Dynamic Reusability Prediction Model for SMEs Based on Realtime Constraints

Dr. P. Mangayarkarasi, Dr. R. Selvarani

Associate Professor, Dept. of ISE, New Horizon College of Engineering, Bangalore, India
Program Director - Doctoral program Professor, Dept. of CSE, Alliance University, Bangalore, India

mangaivelu18@gmail.com, selvarani.riic@gmail.com

Abstract *— With the increasing cutting-edge competition in the IT industry, it has become essential to offer maximum quality in the dispatched software along with cost-effective productivity. Although high-end IT sectors can adopt different quality management schemes to offer more enhancement in productivity, they cannot be deployed so quickly among Small Medium Enterprises (SME). The prime reason for such impediment is mainly the different forms of constraints associated with SMEs. At present, SME uses software reusability which is different from design reusability owing to the absence of any benchmarked or proven model in this regards. The claim in existing studies is that if the software metrics can be amended, then it can compute efficient software reusability. However, there is no connection of calculating and ensuring software reusability in such way. The issue is entirely a computational problem which is solved in the form of mathematical modeling in the proposed system. The proposed study mathematically formulates near real-time constraints and different objective function along with a mathematical expression for computing design reusability. Hence, the contribution of the proposed study is not only to calculate design reusability but also to ensure proper implementation of design reusability. A machine learning approach was used which is responsible for obtaining all the best cases of deployment of design reusability in compliance with all the constraints. The outcome of the proposed model proves that it not only offers a cost-effective solution to compute design reusability but significantly assists the stakeholders of software projects to make some critical decision for the successful dispatch of software projects to clients.*

Keywords *— Cost Effective, Design-Reusability, Object-Oriented Design, Software Project , Scheduling*

I. INTRODUCTION

The object-oriented design methodologies play a significant role in the software industry. Various design aspects used in object-oriented systems are considered as potential attributes for measuring the eminence factors of the software under fabrication. In the recent times, the usage of software metric is added by the software engineers to assess the demanded components of the model for any specific software project. The software metric provides an efficient platform to determine the design pattern performance. Adherence to such assessment, the software reliability and sustainability will be well quantified.

Once a software development firm receives a new requirement from the clients, it is essential to formulate an efficient design. These new design strategies are subjected to many iterations to fit the expected level of the requirement by the clients. Finally, the development team receives conformed design details for the process. The risk factor here is that even a minor error in the design formulation will cost more for the production team regarding point in time, vigor, and money. After a product dispatched to its client, it is unethical to reuse the similar code of the same software project as it is an intellectual property of its legitimate client. The question of code reusability comes in the picture as an organization can expect a similar form of requirement from another client. In such aspect, it cannot use the old code. Hence, code reusability is not permissible here. In such cases, the production team has to start their coding right from the scratch which is again an expensive process involved in software development cycle. Hence, a concept called design reusability can be utilized to avoid such circumstances [1]. By adhering to the process of design reusability, the production skills enhance as well as it becomes feasible for ensuring the project delivery on time. The advantages of adopting design reusability are: i) it save production time, ii) it focuses on evolving design patterns, iii) it ensures minimization of project schedule slippage, iv) it enhances the skills of software developers [2] and, v) it reduces the development cost.

The remaining paper is organized as follows: In the next section, Literature Survey is presented. Proposed system and Research methodology is described in section 3 and 4 respectively. Section 5 describes result analysis and followed by the application of proposed approach in section 6 and finally section 7 presents the conclusion.

II. LITERATURE SURVEY

Various research works carried out towards design reusability in software engineering and its implementation using different forms of tools and technologies. This section will briefly outline the research contribution made so far in the area of software reusability.

Nair and Selvarani [3] proposed a framework with a capability to forecast the reusability index considering three metrics in Chidamber and Kemerer metrics viz.: DIT, RFC, and WMC. They exposed the strong relationship that exists between the design parameters and reusability factors in developing a reusability estimation model.

Khoshkbarforoushha et al. [4] presented a quantitative approach for investigating the software reusability associated with the standard process management. A unique metric formulation is mechanized for forming probability of mismatch while analyzing the process flow system. The investigation bears potentially resemble expert's opinion and the outcome of the software metric. The presented empirical approach offers a better measure of the process reusability. The study also gave practical as well as analytical validation. The study outcome shows significance value of correlation.

Themistoklis et al. [5] presented a technique that can assist in suggesting reusability factor involved in the component of the source code. The suggestions offered from both quality and functional viewpoint of the software component. The system also provides a ranking mechanism to highlights the enrichment towards the software components. The software metric adopted by the author consists of coupling, complexity, volume, and cohesion. The study outcome shows the accomplishment of reusability score of 89% and precision of 89% approximately.

Demraoui et al. [6] presented a research-based technique where the reasoning capability of the case-specific is introduced to enhance the reusability of the software framework developed. The reusability framework reflects a knowledge-based case study for the consideration. A decision process proposed novel approaches in association with the context, objective, and preference. The authors deployed a system for retrieving cases based on defined relations by the principle of the knowledge base.

Sham and Bakar [7] investigated the effectiveness of CK metric by implementing it on the programs scripted using C++. They examined CK metric for Spearman correlation to understand the extent of a software defect. The study outcome shows that RFC and NOC (others are CBO, WMC, LCOM, and DIT) are the dominant CK metric that can be used reliably for forecasting the software defects density residing with the system.

Awad et al. [8] introduced a discussion of the software model for enhancing the reusability. A workbench presented for representing various functionalities of the interfaces, software applications, different attributes, etc. The architecture performs encapsulation of development as well as cloud layer where the communication bridge exists between the virtual development environment and integration platform. The complete architecture is claimed to offer significant improvement in the reusability demands.

Munk [9] presented a discussion about the applicability of reusable components. According to the author, existing software representation are the dependable parameters of the interoperability depiction; however, such forms of the solution are associated with problems. According to the study findings, it says that supportability of Multilateral Interoperability Program-based solution is less significant as compared to a standard NATO-based solution regarding interoperability. At the same time, the solutions offered by Semantic Interoperability Logical Framework-based approach failed to provide support for semantic utilization.

Selvarani [10] presented an extensive evaluation framework for assessing the impact of defects in software using data-driven techniques. The study was conducted in the direction of defect evaluation in the design stage of Object Oriented programs. The outcome of the study shows better efficiency in the existing development lifecycle of software.

Nair and Selvarani [11] presented a framework for estimating the software reusability with the aid of different software parameters. A typical method introduced by the authors that perform analysis of the design aspect of object-oriented software for computing the level of reusability associated with the software. The presented system also calculates the structural characteristics of the software with the aid of design metric and then performs transformation of reusable measures to linked with the software package. The system also presents a data-driven approach for forecasting the software reusability for enhancing the software design.

