ELEMENTS OF LEGACY PROGRAM COMPLEXITY

Harmeet Kaur[1], Shahanawaj Ahamad[2], Gurvinder N. Verma[3]

[1]Ph.D. (Computer Applications) Research Scholar, Punjab Technical University, Jalandhar, Punjab, India
[2]Assistant Professor, Dept. of CS & SWE, College of Computer Sc. & Engg., University of Ha'il, Ha'il, Saudi Arabia
[3]Professor and Principal, Shri Sukhmani Institute of Engineering & Technology Derabassi, Punjab, India

Abstract

Researchers in the field of software engineering, business process improvement and information engineering all want to drastically modernize software life-cycle processes and technologies to correct the problems and to improve the quality of software. Research goals have included ancillary issues, such as improving user services through conversion to new platforms and facilitating software processes by adopting automated tools. Automated tools for software development, understanding, maintenance, and documentation add to process maturity, leading to better quality and reliability of computer services and greater customer satisfaction. This paper focuses on critical issues of legacy program improvement. The program improvement needs the estimation of program from various perspectives. The paper highlights various elements of legacy program complexity which further can be taken in account for further program development.

Keywords: *Legacy, Program, Software complexity, Code, Integration.*

----------***----------

1. PREFACE

This section will explain the contribution and motivation of research.

1.1 Motivation

As indicated by research study conducted earlier, the best future test for organizations is in managing complexity. As per the present scenario the markets are more volatile, more uncertain and complex, constraining organizations to respond to market changes quickly. The heads of the companies come up against these dangers by empowering their organizations to rapidly respond to these changes. Moreover, the organization's software systems must be empowered to catch up to the fast changes also.

1.2 Contribution of the Paper

This paper introduces an approach to identify elements of legacy program complexity for effective reengineering of the legacy program. The literature has been explored for mapping the elements responsible for enhancing the complexity of the software and also carried a research to know the elements responsible for increasing the complexity of legacy program. The paper is divided into four sections. Section-I entails introduction and literature survey. Section-II describes how to map elements enhancing the complexity of legacy program. Section-III introduces various elements and provides the description of each element. Section-IV summarizes the paper and provides an Outlook on open research issues.

2. INTRODUCTION

The legacy program is an essential component of legacy systems. The legacy systems are those systems which were developed before the invention of advanced software engineering techniques. These legacy systems are still workable and usable but need improvement to accommodate the changes according to current computing needs. The work considers that legacy programs were developed in earlier languages, but even now the program developed in VB and Java in early 90s are also legacy. Legacy software may be characterized as software we don't recognize what to do with however it is performing a valuable job. The inference is that the ideal solution is to abandon the software totally and begin again with new software. This may not be suitable in all cases for instance

A. The software shows years of cumulative experience which is unrepresented somewhere else so discarding the software will likewise abandon this knowledge however clumsily it is shown.
B. The manual system which was replaced by the software no more exists so system analysis must be embraced on the software itself.
C. The software may truly function admirably and its behavior may be well comprehended. A new system may perform significantly poorly in the beginning. Consequently it might be worth recovering a portion of the features of the legacy system.
D. A normal legacy software system has numerous users who normally have misused undocumented features and negative effects in the software. It may not be adequate to request that clients attempt a significant rework for no discernable benefit. Therefore it might be critical to hold the interfaces and definite usefulness of the legacy code both unequivocal and implied.
E. Users may favor an evolutionary instead of a progressive methodology.

Notwithstanding, moving from present software practice dominated by extensive, complex legacy applications with advancement and support accumulations to a future focused around software reuse and automatic project generation has proven to be exceptionally troublesome. One issue is that the advocated transition techniques require dedicated professionals with extensive knowledge of both the application domain and the software technology.

These professionals are charged with continuously making computerized knowledge base of potential application requirements. Such a knowledge base would be supported by a library of versatile, reusable software components and a set of decision rules and computerized methods for selecting and collecting the segments into the desired software application. This methodology, which creates a domain and software reuse environment from the very beginning, has so far demonstrated expensive and unsafe, with long-delayed payback periods for recuperating the initial investments.

3. METHODOLOGY

To study the elements which are responsible for legacy program complexity we have gone through the plethora of literature which is available online as well as in the libraries also. We have consulted the books on this aspect. Through the in-depth study of the literature as well as books we were able to find the elements which are responsible in enhancing the complexity of the legacy program. It took us few months to find and explore the elements of legacy program complexity.

4. ELEMENTS OF LEGACY PROGRAM COMPLEXITY

Following are the elements which are responsible for legacy program complexity.

