ISSN 2278-3091

Volume 9, No.1, January – February 2020

International Journal of Advanced Trends in Computer Science and Engineering

Available Online at http://www.warse.org/IJATCSE/static/pdf/file/ijatcse21912020.pdf

https://doi.org/10.30534/ijatcse/2020/21912020

Impact Analysis of Legacy System Migration to the Cloud Environment: A Focused Study

H. SeetharamaTantry[1], Murulidhar N.N[2]., K. Chandrasekaran[3]

[1] Department of Mathematics & Computational Sciences, National Institute of Technology Karnataka, Surathkal, India, e-mail: hstantry@gmail.com

[2] Department of Mathematics & Computational Sciences, National Institute of Technology Karnataka, Surathkal, India, e-mail: murulidharnn@gmail.com

[3] Department of Computer Science and Engineering, National Institute of Technology Karnataka, Surathkal, India, e-mail: kch@nitk.ac.in

ABSTRACT

Although “Legacy System” frameworks are frequently worked over numerous years by a blend of IT and business specialists. They stay inflexible inside the authoritative setting and business speculation made by practical applications. Semantic Design provides highly automated tools and services to migrate legacy system to a new platform. The proposed migration of the legacy software life cycle model is used for legacy migration. Legacy modernization encourages the organization to rejoin to the changing interest in the business world. Manufactured goods conveyed through the cloud give the organization both new chances to expand their business and to manage expenses. The researchers concurred that the different legacy programming languages are discussed as in the result section. Both interoperability considerations and incorporation, expand upon measures for SaaS, PaaS, and IaaS in IT Business in the cloud market, has increased.

Key words: Legacy System, Cloud Adopt, Cloud Computing, Migration, Software Modernization.

1. INTRODUCTION

The word “Legacy System” is intended and executed for an effective system and mounted by the current trends and Telecommunications Technology strategies. Even though these frameworks are frequently worked over numerous years by a blend of IT and business specialists, they stay inactivity inside the authoritative setting and business speculation made by practical applications. A few elective meanings of what precisely a legacy system is presented here. (Bennett *et al.* 1995) Alluded to the legacy system as *"vast software frameworks that we do not have of the foggiest idea how to adapt. However, that is fundamental to the organization."* This is fundamentally unique about the first, yet it contains significant usefulness and information from the legacy system. The expected maintenance activity can never meet the required level in due course of time. Legacy Software Modernization is the way towards advancing existing software frameworks by substituting, re-creating, and recycling the software modules and platforms when expected maintenance follow-ups can never again accomplish the ideal framework properties. The essential point of software modernization [1] is to decrease support cost and increase adaptability. The majority of these frameworks were created years back and have kept on advancing. New necessities have repeated changes on these legacy system bringing about unstructured source code that is hard to keep up. These issues have been perceived by the software designing network and sufficient legacy system modernization methodologies.

Today, an ever-increasing number of organizations prefer to reposition their information, applications, and foundation to Cloud Computing. Cloud Computing is an on-request provision model that enables clients to exploit a few advantages, like, adaptability and decreased cost. Cloud Migration is the way towards moving all or part of an organization’s data and software framework from an existing domain to the cloud state. To date, there has not been a Systematic Literature Review (SLR) of research on cloud migration, making it difficult to evaluate the growth in general and identifying trends, research gaps, and future dimensions of cloud migration in particular. A SLR identifies, classifies, and creates a comparative overview of state-of-the-research and enables knowledge transfer in the research community.

The setting of the fundamental goal in this present work is to examine some underlying issues which to some extent the existence cycle of legacy software framework. This study expects to the accompanying commitments by leading methodological research of existing exploration:

- Dynamic motivations in legacy software framework modernization and migration on to the cloud.
- A novel migration of Legacy Software Life Cycle
- Existing approaches and methods to empower the relocation of the legacy software framework to the cloud.
- Research benchmark study between Legacy and Cloud Migration.

This paper presents the establishment of migrating legacy software frameworks in the higher setting of software maintenance and reclaim. Also it shows an outline of methodologies for moving to Cloud Computing platform and related service-oriented methodologies.

