2021 2nd International Conference on Computation, Automation and Knowledge Management (ICCAKM)
Amity University

Integration of Artificial Intelligence into software reuse: An overview of Software Intelligence

Amandeep Kaur Sandhu
School of Computer Science & Engineering)
Lovely Professional University
Punjab,India
amandeep.22162@lpu.co.in

Ranbir Singh Batth
School of Computer Science & Engineering)
Lovely Professional University
Punjab,India
ranbir.21123@lpu.co.in

Abstract— Software reuse not only reduces time-to-market but also improve productivity and reliability of the software. Developers can take the benefits of reusable modules of the software by simply accessing the software repository. To create these repositories Data Mining techniques are used and later to predict the reusability factor of the modules Artificial Intelligence comes into light. This integration of Data Mining with Artificial Intelligence to perform software engineering task is termed as software Intelligence. This paper provides a roadmap of conventional reuse approaches to Software Intelligence approaches. Intense survey is conducted on various Artificial Intelligence algorithms applied to measure reusability of the extracted component. From the survey it is concluded that Components Based Development fits best for modern applications as it supports object-oriented paradigm. Besides, the existing Artificial intelligence approaches lack is efficient prediction of reusable module.

Keywords—Software Intelligence, Data mining, Artificial intelligence, Component based development, software metrics

I. INTRODUCTION

Software plays a significant role in our life. In today's world internet is considered as the primary means of communication. To access the required information from internet, software's are used. For making a software from scratch it has to go through the Software Development Life Cycle (SDLC). A large amount of data is gathered in every phase of SDLC, which is both cost and time consuming. Instead of creating the software from beginning the data present in the repository can be reused. Software reuse term was first discussed in the conference of NATO in 1968. Reuse helps in both reducing the cost as well as time. Not limited to this, reuse also increase the productivity, reliability and quality of the software. Software reuse process not only reuse source code but also reuses the design of the product or software. Various techniques for software reuse are Software Product Lines (SPL) and Domain Engineering (DE), Commercial off the Shelf development (COTSD), Model Based Development (MBD), and Component Based Development (CBD). These techniques are explained in section III. Although, reuse has vast benefits, it faced various issues in implementation. While reusing the software components from existing source code, the reusable component might not fit or fail to provide the desired functionalities. To address the issues researchers used the concept of Data mining to extract the meaningful data from the given repository. After extracting the useful data researches integrated Artificial Intelligence to make decision whether the extracted data is reusable or not. This concept of combining data mining with artificial intelligence is termed as Software Intelligence to perform various software engineering task. This paper will brief a pathway from selecting an approach to extracting the meaningful data and then making decision for reusability.

Section II give a related work. The working of reuse process is summarized in section III. A survey on these approaches is done in the same section. Various software metrics are presented in the section. In section IV artificial intelligence linked to software reuse is explained. In section V various approaches available for reusing the software component is explained. And at last paper is concluded in section VII after a brief discussion in section VI.

II. RELATED WORK

Based on various approaches used for software reuse, Component Based Development (CBD) technique has gained the attention of researchers and industrialist. The reason for the popularity of this approach is the extraction of code from object-oriented application. Other popular techniques are mentioned in the next section. Vast research work is done in this field. Some of the work is listed in this section.

