International Journal of Scientific Research & Engineering Trends
Volume 9, Issue 6, Nov-Dec-2023, ISSN (Online): 2395-566X

A Study and Analysis of Software Metrics Components

Research Scholar Sandhya, Professor Mukesh Kumar
NIILM University, Kaithal

Abstract- In software programming, Component-based programming is one of the most well-organized and dependable factor to improve software development capabilities. This kind of programming uses the existing components or program blocks to generate new programs. The reusable components not only speed up the development process but also increase the software's reliability. But this reliability and efficiency depend on the number of components used along with interfacing with new components. There is the requirement to study the complexity of the inclusion of these new components in the software system so that the complete software analysis will be performed. In this present paper, module component analysis and the integration analysis approach is done to analyze the software system. In this paper, a weighted approach is defined to perform the analysis and to identify the effectiveness of software reusability.

Index Terms- Software components, Software Metrics

I. INTRODUCTION

Today most of the available software systems are defined in modular form. These modules are defined in the form of a method or component. These components are being used in a software system as the essential software part based on which software complexity analysis can be performed (Da-Wei, E., 2007)

This kind of software system can be analyzed under different vectors such as module interconnectivity analysis and component reusability analysis. There is another requirement to analyze the requirement of software component reusability in a software system. This usability analysis also depends on multiple parameters such as the criticality of the components, Number of variables or methods being shared, interactivity with external or internal files etc. Based on these all vectors, software complexity analysis to perform reliable software delivery (Koh, T. W.et al., 2008)

1. Software Components

- A software module or component can be described along with specific properties.
- A Software module, block, function, or class can be a Software component
- These modules can be dependent on the language or can be neutral and generalized so that can be embedded in any language (Xiao, H. et al.,2009)
- The software components can be extended to generate the software product.
- These are defined conceptually with interface specifications so that the interconnectivity description can be provided (Selvarani, R. et al., 2009).
- The components can be some end product to other applications.
- The components can be freeware or the registered components.
- These components can be application or database-specific These components can be online or offline components (Lang, A. B. et al., 2021).
- A software component is an end product or it can be extensible.
- A software component is the part of interfacing that conceptually identifies internal and external interfacing with the main application (Abbas, M. et al., 2021).
- A deliverable software object can also be considered a software component (Alvi, A. K., &Zulkernine, M., 2021).
- Any online or offline product or code can be a software component (Sandhu, A. K., &Batth, R. S.,2021; Aggarwal, A.et al.,2021)

The above-mentioned points about software components must be supported by all the available languages

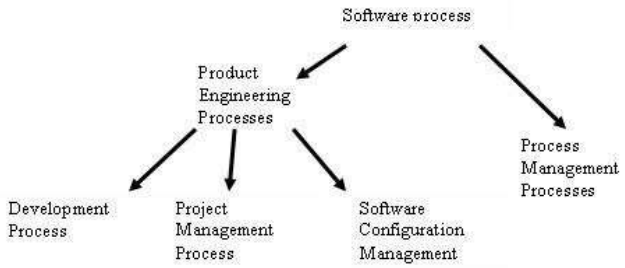

Fig1. Software Components

2. Software Metrics

As the number of components available on the market increases, it is becoming more important to devise software metrics to quantify the various characteristics of components and their usage. Software metrics are intended to measure the software quality and performance characteristics quantitatively, encountered during the planning and execution of software development. These can serve as measures of software products for the purpose of comparison, cost estimation, fault prediction, and forecasting. Metrics can also be used in guiding decisions throughout the life cycle, determining whether software quality improvement initiatives are financially worthwhile (Sedigh et al., 2001). A lot of research has been conducted on software metrics and their applications. Most of the metrics proposed in the literature are based on the source code of the application. However, these metrics cannot be applied to components and component-based systems as the source code of the components is not available to application developers. Therefore, a different set of metrics is required to measure various aspects for component-based systems and their quality issues.

