An implementation Analysis of Risk Mitigation in Software Reusability using Matrix Approach

Alankrita Aggarwal[1], Kanwalvir S. Dhindsa[2], P. K. Suri[3]and Pardeep Singh[4]

[1,2] *IKGPTU, Kapurthala,Jalandhar-Kapurthala, Highway, Vpo Ibban, Punjab 144603, India*
[3]*Kurukshetra University,Thanesar, Kurukshtera, Haryana 136119, India*
[4]*University of Petroleum and Energy Studies, Energy Acres, UPES, Bidholi, via, Prem Nagar, Dehradun, Uttarakhand-248007, India*

Abstract
Component-based programming is one of the most efficient and reliable parameters to improve software development capabilities. 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. In this work, a weighted approach is defined to perform the analysis and to identify the effectiveness and risk minimization of software reusability and done using Matlab simulations.

Keywords
Software Components, Risk Mitigation, Matrix analysis, Complexity Analysis

1. 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. This usability analysis also depends on multiple parameters such as the criticality of the components, the 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 [1][2].

1.1SoftwareComponents

A software module or component can be described along with specific properties

- A Software module, the block, function, or the 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
- These components can be application or database-specific these components can be an online or offline component. An end product or it can be if extensible can be considered a software component.
- An interface that conceptually identifies internal and external interface with the main application will be also a part of a software component

International Conference on Emerging Technologies: AI, IoT, and CPS for Science & Technology Applications, September 06–07, 2021, NITTTR Chandigarh, India
EMAIL:alankrita.agg@gmail.com (A.1); kdhindsa@gmail.com (A. 2); pksurikuk@gmail@mail.com (A. 3),pardeep.maan@gmail.com(A.4)
ORCID: 0000-0002-0931-1118 (A. 1); 0000-0002-7911-9244(A. 2); 0000-0002-0368-4757(A.4)

- A deliverable software object can also be considered a software component. The above-mentioned points about the software components must be supported by all the available languages [3][4]

1.2 SoftwareMetrics

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. Metrics can also be used in guiding decisions throughout the life cycle, determining whether software quality improvement initiatives are financially worthwhile. Therefore, a different set of metrics is required to measure various aspects for component-based systems and their quality issues. [5][6]

2. Literature Review

Defined work on the analysis of the software system for different structural and object-oriented metrics. The authors discussed the metrics such as LOC, cyclomatic complexity, cohesion, and coupling.

- Authors proposed estimation on software products to analyze the software system under defect analysis for an object-oriented software system [7].
- Presented a resource-based software estimation scheme for software quality analysis. The author defined a budget analysis approach to improve software product analysis and also performed analysis under different testing aspects.[8]
- The proposed work introduced an improved metrics-based complexity model for object-oriented programming and the complexity analysis under multiple aspects so that effective software development under method analysis will be performed [9].
- Proposed the structural complexity model under an integral factor so that the development effort will be reduced and also proposed the size and complexity-based model for the development of software systems under cost estimation [10].
- Proposed and defined a computational system for the software system under the software development rules for cost, timeline, and quality analysis [11].
- The idea developed computational work to perform software development and software product analysis under cost and time analysis. Here all works are done to define a parametric analysis on software system under software quality and software history analysis [12][13].
- The work 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 [14].
- Authors presented the entity analysis of the components of the software and analyzed the effective path generation so that effective software measurement and testing will be applied over it [15][16].
- The proposed work explained the reuse of software by using combining RF and gradient boosting machine learning methods [17].
- Proposed to minimize the risk by using Random ForestAlgorithm [18][19].
- Explained the methods to increase the quality of software by bugs detection and prediction methods[20[21][22].

3. Proposed Work

Risk mitigation is the important criterion in a software plan while performing software cost analysis and software project scheduling. 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. The presented work is divided into four main stages. During the first stage, software analysis will be done under the metric-based estimation. In the second stage; the module requirement will be defined to represent reusable modules. In the final stage, all representations of reusable modules will get identified in terms of cost estimation with the inclusion and testing of 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 work begins with the selection of a complete software system. These stages are defined in figure 1. As shown in the figure.

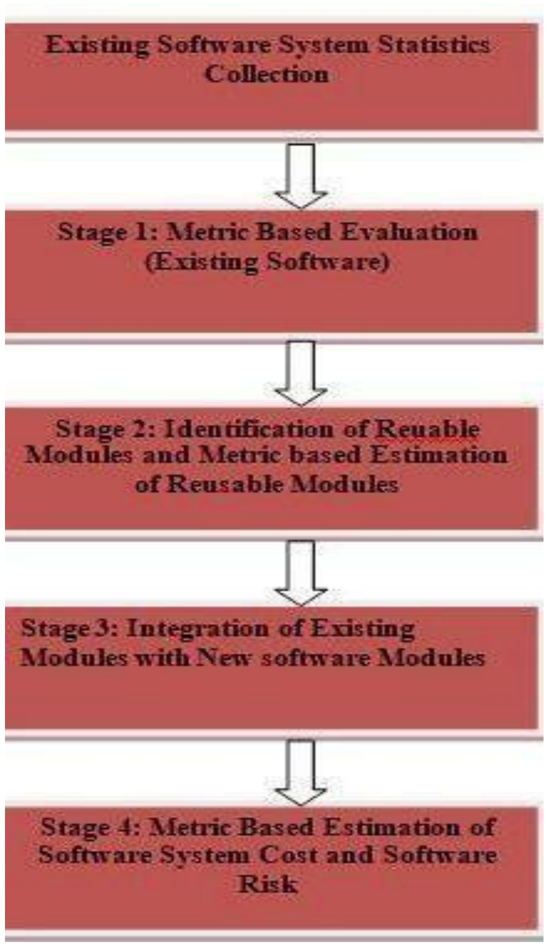
Existing Software System Statistics Collection
Stage 1: Metric Based Evaluation (Existing Software)
Stage 2: Identification of Reuable Modules and Metric based Estimation of Reusable Modules
Stage 3: Integration of Existing Modules with New software Modules
Stage 4: Metric Based Estimation of Software System Cost and Software Risk