Hironori et al. [19] worked on objected-oriented software systems and introduced an approach which extracted the components from existing software into JavaBean reusable components. Dependency Inversion Principle with refactoring was used by Andreas et al. [20] to identify the reusable components. Xinyu et al. [21] used hierarchal clustering algorithm by using weighted graph for component identification. Search based optimization was first suggested by Mancoridis et al. [22] Authors used genetic and hill-climbing algorithm to find the optimal solution using the module relation from source code. Rathee et al. [23] proposed a search based optimization technique to identify the reusable components. Authors used NSGA-III algorithm by using the concept of obоject oriented metrics of cohesion and coupling. Cohesion and coupling were used by Praditwong et al. [24] in their work to improve the working of search based module clustering. Kabir et al. [25] proposed hierarchal clustering technique to find components from the source code. Integrated Genetic Algorithm was used by Bavota et al. [29] focusing on re-modularization keeping developers perspective with minimum feedback. Semi-heuristic framework was proposed by Shimin et al [26] which extracted reusable components form the software. The extracted components were arranged in hierarchal order to get groups. . Matej et al. [27] Studied two factors of search-based named as exploitation and the other exploration in evolutionary algorithm surveying 100 works. Janez et at. [28] proposed differential evolution algorithm which provided provision from self-adaptive control parameters setting (especially numerical problems). A technique was proposed by Alexandre et al. [30] which worked on unbalanced datasets. Matthew et al. [31] proposed a new approach (LANLAN), using word embeddings and machine learning, to harness information available in Stack Overflow. Kazi et al. [32] proposed a set of metrics at two lower granularity levels and provide evidence for their usefulness during vulnerability prediction (which will help in maintaining secure code and ensure secure software evolution). Diago et al.[33] used metamorphic relations to verify and validate systems for understanding the relations between their inputs and outputs.

III. APPROACHES OF REUSE

Software Engineering gained a lot of attention of researchers when it came to using the code of existing software to a new. Software Reuse is categorized in two parts. First 'Reuse-with' and the other is 'Reuse-For'. Reuse-for identify the reusable knowledge from the existing application and store in 'knowledge base'. On the flipside, Reuse-with search for the reusable knowledge and modify the reusable knowledge according to the requirement [1]. There are five main artifacts of reuse presented in Figure. 1.

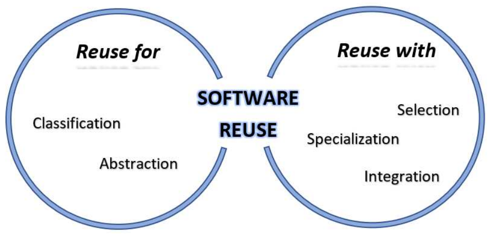

Figure 1. Dimensions of Software reuse

Table 1: Description of software reuse dimensions.

Reuse Type	Software artifacts	Description
Reuse for	Classification	Identification of reusable knowledge and presentation in abstract form.
Reuse for	Abstraction	Storing of the identified reusable knowledge according to their characteristics.
Reuse with	Selection	Search of appropriate reusable component from the library.
Reuse with	Specification	Modification of the selected component into required form.
Reuse with	Integration	Combination of the reusable component with the application being developed.

Techniques covering reuse-for are Software Product Lines (SPL) and Domain Engineering (DE). And the techniques named Commercial off the Shelf development (COTSD), Model Based Development (MBD), and Component Based Development (CBD) are covered under 'Reuse-with'. [2].

These approaches are summarized below:

A. Commercial off the Shelf development (COTSD):

A new application is created using the components of existing applications. These components are leased, licensed and/or sold to the public [3]. COTSD systems usually offer Application Programming Interface (API). To exemplify, the functionality of Operating Systems such as memory management, device drivers of various peripherals, Information acquisition applications and many more are some of the COTSD products.

B. Model Based Development (MBD):

In model-based development models or prototype are created which can be reused for other applications. Reusing the models provide abstraction and less complexity.

C. Software Product Lines (SPL):

"consists of a set of products and/or services sharing explicitly defined and managed common and variable features and relying on the same domain architecture to meet the common and variable needs of specific market." [4]. SPL act upon prediction rather than opportunity. This implies that instead of putting the components of software into a library call software artifacts in product line. [5] .

D. Component Based Development (CBD):

The existing components of software are implemented and integrated in new system in Component Based Development. Both the cases are true in CBD that is either the components used are specifically developed for reuse (reuse-for) or are used from existing system (reuse-with) [6]. This technique fits best for object-oriented applications. Based on these techniques a survey is done which concluded that the Component Based Development for Software Reuse is the most effective technique to be implemented [18].