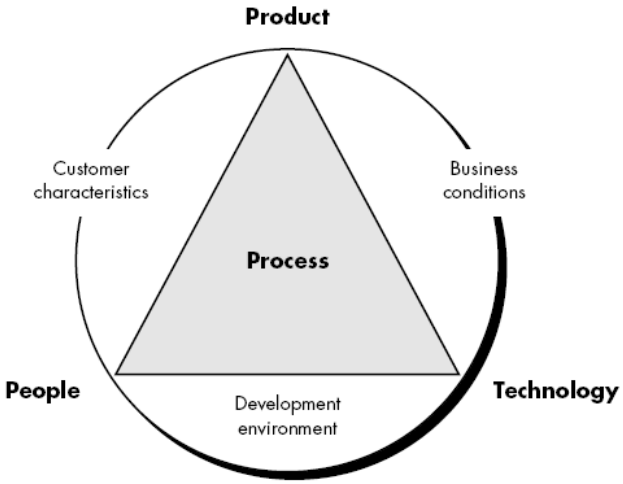

Fig2. Software Metrics

II. RELATED WORK

Prasad, L., & Nagar, A., 2009 has defined work on the analysis of the software system for different structural and object-oriented metrics. This metric-based component estimation along with relationship analysis is performed under the procedure-based analysis. The authors discussed the metrics such as LOC, cyclomatic complexity, cohesion, and coupling metrics and also defined the analysis under the class level analysis so that the software product error detection and correction can be performed over the system. Estimation of the software system under operational measures is also performed to analyze the software quality. The operational features of the project and product are also discussed along with coupling analysis so that the estimation of software-related metrics will be obtained. The paper has also discussed the software coupling measurement under structural analysis.

Rana, Z. A. et al., 2009 has defined estimation on software products to analyze the software system under defect analysis for an object-oriented software system. The author defined the work to perform the defect prediction in the software system and analyze the software system effectiveness under module-level defect analysis. The defects are analyzed under different vectors such as a number of defects. The author defined the coupling-based analysis, association analysis, dependency analysis, and interface analysis. The class and inheritance analysis is performed to analyze the software product effectively. The UML diagrams are constructed to establish the relationship between classes. This kind of software system is also able to present the static view of the system so that the decision regarding the software quality can be done. Software system analysis is performed under the complexity metrics-based analysis.

Boehm, B. W., 1988 presented a resource-based software estimation scheme for software quality analysis. Author-defined budget analysis approach to improve the software product analysis. The author performed analysis under different testing aspects.

Da-Wei, E., 2007 has defined an improved metrics-based complexity model for object- oriented programming. The author defined the complexity analysis under multiple aspects so that effective software development under method analysis will be performed. The author defined the analysis under the software complexity and method analysis so that the software design model will be improved. The author defined the development process with the specification of the cost model under the size and volume analysis so that the prediction to the software system will be done effectively.

Shatnawi, R., 2010 proposed the structural complexity model under integral factor so that the development effort will be reduced and also proposed a size and complexity-based model for the development of software systems under cost estimation. He also worked on the object- oriented program so that effective software development will be done as well as a predictive software analysis under quality modeling so that the software system analysis will be done.

Krishnapriya, V. &Ramar, K., 2010 worked on an effective software evaluation and measurement for software systems under traditional metrics analysis. The author defined a metrics suite with coupling, cohesion so that the software cost estimation will be done effectively.

Krishnapriya, V., &Ramar, K., 2010 proposed and defined a computational system for the software system under the

software development rules for cost, timeline, and quality analysis. The author defined the software development under software modeling and analysis and also defined a professional practice analysis and model for the development process so that an effective development model will be presented.

Kulkarni, U. L., et al., 2010 developed a computational work to perform the software development and software product analysis under cost and time analysis was done by the researcher. Here all works are done to define a parametric analysis on software system under software quality and software history analysis.

