International Journal of Engineering and Advanced Technology (IJEAT)
ISSN: 2249-8958 (Online), Volume-9 Issue-6, August 2020

A Metrics-Based Fuzzy Logic Model for Predicting the Reusability of Object-Oriented Software

Kevin Agina Onyango, Geoffrey Muchiri Muketha, Elyjoy Muthoni Micheni

Abstract: Software reusability facilitates the engineering of new software or systems functionalities without having to start coding from scratch. This software quality provides numerous merits to the software developers including coming up with larger systems within a short time, with reduced development cost and reduced developer effort. However, software reusability is an external software quality attribute that is not measurable directly, and this is a major hindrance factor to its adoption in the software engineering process. This indicates that there is a need to come up with an approach to measure or predict software reusability to give advisories to the software developers on the prediction of software reusability. Therefore, this study aims at coming up with a prediction model to estimate software reusability by employing the use of a fuzzy logic approach, the study focused on Object-Oriented software. The research followed a guideline of four objectives and assumed descriptive and diagnostic methodology. It started by considering the existing ISO/IEC 25010 software quality model to determine and describe the main factors affecting the reusability of Object-Oriented software. The main factors identified were Abstraction, Cohesion and Coupling, thereafter, it was followed by the definition of metrics to measure each of the three factors identified, to operationalize these metrics, they were then validated using Weyuker's nine properties. After this, the researchers developed a metrics tool to automate the process of computing the metrics values of the defined metrics. Finally, a fuzzy logic prediction model that predicts the reusability of Object-Oriented software was developed based on the metrics values computed by the metrics tool. This model was then validated using the Analytical Hierarchical Process (AHP) framework. The outcome of the four objectives was presented, discussed and future works of this study outlines. This research work is a contribution to the software development industry where the software developers can apply it to engineer reusable software.

Keywords: Analytical Hierarchical Process, Fuzzy Logic, Metrics Tooling, Object-Oriented Software (OOS), Software Metrics, Software Reusability.

Revised Manuscript Received on August 15, 2020.
* Correspondence Author

Kevin Agina Onyango*, Department of Information Technology, Murang'a University of Technology, Murang'a, Kenya. Email: konyango@mut.ac.ke

Geoffrey Muchiri Muketha, Department of Computer Science, Murang'a University of Technology, Murang'a, Kenya. Email: gmuchiri@mut.ac.ke

Elyjoy Muthoni Micheni, Department of Management Science and Technology, The Technical University of Kenya, Nairobi, Kenya. Email: elyjoy.micheni@tukenya.ac.ke.

I. INTRODUCTION

Software is becoming inevitable in our daily operations, it has penetrated all the industries including telecommunication, automobile, personal entertainment and many others. Due to this, software developers are also in the run to try to increase their work to sustain this widespread adoption and demand for software. They incorporate the concept of software reusability where they are reusing existing legacy systems to engineer sophisticated systems in a faster way to meet this increasing demand for the software in various industries [1].

Software reusability refers to the ease in which the software developers can use an existing software asset, again and again, to engineer a new and advanced system or software functionality with little or no modification without starting the development process from scratch [2], [3]. The software assets in this scenario may include products and by-products of software development life-cycle e.g. sections of code, modules, test suites, design, interfaces, documentation [4]. Software reusability saves programmers from copy, paste and edit a block of a similar section of code during the development of large and sophisticated systems [1]. Other relevance associated with software reusability is that the software developers are in a position to come up with large systems within a short and limited schedule and with reduced developer effort. Industrial evaluation of this quality attribute also depicts that in software development, the reuse strategy saves up to 20% of the software development cost [5].

Many software methodologies including Module-Oriented, Aspect-Oriented, Component-Based as well as Object-Oriented methods have emerged in support of software reusability with diverse features. Object-Oriented (OO) approach stands out as among the suitable approach of software development methodology supporting reusability because it's modular structural composition [6]. This informed the choice of the methodology of concern in this study. In Object-Oriented Software, for a section of a code to be reused, it must be of high quality [4]. However, software reusability is an external software quality attribute that is not measurable directly making it impossible to directly determine the quality of software to be considered reusable in software [7].

This is one of the hindrance factors to the adoption of software reusability in many software development industries despite the rapid advancements in software coverage and consumption in the last decade [6], [8], [9]. Attributes like reusability are not measurable directly it is advisable to consider their internal quality attributes that are measurable directly.

In software engineering, the best approach to determine the internal software quality attributes is through the use of software quality models. This study aligns itself with ISO/IEC 25010:2011 Software Quality Model, an improved version of its predecessor ISO/IEC 9126 Software Quality Model to identify the factors internal factors affecting reusability. The internal quality attributes are therefore measurable directly to get an estimation of the software reusability. According to this software quality model, Abstraction, Cohesion and Coupling are the three major factors that contribute to software reusability in an Object-Oriented environment [2], [10]. Metrics definition was then done following Fenton's Entity-Attribute Model to capture the manifestation of these three factors in Object-Oriented software [11]. This resulted in the definition of three metrics to measure Abstraction, Cohesion and Coupling. These metrics were then validated using Weyuker's nine properties and the validation results showed that metrics conform to the intended study as they reported 77.78% compliance [12]. Metrics computation is a tedious process and can result in human-error when done manually, the researchers made effort and developed a metrics tool called Abstraction-Cohesion and Coupling Metric Tool-ACCMT that automates the computation of the metrics values for the defined metrics. The metrics tool starts by accepting the project to be analyzed then captures the metrics of concern from the block of project code and records the metric values, the tool makes this process to be more accurate and efficient [13], [14]. Finally, the output metrics values from the metrics tool formed the input variables for the proposed fuzzy logic model that uses these metrics values to predict the reusability of a given Object-Oriented software in a simulated experimented conducted in MATLAB [15]. The implementation of the proposed model was done using selected projects developed using Java programming language and the results depicted that metrics are suitable in the prediction of reusability in OOS. To affirm the validity and consistency of the prediction result, the proposed fuzzy logic model was validated using the Analytical Hierarchical Process (AHP) framework [16].