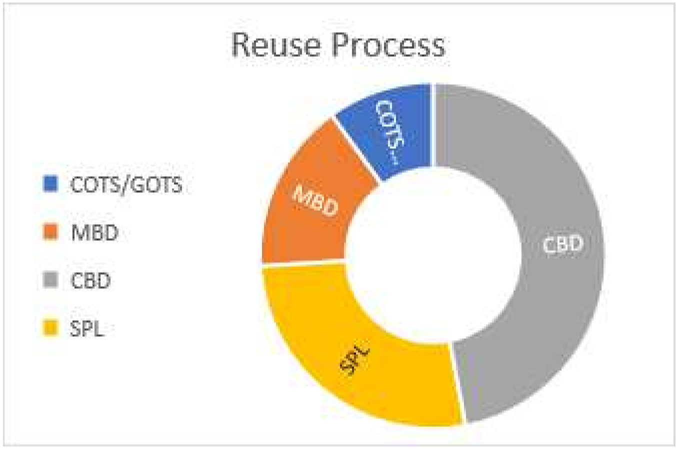

Figure 2: Pie chart depicts research work done using various reuse approaches

From the Table 2. it was concluded that Component Based Development fits best for the software reusing. As CBD provides minimum development cost, less maintenance efforts and higher feature extraction form the Source Code. Fig. 2 [7] shows the evolution of the quantity of papers mentioning reuse processes. [7] in their study surveyed 65 papers from year 1990 to 2016 and concluded 47% of researchers contributed towards CBD for reuse followed by

SPL with 27%. MBD and COTS approaches where the least with 16% and 10% respectively.

Table 2. Survey Based of Technique used for Software Reuse

Study	*Technique*	*Benefits*	*Development Cost*	*Maintenance Efforts*	*Feature Extraction*
[33]	MBD	Collaborating modeling support and tool interoperability	Low	Low	Low
[34]	MBD	High quality assurance	Low	Low	Low
[35]	SPL	Search based optimization technique	High	High	High
[36]	MBD	Faster prototyping robot operating system	High	High	High
[37]	MBD	Reuse UI specification at development time which benefit repository sharing of UI elements at run time	Low	High	Low
[38]	SPL	Cost estimation perspective	Low	Low	Low
[39]	SPL	Feature extraction for natural language for reuse	Low	High	High
[40]	SPL	Extract product line using bottom-up technology for reuse	High	Low	High
[41]	SPL	Run time adaptive system	High	Low	Low
[42]	CBD	Improve efficiency of IOT	High	Low	High
[43]	CBD	Modeling methodology for parametric dependencies and a corresponding graph-based resolution algorithm.	Low	Low	High
[44]	CBD	Use of pattern mining for software libraries reusability	Low	Low	High

IV. SOFTWARE REUSE PROCESS

The Software reuse is a four-step process. [8]. Searching of the component from the software repository is the very first step. It can be internal memory or external library search. In the second step of software reuse process the design structure of the program is analyzed in detail. And then the reusable components are extracted using various reuse techniques in the third step. In final step results are obtained by evaluating and testing. It consists of code cloning and code invocation.

Data Mining is a computer-assisted procedure to analyze large number of datasets [8]. The core motive is to extract meaningful data from provided input. Data mining process is mostly termed as Knowledge Discovery (KD). In Figure 3 KD process is explained. KD starts with the collection and storing of relevant data into database.

