A Comparitive Analysis Between Open Source And Closed Source Software in Terms of Complexity and Quality Factors

Mehreen Sirshar
Faculty of Software Engineering, Fatima Jinnah Women University, Rawalpindi, Pakistan;
mehreensirshar@fjwu.edu.pk

Asma Ali
Department of Software Engineering, Fatima Jinnah Women University, Rawalpindi, Pakistan;
aasma6038@gmail.com

Sara Ibrahim
Department of Software Engineering, Fatima Jinnah Women University, Rawalpindi, Pakistan;
saraibrahim7479@gmail.com

Abstract: The complexity of software is increasing day by day due to the increase in the size of the projects being developed. For better planning and management of large software projects, estimation of software quality is important. During the development processes, complexity metrics are used for the indication of the attributes/characteristics of the quality software. There are many studies about the effect of the complexity of the software on the cost and quality. In this study, we discussed the effects of software complexity on the quality attributes of the software for open source and closed source software. Though, the quality metrics for open and closed source software are not distinct from each other. In this paper, we comparatively analyzed the impact of complexity metrics on open source and private software. We also presented various models for the management of the project complexity such as William's Model, Stacey's Agreement and Certainty matrix, Kahane's Approach and UCP Model. Quality metrics here refer to the standards for the measurement of the quality of software which contains certain attributes or characteristics of the software that are related to the quality of the software. Certain quality attributes addressed in this study are Usability, Reliability, Security, Portability, Maintainability, Efficiency, Cost, Standards and Availability, etc. Both Open source and Closed source software are evaluated on the basis of these quality attributes. This study also recommended future approaches to manage the quality of project Open source and Closed source software and specify which one of them is mostly used in the industry.

Keywords Software Quality Metrics; closed source software; open source software; Kahane's Approach; UCP (Use Case Points) model and William's Models

1. Introduction

Complexity is one of the most important factor in the management of a project. In every organization which develop software have different perspective and definition of complexity of the software. Every organization has variant complexity factors such as Project Management Institute has a different view of complexity factors in comparison with the System of system view. There are certain attributes or characteristics upon which complexity of a software is defined and evaluated. Such as the most common complexity factors involved in almost all software projects are dependencies between the systems and the subsystems, software dependent on the cost and time constraints of the projects, project specified requirements and another factor is technology, tools and techniques used in the project development.

For the better planning and management of the software product early analysis of the complexity of the software is important. Software complexity is considered to be the most effecting factor in the development of the software and also contributes in the maintenance of the software. With the increase in the quality and size of the software product software complexity is also increasing and it became essential to control and maintain the software complexity.

Software metrics tools are used for the measurement of the progress in the development of the software and to analyze the different types of defects and loopholes in the software and these metrics help in the maintenance of the software. Software Quality metrics are used for the measurement analysis of the performance of the software and determine performance on the basis of certain factors such that maintainability, reliability, usability, satisfaction of customer and security that affects the quality directly or indirectly. Tom Demarco assured that "You can't control what you can't measure". So to control the complexity and quality of a software it is better to measure all the vulnerabilities first by software quality and complexity metrics.

Software Complexity metrics that are being proposed in the literature are discussed in this study to maintain the software complexity. Complexity metrics are used to check each type of complexity either it is structural, internal or design complexity. There are also certain types of complexity like complexity in the design and architecture of the software like UML diagrams etc. and also complexity in the source code of the software. Software complexity metrics like William's Model, Stacey's Agreement and Certainty matrix, Kahane's Approach and UCP Model are used to analyze the performance and help in maintaining the complexity of the software that leads to the improvement of the quality of the code, reduces cost, increase productivity and architectural standards.

In this study software complexity factors are discussed for both Open Source and Closed Source software. However there is no much difference in the complexity metrices of the both software but the complexity factors has distinct influence on both software. Open Source Software are now widely used in software industry because of certain quality factors like innovation, code improvement, cost reduction etc. Open source software is reviewed as the effective software in software industry. Although, there are certain negative traits of Open source software are also mentioned but it is widely influencing today's software industry.

In this paper we discussed that main causing factors of the complexity, different types of complexity and how complexity of a software influences a particular software products and which factors are involved in the complexity of Open source and private software. We also discussed how these factors influences differ in different type of software.