The rest of this paper is organized as follows. Section 2 throws some light on need of legacy system modernization. Section 3 gives overall picture of legacy system migration challenges, methods and the actions to be taken to protect the legacy systems. Section 4 discusses the need of migration of legacy system to cloud. Section 5 proposes a model for legacy system migraton. Section 6 reveals results of study carried out and discusses various reports. Finally the paper concludes with the various studies carried out during the process.

2.BACKGROUND

A legacy system is a business specific software framework that fundamentally faces modification, and their absence can seriously affect the business [2]. The essential point of software modernization is to moderate support cost and increment adaptability. software modernization is the way toward advancing existing software frameworks by replacing, re-creating, reusing, or relocating the software segments and phases when routine maintenance practices can never again accomplish the ideal framework properties. Following many years of legacy system modernization research, it might come as an unexpected fact that numerous legacy system are still in everyday's task. Novel requirements have provoked frequent changes of these legacy system bringing about unstructured source code that is hard to keep up. Besides, information about the legacy system is not available fully as unique developers leave the organization or resign, and documentation is generally lost [3]. This exploration, linked with the modernization procedure has performed to reuse and incorporate legacy valid codes.

Organization's frameworks are required to be increasingly agile, flexible, high combination, and versatile to remain to adapt to the present business need. Organizations are forced to give their clients the objects and administrations as fast as could be expected under the circumstances. What they need is the frameworks with adaptability, methods, and tools that give better considerations to code and new models which encourage interoperability. The perfect answer to overcome issues brought about by legacy system is legacy modernization. Legacy modernization encourages the organization to rejoin to the changing interest in the business world. [4] characterizes legacy modernization as an endeavor to progress a legacy system, when routine practices, for example, maintenance and improvement, can never again accomplish preferred framework properties.

Several studies explored in the academic world have been conversing about legacy systems and how would they make issues for the organizations in reality. Issues, complications, and many negative impressions about legacy system are controlling the journals, research articles, and some other reports in the educational domain. The study by Gartner Group detailed that 88% of the world's business kept running on COBOL with more than 200 billion lines of code at present [5]. Likewise, support that report and expresses that there assessed 6 billion lines of new COBOL code every year. Moreover, the legacy system TIOBE file likewise reports that COBOL as the most standard code as ever used [6]. In recent research, researchers differentiate IT implementation and relocation since movement occurs from a binding IT to a temporary IT, while implementation does not expect a current IT [7].

3. THE LEGACY SOFTWARE SYSTEM

Software frameworks are never considered as tomorrow's legacy system when they are fabricated, and their requirement for change isn't considered. [8] communicates a similar conclusion about software preservation. As indicated by this researcher, the principle issue of support is that software frameworks are not intended to sustain, and if circumstance does not change, their dependability happen issues with maintenance of the frameworks in future.

3.1 The Key challenges of Legacy Software System

Software change is a particular portion of software improvement. In this method, maintenance and advancement are considered during the beginning of improvement of the software framework. While selecting technologies, total lifetime costs, not just starting improvement costs, surveyed, and probable maintenance issues with this specific innovation, should be considered [9]. Also, the software framework should be sufficiently adaptable for development, as new client necessities could increase.

As indicated by [10], starting advancement, yet additionally, development should be controlled. The Managed Evolution approach, in which all upgrades to the framework is isolated into two classifications: one expands business esteem, and another improves the agility and spreads enhancements in such regions as architecture, technologies, and codebase.

3.2. Legacy Software Modernization Approaches

Framework advancement covers an expansive scope of improvement training from adding a line of code to re-execute the framework. In [17], framework development training partitioned as follows:

- **Maintenance** is a steady and iterative procedure in which little changes made to a framework. These progressions are regularly bug fixes or little practical improvements that does not include significant auxiliary changes.
- **Modernization** includes more broad changes than support; however, it preserves a critical bit of the current framework. These progressions may incorporate rebuilding the framework.

- **Replacement** requires remaking the framework starting with no external support. Frameworks replaced by utilizing the “big-bang” approach.

In such a manner, Lehman's first law states that software must be consistently maintained, or it will turn out to be logically less acceptable. Modernizations, for the most part, suggest massive scale changes that help to expand the software's lifetime. The modernization methodologies ordered into two distinct classes: “Black-Box” Modernization and “White-Box” Modernization [21].