4.1 Difficulty in Understanding the Code

Difficulty in understanding the code of the legacy systems makes the legacy system complex. Sneed [6] stated that legacy code is difficult to understand and maintain. It will lead to low quality of the software. The difficulty in understanding the code leads to complexity enhancement and thus lowers the quality of the software or system. Bernard [8] showed that the legacy code is written in assembly language or using any of the third generation languages although the system might be doing beneficial job for the organization what it is hard to understand the code. However, in many cases, constant changes cause the code to become convoluted, without organization or structure, adhering to no standards [6], [36], [10]. The resulting code is often called spaghetti code and is virtually impossible to unravel.

4.2 Cost

As the legacy system is large and complex so redesigning or replacing, upgrading the system is very costly. If legacy runs on obsolete hardware, the expense of maintaining the system might inevitably exceed the expense of replacing both the software and hardware unless some type of imitating or regressive similarity permits the software to run on new hardware [38].As matter of fact, the elevated amount of entropy joined with uncertain documentation about the design and architecture make their maintenance more complicated, time consuming, and expensive. Besides, these systems have significant economical value and many are vital to their owners [3]. For the high cost of lost previous investment and business knowledge implanted in those systems, in some cases replacing the legacy systems are replace with new systems which are developed by using new technology is not right decisions as these systems are important assets of the organization.

4.3 Size

The size of the system is also one of the elements that affect the complexity of the legacy system. As the size of the software increases, the complexity of the system also increases. As far as size is concerned there are two factors which are to be considered while dealing with the complexity of the software.

4.4 No of Control Structures

The number of control structures also impacts the complexity of the system. If the program is large in size but no of control structures its complexity will be less on the other hand if the size is small but the number of control structures is large it will increase the complexity.

4.5 Recursive Functions

The size of new projects will basically rely upon the quantity of inputs, outputs and the interfaces, the system has. As their number increases so is the complexity of the legacy program or vice-versa. The attributes of the data structure and procedures in the software makes the software complex and hard to understand.(Curtis et.al,1979).

4.6 Design

The poor design of the system makes it more complex and hence it is hard to change and expensive to support the program. The objective behind every system, including data information system, is to bring about required data information based on input data and its processing. A long time of experience in creating information systems brought about understanding that the amount of documents and the complexity of a given document inside a business system by one means or another establish the complexity of designing, as well as creating a information system. By and large, to completely comprehend them, more time and effort is required for systems with various documents, than for those with fewer documents. Relatively, the project development is more intricate and tedious.

4.7 Integration

Since the legacy systems are based on old platforms that is legacy platforms these platforms are difficult to understand

and hence increases complexity and it is very difficult to integrate these systems with the new platforms is very complex. Incorporation crosswise technique is truly common in computing, yet integration between new technologies and substantially older ones is not normal. There may basically not be sufficient interest for integration technology to be created. Some of this "glue" code is seldom created by sellers and enthusiasts of specific legacy technologies.

4.8 Security

As the legacy systems have older operating system it may be susceptible to attacks due to lack of security patches which are being applied the security issues may be caused by production setups. These issues can put the legacy systems at danger of being traded off by attackers or knowledgeable insiders [4].As legacy systems are based on obsolete technology they are complex and are liable to be less secure that is complexity hinders the security of the legacy programs. The code which is complex ought to be larger than the simple code that means there are more chances for accidents, omissions and manifestation of code errors. As complexity increases, it is common to just affirm that a system or product is secure as it becomes less and less conceivable to really make it secure despite complexity. Regardless of an abundance of testing tools that assert to grab bugs, the complexity of software makes security imperfections and errors unavoidable and progressively normal.

4.9 Performance

Performance is either a moderate degradation of software performance about whether or its decreasing responsiveness that will in the long run lead to software getting to be flawed, unusable, or overall called "legacy" and in need of updating. This is not a physical phenomenon: the software does not really rot, yet rather experiences an absence of being responsive and redesigned as for the changing environment in which it operates. software can fall apart in "performance" over the time and becomes "legacy" as it runs and becomes error prone; this is not for the most part considered software decay, however it may have a portion of the same outcomes. Generally, such a state can be overcome by totally reinitializing its state (as by a complete reinstallation of all applicable programming segments, perhaps including working framework programming). The another element which is responsible for making software or program legacy is the performance as the software deteriorates slowly over the period of time its responsiveness diminishes and makes the software error prone, useless or legacy which needs up-gradation to meet the current technological requirements. As the time passes the software detoriates and its performance decreases and is not able to meet the expectations of the user so it is required that the software must be upgraded rather than discarding it totally. To enhance the performance of the software install the new software's or modifies the code.