2. Literature Review

In [1] Nguyen described that for the better planning and maintainability of the software, early prediction of software quality is important. He reviewed the impact of software complexity metrics on quality attributes and examined the two different types of complexity i.e. design object complexity and source code complexity. He adapted Goal-Question-Metric(GQM) for the validation of design complexity by investigating its relationship with external quality. He proposed four research questions from the literature review and used two approaches to summarize the results of the systematic review i.e. vote counting and meta-analysis.

In [1] Partap et al., studied the analysis of the design complexity metrics of Object Oriented software in relation with reusability. They proposed that how design metrics like CK(Chidamber and Kemerer) metric provide support in estimation of the reusability. They compared four regression techniques to find out which one will give the appropriate estimation for the reusability of the software products i.e. Multiple Linear Regression(MLR), Model Tree(M5P), Meta Learning Addictive Regression and Isotonic Regression(IR). The data set on which these techniques are applied is taken from the real world project and the tool used for the implementation of these techniques in "WEKA". The comparison between these Regression algorithm demonstrates that Additive Regression with Model Tree(M5P) is the best one for the estimation among the three other algorithms implemented.

In [3] Andrea et al., evaluated the BPMS(Business Process Management System) that contains specific needs/characteristic of each organization. They provide a systematic approach for evaluating

Business Process Management System Tools both open source and closed source which includes some key characteristics of the software and evaluate them by the mean of test cases and case study. They also proposed that in the future work they are going to evaluate the open source software by including some other aspects to the evaluation e.g. Non Functional Aspects.

In [4] Michael Sacks analyzed the comparison of public and private software by the evaluation of a competition between a firm developing closed source software and a community developing open source software. He proposed that developer of OSS and developer of Private software should not compete over the same market. Open Source software leads to the self-selection whereas the Closed source software leads to market.

In [5] Henrik et al., proposed a novel approach for the impact or security assessment based on the analysis of the vulnerabilities occur because of the code changes.. They described the approach using an example of the software and perform the evaluation/comparison on the basis of both closed source and the open source software. They also compared the results of our proposed approach with the three state-of-art tools with the similia results. Several tools were used to detect the usage of the libraries that are known to be vulnerability such as OWASP Dependency check or Victims Project.

In [6] Nicholas et al., explained that the quality assessment models are classified into five categories: single attribute, rounded category, non-community attribute, community-only attribute, and non-quality. They proposed that the hierarchical structure is used to be the best selection methods among all models. They analyzed all the software assessment models from the previous study and described that it will help in the development of newer assessment models for the evaluation of the software. They also analyzed all the key characteristics of Open source software like usability, maintainability, safety and satisfaction and specified that maintainability and usability as the most valuable quality characteristic. This research will help the researchers regarding the development of new software quality assessment models.

In [7] Vyron et al., proposed that the project management is a management method which used widely today in order to manage the complex problems. In this paper they analyzed he factors involved in the complexity of software that are related to project time, cost and quality. Then they applied these factors to selected projects and evaluate these factors on each selected project separately. These factors evaluation provide help in the development of new models for managing and controlling the complexity of the software product.

In [8] Gauri et al., compared the private and open source software to better understand the factors that involves in the key characteristics of software complexity and quality. The main focus of this paper is the review on Open Source Software. They evaluated the comparison of the open source and closed source software in terms of cost, time, security, usability, maintainability, availability and reliability. This paper will give the solid study of Open Source Software.

In [9] Jane et al., studied the effect of project complexity on a case of Orange money project. The objectives of this study is to analyze the impact of project team, project planning, management support, IT infrastructure on implementation of Orange Money project. They used the descriptive research designs to analyze project complexity. They used Correlation analysis, regression analysis to specify the most critical factor in the complexity of the software and find that the management support is the most critical factor.

W.Aslam et al., [10] uses agile software development approach for assigning the appropriate task to particular project member. This approach facilitates the project management activities, lessens the complexities, and influences the chance of project success. They use a task allocation framework that consist of two phases; one identifying the factors and dependencies that strongly influence the task allocation decision; two, proposing a quantitative method that allocate the task to team members who best match the task requirements. In this way understanding of project increases that reduce the complexities issues within the project development process.