Figure 1: Example figure

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.

3. Results & Discussion

The presented work is implemented in a Matlab environment to the process of data analysis from obtaining data from external sources and databases, doing pre-processing, related visualization, numerical analysis to produce quality output presentation tested on a dummy.

4. Conclusion & Future Scope

In this work, system complexity analysis is defined respective to software usability under different parameters such as inter-communication analysis, and risk is minimized. The work can be analyzed with the help of various machine learning methods which will be giving promising results rather than traditional methods. The paper applies to the industry as the component-based

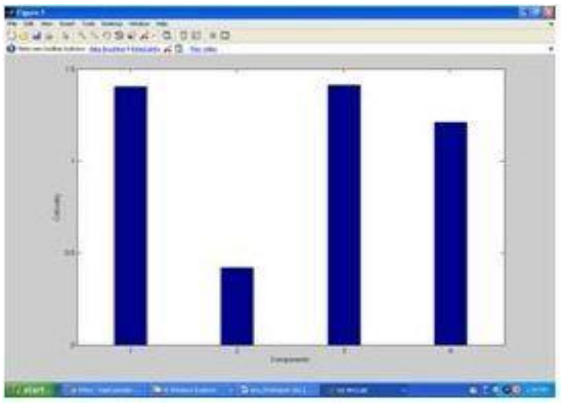

Figure 2: Individual Module Analysis

risk mitigation will minimize the risk and will be useful to the industry as well. module-based project representation. The results obtained from the work are given here under.Here figure 2 the individual. Module analysis dependent on module interface complexity is shown. Here x-axis is representing the modules and the y-axis is representing complexity analysis.In figure 3 the internal component analysis respective to modules is shown. Here x-axis is representing the modules and the y-axis is representing complexity analysis.

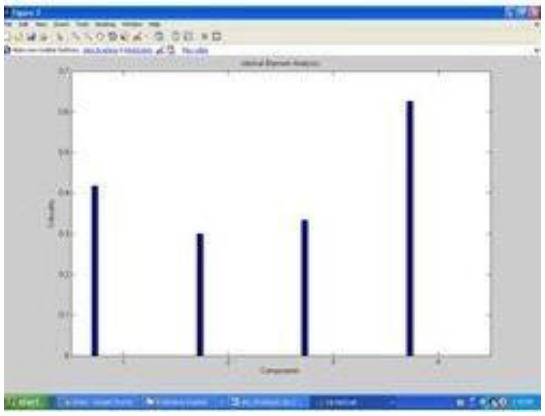

Figure3: Internal Component Analysis

5. Acknowledgment

We are thankful to the research department of IKG Punjab Technical University, Kapurthala (Punjab) for their assistance. No other person or organization is associated with our work in this manuscript.

6. 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] Lang, A. B., Debenham, C. J., &DeLaurentis, D. A. (2021). Enabling reusability of a spacecraftdesigntoolset via MBSE. In *AIAA Scitech 2021 Forum* (p. 0095).

[17] 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.

[18] 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.

[19] Aggarwal, Alankrita, Kanwalvir Singh Dhindsa, and P. K. Suri. "Enhancing Software Quality Assurance by Using Knowledge Discovery and Bug Prediction Techniques." In *Soft Computing for Intelligent Systems*, pp. 97-118. Springer, Singapore,2021.

[20] Hasan, M. Mahmudul, George Kousiouris, DimosthenisAnagnostopoulos, TetaStamati, Peri zoucopoulos, and Mara Nikolaidou. "CISMET: A Semantic Ontology Framework for Regulatory-Requirements-Compliant Information Systems Development and Its Application in the GDPR Case." *International Journal on Semantic Web and Information Systems (IJSWIS)* 17, no. 1 (2021):1- 24.

[21] Abayomi-Alli, Adebayo Adewumi, Sanjay Misra, MulkahOpeyemiAkala, Abiodun MotunrayoIkotun, and BolanleAdefowokeOjokoh. "An Ontology-Based Information Extraction System for Organic Farming." *International Journal on Semantic Web and Information Systems (IJSWIS)* 17, no. 2 (2021): 79-99.

[22] DK ShivaniGaba, ShifaliSingla.A Genetic Improved Quantum Cryptography Model to Optimize Network Communication," Special Issue, vol. 8, no. 9S, pp. 256–259, Aug. 2019.