3.3 The Modernization Process of Legacy Software System

Semantic Designs provides highly automated tools and services to help migrate legacy system to new languages such as Object-Oriented Programming languages (Java and C#). Legacy systems are effective and progress along these lines, and likely to be in field for an extensive span of time. An outcome is that legacy software by utilizing equipments accessible at the time it developed, rather than the most current software developments. More experienced technologies are increasingly hard to keep up, and this is a crucial purpose of legacy framework administrators. The primary alternate is called redevelopment [11]. Redevelopment is one method for modernization where the target framework built from scratch. The legacy system is remade and supplant with the new target framework. Towards the finish of the execution, the new framework has a new design, so it is simpler to maintain, expand, and build up the framework after modernization.

Service-Oriented Architecture (SOA) [12] is a compositional world view that communicates to an open, extensible, and compostable software design assembled from reusable software parts known as services. SOA concentrates around the reusability of the modules by isolating the interface from the interior execution. The fundamental rules that advance SOA incorporate free coupling, a reflection of necessary foundation, agility, adaptability, reusability, dependence, statelessness, and discoverability. From an software modernization point of view, SOA guarantees to reuse the prior legacy resources by representing them as included esteem organizations. While recognizing the issues of legacy modernization [13], the followed measures are complexity and changeability.

3.4 Necessary Changes in Legacy Systems

Software needs to empower business tasks in a domain portrayed by globalization, expanded challenge, and portability. This expects initiatives to adjust to changes in the business atmosphere. For inviting new clients and supporting a fast development, achievements, and the data frameworks utilized by undertakings should have the capacity to adjust to new principles, changes in working conditions, and upgrade of business procedures. Software necessities are as yet subject to Lehman laws, including [14]:

- “Continuous Change” of necessities, locations, and business rules;
- "Expanding Complexity Nature" of software frameworks;
- "Authoritative Security," decided in administration;
- "Proceeding with Development" of the back-end framework; and
- "Declining Quality" without proactive measures.

3.5 Legacy Software Systems Protection

The advancement to totally new frameworks, adjusted to the existing trends, would include complex improvement cycles, staggering expenses, in addition to the need to keep up the old framework amid this undertaking. These states to an exertion that can't be sustained by the individuals who have just put resources into their data frameworks and have qualified personnel to work with them. The need to protect legacy system contains various perspectives that are regular to the benefits of reuse: exploiting software that has been widely tried, in actuality, decreasing risk, preserving valid area information, and accelerating the procedure for achieving current business targets. It frequently symbolizes to a low-scale reuse, performed for making a new version for reusing at a bigger scale.

3.6 Legacy Software Migration

Migration is regularly low for installed software frameworks due to the huge expenses of retrofitting, for example, the smart home appliances with new software. The software maintenance modules as per the arranged model of software life cycle [15], after the underlying improvement, maintenance is made out of three stages: (1) Evolution: When adjustment and improvement managed without rebuilding or significant changes; (2) Servicing: When patches and wrappers presented with the unavoidable impact of harming the building reliability; (3) Phase-out: When it is just likely to work around for protection of the application used.

4. LEGACY SOFTWARE MIGRATION TO CLOUD

Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable resources that can be rapidly provisioned and released with minimal management effort or service provider’s interaction defined by US National Institute of Standards and Technology (NIST) [18].

Software migration has been the enthusiasm of the software engineers and somewhat the business side of the organization. The migration to the cloud puts more attention on the business side of the organization and includes a more significant number of stakeholders than the designers. Manufactured goods conveyed through the cloud give the organization both new chances to expand their business and to maintain costs.

There are variety of reasons that a migration of a legacy system may be needed:

- The legacy system languages and maintenance tools [24] which are expected to help the legacy system are progressively troublesome or costly to acquire.
- Persons who knows the legacy system languages and maintenance work are getting to be hard to discover and hold. Moreover, younger generation is hesitant to learn legacy system that it doesn't seem to propel their long-term profession [22][23].
- Numerous legacy system keep running on legacy system hardware frameworks. Such equipment frameworks are winding up increasingly, costly to maintain, and workforce that realizes these frameworks are likewise progressively hard to discover.
- The software engineering of legacy system languages frequently does not fit its structure platforms to other IT frameworks that have grown up around it.