II. RELATED WORKS

It is noticeable that several studies have been done revolving around this research area where several researchers conduct studies to determine the software reusability of different paradigms. For instance: Dhand et al. [4] did a study on how to estimate software reusability from Object-Oriented metrics using a fuzzy logic model. This study considered only the six CK metrics set, which were further categorized into two; the first category included Coupling between Object Classes (CBO), NOC (number of children), FRC (response for a class) and WMC (weighted methods per class), the second category had DIP (depth of inheritance tree) and LCOM (lack of cohesion of methods). In the fuzzy logic model, it demonstrated that the first category measured the modularity aspect of coupling while the second category of metrics measured the modularity aspect of cohesion resulting in the modeling of two fuzzy controllers for each set of the categories. These metrics formed the input variables in the fuzzy model while the reuse factor formed the output variable. The result of the prediction by their model revealed that coupling metrics e.g. CBO has a negative correlation with reuse of software whereas cohesion metrics e.g. LCOM has a positive correlation with the reuse of software in OOP. The study concluded that for a system to be reusable the developer should lower the coupling and tighten the cohesion. However, this study only considered cohesion and coupling as the factors affecting reusability overlooking other factors that affect reusability in OOS. Gui and Scott [17] conducted a study to measure software reusability by cohesion and coupling metrics. This work categorized coupling metrics as coupling between object classes (CBO), RFC, coupling factor (CF), data abstraction coupling (DAC) and cohesion metrics as LCOM, improved of lack cohesion methods (LCOM3), ratio of lack of cohesion of methods (RLCOM), and TCC (Tight Class Cohesion). They carried out Linear Regression to show how to measure software reusability using cohesion and coupling metrics, the conclusion depicted that to have high reusable software, the developer should lower the usage of coupling and increase the usage of cohesion in its software development. Singh et al. [18] carried out a study to evaluate software reusability using software metrics through fuzzy logic. The study focused on Component-Based Systems and they used customizability, interface, complexity, portability, documentation level, and observability as the metrics of concern. They concluded that for external quality prediction or evaluation, fuzzy logic is one of the best approaches to adopt. Singh et al. [6] proposed a framework for evaluating software reusability utilizing the fuzzy logic strategy for AOS (Aspect-Oriented Software). This study listed separation of cohesion, concern, size, and coupling metrics as their metrics of concern. These metrics formed the input variables in modeling their Fuzzy Inference System (FIS). They also concluded that fuzzy logic gives a better approach in assessing software reusability, on the same note, their results showed that low coupling and high cohesion combination leads to high reusability. Singh and Tomar [19] proposed a model implementing a fuzzy logic perspective to measuring the complexity of component-based software in their study. In this study, cohesion metrics, coupling metrics, and newly defined interface complexity formed the metrics of concern. The result of this proposed model concluded to show that the fuzzy logic approach is suitable to predict factors like software complexity. Singh et al. [5] presented a study on an approximation of software reusability making use of the fuzzy logic strategy based on component-based systems. In this study, they considered Interface, Modularity, complexity, Adaptability, Maintainability, and Flexibility, as the main factors that affect software reusability. The study concluded that the fuzzy logic model is a good prediction that can be used to predict the reusability of applications. In their future works, they proposed an extension of this study to cover other domains like Object-Oriented. Singh et al. [20] conducted a study on software reusability evaluation implementing soft computing approaches.

This study assessed the reusability of component-based systems by considering the complexity, documentation quality, interface, software understandability, and changeability, as the main metrics for the study. In their modeling, the researchers did the prediction using various soft computing approaches, they used Neural-Network, Fuzzy logic, and Neural Fuzzy. They concluded that these techniques can be best used to assess software quality attributes from metrics values. However, they noted that fuzzy logic gives a simpler interpretation of the prediction results compared to the other soft computing techniques used.

III. MATERIALS AND METHODS

The proposed model in this study considered reusability as a function of directly measurable factors as documented in the ISO/IEC 25010 Software Quality Model which is a successor of ISO 9126 software quality model [2], [3]. These factors are attributes of source-code that identify the reusability of Object-Oriented software and they include Abstraction, Cohesion and Coupling [2], [7], [10]. Three metrics were defined to measure each factor identified resulting in Abstraction Complexity Metric, Cohesion Metric and Coupling Metric. Weyuker's nine properties were then used to validate these metrics and they showed a high level of compliance for the intended study [12]. After this, the researchers developed a metric tool called Abstraction, Cohesion and Coupling Metric Tool - ACCMT to automate the process of computing the metrics values of the metrics defined from the selected projects. Finally, the output metrics values from the ACCMT metrics tool formed the input variables for the proposed fuzzy logic model which simulates them into one output function forming the predicted reusability of the projects evaluated [15], [21].

Therefore, the methodology used in this study followed the following steps:

- Identification of the factors affecting the reusability of Object-Oriented software.
- Metrics definition to measure each of the factors identified.
- Metrics tooling to automate the computation of the metrics values of the defined metrics.
- Development of a fuzzy logic model based on the metrics values to predict the reusability of Object-Oriented software.

A. Factors affecting Reusability of Object-Oriented Software

One of the most suitable software paradigms that support reusability is Object-Oriented. Programming languages under this paradigm like Java uses modular programming style, where modules form the building blocks of software development. However, modules can affect the reusability of systems developed using this approach. For instance, the inter and intra-dependencies of modules in the system can make such system fragile and rigid, lowering the ease of its reuse since alteration of one module has a ripple-effort to the other modules that may, in turn, compromise the operation of the entire system. To overcome this, the developers must make sure that the reusable modules are of higher quality, however, reusability is an external software quality attribute that is not measurable directly. Therefore, for better measurement of such attributes, their internal attributes that are measurable directly are considered [1]. This study considered ISO/IEC 25010 software quality model to identify the internal software quality attributes that affect reusability. This model documents Abstraction, Cohesion and Coupling as the three main factors affecting software reusability in an Object-Oriented environment [2], [7], [10]. Figure 1 shows an excerpt of this model.

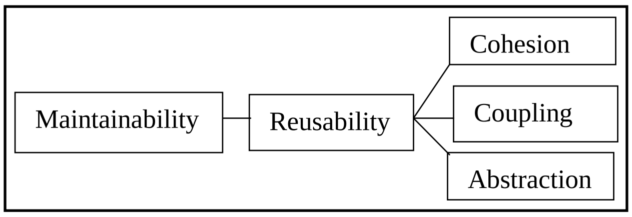


Figure 1: An Extract of ISO/IEC 25010:2011 Software Quality Model [2]

These three factors measure the reusability of Object-Oriented software at the source-code level. For instance, when using Abstraction, a programmer will define abstract methods in an abstract class which cannot be instantiated but will act as a code template (or parent classes) to be extended and implemented repeatedly by concrete methods in the concrete classes (child classes) in future to offer different functionality of the system with no or little modification without having to rewrite the entire code again, this will save the programmer a lot of time and effort during the software development process, hence achieving reusability [2], [22].