Du, Q., & Wang, F., 2010 developed the software process and software design mechanism to analyze the software system under software evolution and software management analysis. Work done also an emphasis on the specific development model so that the development process will be improved and to define a set of management approaches to analyze the software system under software quality analysis. The work defined the management analysis program under the specification of research analysis and development by validating the software management under software improvement.

Du, Q., & Wang, F., 2010 analyzed the software measurement under the defined framework so that the software measurement validation is performed along with the structural model for software development so that the attribute relation analysis will be performed.

Chen, J. et al., 2011 presented the entity analysis so that the software development and validation will be performed and the software measurement-based program to analyze the software system under rule specification and to analyze the program under different metrics. 2.11 Thapaliyal, M., & Verma, G., 2010 analyzed the effective path generation so that effective software measurement and testing will be applied over it.

III. PROBLEM FORMULATION

Risk is one of the important factors to identify the criticality of a software project. Risk is the important criterion in a software plan while performing software cost analysis and software project scheduling. There is a number of factors that help to analyze the software risk such as availability of resources, prediction vector, software faults, etc. Another important concern of software systems is software reusability. Software reusability is about designing a software system by using some existing software or the module. In this work, we are combining the software reusability vector with software risk management. As a known concept, if a software is designed using some existing modules or libraries, the risk vector is reduced. The presented work is divided into four main stages. During the first stage, software analysis will be done under the metric-based estimation. This estimation includes the software analysis under the individual module analysis, module interaction analysis, and software system analysis. In the second stage, the module requirement will be defined to represent reusable modules.

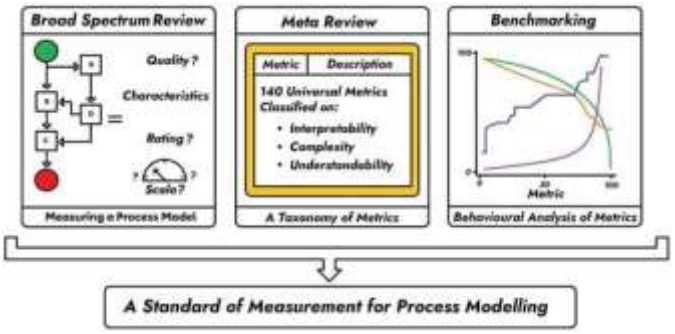

Figure 3: Software Process Flow

The metric-based estimation will be done with these modules under different metrics in terms of individual module analysis, module requirement analysis, and module interaction analysis. In the third stage, the inclusion of new modules in the software system will be done along with the interactivity definition of reused modules.

After setting up the interaction, analysis of the complete system will be done in terms of module interaction analysis, individual module analysis and system analysis. In the final stage, all representations of reusable modules will get identified in terms of cost estimation with the inclusion and testing of the reused modules will be performed. Based on this analysis, the system cost and the risk estimation will be identified and presented as the final result.

The presented work estimates the software risk as well as software risk by analyzing existing software metric analysis and its inclusion with new software modules. The presented work is divided into four main modules. The work begins with the selection of a complete software system. These stages are defined in figure 1. As shown in the figure, the first stage of the work is the estimation of the existing software system under the metric-based analysis.

This analysis work will be divided into three main parts. The parts are the evaluation of individual software modules, module interaction analysis, and the complete system analysis. This stage will be able to perform the software representation in terms of software statistical analysis. In the second stage, the software system will be analyzed respectively to the new software system with which, the existing software system will be integrated.