5. A PROPOSED MODEL FOR MIGRATION OF LEGACY SOFTWARE LIFE CYCLE

The proposed Migration of legacy Software Life Cycle Model worked whereby addressed by circles contrast with learning states while changes (addressed by square shapes) identify with strategies [16]. The legacy software life cycle model depicted in figure 1.

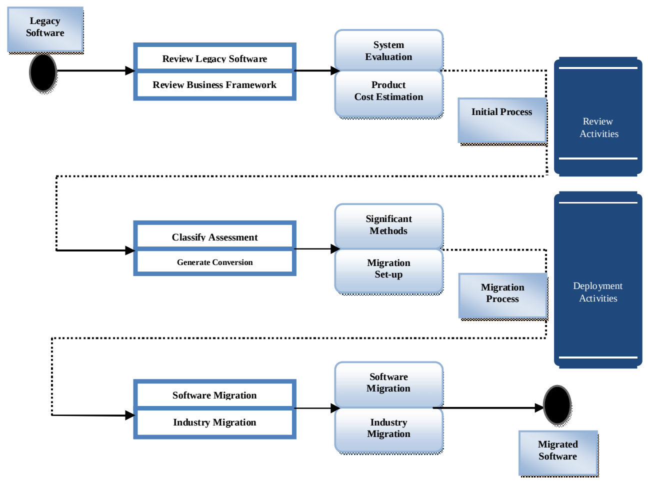

Figure 1: Migration of Legacy Software Life Cycle Model

It is essential to keep learning about the current scheme objectives and how they are accomplished through the present undertaking ways just as secure information about the usefulness of the legacy system to the degree essential for the relocation. The model continues to characterize an organized method for distinguishing and assessing the different advances for the legacy system movement. The plan and advancement of the objective framework can extend from the easy to the complex. It includes the actual relocation process for both legacy system and business forms.

6. RESULT AND DISCUSSIONS

The finding pursued a Grounded Theory (GT) [19][20] way to deal with examining 50 in and out semi-organized reviews, for example, a subjective procedure, to distinguish how legacy system frameworks and their modernization agreed in the industry. In an observational investigation, for example, this investigation, the utilization of different research procedures (subjective and quantitative systems in this situation) expands the certainty that the outcomes are dependable. GT is an explorative research strategy that goes for finding new points of view and bits of knowledge, as opposed to affirming existing ones. To begin with a progression of surveys directed with 50 researchers (distinguished as R1-R50 in this research, refer table 1). The witnesses were chosen on the two criteria that they have involvement with legacy system, and with legacy system modernization projects.

A strategy that utilizes more than one information source, or gathers similar information at various events is usually used to build (decline) trust in finding by giving affirming (opposing) proof and improves the legitimacy of the discoveries of an observational investigation. Designers framed the biggest gathering of supporters (22%), trailed by IT Managers (14%) and Scientists (12%); they originate from different areas, for example, software advancement organizations (28%), counseling organizations (21%), specialist organizations (11%), and money related establishments (9%).

Table 1: The utilization of different research procedures

Researchers	Service Domain
R2,R11,R12,R20,R21	Info Knowledge Services
R1,R15,R17,R22	Economic Service Provider
R4,R5,R25,R26,R39	Management Organization
R7,R8,R18,R19,R40	Software Change Business
R6,R10,R24,R27,R50	Consultancy Firm
R3,R28,R34,R41,R42,R43	Aeronautics Manufacturing
R9,R32,R33,R44	Security Enterprise
R16,R30,R31,R45	Manufacturing Enterprise
R13,R35,R36,R46	Poultry
R23,R37,R47,R48	Nutrition
R14,R29,R38,R49	Apparatus

6.1 Observed Features of Legacy Systems

This review displays the consequences of the investigation of the data of legacy systems, also, an overview is signified to the features: a) Industry Significant b) Technology c) Reliability and d) Performance showed in figure 2.