Singh et al. [12] presented a soft-computational framework for evaluating the potential of reusability. The authors introduced multiple software metric-based factors, e.g., the complexity of interface, modularity, flexibility, maintainability, adaptability, etc. Different soft-computation based approach, e.g., fuzzy logic has used with varying forms of membership function to assess the effectiveness. The study outcome evaluates the output in the form of reusability as impacted due to various types of the membership function.

III. PROPOSED SYSTEM

The proposed study is to perform modeling of design reusability and explore its possible effect in software engineering for a given set of real-world constraint. The formulation of this problem statement evolved from the real-world limitations of SMEs (Small and Medium Enterprises). The proposed study considers

the facts that SME is the best end-user to opt for design reusability implementation as they have a lesser bandwidth of resources as well as they have the lower liquidity to adopt specific expensive enhancement plans of their productivity. The system design of the proposed system is carried out using an n-tier architecture where the development of the model is carried out using mathematical approach.

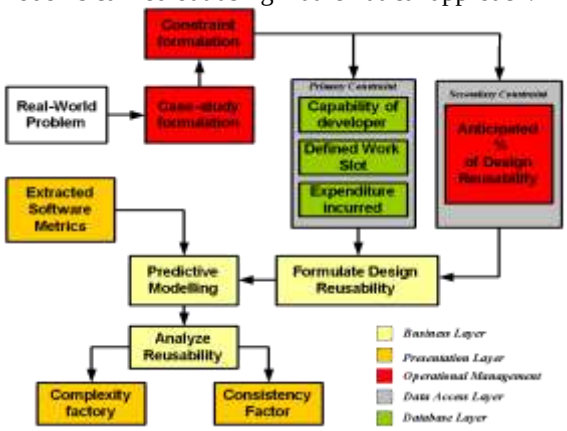

Figure 1 Proposed System Architecture

Fig.1 highlights the proposed system architecture, which is designed using *n*-tier architecture approach. This is an optimized frame work designed for quantifying the reusability of the design of any software through a predictive approach. Following are the detailed information about the essential components included in the proposed system architecture.

The proposed study considers a case study to shape the real-time implementation by considering essential attributes of intrinsic and extrinsic factors, i.e., number of human resources involved in a particular software project, working hours allotted for the human resources, cumulative expenditure involved in the development, and degree of uncertainty involved in software project development. These attributes are found universally in every organization and will assist in constructing n-tier architecture under *operational management layer*. The *first constraint* introduced in the system architecture is *capability of the developer*. This effort measures the maximum number of software projects delivered in sole handed. The *second constraint* is related to *defined working slot allocated* to the developer for doing the same task equivalent to effort offered by them. The *third constraint* is related to *expenditure incurred* for complete team management to dispatch a software project to the client successfully without violating any terms and condition.

The core component of a proposed system model is a novel mathematical model for computing design reusability. This component designed is executed over a *business layer* of proposed system architecture. The evolution of the mathematical model initiates from the constraint design using the layer-based approach in n-tier architecture. The formulation of design reusability is carried out by considering *secondary constraints* viz. An outcome of the fixed proportion reusability is set aside for further usage. The proposed system has to satisfies this constraint to offers optimization towards design reusability. It will mean that proposed scheme not only computes design reusability considering all real-world constraint but also ensures that outcome of design should also ensure certain specific level of design reusability. In reality, the information about the constraint is generated from the *database layer* by *business layer* via *data access layer*. Hence, it is more probable that the databases associated with the constraints are quite massive and distributed. Consequently, this type of modeling will assist the proposed *business layer* to compute design reusability score as all the constraints are specifically defined well in the *database layer*. However, a stakeholder will never be able to assess the value of this database for its future clients. Therefore, a system is required to access the database and is capable of computing the design reusability considering future prospective of client requirement of software design. Hence, the proposed method applies predictive modeling as a part of *utilities* in a *business layer* with the help of machine learning approach. This component takes two different inputs, i.e., the stored value of design reusability as a standard score in the database and probabilistic numerical value of software project with object-oriented logic. The processing results in the analysis of final reusability scores. The extraction of software metric score component is one of the prime processes to carry out validation of the proposed design reusability model, and it checks the effectiveness of its claimed optimization process. This component initially takes the input of a complete software project which complies with object-oriented logic and associated with different forms of frequently used enterprise applications. The reason for the inclusion of enterprise applications, as it possesses a large number of class files, interfaces, functions, Application Programming Interface (API), etc. that offers ample input for one software project. The proposed system uses existing tools to extract standard software metrics that will give the experimental value of design reusability. The extracted software metrics are then fed to the predictive model to compute the statistical outcome of design reusability. This component mainly runs on the presentation layer as it could be easily integrated with any form of an integrated development environment to capture the standard software metric.

The proposed system applies machine learning to extract the elite outcome of the best possible value of design reusability to the offered design for the given set of problems. Hence, the last stage of design

reusability score will also ensure to provide similar performance (for maintaining the related trend of design reusability) for future clients. Therefore, two performance parameters, e.g., complexity factor and consistency factor are used to deliberate the final analysis of design reusability. The complexity factor will offer extensive information of the degree of computational complexity associated with the operation of the proposed technique as it includes certain iterative optimization process in machine learning. The consistency factor will give the tentative degree of adherence or violation of the allocated secondary constraint for upcoming future prospective clients. Hence, the proposed system architecture offers an efficient optimization of design reusability using an effective validation process.

IV. RESEARCH METHODOLOGY

The methodology implemented for the design of proposed system is an analytical approach which initiates with a specific case study where the technical adoption of the model is analyzed. The idea is mainly to develop a mathematical model motivated from the near real-time problems in existing software development cycle. Another core motivation of this design is also to assess the improvement in the production that could be possibly caused by merely incorporating design reusability. Hence, the elaboration of the design methodology is carried out by concerning a simplified mathematical model followed by design optimization in the proposed system. Following is the illustrative description of the proposed methods adopted to achieve the proposed design goal.