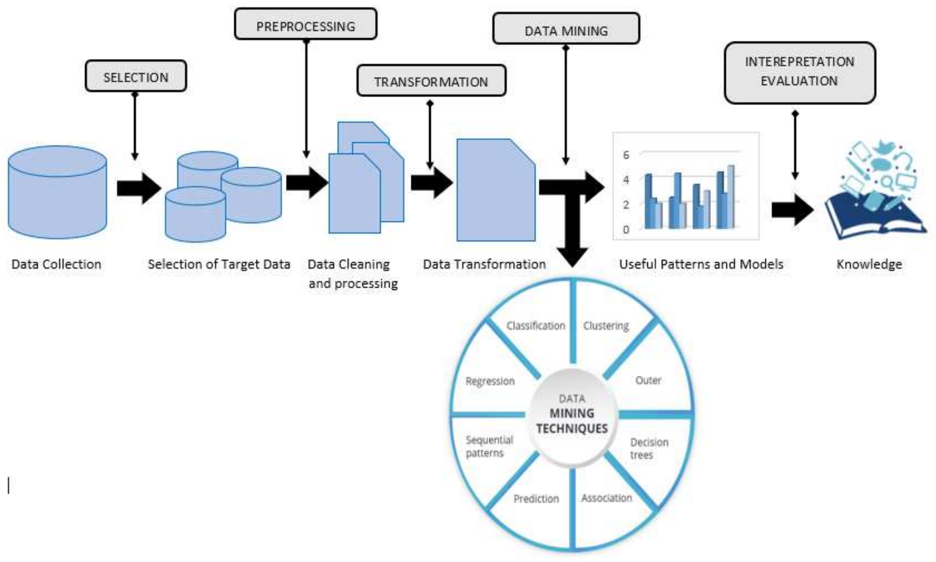

Figure 3. Core Process of Knowledge Discovery

Next, Preprocessing on the data is done to remove unwanted noise. Transformation comes after preprocessing in which data is transformed into suitable type so that required operations can be performed. Then comes the most important and crucial part of Knowledge discovery that is Data Mining. Algorithms are used to obtain meaningful patterns from the transformed data. Rules are also created along with patterns to further make predictive models. Then from these patterns are rules Machine Learning algorithms are used to make decisions. In the last phase of Knowledge Discovery Interpretation and evaluation of results is done to make useful decisions.

A. Software Metrices

To predict whether the given component of software is reusable or not some parameters are required. These measurable parameters are taken into account by the predictive models to make decision from the given data [9]. In programming these measurable attributes are named as Metrics. Various studies are conducted on these metrics [10-16]. From the source code, the classes, relation between the classes, interlinking of classes with functions, their attributes, are evaluated to give values to these metrics. Variations in metrics can appear from object oriented to procedure-oriented languages. Various parameters are reported by the researchers. The most significant metrices to evaluate the reusability [18] of the provided component are described in the Table 4.

Table 4. Provides the general description of these metrices.

Software Metric	**Description**
Coupling	It represents the dependency of on software module on other. The more one module depend on other the lesser that component will be reusable.
Cohesion	It depicts the independency of the software module to perform a task. That is how strong a module can work to perform a specified task, which in return give high reusability.
Number of children (NOC)	It defines the number of child class being inherited from the provided module. If the NOC factor is large the reusability will increase.
Weighted Method per Class **(WMPC)**	It given the class complexity. If the class have high complexity it become difficult to integrate it with new environment, so it needs to be low.
Cyclomatic Complexity **(CC)**	It defines the complexity of overall program. More the complexity lesser the chances of being used again as testing become difficult.
Depth Inheritance Tree **(DIT)**	In the inheritance hierarchy level of the class is depicted by DIT factor.
BUGS	Bugs are the incorrect codes that halter the smooth working of application, which should be low to increase reusability.

V. ARTIFICIAL INTELLIGENCE AND SOFTARE REUSE

After extracting the useful component from the software repository using Data mining the component are used in the new applications to be reused. Most of the time these reusable components fail to fit in the new application or do not provide the desired functionalities. Here, Artificial Intelligence comes to light. It helps to predict whether the extracted component is reusable or not. Besides, it helps to predict the quality of the component which further helps to increase the efficiency and adaptability of the software hence, reducing the cost and the development time of overall software [17]. CBD gained much interest in Software Reuse. A lot of research work is done in this field. Some of the work is listed in the section below.

VI. DISCUSSION

After conducting the comprehensive survey followings research gaps have been formulated that the majority of the existing software reusable components identification techniques suffer from the hyper-parameter tuning issues. Therefore, meta-heuristic techniques are required to optimize the hyper parameters efficiently. It has been also been observed that majority of existing meta-heuristic-based techniques such as genetic algorithm, differential evolution, particle Swarm optimization etc. suffer from pre-mature convergence issue. It limits the performance of reusable software components identification techniques. Besides, majority of techniques suffers from poor computational speed.