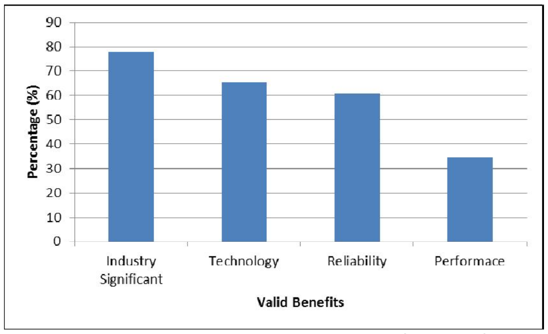

Figure 2: Research review responses for benefits

6.2 Legacy Programming Languages

The more significant part of the sources concurred that legacy systems are *"main"* frameworks. Furthermore, the observers called attention to that legacy system are *"center"* frameworks that have been demonstrated to work accurately in a construction atmosphere for a considerable length of time (refer table 2).

Table 2: Demonstrated work by the researchers

Researchers	Solution
R1,R3,R6,R9,R13,R14,R18,R19,,R22,R23,R24,R36R33,R34,R35,R46,R49	The greater part of the legacy system is more seasoned than 3 to 4 decades.
R2,R5,R8,R11,R16,R17,R25,R26,R27,R32,R37,R38,R39,R42,R43,R47,R50	Legacy system is an old framework; a great deal of legacy system is the center framework
R4,R7,R10,R12,R15,R20,R21,R28,R29,R30,R31,R40,R41,R44,R45,R48	A legacy system is a framework that do not have a place with in vital innovation objectives.

In the following examination, the researchers were inquired as to whether programming language is a deciding element for a framework being legacy system and obtained varied results. Such a different conclusion additionally perceived from the study of the surveys from legacy system (COBOL, PASCAL, FORTRAN, BASIC, ASSEMBLER, and VB). The respondents concurred that the languages like COBOL is 38%, PASCAL is 12%, FORTRAN is 10%, BASIC is 11%, ASSEMBLER is 8%, and VB is 21% do decide whether a framework is legacy system, which is depicted in figure 3.

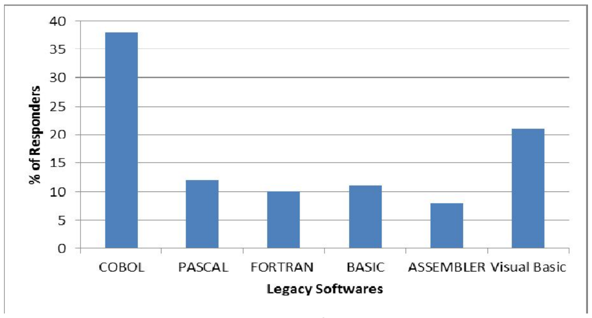

Figure 3: Review analysis of Legacy Programming

6.3 The Legacy Programming Language Review

In summary, it gives complete understanding and analysis of legacy system proposed by various researchers (refer table 3).

Table 3: Review of legacy programming language

Proposed Techniques	Tools / Methods Approached	Researchers
Reverse Engineering and Software Architecture	UML Mapping	R1,R3,R6,R9,R13,R14,R18,R19,R22
Source Code Assessments	API	R2,R5,R8,R11,R16,R17,R25,R26,R27
Development Modeling	USE Case Graph	R4,R7,R10,R12,R15,R20,R21,R28,R29
Documentation	SMART Tool	R23,R24,R36R33,R34,R35,R46,R49
Functional Requests	Tool: Reverse Engineering	R32,R37,R38,R39,R42,R43,R47,R50
Test Circumstances	Tool: Code Reporting	R30,R31,R40,R41,R44,R45,R48

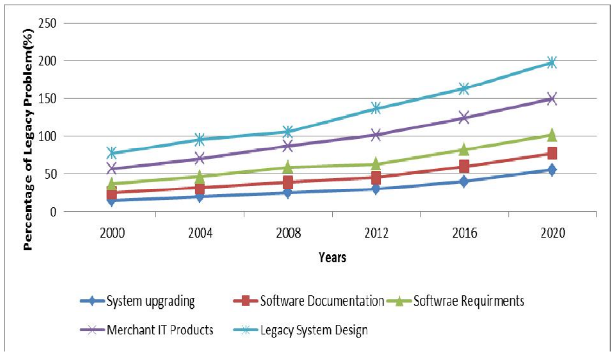

Figure 4: Problems in Legacy Systems

Figure 4 portrays the problems of legacy system over the range of 3 decades. It perceived that the documentation, framework plan necessities represents the difficulties for the support of legacy system. In Figure 4, it additionally refers to System Upgrading, Software Documentation, Software Requirements, Merchant IT Products and legacy system Design.