A) Definition of Actors

Following are the essential actors of the proposed system for design reusability:

i) Software Developer

The proposed study considers a software developer to be a person responsible for constructing the complete software project as per the specification of the client. The study uses the variable S to represent software developer and is always of the form of an integer Z.

$$\therefore S \to Z \tag{4.1}$$

The proposed study also considers that there are two designations of software developer, i.e., junior software developer S_j and senior software developer S_s. Hence, a mathematical representation of software developer is,

$$S = (S_j \parallel S_s), \text{ where } S_j, S_s \to Z \tag{4.2}$$

ii) Software Project

The organization gives the term software project as the assignment to the software developers which are outsourced by the client. The proposed study considers that *Software Projects* are designed using object-oriented logic and consumes the discrete amount of duration within that the project is to be completed. The study allocates a variable ϕ to represents software project and also defines its characteristics to depict the strength of the software project. The exposed mathematical representation of the software project is shown as,

$$[\phi_r, \phi_m, \phi_h] \subset \phi$$

where, $\phi \to Z$, & $(\phi_r, \phi_m, \phi_h) \to [0\ 1]$ (4.3)

The above expression shows that the software projects are categorized by three grades of complexities, i.e., software project of regular complexity (ϕ_r), medium complexity (ϕ_m), and higher complexity (ϕ_h). As ϕ represents number of software project, hence, it is of the form of integer numbers. However, the proposed study allocates the complexity of the software project regarding probability range of (0, 1). This will mean that possible range of allocation of complexities are $0<\phi_r<0.03$, $0.03<\phi_r<0.05$, and $0.05<\phi_h<0.07$ on the basis of general statistical concept.

iii) Project Duration

The project duration is different from the typical and conventional representation of software project development time. As the proposed study investigates the impact of design reusability on the productivity as well as on software development life cycle, the project duration represented as exactly the duration that an employee spends to incorporate design reusability. The mathematical representation of the project duration δ is as follows,

$$\delta = [\delta_n\ \delta_r] \quad \&, \quad \delta_{min} < \delta < \delta_{max} \tag{4.4}$$

The above mathematical expression of project duration δ shows that there are two types of duration considered viz. normal project duration δ_n (duration without reusability) and project duration where an employee works only on design reusability δ_r. It is also stated that scope of project duration lies within the range of minimum and maximum value of δ_{min} and δ_{max} respectively.

iii) Cost

The term *cost* is a direct representation of all the possible forms of resources that are deployed to achieve the design objectives during the software project development. For better study effectiveness, the term cost ψ is kept under the statistical scope of *p*-significance to obtain a much clear indication of actual cost involved in incorporating design reusability. The next section further elaborates

various forms of constraints evolved from all the restrictions mentioned above the real-world environment.

B) Constraint Formulation

There are a specific set of actions that are performed by the actors in the conventional cases adhering to software development life cycle. A specific set of strategies are formulated by such set of activities to deliver the product or services in software project development. These sets of strategies are essential to consider or else the proposed modeling would become more hypothetical with less applicability in a real-time environment. Such set of characteristics combine to form constraint. Each actor of the proposed study is characterized by specific near real-time constraints that allow the modeling of design reusability to be more practical. This section discusses different forms of the constraints formulated from the case study of SME.

i) Constraint related to Software Developer

The proposed study generically categorized the Software Developer S into two types, i.e., Junior Software Developer S_j and Senior Software Developer S_j. Hence both S_j and S_s possess different capabilities concerning the software project development. The capabilities defined are concerned with number of software projects that can be handle by the developer standalone and duration of project development. The mathematical expression of the capabilities C_j and C_s of junior and software developers is represented in the equation 4.5

$$C_j \rightarrow (\phi_j, \delta_j) \quad , \quad C_s \rightarrow (\phi_s, \delta_s) \tag{4.5}$$

The proposed study defines that senior software developer possesses more capability as compared to the junior software developer. The proposed study constructs an objective function of this capability construct as,

$$f_1(x) \rightarrow [(\phi_j < \phi_s) \ \& \ (\delta_j = \delta_s)] \tag{4.6}$$

The above mathematical expression of $f_1(x)$ shows that junior software developer is capable of delivering a ϕ_j number of software projects is less as compared to a senior software developer who can provide a ϕ_s amount of a software project. Such that $\phi_j < \phi_s$. It has ensured that both the types of developer consume similar project duration to deliver a different number of software projects, i.e., $\delta_j = \delta_s$. For better optimization in design reusability, both the condition of above objective function has to be satisfied.

ii) Constraint related to Software Projects

The mathematical representation of the constraint associated with the software project is,

$$\alpha \rightarrow \{\phi_r, \phi_m, \phi_h\} \tag{4.7}$$

Where, α represents the degree of complexity associated with three different types of software projects, i.e., software projects with regular complexity ϕ_r, medium complexity ϕ_m, and higher complexity ϕ_h, Hence, the expression 5.5 has to remodeled as,

$$C_j \rightarrow \{(\phi_r, \phi_m), \delta_j\} \ , \ \ C_s \rightarrow \{(\phi_m, \phi_h), \delta_s\} \tag{4.8}$$

A closer look into the above expression shows that a junior software developer is capable of handling software projects with regular and medium complexity, i.e. $(\phi_r, \phi_m\}$ whereas a senior software developer is capable of handling software projects with medium and higher complexities, i.e. $(\phi_m, \phi_h\}$ considering $\delta_j = \delta_s$. The objective function $f_2(x)$ formulated for the software project construct as,

$$f_2(x) \rightarrow [\arg_{min}(\alpha) \ \& \ \arg_{max}(\phi)] \tag{4.9}$$

According to the mathematically formulated objective function above, it is exhibited that proposed study targets to minimize the complexities associated with the software projects and maximize the number of software projects delivery. For a valid optimization principle for design reusability, it is necessary to assure that the extent of design quality increases with a number of a delivered software project. This process not only increases productivity but also enhances the skillset and capability of the software developer.

iii) Constraint related to Project Duration

Project duration is one of the core deciding factors to ensure the product effectiveness. Consumption of lesser score of project duration is fundamentally essential as well as challenging task for the developer as there is the presence of software projects of multiple complexities (α of expression 4.7). The junior developer takes δ_j duration for developing software project bearing complexities of ϕ_r and ϕ_m respectively. Hence, the proposed system constructs a condition as follows,

$$\text{Schedule slippage} = \delta > \delta_{max} \tag{4.10}$$

The proposed system considers that condition of project schedule slippage surfaces when it surpasses the maximum duration δ_{max} of software project development. At the same time, the proposed system assumes that a software developer is always in the work direction to increase their productivity, this can achieve if they focus on spending more time to develop the reusable design. For this purpose, the study develops an objective function for project duration to ensure optimal quality of software projects. The objective function has represented below as,

$$f_3(x) \rightarrow arg_{min}(\delta_n) \;\&\; \arg_{max}(\delta_r) \quad (4.11)$$

The above objective function states that proposed system should ensure more dedication towards project duration with design reusability δ_r rather than one with standard development strategy, i.e., δ_n. Although the optimization principle should target to increase δ_r, there is a limit set for δ_r, i.e., $\delta_r < \delta_{max.}$ Similarly, the optimization principle cannot minimize δ_n to zero, and a minimum limit is to fixed as $\delta_n > \delta_{min.}$ The complete focus is laid to ensure that in many cases, the software project with more reusable factors can incorporate into the scheduled time of project delivery to the client.

iv) Constraint related to Cost

Cost is the final constraint of the proposed study where it is associated to multiple actors involved in the entire process of software project development. As various factors control the cost, hence, it is challenging to model the cost using single attribute. Lower cost is required to be targeted to prove better software project ϕ_h should be equivalent to that of medium complex software project ϕ_m. In parallel, this objective function also ensures that cost involved in the complex project should not be more than usual optimization capabilities. The proposed system formulates two objective functions for this reason to ensure a balance between complexity and delivery schedule. The first objective function of the formulation $f_4(x)$ is

$$f_4(x) \rightarrow \psi(\phi_h) \approx \psi(\phi_m) \;\&\; \psi(\phi_h) \leq \Psi(\phi_r) \quad (4.12)$$