In [11] Massacci et al., presented the work need to estimate the effect of security while consuming free and Open source software objects. They proposed three different model for the cost estimation i.e. centralized, distributed and hybrid. They also presented some other factors like commits, code size to estimate that which one has the greatest effect on the security effort of using Free and open source software. They uses SAP product for the implementation of these security factors and analyzing the critical factor among them. They uses seven different data sources/ data sets i.e. National Vulnerability database, Open sourced Vulnerability Database etc. for the implementation and also establish the FOSS project metrics.

In [12] Hilton et al., used the three complementary methods to study the use of Continuous Integrity(CI) in open source software products. They used 34,544 open source projects from the GitHub to analyze which CI systems developer use. They used 1,529,291 from the most commonly used CI system to analyze that how developers used CI. With the help of the data analyzed from answers of these questions related to cost and benefits of CI.

In [13] Sindhu et al., examined the Free and Open source software(FOSS) in terms of the key characteristics i.e. Usability and Deployment. Different attributes are identified from the literature and convey them to this study to evaluate the empirical analysis of the FOSS. They also specified different types of user involved in the FOSS. Moreover, other important characteristics like Functionality, Efficiency and Reliability of FOSS.

In [14] Hutchesson et al., discussed about the tool for the private UML modeling(PTC Integrity Modeller) which is used to design the model based development of critical systems and for open source software the tool they used was Epsilon. They evaluated the open source software implementation in Epsilon with IM on the basis of performance, incrementality and interoperability.

In [15] Carral et al., gave an overview that how complexity of the software can be observed by the project management team to manage and control the complexity impacts in a project. The main complexity models are helpful for the project management team. They list down all the factors involved in the software complexity i.e. size, interdependence, goals and objectives, user requirements, maintainability, technology and ambiguity etc. They also categorized all the types of project complexity and the project complexity models such that goals and method matrix, Stacey's Agreement and Certainty Matrix, kahane's Approach and Cynefin Decision-Making framework etc.

In [16] Ahmed et al., regarded Open Source Software as one of the popular trends in Information Technology. In this study they determined the factors that affect the Open source software in desktop environment. They used usability, compatibility, quality and support factors as the affecting factors. In order to increase the usage of Open Software System they concentrate on usability and compatibility factors.

In [17] Thomas et al., proposed Intuition and Reflexivity as the important variables in management theories. They collected the data from project managers of different software development projects and using the questionnaire survey method. The results of this study is that intuition and reflexivity are used to improve the management processes when there are technical complexities. Intuition and Reflexivity will be helpful in the better management of decision-making.

In [18] Zimmermann et al., studied the transition from the closed to open source systems. They provide guidance for other teams to make their project open source. And help them in avoidance with the pitfails during transition from closed source to open source. They studied five different Microsoft systems which are open sourced i.e. entity framework, MVC, Orleans etc.

In [19] Javad et al., proposed the clarification of the project complexity by the understanding of the previous research about the complexities of the software products. They found that there are three different models of project complexity i.e. Project management institute view, System View and Complexity theory views. They differentiate the different complexity theories aspects of the software for simple and complex projects.

3. Proposed Methodology

In this paper we proposed our perspective of what a software complexity is and how the factors affect the complexity of a software and results in the poor performance of software. We specified two particular types of software currently available in Information technology industry which are Open Source Software and Private software. We proposed some models for the management of the complexity of the software product which we specified from the literature. We also discussed which one of the software is more secure and less complicated and how it can be effective in usage with less complexity and upright performance.

In Future work we proposed that how we can the models for measuring the complexity of the software because complexity is the main factor of the software development. Complexity analysis should have to be initiated with the finalization of requirement gathering. So that the complexity can be addressed during each phase of the development and it can be improved at that phase without any delay of causing more difficulty to other modules.

Project Complexity

According to the standards of IEEE software complexity is "the degree to which a system or component has a design or implementation that is difficult to understand and verify". According to this definition of IEEE software complexity is divided into two different kinds of complexities: complexity in the design of a project and complexity in the implementation of a project.

There are different definitions of software complexity but most of the researcher used some of the keywords to define complexity of the software such as: dependencies between the modules of projects, Operational interdependency, technological complexities, difficulty in understanding project etc. These factors are the main contributors in the complexity of a software.

A. Complexity Sources/Factors

Factors that contribute in the increasing and decreasing of the severity of a project. There are numerous factors that can contribute in making a project difficult.

Table 1 shows the sources that are the cause of complexity in a project.

Table 1. Factors Affecting Complexity of Software.