Figure 5 outlines the examination of different programming languages used to create business applications concerning software measures, for example, Code Reusability, Methods, Tools and Modularity. It is mentioned that the common legacy system programming languages, for example, COBOL, PASCAL, FORTRAN, BASIC, ASSEMBLER, and VB, give increasingly complex techniques contrasted with the other present-day object oriented programming languages, like, JAVA, C#, .NET, and PHP. With regard to the reusability metrics, legacy system code in JAVA, C#, .NET, and PHP have a more reusable rate than COBOL, PASCAL. Further, it is represented in Figure 5 that instruments, particularly measures in JAVA, C#, and .NET contribute preferred remarks over COBOL, PASCAL, FORTRAN, BASIC, ASSEMBLER, and VB.

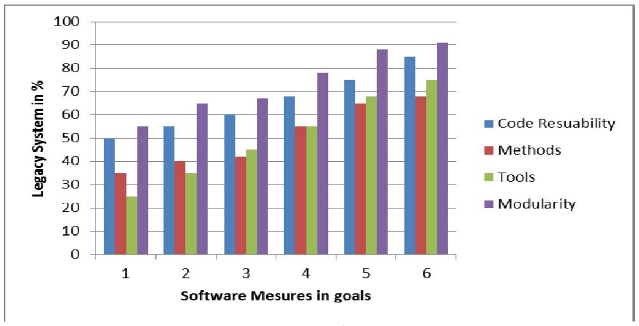

Figure 5: Legacy Software measures

6.4. Security Analysis in Migration to Cloud Environment

To sum up the results of the systematic review, we present in Table 4, a summary of the number of studies by the initiative. The initiatives obtained from the main topics found on the approaches analyzed of the review carried out about migration processes to the Cloud. The advantages are analyzed by the frameworks if these approaches are focused on software structure, ethics, renovations, security analysis and tools, and reengineering propose transformations of models in the migration process if security is considered in these approaches. This migration study among comprehensive technology and cloud providers, is to provide interesting aspects to take into account the migration of legacy systems to Cloud computing.

Table 4: Summary of the number of studies by initiative

Researchers	Boundaries
R1,R3,R6,R9R23,R24,R36R33, R34,R35,R46,R49	Software Structure
R2,R5,R8,R11,R32,R37,R38, R39,R42,R43,R47,R50	Software Ethics
R4,R7,R10,R12,R15,R20, R41,R44,R45,R48	S/W Renovations
R13,R14,R18,R19,R22	Security Analysis
R16,R17,R25,R26,R27,	Case Study
R21,R28,R29,R30,R31,R40	Tools & Reengineering

6.4.1 Advantage of Cloud Computing

In another examination about Cloud Computing, most of the researchers anticipate six fundamental drivers of Cloud computing: Flexibility, Cost estimation, Scalability, Business promotion, Security, Latest Technology in the IT industry. Cloud Computing benefits by the faster sending off applications for lesser expense. In this equivalent examination, overpowering the mainstream of members, considering security issues, to be their principal concern considering the utilization of Cloud computing (refer fig. 6).

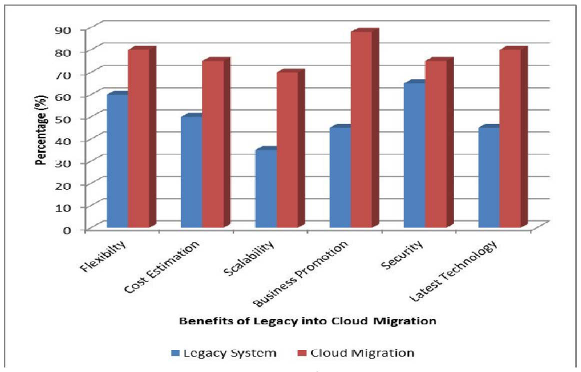

Figure 6: Advantages of Cloud computing

6.4.2. Threats Assessment of Cloud Computing

Likewise, security, legal, privacy, integration, loss of performance, lack of functions viewed as regions of threats. The review results indicated in figure 7. It creates the impression that they are not stressed principally over the absence of safety efforts in themselves, yet about the absence of straightforwardness in favor of sellers. A littler, cost-controlled association may find that a cloud organization enables them to exploit tremendous scale framework safety efforts that they couldn't generally manage.