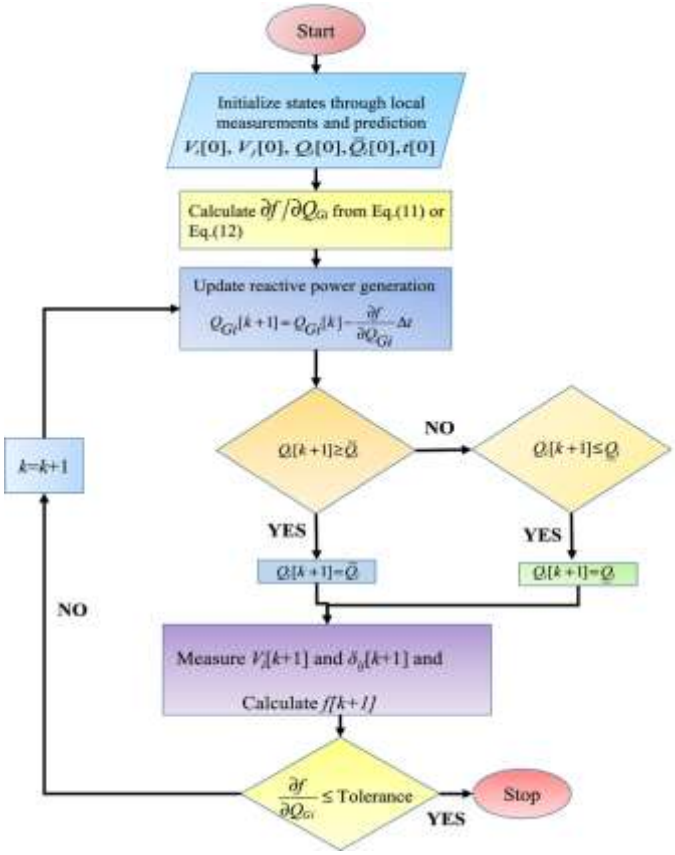

Figure 4: Algorithmic flow of the proposed work

The identification of the existing software modules will be identified and these modules will be considered reusable modules. The estimation of these modules will be done under different metrics. In the third stage, the actual integration will be done in terms of new module inter-connectivity and the structural design of the newer software system will be done. In four stages, the estimation of the software system cost, reusability cost, and the software risk will be identified.

IV. CONCLUSION

Software metrics are measurable characteristics of a software program that can be used to evaluate and improve software quality. They can be used to:

- **Plan and Forecast:** Provide a quantitative basis for software development planning and forecasting
- **Monitor Quality:** Provide a way to monitor and improve software quality
- **Manage Projects:** Help identify, prioritize, and track issues to improve team productivity
- **Communicate Status:** Help development teams communicate the status of projects and address issues
- **Estimate Costs:** Provide information for estimating costs
- **Manage Production:** Help manage the production process by providing visibility into numbers and trends

REFERENCES

1. Sedigh-Ali, S., Ghafoor, A., & Paul, R. A. (2001). Software engineering metrics for COTS- based systems. Computer, 34(5), 44-50.
2. Prasad, L., & Nagar, A. (2009, July). Experimental analysis of different metrics (object- oriented and structural) of software. In 2009 First International Conference on Computational Intelligence, Communication Systems and Networks (pp. 235-240). IEEE.
3. Rana, Z. A., Shamail, S., &Awais, M. M. (2009, May). Ineffectiveness of use of software science metrics as predictors of defects in object oriented software. In 2009 Wri World Congress on Software Engineering (Vol. 4, pp. 3-7). IEEE.
4. Boehm, B. W. (1988). A spiral model of software development and enhancement. Computer, 21(5), 61-72.
5. Da-Wei, E. (2007, April). The software complexity model and metrics for object-oriented. In 2007 International Workshop on Anti-Counterfeiting, Security and Identification (ASID) (pp. 464-469). IEEE
6. Shatnawi, R. (2010). A quantitative investigation of the acceptable risk levels of object- oriented metrics in open-source systems. IEEE Transactions on software engineering, 36(2), 216-225.
7. Krishnapriya, V., &Ramar, K. (2010, June). Exploring the difference between object oriented class inheritance and interfaces using coupling measures. In 2010 International Conference on Advances in Computer Engineering (pp. 207-211). IEEE.
8. Kulkarni, U. L., Kalshetty, Y. R., &Arde, V. G. (2010, November). Validation of ck metrics for object oriented design measurement. In 2010 3rd international conference on emerging trends in engineering and technology (pp. 646-651). IEEE.
9. Du, Q., & Wang, F. (2010, December). Software Power: a new approach to software complexity metrics. In 2010 Second World Congress on Software Engineering (Vol. 2, pp. 165-168). IEEE.
10. Chen, J., Wang, H., Zhou, Y., &Bruda, S. D. (2011). Complexity metrics for component- based software systems. International Journal of Digital Content Technology and its Applications, 5(3), 235-244.
11. Thapaliyal, M., & Verma, G. (2010). Software defects and object-oriented metrics-an empirical analysis. International Journal of Computer Applications, 9(5), 41-44.
12. Da-Wei, E. (2007, April). The software complexity model and metrics for object-oriented. In 2007 International Workshop on Anti-Counterfeiting, Security and Identification (ASID) (pp. 464-469). IEEE.

