DOI: http://dx.doi.org/10.26483/ijarcs.v9i2.5853

ISSN No. 0976-5697

Volume 9, No. 2, March-April 2018

International Journal of Advanced Research in Computer Science

REVIEW ARTICLE

Available Online at www.ijarcs.info

LITERATURE REVIEW ON SOFTWARE COMPLEXITY, SOFTWARE USABILITY AND SOFTWARE DELIVERABILITY

Mohd. Kamran Khan
Department of Computer Science & Engineering
Integral University
Lucknow, Uttar Pradesh, India

Mr. Faiyaz Ahmad
Department of Computer Science & Engineering
Integral University
Lucknow, Uttar Pradesh, India

Dr. Mohammadi Akheela Khanum
Head, Department of Computer Science & Engineering
Integral University
Lucknow, Uttar Pradesh, India

Abstract: This paper is a literature survey (Review Paper) of comparative analysis on various definitions and concept explanations on software complexity, software deliverability and Software usability published recently (till date) in different international papers and books. This paper also includes the basic introduction of soft computing and its approaches. This paper explains how the software usability gets affected by the software complexity.

Keywords: Software Complexity, Software Usability, Software Deliverability and Soft Computing

I. INTRODUCTION

With the ever-increasing expansion of business forces, it is becoming very hard to model the software in a decentralized and distributed atmosphere. If we model the complete business process, it leads to an extremely complex and demanding software application which means complexity in the usage. The major aim of the software gets defeated if its user finds it very hard to operate or cannot contribute considerably to the productivity. The huge amount of data and functionalities required by the current enterprise systems imposes numerous challenges on software developers. The most important difficulty arises to maintain a balance between Software Complexity and Usability, as both the qualities of the software is very much inter-related. The main reason of choosing this study is the companies like SAP, Oracle are now losing their market because of the cost and complexity of their software product.

II. SOFTWARE COMPLEXITY

Complexity has been a frequent word in recent scientific literature, in different fields and with diverse meanings. Sometimes complexity appears as a precise concept, at other times as vague ideas. Many organizations are seeking to upgrade their business with the help of Software. These organizations are looking out for software supplier which can provide a tool that meets their organizational needs. Although there are number of tools available in the market, but the sole reason of selecting criteria over completing product is the ease of use [2,7]

The process of software development, including documentation, design, program, test, and maintenance can be measured statistically. Therefore, the quality of software can be monitored efficiently. Software metrics is very important in research of software engineering and it has developed gradually. In this paper, software metrics definition was given and the history of and the types of software metrics were overviewed. Software complexity measuring is the important constituent of software metrics and it is concerning the cost of software development and maintenance. In order to improve the software quality and the project controllability, it is necessary to control the software complexity by measuring the related aspects [1, 2].

The software products that are technologically advancedand developed based on prioritized requirements can be anticipated to have a minor probability of being rejected. In order to rankthe requirements, participants will have to relate them in order to limit their relative status through a weighting or scoring arrangement [10]. These comparisons turn out to be complex with rise in the number of requirements [11]. Hence, the intention of this learning is to chronologically select and review published literature and present a holistic outline of present techniques used in ordering software requirements. To the greatest of the authors' knowledge, there is no present SLR that emphases on prioritization techniques with respect to their explicit boundaries, taxonomies, and procedures. Instantly, the crux of this SLR is to abridge and simplify the existing evidences concerning (1) the existing prioritization procedures, (2) their restrictions, (3) procedures and (4) taxonomies. This SLR will thusdeliver insight for both researchers as well as practitioners in the businesses and academic world in their pursuit to develop and employenhanced techniques [2,3].

With the evolution of the software development, the scale of the software is increasingly growing to the extent that we cannot hand it easily. Some metrics are proposed to measure the complexity of software in last a few years. This article aims at a comprehensive survey of the metric of software complexity. Some classic and efficient software complexity metrics, such as Lines of Codes (LOC), Halstead Complexity Metric (HCM) and Cyclomatic Complexity Metric (CCM), are discussed and analyzed first[4,9].