Group of complexity factors	**Factors**
Objective, Requirements and expectations	Clear statement of requirements Realistic expectations Clear strategic objectives Uncertain and changing Regulatory requirements
Cost	Consistent Changes in the budget of a project External source for the finance of a project Lak of project planning
Stakeholders	Involvement of participants during development. Support for the maintainability and Support of project Sponsors.
Team Management and decision making	External teams involved All the team members that are directly or indirectly involved in the development and management should have to be familiar with all aspects of projects.
Time	Large number of deliverables Insufficient resources dependencies between modules of project. Critical Activities the activity that takes larger time in implementation.
Technology	Incompetence in understanding because of New technology used for development

	IT project management support
Size	Long Project durations Large number of activities/function, deliverables and dependencies/interconnection.

B. Types of Complexity

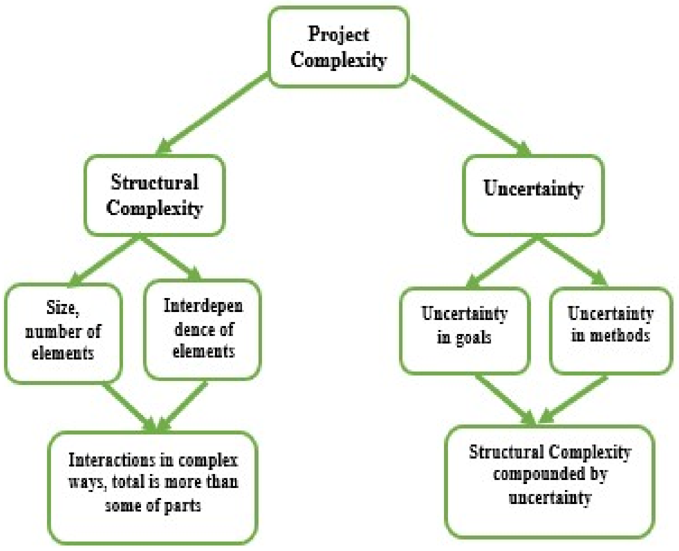
Project Complexity
Structural Complexity
Uncertainty
Size, number of elements
Interdependence of elements
Uncertainty in goals
Uncertainty in methods
Interactions in complex ways, total is more than some of parts
Structural Complexity compounded by uncertainty

Figure 1. Williams Model for Complexity.

According to Pollack and Remington, they proposed four types of complexity in their research which are as follow:

✓ **Technical Complexity**

Technical complexity refers to the difficulty in the design and architecture of the project. These types of complexity mostly occur in new projects which is not previously implemented because for new project sufficient architecture and design details are not available. It is Further divided into three type: (i).Pooled (ii) Sequential (iii) Reciprocal

Pooled: In this every element of the project gives discrete contribution in a project.

Sequential: In which output one element of the project becomes the input of other element of a project.

Reciprocal: In which output of each element becomes the input of other elements.

✓ **Structural Complexity**

Structural Complexity refers to the difficulties that occur because of the dependency or interconnection of the activities/modules in a project. It mostly appears in large scale projects where the project is divided into different small modules or activities. In order to avoid this type of complexity managers must have to keep track of all the interconnected activities.

✓ **Temporal Complexity**

Temporal complexity refers to the difficulties that occur due to the uncertainty in the project. Uncertain environment of a project includes unexpected legislatives changes in the project. It appears because of the inconsistent changes in the technology used in the project.

✓ **Directional Complexity**

Directional Complexity refers to the difficulties in which objectives and goals of a project is not understood well. The scope of the project is too vague to understand the goals and objectives. It occur because of the inconsistent changes in the project. It can be avoided by clarifying the objectives and goals of the project.

✓ **Cognitive Complexity**

Cognitive complexity is the type of complexity that is divided into two types of complexities: (i)Complexity in the design and architecture of the project (ii) Complexity in the source code of a project while implementation phase.

Project Complexity Management

Figure 2. shows the framework for the management of the complexity in a project.

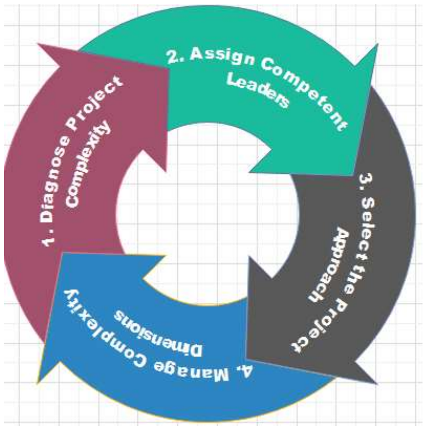
1. Diagnose Project Complexity
2. Assign Competent Leaders
3. Select the Project Approach
4. Manage Complexity Dimensions

Figure 2. Complexity Management Framework.