The above mathematical expression exhibits that cost ψ involved in developing highly complex software project ϕ_h should be equivalent to that of medium complex software project ϕ_m. In parallel, this objective function also ensures that cost involved for highly complex project should not be more than normal software project ϕ_r or else that cannot be termed as an optimization. When these two conditions are satisfied together than only cost optimization is said to be taken place to enhance the productivity. The second objective function $f_5(x)$ is formulated as

$$f_5(x) \rightarrow \psi(\delta_r) \leq \Psi(\delta_n) \quad (4.13)$$

According to the objective function shown in above expression, it has seen that proposed system ensures the cost involved in consuming project duration corresponding to incorporating reusable design should be less than that of typical design. There are various rationale and justification behind this objective function as viz. i) developing a reusable design from scratch also consumes a certain amount of cost and the end product should always consider that it should have certain limit of reusability in its design aspect even after the project is completed and delivered to the client. Hence, there is always a certain level of work that has to designed from scratch, i.e., δ_n for any future client's requirement. For this purpose, if the cost involved in developing reusable components keeps on increasing in every iteration of new client's design requirement then it would gradually minimize the effectiveness of optimization concept in software design. Therefore, $\psi(\delta_r)$ has always kept as low as possible, ii) the duration actor δ_r also governed by the complexities associated with it for an upcoming future client where the worst possibilities could be ϕ_h. Hence, if $\psi(\delta_r)$ is kept low, then it will mean that it offers the consistent performance of design reusability for upcoming client's requirement showing the presence of deterministic characteristics of cost constraint.

Therefore to obtain design reusability experimentally, there is a dependency on the higher number of software projects compliant with object-oriented logic. However, it is beyond the scope of assuming the highest number of software project considers to obtaining the experimental form of mathematical modeling for design reusability. Moreover, without the consideration of highest number of software project. This problem has addressed by adopting probability theory. For this purpose, the initial step is to compute the best probability of calculating design reusability by incorporating the number of days with design reusability consideration. The mathematical expression of this duration with design reusability τ has represented in the below equation.

$$\tau = \sum_{c=1}^{\delta} \gamma.c.\rho \quad (4.14)$$

The above mathematical expression shows the mechanism to compute the best probability τ considering total project duration allotted δ starting from c=1 where c is number of days that a software developer works with design reusability. The formulation considers the mean value of the design reusability using γ (that corresponds to T). It also considers the total duration of an effort per day ρ by the software developer where the developer is assumed to incorporate design reusability. The similar expression of duration μ can be represented by considering the exclusion of the durations via including design reusability as,

$$\mu = \sum_{c=1}^{\delta} \gamma.(c-1).\rho \quad (4.15)$$

A closer look at mathematical expression 4.14 and 4.15 shows that it represents a mechanism to compute the project duration with and without the inclusion of effort associated to incorporate design reusability in project development respectively. Hence, the favorable outcome F_{out} of project duration has expressed below,

$$F_{out} = (\tau - \mu) \quad (4.16)$$

The total outcome T_{out} of software project duration has expressed beneath,

$$T_{out} = \delta - |\omega| \quad (4.17)$$

The computation of the total outcome of the software project duration can be calculated by subtracting total project duration δ with the absolute value of actual project duration ω. It will also mean that there is involvement of δ number of duration that is demanded to deliver the software project to the client satisfactorily. The variable ρ corresponds to 8 hours of working time for each developer. For effective outcome of the study, it is considered that actual duration of project development ω should always be less than the total allocated duration, i.e., δ. This condition is necessarily required to be satisfied for every iteration of computation of design reusability in the proposed system. Hence,

$$\text{Effective Duration} = \delta > \omega \quad (4.18)$$

By applying the concept of probability theory, the numerical value of probability can be obtained by dividing favorable outcome with total outcome. Hence, the final mathematical formulation of probability of design reusability is,

$$\gamma = \frac{F_{out}}{T_{out}} \quad (4.19)$$

Therefore, the proposed mathematical model contributes to providing information about optimal project development duration. This information significantly assists the project leaders as well as the stakeholders to assess the experimental value of design reusability . An effective convergence test is carried out using hypothetical and experiment outcomes. An optimal convergence outcome will offer positive insight to the development team while the negative convergence outcome will save the development team from meeting any maximum circumstances that associated with the degradation of project quality standards.

Therefore, the prime motivation for developing the methodology for constructing optimization policy is to address the challenges associated with non-inclusion of uncertainty or iterative steps or predictive process or validation. All these four research challenges towards optimization of a proposed mathematical model of design reusability are addressed using just one form of machine learning approach, i.e., neural network.

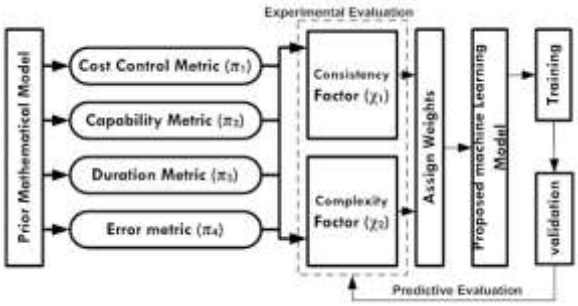

Figure 2 Schematic Diagram of Proposed Optimization [14]

Analytical research methodology is carried out for optimizing the proposed work. With reference to the Fig.2 the study has considered two forms of input for its optimization model, i.e., primary input and secondary input. The primary input consists of the all the three factors of optimization principle, e.g., cost control metric (π_1), capability metric (π_2), and duration metric (π_3). The secondary input consists of an error metric (π_4). From the discussion of research challenges associated with the mathematical model, it sensed that one simpler mechanism has to be modeled where all forms of unaddressed problems are to substitute for the user-defined range of errors. Hence, the optimization model will be capable of performing learning mechanism with different forms of analytical errors associated with the mathematical modeling. Therefore, an error is an abstract input that represents all sorts of the unaddressed problems related to the model.