Cohesion measures the intra-dependency within and among the module's internal elements. High cohesion is desirable in software development since it enhances modules to have a single and well-focused purpose. This makes such a module reusable since a change in that module will not have a ripple effect on the other modules in the system [17], [23].

Coupling is a complete inverse of Cohesion since it shows the inter-dependency between system modules i.e. the measure of how close two modules are connected. Programmers and software developers are advised to employ loose Coupling in software development [8].

This is because having a tight Coupling means that a change in one module enforces a change in other dependent modules in that system and this compromises the operation of the whole system hence modification, maintainability and reusability of such a system are also reduced [8], [17].

B. Metrics for the Study

Three metrics were then defined to measure each of the factors identified to be affecting software reusability. The metrics definition process followed Fenton's Entity-Attribute-Metric (EAM) model [11].

The first metric defined was Abstraction Complexity Metric (ACM) which is an extension of the Method Hiding Factor of the MOOD's metric suite [24], [25], [26].

ACM is computed by calculating the number of abstract strategies in all Java classes then dividing them with the count of all the total number of methods (both abstract and concrete methods) characterized in all the classes.

The value obtained from ACM gives the overall ratio which helps in giving advisories to the Java programmers and system designers on the complexity of abstraction in any given Java program at the system level. The following equation shows the definition of the ACM metric.

$$\text{ACM} = \frac{\sum_{i=1}^{n}\{NAM_{(Ci)}\}}{\sum_{i=i}^{n}\{NAM_{(Ci)}+NCM_{(Ci)}\}} \quad (1)$$

Where ACM is the Aggregate Abstraction Complexity Metric value for the system, NAM (ci) is the Number of Abstract Methods in a class and NCM (ci) is the Number of Concrete Methods in a class.

Cohesion Metrics (CohM) was the second defined metric that extends TCC (Tight Class Cohesion) of the Class Cohesion Metrics by Bieman and Kang [23], [27], [28]. CohM metric considers modules or classes that can execute independently (independent modules) and those that depend on others to execute (dependent modules). To get the overall metric value to give the cohesion value at the system level. The defined metric computes the ratio of the total number of classes/modules that are independent (NIM) to the total number of classes/ modules (TNM) defined in a system. The equation of the metric is as shown below.

$$\text{Cohesion Metric (CohM)} = \frac{\sum_{i=1}^{n}\{NIM\}}{\sum_{i=1}^{n}\{TNM\}} \quad (2)$$

Where CohM is the cohesion metric value for the system, NIM is the number of independent modules in the system and TNM is the total number of modules defined in the system both dependent and independent.

The last metric defined was Coupling Metrics (CopM) which extends CBO (coupling between object classes) of the CK metrics suite [23], [28], [29], [30]. CopM metric considers modules or classes that depend on other modules for their execution (dependent modules) and those that do not depend on others for execution (independent modules). Therefore, to get the overall metric value to give the coupling value in a system, the defined metric will compute the proportion of the total number of classes/modules that are depending on other modules (NDM) to the total number of classes/ modules (TNM) defined in a system, the equation of the metric is as shown below.

$$\text{Coupling Metric (CopM)} = \frac{\sum_{i=1}^{n}\{NDM\}}{\sum_{i=1}^{n}\{TNM\}} \quad (3)$$

Where CopM is the coupling metric value for the system, NDM is the number of dependent modules in the system and TNM is the total number of modules defined in the system both dependent and independent.

Project scenarios were then used to demonstrate the computation of each of the defined metrics.

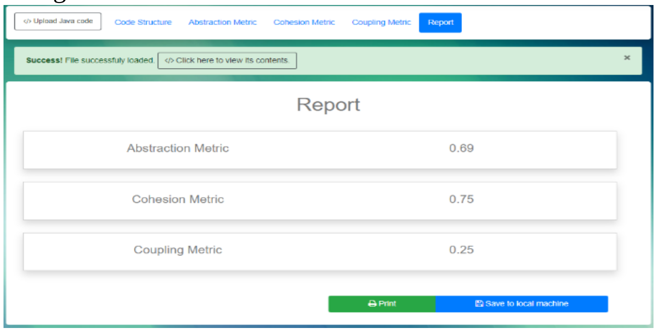
Finally, the metrics were then validated using Weyuker's nine properties and the validation results showed that metrics conform to the intended study giving a compliance rate of 77.78% which is a good indication that the defined metrics are theoretically sound and can be used for the intended purpose [12].

C. Metrics Tooling

This study uses a web-based programming language to develop a metrics tool called Abstraction Cohesion and Coupling Metric Tool – ACCMT. This tool computes the metrics values for Abstraction, Cohesion and Coupling of the selected projects for this study that were developed using Java programming language. It is a good practice to develop a static metric analyzer tool for newly defined metrics. This practice boosts the acceptance level of the programmers to newly defined metrics in the software industry [14], [31]. Research shows that several metrics developed without supporting tools to automate their metric value computation suffer from several critics during their applications. Therefore, ACCMT is in line with the proposed metrics of ACM, CohM and CopM.

This tool accepts projects with Java syntax then tokenize the program and count the tokens that meet the structures of concerned then record the values as the metrics values for Abstraction, Cohesion and Coupling in independent tabs of the tool interface. The final tab of the tool presents the summary report of the three metric values of concern as the output of the tool.

During the implementation of the tool, nine projects having different complexity levels were analyzed by the tool. The results show that ACCMT can give accurate output and within a very short time, hence, it is an indication that this tool is suitable to automate the computation of the proposed metrics. A sample demonstration of the output from the tool is shown in Figure 2.



Success! File successfuly loaded.

Report

Abstraction Metric	0.69
Cohesion Metric	0.75
Coupling Metric	0.25

Figure 2: Computed Metrics by ACCMT Metric Tool

D. Development of a Fuzzy Logic Model

Fuzzy logic is a superset of the traditional Boolean logic that uses the rule-based logic to come up with multi-valued logic and the truth value ranges anywhere between completely false and completely true [15], [32]. When modeling the proposed fuzzy logic model, the computation was done using MATLAB r2020a while the modeling was performed making use of the Fuzzy Logic Toolbox following the steps outlined below:

1) Definition of the Linguistic Terms

Defining the proposed fuzzy logic model considered three input variables viz Abstraction, Cohesion and Coupling. Each of these input variables was denoted with three linguistic terms viz high, medium and low. The model, conversely had one output variable named Reusability described using five linguistic terms viz very high, high, medium, low and very low.