V. CONCLUSION AND FUTURE SCOPE

Software reuse is a domain in software engineering which deals with development of new software using the features of existing software. Software Reuse has gained a lot of attention over years due to its functionalities as it gives faster development, utilize fewer resources, less efforts and leads better performance and reliability. Various approaches such as COTS, MBD, SPL, CBD has been introduced by researchers to implement reusability. These approaches have been depicted in the paper. It was concluded from the survey that CBD fits best for the current object-oriented software. The use of data mining for extracting the meaningful data from the source code was next discussed in the paper. At last, a discussion on predictive models was done. From the intense survey it was concluded that the artificial intelligence algorithm used to predict whether the component is reusable lack in computational speed and low performance. A multi objective search-based optimization technique will be proposed in future work to overcome the current techniques short comes.

REFERENCES

[1] Charles W. Krueger, “Software Reuse”, ACM Computing Surveys, Vol. 24, No. 2, June 1992.

[2] Eduardo Santana de Almeida, Alexandre Alvaro, Daniel Lucrédio, Vinicius Cardoso Garcia, Silvio Romero de Lemos Meira, “A Survey on Software Reuse Processes”, IEEE, 2005.

[3] Jayasudha, R., Subramanian, S. "Enhancing the level of software reuse to improve software development", Anna University, Chennai, 2015.

[4] International Organization for Standardization, “Software and Systems Engineering – Reference Model for Product Line Engineering And Management”, ISO/IEC 26550:2015.

[5] Charles W. Krueger, "Introduction to the Emerging Practice of Software Product Line Development", Software Development Magazine - Project Management, Programming, Software Testing,2006.

[6] I. Crnkovic, M.P.H. Larsson, "Building reliable component-based software systems”, Artech House, 2002.

[7] José L. Barros-Justo, Fernando Pinciroli, Santiago Matalonga, Nelson Martínez-Araujoa, "What software reuse benefits have been transferred to the industry? A systematic mapping study", Information and Software Technology 103 pp 1-21,2018.

[8] B V Ajay Prakash, D V Ashoka, V N Manjunath Aradhya, "Application of Data Mining Techniques for Software Reuse Process", Procedia Technology 4 (2012) 384 – 389Li, J.R., Tao, F., Cheng, Y., Zhao, L.J., 2015. Big Data in product lifecycle management. Int. J. Adv. Manuf. Tech. 84(1-4), 667-684.

[9] Brahmaleen Kaur Sidhu, Kawaljeet Singh & Neeraj Sharma (2020): A machine learning approach to software model refactoring, International Journal of Computers and Applications, DOI: 10.1080/1206212X.2020.1711616.

[10] Murillo-Luna, J.L., Garces-Ayerbe, C., Rivera-Torres, P., 2011. Barriers to the adoption of proactive environmental strategies. J. Clean. Prod. 19 (13), 1417-1425.

[11] Ngai, E. W., Xiu, L., Chau, D. C., 2009. Application of data mining techniques in customer relationship management: A literature review and classification. Expert. Syst. Appl. 36(2), 2592-2602.

[12] Rabl, T., Gómez-Villamor, S., Sadoghi, M., Muntés-Mulero, V., Jacobsen, H.-A., Mankovskii, S., 2012. Solving big data challenges for enterprise application performance management. Proc. VLDB Endow. 5 (12), 1724- 1735.

[13] Silva, D.A., Delai, I., Castro, M.A., Ometto, A.R., 2013. Quality Tools Applied to Cleaner Production Programs: A First Approach Towards a New Methodology. J. Clean. Prod. 47, 174-187.

[14] Vinodh, S., Prakash, N.H., Selvan, K.E., 2011. Evaluation of agility in supply chains using fuzzy association rules mining. Int. J. Prod. Res. 49(22): 6651-6661.

[15] Wang, H. W., Chen, S., Xie, Y., 2010. An RFID-based digital ware-house management system in the tobacco industry: A case study. Int. J. Prod. Res. 48(9), 2513-2548.

