International Journal of Innovative Technology and Exploring Engineering (IJITEE)
ISSN: 2278-3075 (Online), Volume-9 Issue-5, March 2020

Automated Software Design Reusability using a Unique Machine Learning Technique

P. Mangayarkarasi

Abstract: ***The era of machine learning (ML) has brought significant advancement into the traditional approaches of software development and services. Software reusability and design automation is a key requirement that can be handled through the integration of artificial intelligence (AI) capabilities with the traditional approach of software development lifecycle (SDLC) practices. The study introduces a novel approach of ML, which can assist inappropriate selection of reusable software components, which in the long run, can optimize the operational cost in the context of development practices and also speed up the service delivery performance of software engineering activities. The proposed model is validated through a numerical analysis that shows the effectiveness of the system in terms of both classification accuracy and computational efficiency.***

Keywords : Machine Learning, Software Reusability, Supervised Learning Model, Computational Complexity.

I. INTRODUCTION

'***Software reusability***' is a popular approach for many years in various software development sectors, and the standards are being followed to develop and upgrade their respective software design frameworks. It is considered an important and essential prerequisite to optimizing the design in terms of both time and cost during the entire development lifecycle. And also, in the longer run, it escalates the overall productivity, and reliability of operations of each associative component significantly improves the quality of service delivery models [1] [2]. However, the conventional trends of evolving research approaches in the domain of software engineering and research bring out the possibility to automate the design reusability approach for different components where the scope of machine learning (ML) is found to optimize the development cost with lesser incorporation of human-assisted efforts [3]. As in the context of ML, the machine itself is programmed in a way where it learns from the data corresponding to the existing repository of reusable software components. Hence it reduces the risk factor in the design of reusable software components, which enhances the scope of applicability of the software in various use-cases.

Revised Manuscript Received on March 30, 2020.
* Correspondence Author
Dr. P. Mangayarkarasi*, Associate Professor, Department of Information Science and Engineering, New Horizon College of Engineering, Email: mangaivtu@gmail.com

There exist various factors that lead to reducing the uncertainty factor associated with an overall cost for the software component development and deployment modules if software components are reused to build a new component for the further development practices and use-cases It is found that learning modules can be integrated into the software design and reusable components to reduce the human-efforts and requirements of resources for coding tasks. This way involvement of many resources to reuse the same software component for different use-cases can be minimized, and it leads to efficient time and resource utilization, which is an essential factor to avoid the situation called as software crisis to a greater extent [4][5][6]. An extensive research effort utilizes different unsupervised and supervised ML approaches to determine the predictive modelings where the classifiers are capable of suggesting whether any particular software component is reusable or not in advance to save the time and cost factors from the viewpoint of operational management [7][8]. There exist a series of classifiers and learning models which can achieve better accuracy in predicting the appropriate reusable software component for specific problem so that the solution approach can be instantaneous and reliable, but most of the techniques lack efficiency in terms of both classification accuracy and complexity which restrict the applicability of software design reusability from many use-cases and also limits the design features of software components during the development lifecycle. The study introduces a machine learning approach to automate the software reusability task without the intervention of external effort to select the proper reusable components. This approach speeds up the production speed and, at the same time, reduces the cost of development and validation procedures, which in longer run found scales up the service delivery models. The formulated approach incorporates a supervised classifier to perform intelligent knowledge discovery from the data corresponds to the reusable software repository (SRR), which get updated simultaneously during the software development lifecycle (SDLC) phase. The outcome of the study shows that the system offers better classification accuracy towards the selection of the appropriate reusable component using intelligent knowledge discovery, and the system envisions for the process maturity at a lower cost of computation within finite steps of implantation which also reduces the overall operational cost.

- *Software reusability:* There exists a variant range of software components from both functional and behavioral aspects of designs. In every type of software component, the functionalities are in-built or can be realized through procedural changes in the subroutines. The objects in the context of object-oriented programming (OOPs) can be utilized for design reusability. However, although there are various existing potential components that can be reused for different use-cases in many cases, it is difficult to measure the degree of reusability with external human effort and software practices, which is often found time and effort consuming task and involve more cost of implementation. Thereby, if between two software component modules connectivity and similarities can be obtained, then it becomes easier to decide the reusability tasks. The common definition of reusability can be given as- The notion of reusability is to decide the appropriate functions or subroutines of software component modules which are available for a specific set of problem and can ensure better solution if applied to a new problem. The following figure shows the common steps which are involved in engineering the procedure of software reusability practices for various use-cases[9][10].