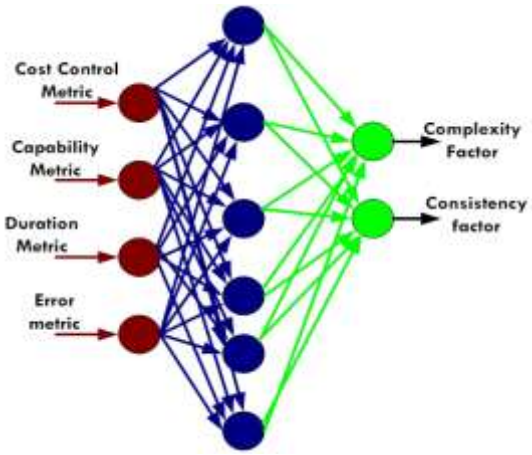

Figure 3 Multi-layer Perception for Proposed Optimization [14]

A multi-layered perceptron (Fig.3) is developed, where four different forms of the inputs are given to the neural network to obtain the output of consistency factor and complexity factor. A higher number of iteration rounds (100-1000) has used for evaluating the different set of optimized problems to check the effectiveness of retaining maximum score of design reusability. One of the advantages of this optimization technique is the iterative process where the system iterates until and unless it obtains the least empirical error during the learning process. Without

having any dependency of any specific thresholding factor, the multi-layer perceptron model is good enough to find better convergence point that not only validates the model but also results in the generation of the elite outcome of design reusability.

. The calculation of weight w has molded as follows:

$$w= (A_n \times B_n) + (B_n \times C_n) \quad (4.20)$$

Following is the briefing of the input to the multi-layer perceptron model:

- The first input to the proposed multi-layer perceptron based optimization is *Cost Control Metric* (π_1) that is responsible for computing cost involvement for constructing a set of reusable components of a new software project. The proposed system performs randomization to yield different forms of the cost involved in various software projects.
- The second input to the proposed multi-layered perceptron model is *Capability Maximization Metric* (π_2) that is responsible for computing different number of software projects with the presence of reusable components of the design. However, for better quality in each deliverable, the study considers only the software projects where the experimental value of design threshold nearly matches with a hypothetical threshold value.
- The third input to the multi-layer perceptron model is *Development Duration Metric* (π_3). As per mathematical modeling, the development duration is deliberated with the inclusion of duration with the incorporation of reusable design components.

- The last input to the optimization model using multilayer perceptron is *Error Metric* (π_4) which is a substitution of all the essential uncertainty factors that has acted as a research challenge for the mathematical modeling of the proposed system.

Therefore, the mathematical representation of the input to proposed multilayer perceptron O_{in} is denoted as,

$$O_{in} = \{\pi_1, \pi_2, \pi_3, \pi_4\} \quad (4.21)$$

The proposed modeling performs error minimization using supervised learning technique that involves comparing the current outcome with the anticipated result. This process progressively minimizes the rate of error leading to the generation of enhanced quality of design reusability. The proposed system utilized a unique and non-conventional mechanism to assess the optimization of design reusability, which has computed on the run-time. It should be noted that abstraction of the numerical value of design reusability has been already calculated by the mathematical model and hence there is no more requirement to re-compute that again. However, the proposed study estimates two discrete performance factor to ensure that proposed system offers a significant amount of predictive characteristics. The briefings of the output of the optimized model are as follows:

- *Consistency Factor* (χ_1): This output is responsible for computing the degree of consistency in the design reusability, which is depicted mathematically as,

$$\chi_1 = \sum \frac{O_{in} - a}{b} .w \quad (4.22)$$

The above mathematical expression uses two constants *a* and *b* which represents lower and higher limit of variances in input respectively. The probability theory was used to initialize the values of a and b. The variable *w* in the expression is assigned weight.

- *Complexity Factor* (χ_2): This output of the multilayer perceptron computes the degree of either variance or uniformity. If the generated outcome results with more uniformity than the analytical finding exhibit more reliability and predictive characteristics.

V. RESULT ANALYSIS

The proposed system uses OFBiz [13] that is an open source standard ERP application and four different software projects associated with CRM and SCM. The complete source codes are in Java which comprises of 151 classes, 2541 methods along with cumulative of 51,000 lines of source code approximately. The study uses a software tool called Metric 1.3.6 [110] to obtain the numerical values of five software metrics viz. CBO, WMC, RFC, DIT, and NOC. The study uses MyEclipse as development environment where Java code of test software projects are analyzed, and plugin of Metric 1.3.6 used to obtain the values that further retained in *csv (comma separated value)* form of data. The study also uses the random forest applied to data obtained from Metric 1.3.6 to obtain hypothetical outcomes. These outcomes are further compared with the experimental values to perform convergence testing, which is essential to prove the correctness of the mathematical modeling of the proposed system. The process of designing this hypothetical data is flexible to consider different forms of constraint in the proposed system.

The part of the analysis was carried out considering that junior software developer can handle a maximum of 2 software projects (i.e., $\phi_{max}(S_j)=2$) while senior software developer can complete a maximum of 3

software projects (i.e., $\phi_{max}(S_s)$=3) in one month with 26 working days practically. Hence, the study considers different forms of the complexity factor associated with the software projects concerning development duration in months, i.e., δ and actual development duration $|\omega|$. .

Table 5.1 Numerical Outcome duration of design reusability

γ_1	c	ρ	τ
6	20	8	960
6	40	8	1920
6	60	8	2880
6	120	8	5760
		τ=	11520

Table 5.1 represents the tabulation of the numerical attributes used as well as computed for the proposed case study of design reusability. The considered parameter under observation is γ_1 (design attribute), c (number of days spend to incorporate design reusability), ρ (duration of single person effort per day in hours), and τ (duration with design reusability in hours).

Table 5.2 Numerical Outcome of duration without design reusability

γ_1	c-1	ρ	μ
6	19	8	912
6	39	8	1872
6	59	8	2832
6	119	8	5712
		μ	11328

Table 5.2 highlights the similar values for γ_1 (design attribute) and ρ (duration of single person effort per day in hours). The study also considers similar step size increment for the days involved in developing software projects without consideration of design reusability. Hence, a notation of (c-1) is used to depict numbers of days spend to develop software project without incorporating design reusability. In the direction of proving the proposed theory, the study considers that initial value of (c-1) is slightly less than one from the value of c and there are good possibilities, which leads to lower values. Achieving the numerical value of the favorable outcomes is entirely dependent on the δ (project duration) and $|\omega|$ (actual project). Computation of this value will eventually give the information about an effective duration involved in developing software projects.

Table 5.3 Total and actual project duration for junior developer

Jr.Dev S_j (ϕ_{max}=2)		Development Duration δ (Months)	Actual Development Duration |ω| (months)	Total Development Duration	Total Actual Development Duration
Client-1	1-Project	6	4		
Client-2	1-Project	8	6	δ	|ω|
	2-Projects	14	10	364	260

Table 5.4 Total and actual project duration for senior developer

Sr. Dev S_j (ϕ_{max}=3)		Development Duration δ (Months)	Actual Development Duration |ω| (months)	Total Development Duration	Total Actual Development Duration
Client-3	1-Project	6	4		
Client-4	1-Project	8	6		
Client-5	1-Project	10	8	δ	|ω|
	3-Project	24	18	624	468

Table 5.3 and Table 5.4 highlights the assessment of effective capabilities of the software developers concerning ϕ (software projects) and δ (project duration). The study considers 1st two clients for junior software developer while rest three clients for a senior software developer as there are five software projects with mixed of ERP, CRM and SCM software project domain.