4.10 Lack of Documentation

These systems are difficult to maintain, enhance, and extend due to absence of understanding of the system; the staffs who were professionals on it have resigned or forgotten what they knew about it, and staff who entered the field after it became "legacy" never learned it in any case. This can be exacerbated by need or loss of documentation. Comair Airline Company blamed its CEO in 2004 because of the failure of an obsolete legacy team scheduling system that ran into a restriction not known to anybody in the company.

4.11 Flexibility

As the legacy systems are working on old or outdated technology thus are not able to compete with the changing technology and are not flexible. It was observed by [10] notes that legacy systems use old technology, are lacking in flexibility, are highly complex and possibly diverge with corporate strategy.

4.12 Time

As less time for processing and more processing speed is the aim for making optimized use of resources. Time is also important element in complexity of the legacy program the programs which takes more time for processing are more complex as compare to the programs which takes less time for processing or completion of the execution. The time of computation in legacy program is high and thus it contributes in enhancing the complexity of the program.

4.13 Lack of Staff

As legacy programs are built using outdated techniques and thus it is not easy to understand this program so the un availability of the staff increases the complexity. The staff either has left the job and they are not interested in teaching the other persons or the staff has retired from the job.

4.14 Reliability

Reliability means ability of the program to perform its intended functions and operations without failure. As the legacy programs are complex so these systems tend not to entirely understand. if the system is difficult to understand than it is difficult to find the ways by which the system can be compromised by the attackers or intruders. It is not easy to prevent insecure operating modes in the legacy systems and that too in cheaply. Generally it has been observed that complex systems are considered to less prone to attacks or they are secure but the fact is although numbers of tools are available to test the legacy programs but the complexity of the legacy programs makes a security flaws and errors nearly unavoidable and normal.

4.15 Bugs or Errors

A software bug is an error, failure, or fault in a computer program or system that makes it to create a wrong or unexpected result, or to behave in unexpected ways. Most

bugs emerge from mistakes and errors made by individuals in either a program's source code or its design, or in systems and operating systems utilized by such programs, and a couple of them are created by compilers creating wrong code. A program that contains number of bugs or the bugs that genuinely intervene with its functionality is said to be buggy. Reports specifying bugs in a system are generally known as bug reports, defect reports, fault reports, problem reports, trouble reports so on and so forth. The outcomes of bugs may be amazingly genuine. In 1996, the European Space Agency's Us$1 billion model Ariane 5 rocket must be crushed short of what a moment after dispatch, in view of a bug in the ready for machine program. In June 1994, a Royal Air Force Chinook impacted the Mull of Kintyre, slaughtering 29. This was from the get go discharged as pilot slip, however an examination by Computer Weekly uncovered sufficient affirmation to induce a House of Lords request that it may have been brought about by a product bug in the airplane's motor control computer. In 2002, a study commissioned by the US Department of Commerce' National Institute of Standards and Technology concluded that "software bugs, or errors, are so prevalent and so detrimental that they cost the US economy an estimated $59 billion annually, or about 0.6 percent of the gross domestic product"[37].

4.16 Unpredictability

It is the inability to know what will happen. As legacy programs are obsolete and are hard to understand.

4.17 Interdependent Parts/Interfaces

The interdependence between various modules of the program makes the program complex. According to Webster's Encyclopedic Unabridged Dictionary, 2001 complexity is characterized by complicated arrangement of interconnected parts. More the number of interconnected parts more will be the complexity of the program and it is difficult to understand the program. Most of legacy programs are complex involving lots of interrelationships or interdependence and changing requirements.

5. CONCLUSION

From the study it was observed that difficulty in understanding the code, size of the legacy program and cost are the factors which are responsible for the complexity of the legacy program. The lack of documentations and interdependence of the paths or interfaces also contributes in enhancing the complexity of the program. Last but not the least the errors or the bugs in the legacy program increases the complexity of the program as it is very difficult to find the errors or bugs in the legacy program as they are very complex. As far as the lack of staff and unpredictability is concerned these also have an impact on increasing the complexity of the legacy program. This study will help the research scholars in predicting the complexity of the program. The study will be further enhanced using mathematical tools in the future using these aspects as per the demand of our research.

REFERENCES

[1] The RENAISSANCE Framework, RENAISSANCE project deliverable, 1997.

[2] Lamb, John (June 2008). "Legacy systems continue to have a place in the enterprise". Computer Weekly. Retrieved 27 October 2014.

[3] Stephanie Overby (2005-05-01). "Comair's Christmas Disaster: Bound To Fail - CIO.com - Business Technology Leadership". CIO.com. Retrieved 2012-04-29.

[4] Razermouse (2011-05-03). "The Danger of Legacy Systems". Mousesecurity.com. Retrieved 2012-04-29.