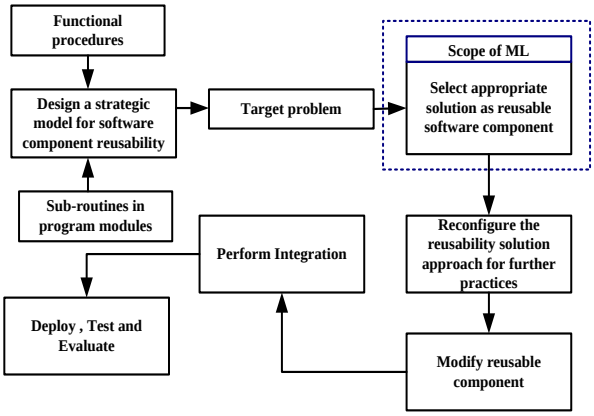

Figure 1 Common step in the approach of software component reusability process during SDLC

Figure 1 exhibits that during SDLC to enhance the productivity in software development practices with the lower cost of implementation and functional operations, the selection of appropriate reusable components is a key-perquisite. Thereby the system has a scope of integration with machine learning (ML) approaches where from the data itself, the system can learn and automate the software reusability task for effective development practices. The consecutive section further shows the related works carried out in the field of software engineering, where the prime focus has been laid towards software reusability practices and design automation using the ML-based approaches [11].

II. LITERATURE REVIEW

Recent works in the field of ML have envisioned constructing interactive optimized programming environments to enhance the software development practices from the perspective of product reusability. A significant work of Walter L. Hill, [10], suggested learning-based computational approaches where specifically the emphasis has been given to deal with the explanation based generalization problems. It also demonstrates the fact that explanation oriented learning paradigm can be subjected to simplifying the program abstractions, which in the long run, promote the software component reuse practices. A learning approach of software reuse is designed, which targets to accomplish two novel specific computational tasks.

i) The learning assists the machine in acquiring specific knowledge about programming and

ii) It also generalizes the programming theory from the viewpoint of abstract data types where interconnections can be made with an explanation based learning approach.

The theoretical approach defines a new shift to the conventional practices of software development based on explanation-based learning to explore the connectivity among various abstract data types modules. It is long-run to assist in enhancing the software component reusability practices [10]. Further, the trend of research evolved towards the mining of software engineering data, which means the approaches are mostly intended to perform knowledge discovery from the data itself. The study of (Wangoo 2018) explored various artificial intelligence (AI) techniques. It concluded their remarks as- AI can improve the mining operations and can pave better direction inefficient reuse of software design components [11]. Similar sort of studies also performed by (Feldt, R., Neto, F. G., & Torkar, R,2018) [12], Hassan, Ahmed E., and Tao Xie [13] where software data mining approaches and their potential to automate the software reusability practices are illustrated from both the theoretical and analytical viewpoint. (Mohammadian M, 2010) [14] introduces a theoretical discussion to outline various AI in the domain of software engineering and also provides a better insight into their respective application areas where efficient reuse of software components with cost-optimized service delivery models can be observed. The study of (Harman, Mark , 2012) [15]also emphasized towards analysing the impact of AI in business intelligence and software design reusability , in similar direction of research goal , other research approaches of (Meziane, Farid, and Sunil Vadera,2012) [16] , (Ammar et al. , 2012) [17] also worked and stated that there exist various ML approaches in the context of data mining such as classification approaches by support vector machine (SVM) , decision tree , K-nearest neighbours (KNN) , neural networks (NN) etc. which are efficient when the software component evaluation and reuse is concerned. However, most of the traditional approaches do not emphasize more in software modeling design pattern exploration perspective from the OOPs viewpoint, and studies, as shown in the following table 1, show that object-oriented design pattern analysis can reveal more insight about the software component and that significantly reduces the effort in software reuse practice during SDLC.