2) Formulation of the Membership Functions

The input variables entered into the model came from the metrics tool in the form of crisp values and the output resulting from the simulation process was then converted back to crisp value to give the reusability prediction. However, the model cannot do the inferencing to give the prediction using the crisp values, therefore, the crisp values from the metrics tool were first converted into fuzzy sets during the fuzzification process to enable the inferencing to take place [33]. In this study, all the membership functions were ranging between 0-1 mapped from the ranges of the metrics values considered for the input space from the metrics tool.

Triangular membership function (trimf) was used to formulate membership functions of the fuzzy sets for both inputs and output variables. Triangular membership function gives the best fitness in the determination of input and output values to the degree to which they belong [5], [34].

Figure 3 indicates the membership responsibility for the first input variable called "Abstraction". Fuzzification of Abstraction input variable having three linguistic variables viz medium, high, and low.

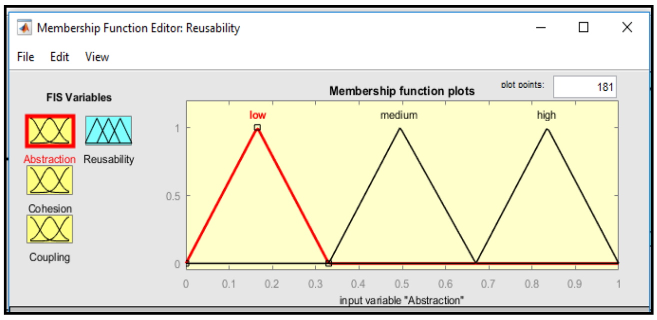


Figure 3: Membership Function for input Variable "Abstraction"

Figure 4 shows the membership function for the second input variable called "Cohesion". Fuzzification of Cohesion input variable having three linguistic variables viz high, medium, and low.

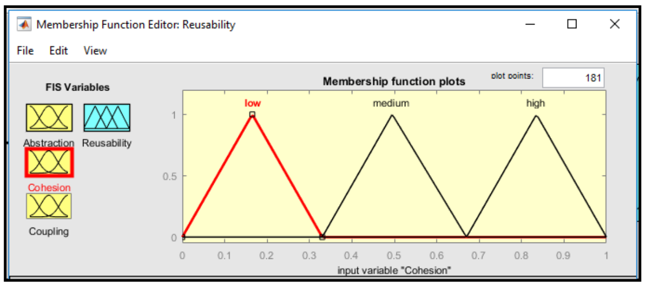


Figure 4: Membership Function for input Variable "Cohesion"

Figure 5 shows the membership function for the last input variable called "Coupling". Fuzzification of Coupling input variable having three linguistic variables viz medium, high, and low.

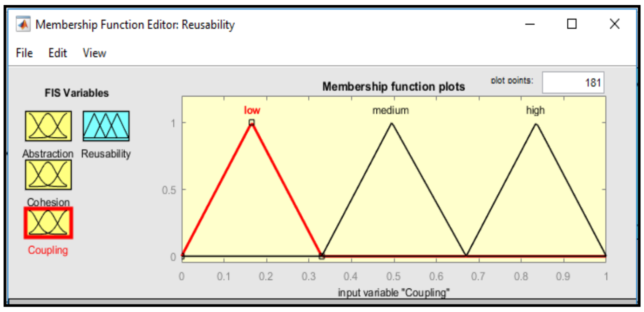


Figure 5: Membership Function for input Variable "Coupling"

Subsequently, the output variable "Reusability" was also modeled assuming five linguistic variables viz very high, high, medium, low, very low as evident in Figure 6

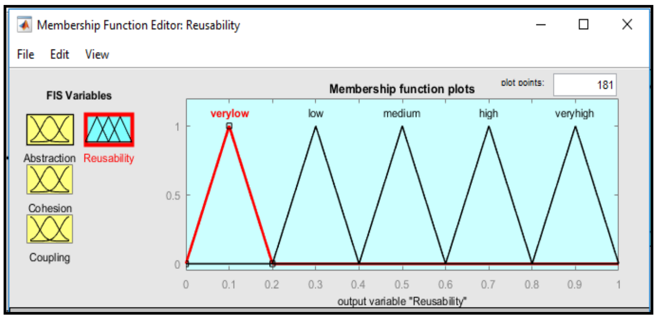
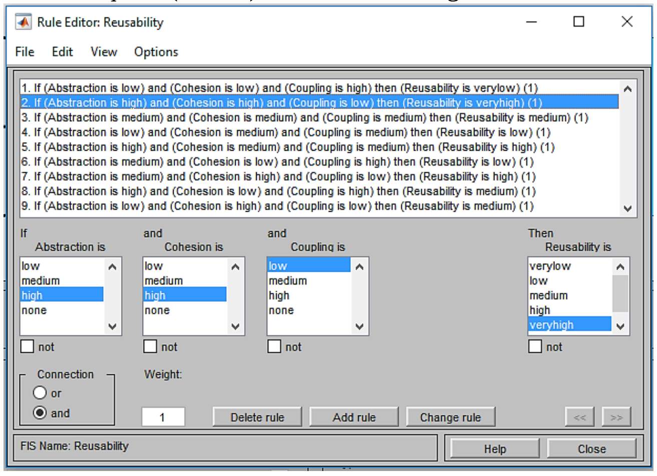


Figure 6: Membership Function for output Variable "Reusability"

3) Formulation of the Knowledge Base

The proposed model is composed of three inputs each formulated into a fuzzy set of three membership functions of low, medium, and high. Therefore, the total possible combination of rules is 27 that is 33 If-Then rules. MAMDANI inference-type was used to create the knowledge base or the inference engine in this study using the MATLAB's "Rule Editor" interface.

The expert domain and the literature as well as other theoretical truisms about the relationship between the variables formed the basis for the construction of the rules. The process to formulate the rules started by choosing a single membership task of the input variable boxes in the antecedent (IF) side and a connector that links the input variables to the corresponding membership function of the output variable in the consequent (THEN) side as seen in Figure 7.