[16] Wei, F. F., 2013. ECL Hadoop: “Big Data” processing based on Hadoop strategy in effective e-commerce logistics. Comput Eng Sci 35(10):65–71.

[17] Divanshi Priyadarshni Wangoo, "Artificial Intelligence Techniques in Software Engineering for Automated Software Reuse and Design", 4th International Conference on Computing Communication and Automation (ICCCA), 2014.

[18] Sandhu AK, Batth RS., “Software reuse analytics using integrated random forest and gradient boosting machine learning algorithm”, Software: Practice and Experience., 2020, 1-13.,wiley https://doi.org/10.1002/spe.2921.

[19] Hironori Washizaki, Yoshiaki Fukazawa, “A technique for automatic component extraction from object-oriented programs by refactoring”, Science of Computer Programming, Vol. 56, Issues 1–2, pp 99-116,April 2005.

[20] Andreas Marx, Stephan Diehl, Fabian Beck, “Computer-Aided Extraction of Software Components”, 17th Working Conference on Reverse Engineering, ISBN: 978-1-4244-8911-4, Oct 2010.

[21] Xinyu Wang, Xiaohu Yang, Jianling Sun and Zhengong Cai, “A New Approach of Component Identification Based on Weighted Connectivity Strength Metrics”, Information Technology Journal, vol.1, Issue.1, pp. 56-62, 2008.

[22] S. Mancoridis, B.S. Mitchell, C. Rorres, Y. Chen, E.R. Gansner, "Using automatic clustering to produce high-level system organizations of source code, Program Comprehension", 1998. IWPC ’98. Proceedings., 6th International Workshop, pp. 45–52, 1998.

[23] Amit Rathee, Jitender Kumar Chhabra, “A multi-objective search based approach to identify reusable software components”. Journal of Computer Languages vol 52, pp. 26–43, 2019.

[24] K. Praditwong, M. Harman, X. Yao, "Software module clustering as a multi-objective search problem", IEEE Trans. Softw. Eng. 37 (2) 264–282, 2011.

[25] Selim Kebir; Abdelhak-Djamel Seriai; Sylvain Chardigny; Allaoua Chaoui, “Quality-Centric Approach for Software Component Identification from Object-Oriented Code”, Joint Working IEEE/IFIP Conference on Software Architecture and European Conference on Software Architecture, October 2012.

[26] Shimin Li; L. Tahvildari. “Comp: A Reuse-Driven Componentization Framework for Java Applications”, 14th IEEE International Conference on Program Comprehension (ICPC'06), ISSN: 1092-8138, June 2006.

[27] Matej Crepinsek, Shih -His Liu, Marjan Mernik, “ Exploration and exploitation in evolutionary algorithms: A survey”, ACM Computing Surveys (CSUR), Vol.45, Issue 3, June 2013.

[28] Janez Brest, Member, Saso Greiner, Borko Bo’skovic , Marjan Mernik, and Viljem Zumer, “ Self-Adapting Control Parameters in Differential Evolution: A Comparative Study on Numerical Benchmark Problems”, IEEE Transactions on Evolutionary computation, vol. 10, pp. 6, December 2006.

[29] G. Bavota, F. Carnevale, A. De Lucia, M. Di Penta, R. Oliveto, "Putting the developer in-the-loop: an interactive ga for software re-modularization", G. Fraser, J. Teixeira de Souza (Eds.), Search Based Software Engineering, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 75–89,2012.

[30] Alexandre M. Nascimento, Lucio F. Vismari, Paulo S. Cugnasca, João B. Camargo, Jorge R. de Almeida, “A cost-sensitive approach to enhance the use of ML classifiers in software testing efforts”, 18th IEEE International Conference on Machine Learning and Applications (ICMLA),2019.

[31] Matthew T. Patrick, “Exploring software reusability metrics with Q&A forum data”, The Journal of Systems & Software, 2020.

[32] Kazi Zakia Sultana, Vaibhav Anu, Tai-Yin Chong, “Using software metrics for predicting vulnerable classes and methods in Java projects: A machine learning approach”, Software: Evolution and Process, 2020.