Table 1 shows that object-oriented design pattern analysis can reveal more insight about software component

Authors	Problem context	Approach
(R. Ferenc, J. Gustafsson,2002)[18]	Recognizing design patterns in software components	Columbus assisted OOPs approach
(E. Gamma et al.,1995)[19]	Design pattern analysis for software	Object-oriented approach
Y.-G. Gu'eh'eneuc et al. (2004)[20]	Design pattern in fingerprinting	Object-oriented approach

A. Research Gap Analysis

From the critical review of the existing trend of research draws the attention in many aspects such as- Even though data mining for knowledge discovery is applied in software engineering aspects, but that does not much ensure the scope of design pattern reusability with definite computational steps. It is also observed that mostly the learning approaches are iterative and computationally expensive, which results in higher false-positive rates while selecting the appropriate design component to be reused with predictor modeling. The study thereby addresses this design limitation of existing approaches and incorporates OOPs based design pattern analysis by means of two key ML approaches such as i) Neural network-based training and classification and ii) Decision tree-based approach.

III. FORMULATED APPROACH: CONCEPTUAL THEORY

The proposed approach performs an efficient software design pattern analysis. It integrates the modules with the ML approaches of i) Decision Tree and ii) Neural Network modules to predict the more accurate and appropriate object of software design pattern, which can assist in effective reuse of the package. The system is designed from an analytical viewpoint and taken the work of [21] as a baseline approach. It performs a reverse engineering process to identify the design pattern in source code objects (sCo) using a graph-based representation and modeling. The framework also borrowed the Columbus module to analyze the design pattern of software components using the mining approach. The theoretical modeling in this context of pattern analysis of software component design initially applies an object-oriented specific strategic schematic approach where inheritable factor (IF) for different strategic interfaces plays a crucial role. The strategy design pattern (SDP) involves three prime attributes, such as ***SDP***-I, ***SDP***-2, and ***SDP***-3, which are interconnected through a common ***algo***(I): Interface design. The design pattern analysis for ***algo***(I) is performed through IF and overriding, but still, to reduce the false-positive rates, a detailed learning-based approach is highly recommended. The following flow design shows the overall process flow overview of the formulated approach from the viewpoint of high-level design.

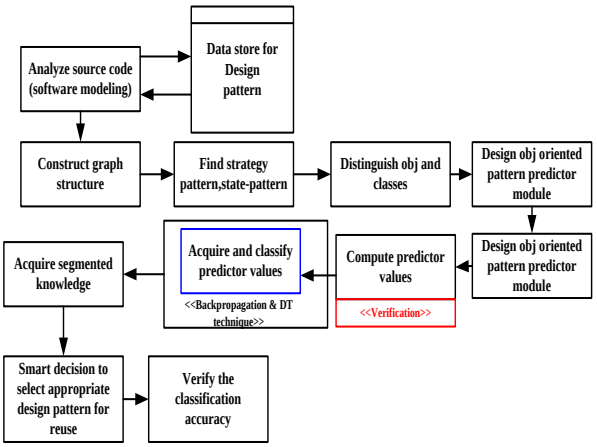

Figure 2 Working flow of the system

Figure 2 clearly shows the working flow, which is constructed to make the proposed system functional from the pattern design analysis viewpoint. It shows that the process adopted graph-based modeling to select the state of strategy for the software component and also distinguish the appropriate object (Obj) and classes. Further, Obj oriented predictor module computes the predictor values which are subjected for classification using the neural network back propagation approach [22] and decision tree approach (DT) [23]. The extracted knowledge laso assist the model in selecting appropriate design pattern, which can be reused for the further development of software modules, with the optimized cost of deployment and operations. The numerical analysis further shows that the system attains higher classification accuracy with lower false positive (FP) cases during the classification of predictor values.

IV. EXPERIMENTAL RESULT

The experimental analysis shows that the formulated approach in the case of DT based analysis attains a processing speed of 3.0056 sec where in the case of neural networks (NN) based approach, the processing speed is quite higher that is 4.568 sec. The following figure shows the outcome obtained for classification accuracy using these two approaches during the predictor value classification, which is comparable with the outcome of [21].

