, T., Gottlob, G. and Voronkov, A., 2001. Complexity and expressive power of logic programming. ACM Computing Surveys (CSUR), 33(3), pp.374-425.

[13] Kim, K., Shin, Y. and Wu, C., 1995, December. Complexity measures for object-oriented program based on the entropy. In Proceedings 1995 Asia Pacific Software Engineering Conference (pp. 127-136). IEEE.

[14] Chemuturi, M., 2010. Mastering software quality assurance: best practices, tools and techniques for software developers. J. Ross Publishing.

[15] A.Yadav, A.Rajavat (2014), “An efficient process to measure the cost of re-engineering project”, Available at : http://www.digitalxplore.org/up_proc/pdf/103-141145157747-52.pdf (Accessed: 25 September 2018).

[16] Gartner (2015) “Digital demand in banking industry and impact on legacy system”. Available at: https://www.gartner.com/en/documents/3348017(Accessed: 01 March 2016).

[17] Mahar, K. and El-Deraa, A., 2006. Software project estimation model using function point analysis with cbr support. In Proceedings of IADIS Conference on Applied Computing.

[18] Azzeh, M., 2013, March. Software cost estimation based on use case points for global software development. In 2013 5th International Conference on Computer Science and Information Technology (pp. 214-218). IEEE.

[19] Seacord, R.C., Plakosh, D. and Lewis, G.A., 2003. Modernizing legacy systems: software technologies, engineering processes, and business practices. Addison-Wesley Professional.

[20] Jones, C., 2012. Software quality metrics: three harmful metrics and two helpful metrics. Namcook Analytic LLC.

[21] Galorath, D., 2006. The 10 step software estimation process for successful software planning, measurement and control.

[22] Thein, K.T. and Khin, T.T., Software Cost Estimation Using Constructive Cost Model (COCOMO II) (Doctoral dissertation, MERAL Portal).

[23] altexsoft software r&d engineering (2020) Whitepaper: Legacy system modernization: How to transform the enterprise for Digital future. Available at: https://drive.google.com/file/d/1hUJcaiPsWSw3JUbBsR800U1DwHKV-qfQ/view?usp=sharing (Accessed: 26 May 2020).