1. Project Complexity diagnoses on the project complexity models
2. Assigning the project to the Leaders who are competent
3. Selection of the approaches used for the management of complexity
4. Management of the complexity dimension present in the project.

Models for the Management of Project Complexity

Here are some of the appropriate complexity models for the management of project

➢ **William's Model:**

In this model William divided the complexity into two different factors: (i)Structural Complexity (ii)Uncertainty. Where structural complexity depends on the design, size, architecture and dependency between the elements of the project. Whereas Uncertainty refers to the uncertainty in the goals and method of the project. Structural complexity occur because of the division of project elements into smaller modules and their dependency on each other.

➢ **Stacey's Agreement and Certainty Matrix**

In this matrix, analysis of complexity depends on two dimensions one is certainty and other is agreement level. Based on these two dimension matrix is developed following these factors: (i).**close to certainty**: Goals of this zone is the identification of the processes for increasing the efficiency and effectiveness of projects.(ii)**close to certainty and far from agreement**: in these types of zones negotiations are used for the solution these type of complexities in a project.

➢ **Kahane's Approach**

This complexity management model is rooted from the social environment. In this model, U-process is used for the identification of different types of complexity present in a project by three methods: (i) /dynamic complexity (ii)generative complexity (iii) social complexity.

When U-process is used in the kahane approach manager of the project involved take three activities: (i) Identification of the current situation of the project (ii) identifying what is going on in a project and what possible solutions should they do (iii) Presenting quick solution to solve it.

➢ **UCP(Use Case Points) Model**

Use case points model categorize projects in terms of complexity and certainty factor of the product. In UCP model uncertainty is further divided into four levels:

(i)Small technology Projects: in which existing technology is used **(ii) Medium Technology Projects**: in which existing technology is used but with few innovative and new features **(iii) High Technology Projects:** that uses existing but new technology **(iv) Super High Projects**: completely new technology that does not exist yet. UCP model categorize complexity into three different system scope: (i)Assembly (ii) System (iii) Array.

- **Assembly:** Collection of the elements of the project in a single module and perform limited functions on the modules.
- **System:** Collection of the complex dependent activities performing inclusive functions.
- **Array:** Huge collection of all the system features together to achieve same goal and objective.

Many other Models/Matrix are also used for the complexity management of a project like: Decision making Framework of Cynefin, Matrix for goals and methods etc.

4. Open Source Vs Closed Source Software

Open Source Software

It refers to the software that is developed and tested openly and anyone can access the source code for it, anyone can update it and can modify the source code according to their need. It can be shared openly with the intended users.

Private software

Closed source software refers to the software that is developed by particular developer or organization and its source code can only be modified by the organization or the individual who developed it. It cannot be shared openly among the intended users.

Table 1. Open Source Software VS Private Software.

Open Source Software	Closed Source Software
No license fee is required for Open Source Software users.	It contain license fee if the user want to use the software
Source code of OSS can be accessed easily by the users.	This software source code can only accessed by the developer of the software.
OSS managed by the open source community of programmers.	Closed source software is managed by individuals who developed it.

OSS provides better flexibility which lead to the innovation in the software.	Focus on very limited scope for innovation because of the restrictions.
Examples: Android, Ubuntu, Firefox etc.	Examples: Windows. Adobe Flash Player, macOS etc.

5. Open Source Vs PrivateSoftware in Terms of Complexity and QuaLity Factors

Open Source and closed source software are two different software terms and are differentiable in terms of various properties.

Table 3. Open Source Vs Closed Source Software in Terms of Complexity and Quality Factors.