13. Koh, T. W., Selamat, M. H., Ghani, A. A. A., & Abdullah, R. (2008). Review of complexity metrics for object oriented software products. International Journal of Computer Science and Network Security, 8(11), 314-320.
14. Xiao, H., Li, S., & Wang, B. (2009, March). A tool for the application of software metrics to UML class diagram. In 2009 First International Workshop on Education Technology and Computer Science (Vol. 1, pp. 181-184). IEEE.
15. Selvarani, R., Nair, T. G., & Prasad, V. K. (2009, May). Estimation of defect proneness using design complexity measurements in object-oriented software. In 2009 International Conference on Signal Processing Systems (pp. 766-770). IEEE. 16.Abbas, M., Ferrari, A., Shatnawi, A., Enoiu, E. P., &Saadatmand, M. (2021). Is Requirements Similarity a Good Proxy for Software Similarity? An Empirical Investigation in Industry.In REFSQ (pp. 3-18).
16. Lang, A. B., Debenham, C. J., & DeLaurentis, D. A. (2021). Enabling reusability of a spacecraft design toolset via MBSE. In AIAA Scitech 2021 Forum (p. 0095).
17. Alvi, A. K., &Zulkernine, M. (2021). A security pattern detection framework for building more secure software. Journal of Systems and Software, 171, 110838.
18. Sandhu, A. K., &Batth, R. S. (2021). Software reuse analytics using integrated random forest and gradient boosting machine learning algorithm. Software: Practice and Experience, 51(4), 735-747.
19. Aggarwal, A., Dhindsa, K. S., & Suri, P. K. (2020). An Empirical Evaluation of Assorted Risk Management Models and Frameworks in Software Development. International Journal of Applied Evolutionary Computation (IJAEC), 11(1), 52-62.
20. Aggarwal, A., Dhindsa, K. S., & Suri, P. K. (2020). Design for Software Risk Management Using Soft Computing and Simulated Biological Approach. International Journal of Security and Privacy in Pervasive Computing (IJSPPC), 12(2), 44-54.
21. Aggarwal, A., Dhindsa, K. S., & Suri, P. K. (2020). A Pragmatic Assessment of Approaches and Paradigms in Software Risk Management Frameworks. International Journal of Natural Computing Research (IJNCR), 9(1), 13-26.
22. Escalated Methods for Software Defects Audit in Repercussion and Effects Construe to Nature Inspired and Behavior Driven Mechanisms. (2019). Regular Issue, 8(6), 1779–1783. doi:10.35940/ijeat. f8442.088619
23. Usage Patterns and Implementation of Random Forest Methods for Software Risk and Bugs Predictions. (2019). Special Issue, 8(9S), 927–932. doi:10.35940/ijitee. i1150.0789s19 [25].Aggarwal, A., Dhindsa, K. S., & Suri, P. K. (2021). Performance-Aware Approach for Software Risk Management Using Random Forest Algorithm. International Journal of Software Innovation (IJSI), 9(1), 12-19.