Software complexity is important to researchers and managers, yet much is unknown about how complexity evolves over the life of a software application and whether different dimensions of software complexity may exhibit similar or differentevolutionary patterns. Using cross-sectional and longitudinal data on a sample of 108 open source projects, this research investigated how the complexity of open source project releases varied throughout the life of the project. Functional data analysis wasapplied to the release histories of the projects and recurring evolutionary patterns were derived. There were projects that saw little evolution, according totheir measures of size and structural complexity. However, projects that displayed some evolution often differed on the pattern of evolution depending onwhether size or structural complexity was examined [5, 11].

III. SOFTWARE USABILITY

The integration of User-Centered Design (UCD) and Agile development processes have been gaining increased interest, in part due to the complementarity of the techniques, the benefits each can apply to the other, and the challenges associated with their combination. The research paper in this contextoutlines a Systematic Literature Review (SLR) that was focused on the Agile as well as UCD integration. The objective of this SLR was to recognizenumerous challenging issues that limit Agile and User Centered Design Integration (AUCDI) and discover the projected practices to handle them. Agile methods are lightweight software development methods that tackle perceived limitations of plan-driven methods via a compromise between absence of a process and excessive process [29]. Agile methodsfocuson dealing with the volatile requirements throughdumping upfront, exactly defined strategies [6,8].

Agile approaches are simpler software development procedures that handles perceived restrictions of plan-driven approachesthrough a negotiationamong absence of a process in addition to excessive process. The Agile processes intent to handle with volatile necessitiesthrough discarding upfront, precisely clear plans. They are basically used to develop software incrementally.

From last few years, the development of software has been considered by two mainstyles: agile software development, that aims to attainimproved flexibility and velocity during the process of development, and user-centered design, that places the objectives and requirements of the system's end-users at the middle of development of software in order to bring software with suitable usability [7].

The Software Usabilityis well-defined by ISO as "the software product's capability to belearned, understood, used and striking to the end user, when used indefinitecircumstances" [1]. This attribute has gained importance as an integral quality aspect of software development [2]. The benefits of usability for users and software companies have been much highlighted in literature [3, 4, 5]. However, poor usability is the single biggest cause of software application failure in practice [6]. For over a decade, the software engineering (SE) community has been actively pursuing different lines of research targeting the incorporation of usability practices into software development [9, 10].

IV. SOFTWARE DELIVERABILITY

The deliverability of the software deliverability can be measured as the degree of the usability factor providing to the handler of the system by the software. The deliverability of softwaremust be high with the intention ofachieveextreme value from thesoftware. The commercial value of any software is vastly affected by the deliverability of software which in futureenforcesnumerouslimitations on the software designers [19,20].

The usability of software is essentially the level of ease or the comfort with which anend-user can work on the software. Like explained earlier the complexity of softwarediffers from one user to anotheruser and from one software to another in a definite and controlledset-up. The usability of software also varies consequently. The higher the software complexity, the lower will be the usability aspect of that specific software in use [12, 17].

High usability factor permitsthe software to capturemaximum market space prospect as comparison to those software products which has comparatively low software usability value. This straight awayimpacts the market position of the companyand the software product.

The well-known term 'software usability' in the perspective of developing software represents a method that place the user, instead of the system, at the center of the business process which is to be developed. This concept is called user-centered design, thatincludes user apprehensions and advocacy from the start of the design procedure and commands that the requirements of the end user must be leading in any design results [14, 21]. The greatest visible characteristic of this method is usability testing, that includesthe work of users and interact with the x interface of the software and share their opinions and worries with the software developers. This paperdeliberates the notion of software usability and why it must be a significant part of the software project [13, 25].

The standard definition of software usability given by ISO is "The degree up to which a software product can be utilized by particular users to attain particular objectives with efficiency, and effectiveness as well as satisfaction in a specified context of usage". The term 'usability' is also refers to the ways for improving the ease-of-use by the user during the software development design process [23, 27].