Rule Editor: Reusability

1. If (Abstraction is low) and (Cohesion is low) and (Coupling is high) then (Reusability is verylow) (1)
2. If (Abstraction is high) and (Cohesion is high) and (Coupling is low) then (Reusability is veryhigh) (1)
3. If (Abstraction is medium) and (Cohesion is medium) and (Coupling is medium) then (Reusability is medium) (1)
4. If (Abstraction is low) and (Cohesion is medium) and (Coupling is medium) then (Reusability is low) (1)
5. If (Abstraction is high) and (Cohesion is medium) and (Coupling is medium) then (Reusability is high) (1)
6. If (Abstraction is medium) and (Cohesion is low) and (Coupling is high) then (Reusability is low) (1)
7. If (Abstraction is medium) and (Cohesion is high) and (Coupling is low) then (Reusability is high) (1)
8. If (Abstraction is high) and (Cohesion is low) and (Coupling is high) then (Reusability is medium) (1)
9. If (Abstraction is low) and (Cohesion is high) and (Coupling is low) then (Reusability is medium) (1)

If Abstraction is: low, medium, [✓ high], none
and Cohesion is: low, medium, [✓ high], none
and Coupling is: [✓ low], medium, high, none
Then Reusability is: verylow, low, medium, high, [✓ veryhigh]

Connection: or [✓ and]
Weight: 1

FIS Name: Reusability

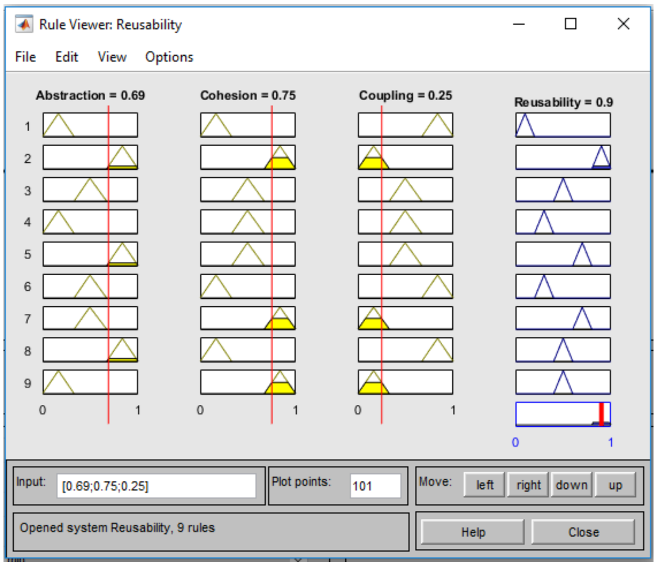
Figure 7: Rule Editor

4) Defuzzification

Centroid defuzzification had been largely used in previous similar studies because of its robustness [6], [16]. This study also adopts the centroid defuzzification to assign the fuzzy sets for both the input and output variables. Using Centroid defuzzification, calculation of the parameter values for input and output variables takes place by considering the central value between the respective variables to get the incremental parameter values. For instance, in this study, the input variables range between 0-1, in this case, 0.33 was the centroid incremental parameter value for the input membership function while incremental parameter value for output membership functions is 0.2. Using this calculation, the input variables ranged as follows: 0-0.33 representing low, 0.33-0.67 falling under medium while 0.67 – 1.0 covered under high. The output variable assumes 0-0.2 as very low, 0.2-0.4 covered under low, 0.4-0.6 falling under medium, 0.6-0.8 is covered under high while 0.8-1.0 is under the very high category.

5) Predicting the Reusability of Selected Projects

The prediction of Reusability occurs by entering the values of all the three input variables for each project evaluated. The input values are the metrics values generated from the analysis of the projects by the ACCMT metric tool. GitHub as the source to get the projects where nine projects developed using Java programming language were selected for the study. When one of the nine projects was analyzed by the ACCMT metric tool for demonstration, it recorded the following metrics values: ACM = 0.69, CohM= 0.75 and CopM= 0.25. To predict the reusability of this project, these output metrics values from the ACCMT tool were entered directly into the fuzzy logic Rule Viewer of the proposed model as crisp input variables. The resulting output of the simulation returned a crisp output value of 0.9. From the calculated parameter values, this output parameter value of the prediction falls under a very high membership function of the output variable. Hence, this simulated indicates that the analyzed project has very high reusability as illustrated in the MATLAB's Rule Viewer interface of the proposed model in Figure 8.


Rule Viewer: Reusability

Abstraction = 0.69
Cohesion = 0.75
Coupling = 0.25
Reusability = 0.9

Input: [0.69;0.75;0.25]
Plot points: 101

Opened system Reusability, 9 rules

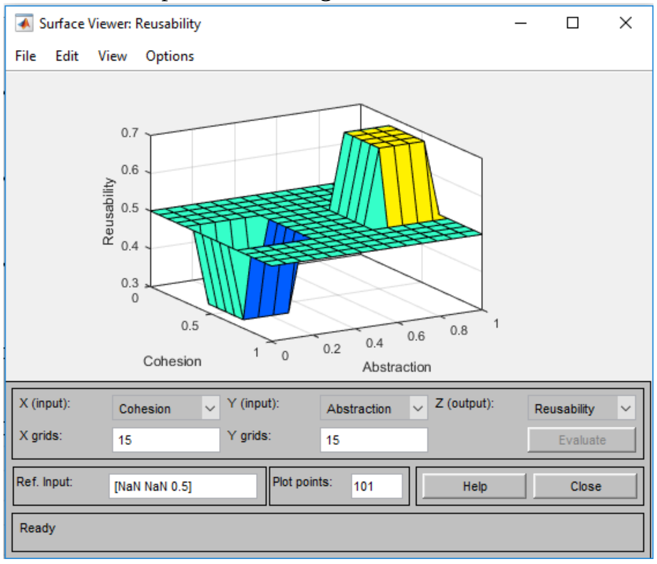
Figure 8: Rule Viewer with the Predicted Result

The prediction output of the experimented results from the nine projects are shown in Table 1.

Table 1: Prediction of the Output Results

Java Projects	Abstraction	Cohesion	Coupling	Reusability
P1	0.25	0.25	0.75	0.10
P2	0.69	0.75	0.25	0.90
P3	0.40	0.50	0.50	0.50
P4	0.25	0.50	0.50	0.30
P5	0.71	0.50	0.50	0.70
P6	0.36	0.25	0.75	0.30
P7	0.50	0.75	0.25	0.70
P8	0.70	0.25	0.75	0.50
P9	0.20	0.75	0.25	0.50

The prediction output analysis depicted that the Abstraction factor runs parallel with Cohesion and Coupling factors and gives a curve as shown in the 3D surface viewer of the MATLAB presented in Figure 9.