Table 5.5 Computed Values of analysis

(a) Duration analysis of Software Developer

Software Developers	δ	|ω|
Jr.Dev	364	260
Sr. Dev	624	468
	988	728

(b) Duration and design reusability analysis

δ	988
|ω|	728
T_{out}	260
γ	0.738461538

Table 5.5 highlights numerical values of δ (project duration), $|\omega|$ (actual project duration), T_{out} (total outcome), and γ (design reusability).

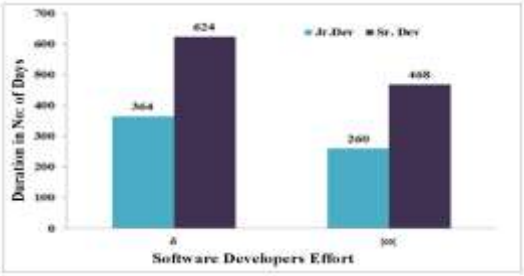

Figure 4. Analysis of Duration Involvement

The complete analysis is carried out considering the process of evaluating the capabilities of software developers working for 8 Hours as an effective hour. Hence, the duration required for the annual production of different software developers has graphically represented in Fig.6.1. It shows that junior software developer will consume 364 days allocation for project development whereas the actual duration involvement is only 260 days in one year. Similarly, the senior software developer will consume 624 days as total project duration but it will consume only 468 days in adherence to its capabilities defined during mathematical modeling.

The computation of favorable outcome F_{out} and total outcome T_{out} assists in finally computing the ultimate reusable probability γ.

$$\gamma = F_{out} / T_{out} = 192/260 = 0.73 \quad 5.1$$

Moreover, from the viewpoint of optimization principle, it is always anticipated that a software developer should offer good design reusability over their standard work shift duration. This effect can observe in Fig.5, where it can understand that increasing the duration ρ to a maximum practical limit of 8 hours per day offers a justifiable increase in the probability of design reusability. The increment of duration ρ more than 8 hours is probably not recommended for better accuracy in computing design reusability, and it strongly affects the productivity.

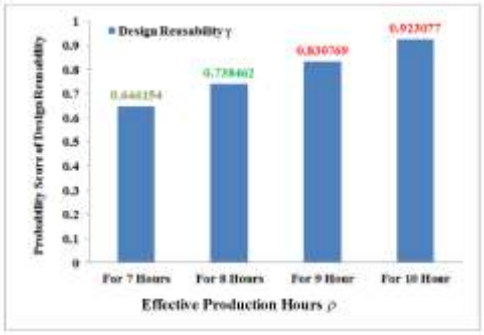

Figure 5 Analysis of Impact of Duration in Design Reusability

Another effective finding of the proposed analysis of the mathematical model is that there are different degrees of impact of different classes when all of them are subjected to a similar concept of design reusability. This analysis also exhibits that the two parameters γ_1 and $\Delta\gamma_1$ are more than enough for analyzing the probable effect of software metrics on design reusability. As the standard of CK metric does not offer much comprehensive information about cohesiveness of specific class, hence, the metric LCOM is not considered in the final analysis of design reusability. The summarized outcome of the software metrics with respect to corresponding classes are highlighted in Table 5.6, which shows that each class has a discrete effect on existing software metric for the object-oriented software projects.

Table 5.6 Summarized outcome of software metric

Test Cases	CBO		RFC		WMC		DIT		NOC	
	γ_1	$\Delta\gamma_1$	γ_1	$\Delta\gamma_1$	γ_1	$\Delta\gamma_1$	γ_1	$\Delta\gamma_1$	γ_1	$\Delta\gamma_1$
Class-1	0	0	0	1	1	-1	0	0	1	0
Class-2	1	-1	1	-1	0	0	0	0	0	0
Class-3	1	0	2	-5	4	0	0	0	0	0
Class-4	1	-1	1.4	-2	2	0	0	0	1	0
Class-5	1	-1	3.5	-2	2	3	0	0	0	0

The final analysis of the proposed study concerning the mathematical model has analyzed concerning three constraints, i.e., software projects, project duration, and development cost. Table 5.7 highlights the numerical relationship between software metrics and three constraints

Table 5.7 Cumulative data

CK-metrics	Software projects	Project Duration	Development Cost
CBO	2.62	5	2.57
RFC	3.36	6	2.44
WMC	1.47	8	2.30
DIT	1.00	4	2.77
NOC	1.63	4	2.77

The numerical and graphical outcome (Fig.6) shows that work schedule is one of the most dominant factors that have a significant effect on the design reusability. On the other hand, it can also state that WMC is one of the indicative software metrics that is found to be highly sensitive towards design reusability. The second software metrics is RFC that offers next better design reusability options while DIT and NOC offer similar reusability performance and hence they could be considered as optional software metric while analyzing reusability.

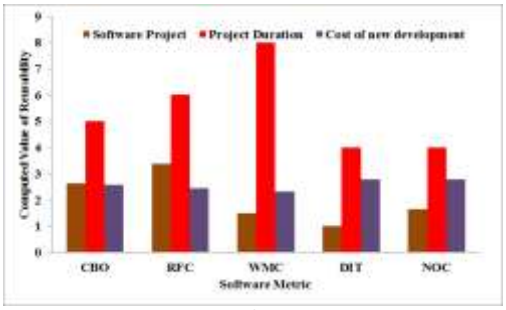

Figure 6 Analysis of Design Reusability

A. Result Analysis of Optimization Method

The proposed system performs optimization with the aid of a neural network. However, the intention of applying neural network is mainly to perform predictive analysis. The initial step of the optimization process using the neural network is to initiate the scopes and scale factor as shown in Table 5.8.

Table 5.8 Considered scaling factors for predictive analysis.

Nature of vector	Parameter	Minimum Value	Maximum Value	Scale Factor
Input Vector	Development Cost	0.3	0.5	0.7
	Software Project	2	3	4
	Project Duration	1	3	3
	Error	1	4	4
Output Vector	Consistency Factor	0.6	0.8	1.3
	Complexity Factor	4.9	13.8	18.7

The proposed study uses MATLAB for coding the logic associated with the curve fitting. The study uses Backpropagation algorithm for performing the training operation which induces a process of

lowering the error for performing prediction. Usually, it is a standard convention that the term *error* in a neural network is obtained by the difference of anticipated outcomes with experimental outcomes. Usually, the scope of the conventional error values is [0 to 0.05] which is also quite justifiable statistically. Hence, for a better scope and practical training implementation, the proposed system initializes the error value to be [0 to 0.04].