The usability consultant Jakob Nielsen and computer science professor Ben Shneiderman have given (individually) regarding a framework of the software system acceptability, where the software usability is an integral part of the 'usefulness' and is composed of following factors:

1. Efficiency
2. Learnability
3. Satisfaction
4. Memorability
5. Errors

The mentioned factors can be explained simply. Like learnability is how effortless is it for users to complete fundamental tasks the opening time they come across the design. In addition, the term efficiency can be explained as once the users have learned the design, how fast can they carry out the everyday jobs [16,18].

The next factor memorability can be defined as when users go back to the design following a period of not using it, how

effortlessly can they re-establish the expertise. The next factor errors can be understood as how many errors do users make, how rigorous are these errors, and how without difficulty can they recuperate from those errors and the last factor satisfaction can be explained as how pleasing is it to utilize the design [22, 26].

Six basic factors of software usability are as follows: -

1. Context shifts
2. Navigational guidance
3. Input parameters
4. System feedback
5. Error feedback
6. New concepts

1. The term 'Context shift'happens when the end user crosses the tool or product boundaries, graphical user interface,so as tocarry out a step.

2. Navigational guidance refers to the support provided to a user for proceedings into a step (from the previous step) and through the step.

3. The Input parameters are basically the data supplied by the end user to executeall the steps.

4. The term 'System feedback'is defined asresponse of the system to the actions of the end user for a given prescribed steps. Instances of system feedback may comprisegrowth indication dialog boxes, verification of execution of commands and reports generated by systems.

5. The term 'Error feedback'can be definedas the response of the system to frequentfault situations the end user may encounter.

6. The term 'New concepts' refer to the background information on a explicitissue that the end user desires to know in order to execute a step and that the end user has encounter for the very first time in the perspective of therecent task(s) [24].

CONCLUSION

Deliverability of the software can be measured as the degree of the usability factor given to the software user. The deliverability of software be high so as toachieveoptimum value from the software in use. The commercial value of any software is majorlyimpacted by the deliverability of software which further imposes numerouslimitations on the software developers.Higher the complexity, lower will be the usability aspect of that particular software product.

V. ACKNOWLEDGMENT

The author is thankful to Dr. Manuj Darbari, Professor, Department of Information Technology, BBDNITM, Lucknow and Dr. Siddharth Lavania, Manager, IFCI Ltd, New Delhi for their kind support and encouragement in this research work.

VI. REFERENCES

[1] Honglei, T., Wei, S., & Yanan, Z. (2009, December). The research on software metrics and software complexity metrics. In Computer Science-Technology and Applications, 2009. IFCSTA'09. International Forum on (Vol. 1, pp. 131-136). IEEE.

[2] Achimugu, P., Selamat, A., Ibrahim, R., & Mahrin, M. N. R. (2014). A systematic literature review of software requirements prioritization research. Information and software technology, 56(6), 568-585K.

[3] Paternoster, N., Giardino, C., Unterkalmsteiner, M., Gorschek, T., & Abrahamsson, P. (2014). Software development in startup companies: A systematic mapping study. Information and Software Technology, 56(10), 1200-1218.

[4] Yu, S., & Zhou, S. (2010, April). A survey on metric of software complexity. In Information Management and Engineering (ICIME), 2010 The 2nd IEEE International Conference on (pp. 352-356). IEEE.

[5] Darcy, D. P., Daniel, S. L., & Stewart, K. J. (2010, January). Exploring complexity in open source software: Evolutionary patterns, antecedents, and outcomes. In System Sciences (HICSS), 2010 43rd Hawaii International Conference on (pp. 1-11). IEEE.

[6] Salah, D., Paige, R. F., & Cairns, P. (2014, May). A systematic literature review for agile development processes and user centred design integration. In Proceedings of the 18th international conference on evaluation and assessment in software engineering (p. 5). ACM.

[7] Brhel, M., Meth, H., Maedche, A., & Werder, K. (2015). Exploring principles of user-centered agile software development: A literature review. Information and Software Technology, 61, 163-181.

[8] Carvajal, L., Moreno, A. M., Sanchez-Segura, M. I., & Seffah, A. (2013). Usability through software design. IEEE Transactions on Software Engineering, 39(11), 1582-1596.