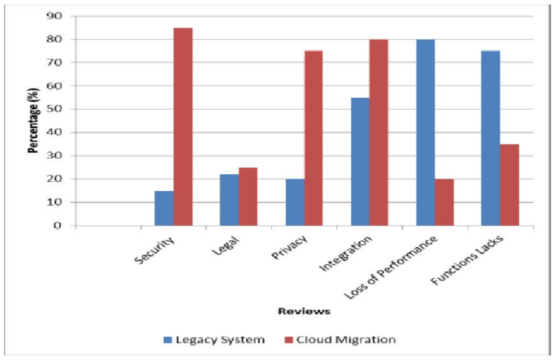

Figure 7: Threats Assessment of Cloud Computing

6.5 IT Business in Cloud Market - Predictions for 2021

IaaS hints at no backing off soon and is relied upon to reach $83.5 billion by 2021. SaaS industry is relied upon to develop by 22.2% to reach $83.6 billion before the end of 2019. As indicated by the equivalent Gartner report, SaaS

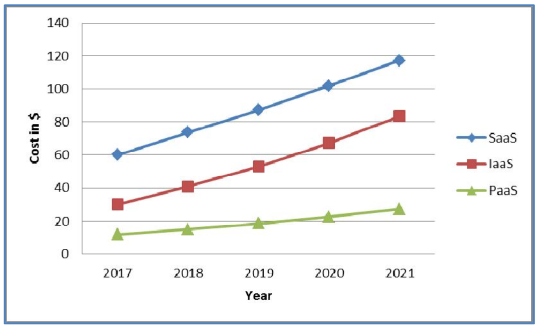

Figure 8: Global Business Income for Cloud Technology

is anticipated to develop to an astonishing $117.1 billion by 2021. Though a portion littler than SaaS, IaaS and PaaS are required to develop at an impressive 26% to reach $15 billion before the end of 2019. By 2021, Gartner expects PaaS to pick up a total market size of $27.3 billion. Industry master predicts that the IaaS market created by 35.9% in 2019 (only a slight decreasing from 38.6% in 2018) to reach $40.8 billion preceding the year ended. The results analyzed in figure 8.

7. CONCLUSION

Legacy systems are progressively turning into a problem for a wide range of IT business. Though explicit difficulties stay, some of them identified with the administrative establishment and others explicit to the proposed methodology of cloud computing. The researchers concurred that the languages like COBOL is 38%, PASCAL is 12%, FORTRAN is 10%, BASIC is 11%, ASSEMBLER is 8%, and VB is 21% do decide whether a framework is legacy system. The threat assessment of cloud computing is also discussed. The investigation of the business method for reasoning on-premise IT administrations to the cloud is brought up by this cloud adoption by Implications of Integration and Interoperability for Cloud applications. In Cloud Interoperability, both Considerations and Incorporation expand upon measures for SaaS, PaaS, and IaaS in IT Business in Cloud Market 2019, and Predictions for 2021 increased.