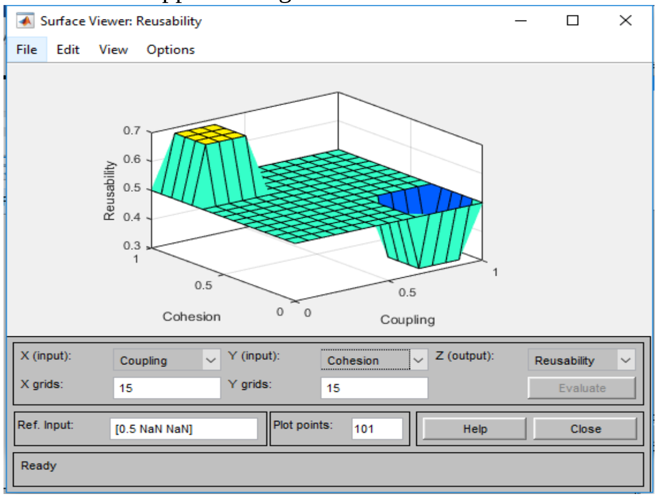

Surface Viewer: Reusability

X (input): Cohesion
Y (input): Abstraction
Z (output): Reusability
X grids: 15
Y grids: 15
Ref. Input: [NaN NaN 0.5]
Plot points: 101

Ready

Figure 9: Correlation of Abstraction and Cohesion with Reusability

The analysis of the output results as well shows that Cohesion and Coupling on the other hand tend to move in a perfect inverse direction. The curve in Figure 10 shows that when Cohesion and Coupling are plotted, the output gets skewed in an opposite diagonal direction.



X (input): Coupling
Y (input): Cohesion
Z (output): Reusability
X grids: 15
Y grids: 15
Ref. Input: [0.5 NaN NaN]
Plot points: 101
Ready

Figure 10: Correlation of Coupling and Coupling with Reusability

IV. VALIDATION OF THE PROPOSED MODEL

Validation is a vital process in any research as it affirms that the results obtained in a study are valid and consistent. Therefore, before discussing the prediction results, validation was done using the Analytical Hierarchical Process (AHP) framework to ascertain that the tool is giving valid prediction results [35].

The validation process of AHP contains three major sequential steps namely: creating a hierarchical structure, creating pair-wise comparison matrices and calculating consistency [16]. These steps were followed when validating the proposed model in this study starting with the creation of the AHP hierarchical structure for reusability prediction, the structure was designed with reusability as the goal in the first layer, Cohesion, Coupling and Abstraction as the criteria is the second layer and the alternatives in the last layer were represented by the nine projects (1-9) whose reusability were predicted. Figure 11 shows the AHP Hierarchical structure for reusability prediction in this study.

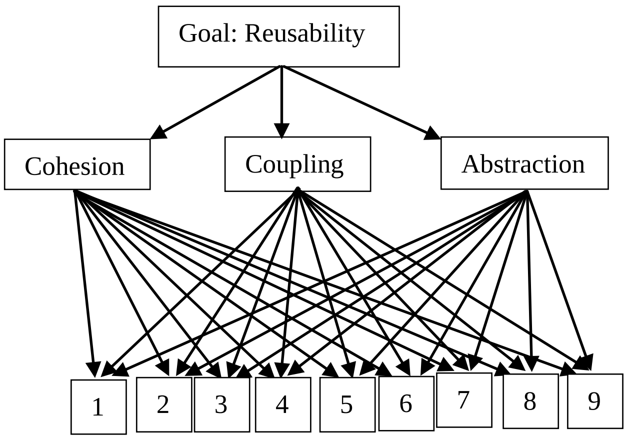


Figure 11: AHP Hierarchical Structure for Reusability Prediction

In step 2 of the validation process, the pair-wise matrix was created to compare different pairs. This matrix gives relative importance of the different attributes or criteria concerning the goal. The relative ranking of the attributes is guided by a Likert scale of 1-9 provided by the Saaty scale [16]. This scale assigns criteria with the extreme importance value of 9, very strong importance is assigned 7, 5 is assigned to criteria with strong importance, 3 is assigned to moderate importance while 1 is assigned to criteria with equal significance. 2,4,6,8 acting on the behalf of the transitional values and 1/3, 1/5, 1/7 and 1/9 covering the values for the inverse comparison of the criteria [16]. When creating a pair-wise comparison matrix, the numerical rating should be aligned with the ranking of the attributes of the model being validated [36]. In this study, the corresponding matrix to compare pair-wise values is the matrix is created by considering the attributes used to model the prediction model as shown in Table 2.

The values in the pair-wise matrix depend upon the decision-making formula used when modeling the model being validated. For instance, when model the proposed model, it was ascertained that Cohesion factor is strongly preferred than Coupling factor when developing a reusable software, Cohesion factor was the as well moderately strong than Abstraction factor, Abstraction factor was equally moderate preferred to Coupling factor when developing a reusable Object-Oriented software, and these were obtained from the expert opinions from the literature. Therefore, relating these judgments to the Saaty scale lead to formation of expression such that: If Coupling = x value then Cohesion = 5x value and if Abstraction = y value then Cohesion = 4y value and if Coupling = z value Abstraction = 2z value.

The following formula is used to populate the pair-wise comparison matrix [35].

$$\frac{\text{Row elements}}{\text{Column elements}} \quad (4)$$

Table 2: Pair-wise Comparison Matrix

	Cohesion	Coupling	Abstraction
Cohesion	1	$\frac{5x}{x} = 5$	$\frac{4y}{y} = 4$
Coupling	$\frac{x}{5x} = \frac{1}{5}$	1	$\frac{z}{2z} = \frac{1}{2}$
Abstraction	$\frac{y}{4y} = \frac{1}{4}$	$\frac{2z}{z} = 2$	1

The sum of each value is calculated to obtain a non-normalized pair-wise comparison matrix. The computation was done and the result is presented as shown in Table 3.

Table 3: Non-normalized pair-wise Comparison Matrix

	Cohesion	Coupling	Abstraction
Cohesion	1	5	4
Coupling	0.2	1	0.5
Abstraction	0.25	2	1
SUM	**1.45**	**8**	**5.5**

The next operation was to create a standard pair-wise comparison matrix. This is achieved by dividing all the elements of the column with the respective sum of the column, this was done and the normalized pair-wise contrasting matrix shown in Table 4.