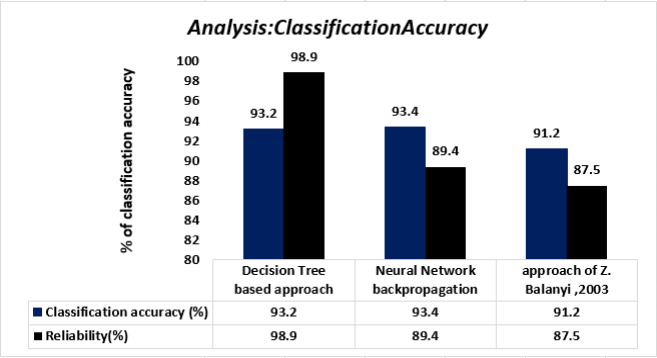

Figure 3 Outcome of the classification accuracy

Figure 3 shows that the formulated framework of DT and NN in the context of ML attains better classification accuracy as

compared to the approach of [21] and also accomplishes very lesser processing time, which reduces the computational effort from the cost viewpoint. Thereby the formulated solution must be quite effective for design pattern analysis, and the schematic learning model can select appropriate reusable features in software component design during the SDLC phase, which in the long run reduces the operational cost and external efforts of coders.

V. CONCLUSION AND FUTURE RESEARCH DIRECTION

The study introduces an ML-based approach to effectively reduce the effort in software component reuse from the perspective of design pattern learning and analysis viewpoint. The proposed design modeling in the form of conceptual theory is presented, and the numerical outcome of the study also simulated with supervised ML approaches. The outcome of the study shows that the approach is quite effective in terms of both classification accuracy and processing time as compared to the existing baseline. It also shows the learning precision of the formulated approach is quite higher, which is ~96% overall.