Factors	**Open Source Software**	**Private Software**
Cost	OSS contain less cost or no cost because OSS free software. There is no need of expensive hardware or software for the development of the software.	Cost of the closed source software varies depending on the complexity of the software. Complexity of the system is dependent on some of the critical activities in software in which much time is needed and for the better solution for complexity the use expensive software and hardware which increase in the cost .
Security	Operating software used for the Open source software are considered the most secured OS. Linux being the best example of OSS. But is considered less secure as compared to the closed source software because during development of OSS in organization they do not consider security feature and source code is also available freely to all so it's less secure.	Security is always the hot topic for debate. Closed source software usually use private operating system for the development. So closed source software is considered as secured software because it is developed in the controlled environment and the code is developed by only one individual or team of an organization and also edited only by the developers which reduces the risk of hacking and reduce bugs in the software.
Usability	Previously Open Source software has been criticized for less usability by the users. They do not use well documented user manual earlier but now Open Source Software are also well documented.	Closed source software are usually tested by the experts because they take into consideration the intended audience of the software. Detailed user manual and documentations are provided for the better understanding to the users of the software.
Reliability	OSS are available on numerous unverified websites and even the modification in the source code of them are may be done by the user. The modification made by them may work perfect individually but it may cause clashes with other components of the	Closed source software are developed by the specialized developers. Only completely finished product is delivered to the users and there is no unauthorized modification made to the software so the closed source software is always reliable in case of any failure.

	software which causes the failure of the system.	
Innovation	Open source software provide the users source code freely and provide flexibility in order to adapt innovation in the software.	As closed source software do not give access to the source code which means that no one can modify the code or made innovations to the system. Source code is only accessed by the developer but now a days several closed source software customized software for specific user.
Transparency	The source code for the Open source software is freely available and the user can easily modify the source. In open source software internal structure of the software is visible to the user.	In Closed source software only interfaces are visible to the user they don't get access or visualize the internal structure of the software. User don't have the internal and other details of the software.
Standards	OSS uses open standard that can be modified by the intended user. Without open source standard it would be difficult to exchange information on the internet.	Most of the closed source software uses private standards that are not available publicly. It can only be controlled and modified by the team of organization who developed it.
Availability	Open Source software are freely available on the internet without any purchasing cost. Some of the OSS are available online 24x7.	They are available only from the company that developed the software that own all the rights of the products. Sometimes for closed source software free trials of 7 days or for a month are available freely.

6. Future Approaches to manage complexity of project

Complexity is the major factor affect the processing and performance of the software. So there is a need to manage the software complexity in order to deliver a valuable product to the intended customer.

There are different type of complexities and different type of metrics are used to manage and control them. Complexity in the design and architecture of the software and complexity in the source code of software. General complexity metrics used for the evaluation of complexity of software are process metrics and product metrices.

Process metrices depend upon the measurement of the properties of the process of development. Process metrics includes the cost, reusability and methodology of the metrics and also consider the time a project take in development and all the error found during the testing of the software.

Whereas product metrics includes the cost, size, time, usability, reliability, functionality and maintainability of software.The metrices used for the evaluation of the complexity are Quantitative metrics: depends on number of instruction per code and number of total function used in the source code, cyclomatic complexity measurement and Halstead metrices.

Metrices used for the complexity in the control flow of the software products are: boundary values metrics, McCabe metrics, Harrison and Magel metrics etc.

Now a days complexity of a software is managed using hybrid metrices by combing two or more metrices for a software.

7. Conclusion

Increase in the size of the software products and advancement in different factors of software development like technology and many other techniques used in the development of the software results in growth of the complexity in software. So there is a need to maintain and control these factors for the development of valuable software products.

In this paper various models/metrices and approaches are defined that helps in the maintenance of the complexity factors like size, cost, technology, reliability and standards of the Open Source and Private software. Although, these factors have different impacts on Open Source and Closed source software but for maintenance of the complexity same metrices are used. Now a days Open Source software are widely trending in the IT industry because of low complexity factors and will be enhanced in the future to make the Open source software more reliable and secure compare to Private software. In Future, to get the better results in term of performance and complexity it can be managed and controlled by using the hybrid metrices and approaches.

I. References

[1] Nguyen, Anh. (2017). The Impact of Software Complexity on Cost and Quality - A Comparative Analysis Between Open Source and PrivateSoftware. International Journal of Software Engineering & Applications. 8. 17-31. 10.5121/ijsea.2017.8202.

[2] A. P. Singh and P. Tomar, "The analysis of software metrics for design complexity and its impact on reusability," *2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom)*, New Delhi, 2016, pp. 3808-3812.