The numerical outcome of Table 5.9 is quite massive to be accommodated for proposed thesis chapter and hence only the significant outcomes have been highlighted. The outcomes were taken for software projects, project duration, cost, and error as an input while output consists of hypothetical and experimental values of consistency and complexity factors. After the accomplishment of the complete training process, it is found that that there is a no much significant difference between the anticipated outcomes and experimental outcomes for both consistency and complexity factor. The trend in the outcomes also exhibits that proposed system successfully minimizes the complexity and maximizes the consistency over an epoch of 2000. The significant contribution is that the better optimization trend can be observed with increasing the value of error. The optimization technique proves the robustness of the proposed mathematical model which also has an excellent predictive capacity.

Table 5.9 Numerical Analysis of Learning Process

Set no	Input				Output			
	C	Q	WS	E	CAF_{exp}	CAF_{pred}	CF_{exp}	CF_{pred}
2	0.325	2	3	1	0.843	0.843	6.8	6.8
18	0.5	2.5	3	1	0.92	0.921	4.9	4.9
31	0.375	2	3	2	0.853	0.850	6.9	6.8
52	0.45	3	3	2	0.852	0.853	7.4	7.3
72	0.3	2.5	3	3	0.876	0.873	7	7.0
95	0.4	2.5	3	4	0.775	0.775	10.5	10.5
108	0.3	3	3	4	0.817	0.817	8.6	8.7
124	0.45	2.5	2	1	0.89	0.891	5.8	5.7
130	0.375	3	2	1	0.838	0.841	7.2	7.2
137	0.325	2	2	2	0.805	0.808	8.2	8.2
153	0.3	2.5	2	2	0.877	0.876	6	5.8
163	0.35	2	2	3	0.789	0.789	8.9	8.9
172	0.3	2.5	2	3	0.748	0.744	11.6	11.4
197	0.475	2	2	4	0.822	0.823	8.2	8.2
204	0.425	2.5	2	4	0.765	0.763	10.1	10.1
219	0.35	2	1	1	0.854	0.854	6.5	6.4
233	0.475	2.5	1	1	0.905	0.904	5.2	5.3
242	0.475	3	1	1	0.888	0.890	6.1	6.1
248	0.4	2	1	2	0.826	0.824	6.9	6.8
252	0.5	2	1	2	0.982	0.982	5.6	5.6
276	0.425	2	1	3	0.805	0.810	8.1	7.9
297	0.5	3	1	3	0.773	0.772	8.5	8.6
307	0.3	2.5	1	4	0.674	0.673	13.9	13.8
324	0.5	3	1	4	0.734	0.737	10.1	10.1

[C=Cost, Q=Capability, WS=Work Schedule, E= % Errors, CAF_{exp}=Consistency Adoptability Factor (Experimental), CAF_{pred}=Consistency Adoptability Factor (Predicted), CF_{exp}=Complexity Factor (Experimental), CF_{pred}=Complexity Factor (Predicted)]

It should be noted that experimental data is captured before applying training algorithm while predicted data is computed after the training algorithm completes its execution process. The computation of the experimental data has been carried out by using data analytics software, e.g., SPSS while the training operation is carried out over the experimental data to generate predictive data which is generated only after the training process. Hence, it is essential to check the internal performance of the proposed system.

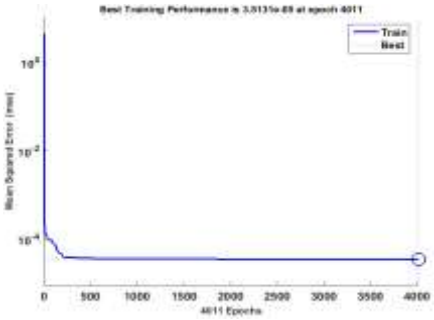

Figure 7 Analysis of mean squared error

Fig 7 highlights the performance of the proposed system concerning mean square error where it can be seen that the training curve coincides with the best fit curve. It also states that proposed modeling confirms a close match for the experimental data and the predicted data stating the higher degree of reliability of the prediction outcome as well as the technical correctness of the predicted outcome concerning accuracy. The observation was carried out for 20,000 epochs to find that there is no significant difference between the training outcome and best-fit curve outcome. This outcome was judged from the viewpoint of Root Mean Square Error (RMSE) as seen in Fig.8, where it states that the RMSE error has witnessed significant reduction and retains reduced error even in the long run.

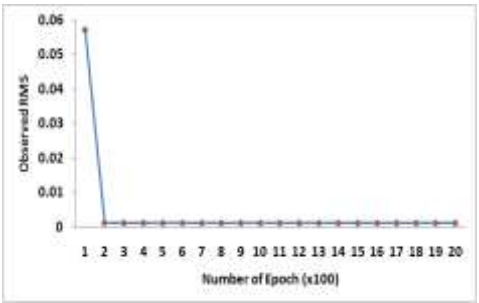

Figure 8 Analysis of Root Mean Square (epoch=20,000)

The study has also performed analysis of the regression factor to find the value of R as 0.99954 where the best fit between the data and the fitness curve has observed. The regression trend is quite smooth and predictive with extremely less occurrences of any sort of variations.

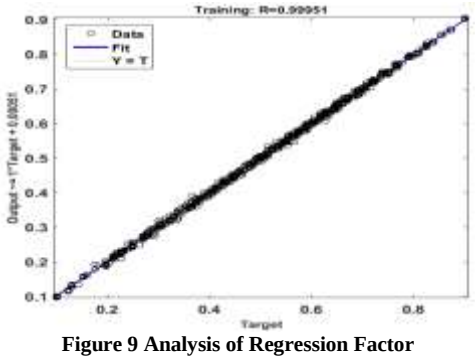

Figure 9 Analysis of Regression Factor

The significant contribution of the outcome obtained from the proposed study is that it establishes an essential relationship between the software metric and near real-time constraint associated with the successful implementation of design reusability in software development life cycle in SMEs. The next essential fact is that an implementation of the Backpropagation-based training approach has significantly assisted in optimizing the design reusability. This training approach was found to increase the precision of the computation as well as the accuracy of the prediction.

A closer look into the adopted methodology of the mathematical modeling will show that there are various forms of objective function to retain the research objectives of design reusability in presence of all the constraint. Therefore, the suitability of the Levenberg-Marquardt algorithm is much better for performing parallel optimization of all condition stated in objective function. Usage of sigmoid function as an activation function also assisted in better optimization that results in minimization of computational (or training) state and more evolution of reusable values. Hence, there is a closer relationship between using proposed machine learning system on optimizing design reusability. The accuracy of the proposed prediction has multiple and heterogeneous dependability factors along with consideration of the uncertainty factor called as an error. The proposed optimization has successfully addressed all the ranges of the practical errors using multi-layered perceptron approach.

At the same time, it was also seen that convergence of the outcome for proposed system is quite faster. The complete execution process to yield the outcome of consistency and complexity factor was found to be with a range of 0.2335 seconds to 0.7162 seconds for a data with more than 350 rows of information associated with the input factors. Although, an initialization of 20,000 epochs has been set, but the proposed system could successfully identify the precise outcome in much less than the initialized epoch. Hence, the proposed system could be termed as computationally cost effective mechanism of calculating design reusability as well as ensuring that it is going to be consistent over certain period of time. The faster response with accurate prediction significantly assists the stakeholder to take necessary decision.