First column: 1/1.45 = 0.6897
0.2/1.45 = 0.1379
0.25/1.45 = 0.1724
Second column: 5/8 = 0.625
1/8 = 0.125
2/8 = 0.25
Third column: 4/5.5= 0.7272
0.5/5.5= 0.0909
1/5.5= 0.1818

Table 4: Normalized pair-wise Comparison Matrix

	Cohesion	Coupling	Abstraction
Cohesion	0.6897	0.625	0.7272
Coupling	0.1379	0.125	0.0909
Abstraction	0.1724	0.25	0.1818

The normalized pair-wise comparison matrix values are used to calculate the criteria weight. The approach weights are sometimes called Eigenvectors; they are calculated by averaging all the elements in a respective row in the normalized pair-wise matrix. This was done as shown in the working below and the result presented in Table 5.

$$\text{Critical Weight} = \frac{(\sum_{r=1}^{3} \text{Row elements of the normalized pair wise matrix})}{3} \quad (5)$$

Where r is the number of rows
For Row 1= (0.6897 + 0.625 + 0.7272) / 3
= 0.6806

For Row 2= (0.1379 + 0.125 + 0.0909) / 3
= 0.1179
For Row 3= (0.1724 + 0.25 + 0.1818) / 3
= 0.2014

Table 5: Normalized pair-wise comparison matrix with criteria weights

	Cohesion	Coupling	Abstraction	Criteria Weight
Cohesion	0.6897	0.625	0.7272	0.6806
Coupling	0.1379	0.125	0.0909	0.1179
Abstraction	0.1724	0.25	0.1818	0.2014

Step 3 is the calculation of Consistency. This is the last step of the validation process; it checks whether the computed values are incorrect or correct. To calculate the consistency, the column values from the non-normalized pair-wise comparison matrix are multiplied by respective Criteria weights or Eigenvector value. This was done as shown in the working below and the result presented in Table 6.

First column: 1 * 0.6806 = 0.6806
0.2 * 0.6806 = 0.1361
0.25 * 0.6806 = 0.1702
Second column: 5 * 0.1179 = 0.5895
1* 0.1179 = 0.1179
2 * 0.1179 = 0.2358
Third column: 4 * 0.2014= 0.8056
0.5 * 0.2014= 0.1007
1 * 0.2014= 0.2014

Table 6: Consistency Matrix

	Cohesion	Coupling	Abstraction
Cohesion	0.6806	0.5895	0.8056
Coupling	0.1361	0.1179	0.1007
Abstraction	0.1702	0.2358	0.2014

The consistency matrix values are used to calculate the weighted Sum value sometimes called the sum of the weighed Eigenvector value. To calculate this, all the values in the row of consistency matrix are added. This was done as shown in the working below and the result presented in Table 7.

$$\text{Weighted Sum Value} = \sum_{r=1}^{3} \text{Row elements of the Consistency Matrix} \quad (6)$$

Where r is the number of rows

First Row = (0.6806 + 0.5895 + 0.8056)
= 2.0757

Second Row = (0.1361+ 0.1179 + 0.1007)
= 0.3547

Third Row = (0.1702+ 0.2358 + 0.2014)
= 0.6074

Table 7: Consistency Matrix with Weighted Sum

	Cohesion	Coupling	Abstraction	Weighted Sum value
Cohesion	0.6806	0.5895	0.8056	2.0757
Coupling	0.1361	0.1179	0.1007	0.3547
Abstraction	0.1702	0.2358	0.2014	0.6074

Weighted sum values are then used to calculate the ratios which give the comparison between the matrices of criteria weight and he weighted sum from the consistency matrix. This was done as shown in the working below and the result presented in Table 8.

$$\text{Ratio} = \frac{\text{Weighted Sum value}}{\text{Criteria weights}} \quad (7)$$

Therefore,

For Row 1= 2.0757 / 0.6806
= 3.0498

For Row 2 = 0.3547 / 0.1179
= 3.0085

For Row 3 = 0.6074 / 0.2014
= 3.0159

Table 8: Consistency Matrix with Ratios

	Cohesion	Coupling	Abstraction	Ratios
Cohesion	0.6806	0.5895	0.8056	3.0498
Coupling	0.1361	0.1179	0.1007	3.0085
Abstraction	0.1702	0.2358	0.2014	3.0159

These ratios are then used to calculate the maximum Eigenvalue demoted as Lambda max (λmax) calculated by getting the average of the ratios as shown in the working below.

$$\lambda max = \frac{(\sum_{c=1}^{3} \text{Column elements of the consistency matrix with ratios})}{3} \quad (8)$$

Where c is the number of columns

λmax = (3.0498 + 3.0085 + 3.0159) / 3

= 9.0715 / 3

λmax = 3.0238

The computed λmax is then used to compute the CI (consistency index). The importance of the Consistency Index is to affirm whether the model being validated gives inconsistent predictions [37]. This helps to make sure that the decisions made by the model are reliable and the decision-maker gives consistent results for the projects whore reusability is being predicted.

For instance, if the decision-maker affirms that when developing a reusable Object-Oriented software, Cohesion factor is more important in the software development process than Abstraction factor and that Abstraction factor is as well more important than Coupling factor, then it would be inconsistent to affirm that Coupling factor is more important that Cohesion factor i.e.:

If X > Y and Y > Z THEN it would be inconsistent to say that Z > X.

The formula for calculating the Consistency Index (CI) is given [36], as:

$$\text{Consistency Index } (CI) = \frac{\lambda max - n}{n - 1} \quad (9)$$

Where n is the number of criteria or the factors used for the prediction, in this study n = 3.

Therefore,

$$(CI) = \frac{\lambda max - n}{n-1}$$

$$= \frac{3.0238-3}{3-1}$$

$$= \frac{0.0238}{2}$$

= 0.0119

Finally, the Consistency Index (CI) was used to calculate the Consistency Ratio (RI), this is used to verify that the Consistency Index (CI) is adequate and that the model is giving valid predictions. The formula is given [16], as:

$$\text{Consistency Ratio (CR)} = \frac{\text{Consistency Index (CI)}}{\text{Random Index (RI)}} \quad (10)$$

The value for the CI is taken from the previous calculation generated from the λmax while the Random Index (RI) is obtained from a standard scale given [16]. RI (random index) is the CI of the randomly created pair-wise matrix. The Consistency Ratio is considered adequate if the resulting value is less than 10% or less than 0.1.i.e.

$$\text{Consistency Ratio (CR)} = \frac{\text{Consistency Index (CI)}}{\text{Random Index(RI)}} < 0.1{\sim}10\%$$

To calculate the Consistency Ration of the proposed model, the Random Index (RI) is picked from the standard predefined scale. The proposed model in this study used 3 criteria which from the standard RI scale is assigned a standard RI of 0.58. Therefore, the Consistency Ratio (CR) was calculated as seen in (10):

$$\text{Consistency Ratio (CR)} = \frac{\text{Consistency Index (CI)}}{\text{Random Index (RI)}}$$

$$= \frac{0.0119}{0.58}$$

= 0.0205 ~ 2.05%

These results from the AHP validation framework affirmed that the proposed fuzzy logic model is giving reasonably consistent and valid prediction results.