[3] Delgado, Andrea & Calegari, Daniel & Milanese, Pablo & Falcon, Renatta. (2015). A Systematic Approach for Evaluating BPM Systems: Case Studies on Open Source and PrivateTools. 451. 81-90. 10.1007/978-3-319-17837-0_8.

[4] Sacks, Michael. (2015). Competition Between Open Source and PrivateSoftware: Strategies for Survival. Journal of Management Information Systems. 32. 268-295. 10.1080/07421222.2015.1099391.

[5] H. Plate, S. E. Ponta and A. Sabetta, "Impact assessment for vulnerabilities in open-source software libraries," 2015 IEEE International Conference on Software Maintenance and Evolution (ICSME), Bremen, 2015, pp. 411-420

[6] Misra, Sanjay & Adewumi, Adewole & Omoregbe, Nicholas & Crawford, Broderick. (2016). A systematic literature review of open source software quality assessment models. SpringerPlus. 2016. 1936. 10.1186/s40064-016-3612-4.

[7] Fitsilis, Panos & Damasiotis, Vyron. (2015). Software Project's Complexity Measurement: A Case Study. Journal of Software Engineering and Applications. 08. 549-556. 10.4236/jsea.2015.810052

[8] Sood, G., Shipra, & Soni, R. (2016). Comparative Study: PrivateSoftware vs. Open Source Software.h

[9] Mwaro, J., Omwenga, J. & Kihonge, E. (2016). Effects of project complexity on project implementation: A case of orange money project at Telkom Kenya limited. International Academic Journal of Information Sciences and Project Management, 2 (1), 1-16

[10] W. Aslam and F. Ijaz, "A Quantitative Framework for Task Allocation in Distributed Agile Software Development," in *IEEE Access*, vol. 6, pp. 15380-15390, 2018. doi: 10.1109/ACCESS.2018.2803685

[11] Dashevskyi, Stanislav & Brucker, Achim & Massacci, Fabio. (2016). On the Security Cost of Using a Free and Open Source Component in a PrivateProduct. 9639. 190-206. 10.1007/978-3-31s9-30806-7_12.

[12] Hilton, M.C., Tunnell, T., Huang, K., Marinov, D., & Dig, D. (2016). Usage, costs, and benefits of continuous integration in open-source projects. *2016 31st IEEE/ACM International Conference on Automated Software Engineering (ASE)*, 426-437.

[13] Sidhu, Harmaninder & Khurmi, Sawtantar. (2017). Analysing Open Source Software in Terms of Its Characteristics and Establishing New Paradigms. International Journal of Information Technology and Computer Science. 9. 17-25. 10.5815/ijitcs.2017.07.02.

[14] A. Zolotas, H. H. Rodriguez, D. S. Kolovos, R. F. Paige and S. Hutchesson, "Bridging PrivateModelling and Open-Source Model Management Tools: The Case of PTC Integrity Modeller and Epsilon," *2017 ACM/IEEE 20th International Conference on Model Driven Engineering Languages and Systems (MODELS)*, Austin, TX, 2017, pp. 237-247.

[15] Cristóbal, José & Carral, Luis & Díaz, Emma & Fraguela, José & Iglesias, Gregorio. (2018). Complexity and Project Management: A General Overview. Complexity. 2018. 1-10. 10.1155/2018/4891286.

[16] Ahmed Majid Taha, Alaa Ahmed Abbood, Atheer Akram Abdul Razzaq and A.S. Al-Bahri, 2018. Identifying the Affecting Factors for Adoption of Open Source Software in it Community. Journal of Engineering and Applied Sciences, 13: 5771-5780.

[17] Pv, Divya & Thomas, Sam. (2018). Role of technological uncertainty, technical complexity, intuition and reflexivity in project planning a study on software development projects. International Journal of Project Organisation and Management. 10. 82. 10.1504/IJPOM.2018.10011409.

[18] Kochhar, Pavneet Singh & Kalliamvakou, Eirini & Nagappan, Nachiappan & Zimmermann, Thomas & Bird, Christian. (2019). Moving from Closed to Open Source: Observations from Six Transitioned Projects to GitHub. IEEE Transactions on Software Engineering. PP. 1-1. 10.1109/TSE.2019.2937025.

[19] Bakhshi, Javad & Ireland, Vernon & Gorod, Alex. (2016). Clarifying the project complexity construct: Past, present and future. International Journal of Project Management.34.1199-1213. 10.1016/j.ijproman.2016.002.