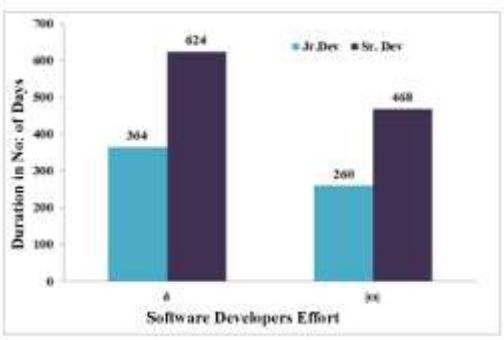

Figure 10 Analysis of Duration Involvement

The complete analysis is carried out considering the process of evaluating the capabilities of software developers working for 8 Hours as an effective hour. Hence, the duration required for the annual production of different software developers has graphically represented in Fig.10. It shows that junior software developer will consume 364 days allocation for project development whereas the actual duration involvement is only 260 days in one year. Similarly, the senior software developer will consume 624 days as total project duration but it will consume only 468 days in adherence to its capabilities defined during mathematical modeling. An organization consists of both junior and senior software developer. Therefore, it will consume 988 days of total duration allocated while its effective days will be 728 days where design reusability has incorporated. This evaluation will assist to compute the total value of the outcome. Hence, the ultimate reusable probability γ is 0.73, which is also an effective probability score.

VI. CONCLUSION AND FUTURE WORK

This paper presented a non-conventional mechanism for constructing a framework to leverage the design reusability for the sole purpose of enhancing software quality. The proposed mathematical model consists of four near real-time constraints, e.g., software developer, software projects, project duration, and cost. The objective functions are designed by considering the situation of SME associated with software project development. In this system neural network with Levenberg-Marquardt algorithm and Backpropagation is used for optimization. The sole emphasis of the work is oriented towards the cost-effectiveness of development strategy with design reusability. The computed outcomes were obtained using the available standard tool on open source Java projects.

REFERENCES

[1] K. Adams, "Non-Functional Requirements In Systems Analysis And *Design*", Springer-Technology & Engineering, 2015

[2] C. Nathan, *"Modular Web Design: Creating Reusable Components For User Experience Design And Documentation*", Pearson Education India, 2010

[3] T.R.Gopalakrishnan Nair and R. Selvarani. 2010. "*Estimation of Software Reusability: An Engineering Approach*", Association for Computing Machinery (ACM) – SIGSOFT, USA, Vol.35, Iss.1

[4] A. Khoshkbarforoushha, P. Jamshidi, M. F. Gholami, L. Wang and R. Ranjan, "*Metrics for BPEL Process Reusability Analysis in a Workflow System*," in IEEE Systems Journal, vol. 10, no. 1, pp. 36-45, March 2016.

[5] T. Diamantopoulos, K. Thomopoulos and A. Symeonidis, "*QualBoa: Reusability-aware Recommendations of Source Code Components*," 2016 IEEE/ACM th Working Conference on Mining Software Repositories (MSR), Austin, TX, 2016, pp. 488-491.

[6] L. Demraoui, H. Behja, E. M. Zemmouri, and R. Ben Abbou, "*A case-based reasoning approach to the reusability of CWM metadata,"* 2016 Third International Conference on Systems of Collaboration (SysCo), Casablanca, 2016, pp. 1-6.

[7] N.S. A.A. Bakar, "*The analysis of object-oriented metrics in c++ programs"*, Lecture Notes on Software Engineering, vol. 4, no. 1, pp.48, 2016

[8] R. Awad, G. Heppner, A. Roennau and M. Bordignon, "*ROS engineering workbench based on semantically enriched app models for improved reusability*," 2016 IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA), Berlin, 2016, pp. 1-9.

[9] S. Munk, "*Reusable interoperability components in military IT*", Technical Sciences, 2015

[10] T.R. Gopalakrishnan Nair, R. Selvarani. 2012. "*Defect proneness estimation and feedback approach for software design quality improvement"*, Information and Software Technology, Elsevier, vol.54, pp. 274–285

[11] G.T.R. Nair and R. Selvarani, "*Software reusability estimation model using metrics governing design architecture*", In Knowledge Engineering for Software Development Life Cycles: Support Technologies and Applications, pp. 196-209, 2011.

[12] C. Singh, A. Pratap and A. Singhal, "*An estimation of software reusability using fuzzy logic technique*," 2014 International Conference on Signal Propagation and Computer Technology (ICSPCT 2014), Ajmer, 2014, pp. 250-256.

[13] https://ofbiz.apache.org/

[14] 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

[15] Kumari, Rashmi, and Raksha Pandey. "*An Efficient Resource Scheduling Algorithm Using Knapsack.*" International Journal of Computer Science Engineering and Information Technology Research (IJCSEITR),7.2 (2017) 25-28.

[16] Shrimali, Arvind Kumar, and Vimlesh Kumar Soni. "*A review on issues of lean manufacturing implementation by small and medium enterprises.*" International Journal of Mechanical and Production Engineering Research and Development (IJMPERD) 7.3 (2017): 283-300.

[17] Gupta, S. L., and R. Ranjan. "*Impact of liberalization on contribution of MSMEs in economic development of India.*" International Journal of Business Management & Research (IJBMR) 4.4 (2014): 11-22.

[18] MEERADEVI, M., NANCY DAS, and P. MARIA DOSS. "*PERFORMANCE OF ICT ON SMALL AND MEDIUM ENTERPRISES IN CUDDALORE DISTRICT*." International Journal of Business Management & Research (IJBMR) 10.1 (2020):53–58

[19] SANJAYA, I. PUTU SUGIARTHA, and Y. HENDRASURYADHARMA. "*ACCOUNTING AND SMALL AND MEDIUM-SIZED ENTERPRISES: CASE FROM THE MEMBERS OF THE EX-MIGRANT WORKER COOPERATION OF KULON PROGOIN DAERAH ISTIMEWA YOGYAKARTA INDONESIA*." International Journal of Accounting and Financial Management Research (IJAFMR) 7.5 (2017):1-10

[20] Moses, Chinonye, Mosunmola Akinbode, and Prosper Onochie. "*Strategies for Financing Small and Medium Enterprises in Nigeria: Concepts and Issues*." International Journal of Business and General Management (IJBGM) 4.5 (2015): 25-38.

[21] TAIRI, HARUN, and NUHI SELA. "*ECONOMIC DEVELOPMENT HOLDERS IN REPUBLIC OF MACEDONIA THROUGH THE BUSINESS DEVELOPMENT*." International Journal of Business and General Management (IJBGM) 2.3 (2013):29-34