[9] Parks, N. E. (2012, October). Testing & quantifying ERP usability. In Proceedings of the 1st Annual conference on Research in information technology (pp. 31-36). ACM.

[10] Albertao, F., Xiao, J., Tian, C., Lu, Y., Zhang, K. Q., & Liu, C. (2010, November). Measuring the sustainability performance of software projects. In e-Business Engineering (ICEBE), 2010 IEEE 7th International Conference on (pp. 369-373). IEEE.

[11] Lavania, S., Darbari, M., Ahuja, N.J., Siddqui, I.A.: Application of computational intelligence in measuring the elasticity between software complexity and deliverability. In: 2014 IEEE International Advance Computing Conference (IACC). IEEE (2014)

[12] Lavania, S., Darbari, M., Ahuja, N.J., Shukla, P.K.: Application of evolutionary algorithm in managing the trade-off between complexity of software and its deliverables. Int. Rev. Comput. Softw. **7**(6), 2899–2903 (2012)

[13] Lavania, S., Darbari, M., Ahuja, N., Siddqui, I.A.: Application of computational intelligence in measuring the elasticity between complexity and deliverability. In: 4th IEEE International Advanced Computing Conference—IACC (2014)

[14] Lavania, S., Darbari, M., Ahuja, N. J., & Shukla, P. K. (2012). Application of evolutionary algorithm in managing the trade-off between complexity of software and its deliverables. International Review on Computers & Software, 7(6), 2899-2903

[15] Lavania, S., Darbari, M., Ahuja, N., Siddqui, I.A.: Genetic algorithms-fuzzy based trade-off adjustment between software complexity and deliverability. In: 9th Annual International Joint Conferences on Computer, Information, Systems Sciences, and Engineering. Springer (2013)

[16] Y.Dia and Keller, "Quantifying the complexity of IT Services Management Processes", IBM Research Report, 2006.

[17] Sobiesiak, R., & O'Keefe, T. (2011, November). Complexity analysis: a quantitative approach to usability engineering. In Proceedings of the 2011 Conference of the Center for Advanced Studies on Collaborative Research (pp. 242-256). IBM Corp.

[18] Y.Dia and R.Sobiesiak, "Quantifying Software Usability through Complexity Analysis", IBM Design: papers and Presentations, 2010.

[19] P K Shukla, S P Tripathi, Interpretability issues in Evolutionary Multi-Objective Fuzzy knowledge Base Systems, 7th International Conference on Bio-Inspired Computing: Theories and Applications (BIC-TA 2012),ABV-IIITM, Gwalior,India,14-16 December, 2012.(Springer AISC SERIES).

[20] Darbari, M., & Yagyasen, D. (2013). Application of Granularised OWL framework for modeling Urban Traffic System. Pensee Multidisciplinary Journal, 75(9).

[21] Ahmad, S. S., Purohit, H., Mohammed, F. N., & Darbari, M. (2013). Information granules for medical infonomics. International Journal of Information and Operations Management Education, 5(3), 205-213.

[22] Diao, Y., & Keller, A. (2006, October). Quantifying the complexity of IT service management processes. In International Workshop on Distributed Systems: Operations and Management (pp. 61-73). Springer, Berlin, Heidelberg.

[23] Nielsen, J., & Mack, R. L. (1994). Usability Inspection Methods John Wiley & Sons. New York.

[24] Neilson Jacob, "Usability Engineering", Boston AP Professional, 1994, ISBN: 0-12-518400-X

[25] D. Dumas, Joseph S., and Janice C. Redish. "A practical guide to Usability Testing", London : Intellect Books, 1999. ISBN: 1841500208.

[26] L. Zadeh, "Fuzzy Sets, Fuzzy Logic, and Fuzzy Systems", Selected Papers by Lotfi A Zadeh edited by: George J Klir (SUNY, Binghamton) edited by: Bo Yuan (SUNY, Binghamton). ISBN: 978-981-02-2421-9.
Reports

[27] Usability in Software Design, "A Report by Microsoft Corporation", 2000