[5] [9].(B. Foote, J. Yoder, "Big Ball of Mud", Pattern Languages of Program Design, Vol. 4, N. Harrison, B. Foote, H. Rohnert, (Eds.), Addison-Wesley, 2000.)

[6] Sneed, H., 1995 "Planning the reengineering of legacy systems" January IEEE Software.

[7] Brooks,F. 1975 The Mythical Man-Month. Addison-Wesley.

[8] Bernard, V.L. 1995. "The Feltham-Ohlson Framework: Implications for Empiricists."

[9] Contemporary Accounting Research 11: 733-747.

[10] Bancroft,N.,H.Seip,A.Sprengel , 1997 Implementing SAP/R3. Manning publications. Also available: http://www.browsebooks.com/Bancroft/Contents.html.

[11] Elliot Chikofski and James H. Cross II "Reverse Engineering and Design Recovery: A Taxonomy" IEEE Software January 1990 7(1):13-17.

[12] V. R. Basili and H. D. Mills "Understanding and Documenting Program" IEEE Transactions on Software Engineering May 1982. SE-8(3):270-283.

[13] Ward, M., "Abstracting **a** Specification from Code", Joumal of Software Maintenance, 5(2), 1993.

[14] J. Cordy, I. Carmichael, and R. Halliday. The txl programming language (version 8). Technical report, Legasys Corp., Kingston, Apr. 1995.

[15] R. S. Arnold, editor. Software Reengineering. IEEE Computer Society Press, 1992.

[16] **Cremer, K.,** "A Tool Supporting the Re-Design of Legacy Applications", in Proceedings of 2"d Euromicro Conference on Software Maintenance and Reengineering - CSMR'98, IEEE. Florence, Italy, 1998, pp. 142-49.

[17] Sneed, H. M., Planning the Reengineering of Legacy Systems, IEEE Software, January 1995, pp. 24-34.

[18] Perspectives on Legacy System Reengineering, Software Engineering Institute, Carnegie Mellon University, 1995.

[19] Assessing the Evolveability of a Legacy System, Software Engineering Institute, Carnegie Mellon University, 1996.

[20] Parnas, D.L. Software Aging. in 16th International Conference on Software Engineering. May 1994. Sorrento, Italy.

[21] Simon, H.A., The Architecture of Complexity. The American Philosophical Society, December 1962.

[22] Henderson, R.M. and K.B. Clark, Architectural Innovation: The Reconfiguration of Existing Product Technologies and the Failure of Established Firms. Administrative Science Quarterly, 1990. 35: p. pp. 9-30.
[23] Hughes, T.P., The Evolution of Large Technological Systems, in The Social Construction of Technological Systems, W.E. Bijker, T.P. Hughes, and T.J. Pinch, Editors. 1987, MIT Press: Cambridge, MA.
[24] Simon, H.A., The Architecture of Complexity. The American Philosophical Society, December 1962.
[25] S Comella-Dorda. K Wallnau, R. Seacord, and J Robert. "A Survey of Legacy System Modernization Approaches". SEI Technical Note CMU/SEI-00-TN-003. Software Engineering Institute, Carnegie Mellon University, Apr. 2000.
[26] S. Tilley, and D Smith, "Legacy System Reengineering, Software Engineering Institute", Carnegie Mellon University, Presented at the International Conference on Software Maintenance, Nov. 4-8, 1996.
[27] L. Raccoon., "The Complexity Gap". SIGSOFT Software Engineering Notes, 20(3), Jul. 1995, pp. 37-44.
[28] N Weiderman, J Bergey, D. Smith, B. Dennis and S Tilley, "Approaches to Legacy System Evolution"
[29] (CMU/SEI-97-TR-014). Software Engineering Institute, Carnegie Mellon University, 1997.
[30] R. Pressman. Software Engineering: A Practitioner's Approach, 4th Edition. New York, NY: McGraw-Hill, 1997.
[31] J. Ransom and I. Warren. "A Method for Assessing Legacy Systems for Evolution," Proceedings, Second Euromicro Conference on Software Maintenance and Reengineering (CSMR98), 1998.
[32] Arnold,R.1989"Software Restructuring" Proceedings of the IEEE, April Vol. 77, No. 4, pp 607 – 617.
[33] "Software bugs cost US economy dear". Web.archive.org. June 10, 2009. Retrieved September 24, 2012.
[34] Bisbal, J.; Lawless, D.; Bing Wu; Grimson, J., "Legacy information systems: issues and directions", Software, IEEE , vol.16, no.5, pp.103-111, Sep/Oct 1999.
[35] http://en.wikipedia.org/wiki/Legacy_system