REFERENCES

1. Poulin, J.S., 1994, November. Measuring software reusability. In Proceedings of 1994 3rd International Conference on Software Reuse (pp. 126-138). IEEE.
2. Gill, N.S., 2006. Importance of software component characterization for better software reusability. ACM SIGSOFT Software Engineering Notes, 31(1), pp.1-3.
3. Wangoo, D.P., 2018, December. Artificial Intelligence Techniques in Software Engineering for Automated Software Reuse and Design. In 2018 4th International Conference on Computing Communication and Automation (ICCCA) (pp. 1-4). IEEE.
4. Shull, Forrest J., and Victor R. Basili. "Developing techniques for using software documents: a series of empirical studies." PhD diss., research directed by Dept. of Computer Science .University of Maryland, College Park, Md., 1998.
5. H.DIETER ROMBACH, VICTOR R. BASILI, RICHARD W. SELBY, "EXPERIMENTAL SOFTWARE ENGINEERING ISSUES: CRITICAL ASSESSMENT AND FUTURE DIRECTIONS. INTERNATIONAL WORKSHOP, DAGSTUHL CASTLE, GERMANY, SEPTEMBER 14-18, 1992. PROCEEDINGS", SPRINGER SCIENCE & BUSINESS MEDIA, PP. 261, 1993
6. William B. Frakes, Christopher J. Fox:Strength and similarity of affix removal stemming algorithms. SIGIR Forum 37(1): 26-30 (2003)
7. Flynn, Barbara B., Baofeng Huo, and Xiande Zhao. "The impact of supply chain integration on performance: A contingency and configuration approach." Journal of operations management 28, no. 1 (2010): 58-71.
8. P.Mangayarkarasi, R.Selvarani, "Reviewing the Significance of Software Metrics for Ensuring Design Reusability in Software Engineering", in International Journal of Computer Science & Communication Networks, IJCSCN,Vol 4(6), and ISSN: 208-213, pp. 208-213, 2015
9. P.Mangayarkarasi, R.Selvarani, "A Dynamic Optimization Technique for Redesigning OO Software for Reusability", ACM SIGSOFT Software Engineering Notes, Vol. 40, No. 2, pp.1-6, 2015
10. R.Selvarani, P.Mangayarkarasi, "oDyRM: Optimized Dynamic Reusability Model for Enhanced Software Consistency", (IJACSA) International Journal of Advanced Computer Science and Applications, Vol. 8, No. 3, 2017
11. R.Selvarani, P.Mangayarkarasi, "Modeling of Reusability Estimation in Software Design with External Constraints. In Software Project Management for Distributed Computing, Springer International Publishing, pp. 3-23, 2017
12. Feldt, R., Neto, F. G., & Torkar, R. "Ways of Applying Artificial Intelligence in Software Engineering". arXiv preprint arXiv:1802.02033. 2018
13. Hassan, Ahmed E., and Tao Xie. "Software intelligence: the future of mining software engineering data." Proceedings of the FSE/SDP workshop on Future of software engineering research. ACM, 2010.
14. Mohammadian M. Innovative Applications of Artificial Intelligence Techniques in Software Engineering. In: Papadopoulos H., Andreou A.S., Bramer M. (eds) Artificial Intelligence Applications and Innovations. AIAI 2010. IFIP Advances in Information and Communication Technology, vol 339. Springer, Berlin, Heidelberg, 2010
15. Harman, Mark. "The role of artificial intelligence in software engineering." Proceedings of the First International Workshop on Realizing AI Synergies in Software Engineering. IEEE Press, 2012.
16. Meziane, Farid, and Sunil Vadera. "Artificial intelligence in software engineering: Current developments and prospects." Machine Learning: Concepts, Methodologies, Tools and Applications, 1215- 1236, IGI Global, 2012
17. Ammar, Hany H., Walid Abdelmoez, and Mohamed Salah Hamdi. "Software engineering using artificial intelligence techniques: Current state and open problems." Proceedings of the First Taibah University International Conference on Computing and Information Technology (ICCIT 2012), Al-Madinah Al-Munawwarah, Saudi Arabia. 2012
18. R. Ferenc, J. Gustafsson, L. M¨uller, and J. Paakki. Recognizing Design Patterns in C++ programs with the integration of Columbus and Maisa. Acta Cybernetica, 15:669–682, 2002.
19. E. Gamma, R. Helm, R. Johnson, and J. Vlissides. Design Patterns : Elements of Reusable Object-Oriented Software. Addison-Wesley Pub Co, 1995.
20. Y.-G. Gu'eh'eneuc, H. Sahraoui, and F. Zaidi. Fingerprinting Design Patterns. In Proceedings of the 11th Working Conference on Reverse Engineering (WCRE 2004), pages 172–181. IEEE Computer Society, 2004.
21. Z. Balanyi and R. Ferenc. Mining Design Patterns from C++ Source Code. In Proceedings of the 19th International Conference on Software Maintenance (ICSM 2003), pages 305–314. IEEE Computer Society, Sept. 2003.
22. C. M. Bishop. Neural Networks for Pattern Recognition. Clarendon Press, Oxford., 1995.
23. J. Quinlan. C4.5: Programs for Machine Learning. Morgan Kaufmann, 1993.

AUTHORS PROFILE

Dr. P.Mangayarkarasi M.Tech[CSE]., PhD[CSE] is Associate Professor of the Department of Information Science and Engineering, New Horizon College of Engineering, (Autonomous Institution, NAAC A grade, NBA- Accredited Institution, ISO Certified), Bangalore. She has to her credit an excellent track record in the fields of academic, research and Quality Assurance. She is author of over 10+ scholarly research/ review papers, 06+ reputed and peer reviewed Journal Scopus/SCI/UGC/IEEE/ACM/Springer/Thomson Reuters) papers with good Citation index and h-index. She has won several research paper awards in different National and International conferences and symposiums. She field around 03 patents. She is Member in 07+ leading professional Societies and Forums. She is reviewer and editorial board member/ Advisory board for 07 reputed/ UGC approved International/ National Journals and has published 2 Technical books in the field of Computer Science. Her current research interests include Data Mining, Machine Learning, Data Science and Analytics, Learning Systems, Algorithm Design for emerging Medical Image and Video, data analysis and Advanced Network and Cyber Security. She is guiding PG & PhD Scholars in the fields of Computer and Information Science and Engineering.