[33] Diogo Moreira, Ana Paula Furtado, Sidney Nogueira, “Testing acoustic scene classifiers using Metamorphic Relations”, IEEE International Conference On Artificial Intelligence Testing (AITest), 2020.

[32] Kaur, A, Singh, P, Singh Batth, R, Peng Lim, C. “Deep-Q learning-based heterogeneous earliest finish time scheduling algorithm for scientific workflows in cloud”, Software: Practice and Experience 2020; 1– 21.,wiley. https:/doi.org/10.1002/spe.2802.

[33] Juri Di Rocco, Davide Di Ruscio, Ludovico Iovino, and Alfonso Pierantonio, “Collaborative Repositories in Model-Driven Engineering”. THE IEEE COMPUTER SOCIETY, 2015.

[34] G. Quilty, M.Ó. Cinneide, Experiences with Software product line development in risk management software, Proceedings of the Fifteenth IEEE International Systems and Software Product.

[35] Roberto E.Lopez-Herrejon, LukasLinsbauer, AlexanderEgyed, “A systematic mapping study of search-based software engineering for software product lines”. Information and Software Technology,Vol. 61, pp. 33-51, 2015.

[36] Yingbing Hua, Stefan Zander, Mirko Bordignon and Bj¨orn Hein, “From AutomationML to ROS: A Model-driven Approach for Software Engineering of Industrial Robotics using Ontological Reasoning”. IEEE, 2015

[37] A. Delgado, A. Estepa, J.A. Troyano, R. Estepa, “Reusing UI elements with Model-Based User Interface Development”. Journal of Human Computer Studies, Vol. 2015.

[38] Jacob Krüger, Wolfram Fenske, Jens Meinicke, Thomas Leich, Gunter Saake, “Extracting software product lines: a cost estimation perspective”. ACM Digital Library, 2016.

[39] Noor HasrinaBakar, Zarinah M.Kasiru, NorsaremahSalleh, “Feature extraction approaches from natural language requirements for reuse in software product lines: A systematic literature review”. Journal of Systems and Software, Vol.106, pp. 132-149. 2015.

[40] Jabier Martinez, Tewfik Ziadi, Tegawendé F. Bissyandé, “Bottom-up technologies for reuse: automated extractive adoption of software product lines”. ACM Digital Library, 2017.

[41] Mahdi Bashari, Ebrahim Bagheri and Weichang Du, “Dynamic Software Product Line Engineering: A Reference Framework”. International Journal of Software Engineering and Knowledge Engineering, Vol. 27, pp.191-234, 2017.

[42] Doohwan Kim, Jae-Young Choi, Jang-Eui Hong, “Evaluating energy efficiency of Internet of Things software architecture based on reusable software components”. International Journal of Distributed Sensor Networks, 2017

[43] Simon Eismann ; Jürgen Walter ; Jóakim von Kistowski ; Samuel Kounev, “Modeling of Parametric Dependencies for Performance Prediction of Component-Based Software Systems at Run-Time”. IEEE International Conference on Software Architecture (ICSA), 2018.

[44] Mohamed Aymen Saied, Ali Ouni, Houari Sahraoui, Raula Gaikovina Kula, Katsuro Inoue, David Lo, “Improving reusability of software libraries through usage pattern mining”. The Journal of Systems & Software, Vol. 145, pp. 164–179, 2018.

[45] H. Sharma, N. Kanwal and R.S. Batth, "An Ontology of Digital Video Forensics: Classification, Research Gaps & Datasets," 2019 International Conference on Computational Intelligence and Knowledge Economy (ICCIKE), Dubai, United Arab Emirates, 2019, pp. 485-491.

[46] K. Sandhu, R. Singh Batth and A. Nagpal, "Improved QoS Using Novel Fault Tolerant Shortest Path Algorithm in Virtual Software Defined Network (VSDN)," 2019 International Conference on Automation, Computational and Technology Management (ICACTM), London, United Kingdom, 2019, pp. 383-388.