REFERENCES

1. Ulrich, W., 1994, **From Legacy Systems to Strategic Architectures,** Software Engineering Strategies, 2(1), 18-30.
2. S.C. Misra, A. Mondal, **Identification of a company's suitability for the adoption of cloud computing and modeling its corresponding return on investment**, Math. Comput. Modell. 53 (2011) 504–521. https://doi.org/10.1016/j.mcm.2010.03.037
3. Ulrich, W. (2004, September). **A status on OMG architecture-driven modernization task force. In Proceedings EDOC Workshop on Model-Driven Evolution of Legacy Systems (MELS),** IEEE Computer Society Digital Library.
4. Khadka, R., Saeidi, A., Idu, A., Hage, J., Jansen, S., (2013), **Legacy to SOA evolution: a systematic literature review. In Migrating Legacy Applications: Challenges in Service Oriented Architecture and Cloud Computing Environments**. A. D. Ionita, M. Litoiu, G. Lewis Editions. IGI Global.
5. Kizior, R. J., Carr, D., & Halpern, P. (2005). **Does COBOL Have a Future?**. Proceedings of the Information Systems Education Conference 2000, P. 126.
6. A. Khajeh-Hosseini, D. Greenwood, J. W. Smith, I. Sommerville, **The cloud adoption toolkit: supporting cloud adoption decisions in the enterprise**, Softw. 42 (2012) 447–465. https://doi.org/10.1002/spe.1072
7. Warren, I & Ransom, J 2002, **Renaissance: a method to support software system evolution**, in Computer Software and Applications Conference, 2002. COMPSAC 2002. Proceedings. 26th Annual International, IEEE, pp. 415-420
8. Stehle, E., Piles, B., Max-Sohmer, J., & Lynch, K. (2008). **Migration of Legacy Software to Service Oriented Architecture**. Department of Computer Science Drexel University Philadelphia, PA, 19104, 2-5.
9. Vu, Q. H.; Asal, R. **Legacy application migration to the cloud: Practicability and methodology**. In: World Congress on Services, 8., 2012, Honolulu. Electronic proceedings. Honolulu: IEEE, 2012. p. 270-277. https://doi.org/10.1109/SERVICES.2012.47
10. P.-J. Maenhaut, H. Moens, V. Ongenae, and F. De Turck, **Migrating Legacy Software to the Cloud: Approach and Verification by means of Two Medical Software Use Cases**. Software: Practice and Experience, 2015. https://doi.org/10.1002/spe.2320
11. Frey, S.; Hasselbring, W. **The clouding approach: Model-based migration of software systems to cloud-optimized applications.** International Journal on Advances in Software, v. 4, n. 3 and 4, p. 342–353, 2011.
12. Lehman, M. M. (1996). **Laws of software evolution revisited**. In C. Montangero (Ed.), Proceedings of the 5th European Workshop on Software Process Technology (EWSPT 1996), (pp. 108-124). London, UK: Springer-Verlag. https://doi.org/10.1007/BFb0017737
13. Sommerville, J. (2006). **Software engineering** (8th ed.). Reading, MA: Addison-Wesley.
14. Bennett, K. H., & Rajlich, V. T. (2000). **Software maintenance and evolution: A roadmap**. In A. Finkelstein (Ed.), Proceedings of the Conference on the Future of Software Engineering, (pp. 3-22). ACM Press. https://doi.org/10.1145/336512.336534
15. L. Peterson, Petri Nets, **Computing Surveys**, Vol 9, No. 3, September 1977 https://doi.org/10.1145/356698.356702
16. Weiderman, N., J. Bergey, D. Smith, Tilley, Scott R., **Approaches to Legacy System Evolution** (CMU/SEI-97-TR-014), Pittsburgh, Pa.: Software Engineering Institute, Carnegie Mellon University, 1997. https://doi.org/10.21236/ADA336213
17. Sneed, H., **Encapsulating Legacy Software for Reuse in Client/Server Systems**, Proc. of WCRE-96, IEEE Press, Monterey, 1996.
18. Chikofsky, E., J. Cross II, **Reverse engineering and design recovery: A taxonomy**, Software Reengineering, IEEE Computer Society Press, 1992, p.54–58.
19. Demeyer, S., S. Ducasse, O. Nierstrasz, **Object-Oriented Reengineering Patterns**, Square Bracket Associates, Switzerland, 2009.

20. V. Andrikopoulos, T. Binz, F. Leymann, and S. Strauch. **How to adapt applications for the cloud environment. Computing**, 95(6):493-535, 2013. https://doi.org/10.1007/s00607-012-0248-2
21. Microsoft Azure. **http://azure.microsoft.com/**. [Visited: May 2014].
22. Google App Engine**. https://cloud.google.com/ products/app-engine/**. [Visited: June 2014].
23. Jeffery, K., & Neidecker-Lutz, B. (2010). **The future of cloud computing: Opportunities for European cloud computing beyond 2010.** Geneva, Switzerland: European Commission, Information Society, and Media.
24. Petcu, D., Craciun, C., Neagul, M., Rak, M., & Lazcanotegui Larrarte, I. (2011). **Building an Interoperability API for sky computing.** In Proceedings of the International Conference on High-Performance Computing and Simulation (HPCS), (pp. 405-411). HPCS. https://doi.org/10.1109/HPCSim.2011.5999853