V. RESULTS AND DISCUSSION

The Consistency Ratio (CR) calculated from the pair-wise comparison matrix from the proposed fuzzy model in this study is CR = 0.0205 ~ 2.05% and the standard CR recommended is that CR should be $< 0.1{\sim}10\%$. This implies that the proposed model produces reasonably consistent prediction results and so it can be implemented in the decision-making process for the process to predict the reusability of OOS.

The validation results weigh the criteria or the factors identified to be affecting reusability and giving the results as shown in Table 9.

Table 9: Criteria Weights for the Factors Affecting Reusability

Criteria Weights	Factors Affecting Reusability
0.6806	Cohesion
0.2014	Abstraction
0.1179	Coupling

The results imply that Cohesion is the most important factor when developing reusable software as it gives the highest contribution weight of 68.06% or 0.6806, the second important factor that contributes to reusability of Object-Oriented software is Abstraction by 20.14 % or 0.2014 and the last factor that should be incorporated when coming up with reusable Object-Oriented software is Coupling since it has the least contribution to reusability with a weight of 11.79% or 0.1179.

VI. CONCLUSION AND FUTURE WORKS

This study reports Coupling to be detrimental to the reusability of Object-Oriented software such that an increase in Coupling lowers the reusability of software. The use of Coupling in software development indicates that most of the modules interlink themselves with one another and therefore a change in one module can alter the result of the dependent module. This makes that software prone to faults and hence becomes difficult to maintain and reuse. Cohesion and Abstraction on the other hand are seen to have a positive impact on reusability. This implies that an increase in Cohesion and Abstraction increases the reusability of Object-Oriented software. High Cohesion in a system implies simplicity and increases reusability, in that such a system has modules that are self-contained and does not depend on other modules for their operations. Modification of one module in such a system does not compromise the execution results of the other modules. Hence it is easier to maintain and reuse systems with high Cohesion factors. High abstraction in a system means that are more abstract methods that are still capable of being extended and implemented in the future hence the reusability of such a system is high.

The proposed model proves to be effective for predicting the reusability of Object-Oriented software, this was affirmed by conducting the validation of the model using the AHP framework. Employing such a model in the early stages of the software development life cycle (SDLC) can ease the software developers from the strenuous development effort and also enable them archives the development target within the recommend time and within the project cost. Reusing components of existing software or a legacy system reports an increase in the productivity of software development. The increased number of open-source software is an indicator that many software development industries have started to appreciate the concept of software reusability.

In conclusion, this study gives an advisory to the software developers that during software development of Object-Oriented software, they should increase the usage of Cohesion and Abstraction and at the same time lowers the usage of Coupling to achieve reusable software. Besides, the study concludes by contributing to knowledge and establishes that metrics are suitable in the prediction of external software quality attributes such as reusability. In the future, the results of this study will be further validated using human experts' experiments (empirical validation) to predict the reusability of OOS.

REFERENCES

1. Ibraheem Y.Y. Ahmaro, Abdallah M. Abualkishik and Mohd Zaliman Mohd Yusoff, (2014). Taxonomy, Definition, Approaches, Benefits, Reusability Levels, Factors and Adaption of Software Reusability: A Review of the Research Literature. Journal of Applied Sciences, 14: 2396-2421.
2. ISO/ IEC CD 25010 (2011). Software Engineering: Software Product Quality Requirements and Evaluation (SQuaRE) Quality Model and guide. International Organization for Standardization, Geneva, Switzerland.
3. ISO/IEC 9126-1 (2001). Software engineering – Product quality. http://www.iso.org/iso/catalogue_detail.htm? csnumber=22749. Accessed 18th July 2014.
4. Pooja Dhand, Parwinder Kaur Dhillon, Jagmohan Mago. (January, 2015)"Estimating Software Reusability from OO Metrics using Fuzzy Logic". Apeejay Journal of Computer Science and Applications, Vol. (3), January, 2015.
5. Charu Singh, Amrendra Pratap and Abhishek Singhal. (2014) "An Estimation of Software Reusability using Fuzzy Logic Technique". Conference Paper · July 2014. https://www.researchgate.net/publication/268195207.
6. Pradeep Kumar Singh, Om Prakash Sangwan, Amar Pal Singh, Amrendra Pratap, (January, 2015) "A Framework for Assessing the Software Reusability using Fuzzy Logic Approach for Aspect Oriented Software".I.J. Information Technology and Computer Science, 2015.
7. Dumke, R. (2008). Software process and product measurement: international conferences IWSM 2008, MetriKon 2008, and Mensura 2008, Munich, Germany, November 18-19, 2008: proceedings. Berlin: Springer-Verlag. (Supporting factors affecting reusability plus ISO 25010 model).
8. Heath, N. (2019). Future of the Java programming language. BS Interactive. Retrieved from https://www.techrepublic.com/article/future-of-java-programming-language-three-major-projects-on-the-horizon/.
9. Sebesta, r. w. (2012). Concepts of Programming languages (10th Ed.). Retrieved from https://www.startertutorials.com/ppl/books/sebesta.pdf.
10. Sammy Olive Nyasente, Prof. Waweru Mwangi, Dr. Stephen Kimani (2014). A Metrics-based Framework for Measuring the Reusability of Object-Oriented Software components. Journal of Information Engineering and Applications www.iiste.org. ISSN 2224-5782 (print) ISSN 2225-0506 (online), Vol.4, No.4, 2014. (Supporting factors affecting reusability plus iso 25010 model).
11. Fenton, N. and Bieman, J. (2014). "Software Metrics: A Rigorous and Practical Approach", 3rd Edition, Chapman & Hall/CRC Innovations in Software Engineering and Software Development Series.
12. Weyuker, E. J. (1988). Evaluating software complexity measures. IEEE Transactions on Software on Software Engineering, 14:(1357-1365).
13. Dhawan, S., & Kiran. (2012). Software Metrics – A Tool for Measuring Complexity. International Journal of Software and Web Sciences (IJSWS), 2(1), 4-7.
14. Ndia, J.G. (2019). Structural Complexity Framework and Metrics for Analyzing the Maintainability of Sassy Cascading Style Sheets (Doctoral dissertation, MMUST).
15. Moore, H. (2017). MATLAB for Engineers. Pearson.
16. Saaty, T. L. (2005). Theory and applications of the analytic network process: Decision making with benefits, opportunities, costs, and risks. Pittsburgh: RWS Publications.
17. Gui Gui and Paul. D. Scott. (Septermber, 2009) "Measuring Software Component Reusability by Coupling and Cohesion Metrics". Journal of Computers, vol. 4, no. 9, September 2009.
18. Manoj Kumar Singh, Govind Kamboj, Avnish Kumar Sharma. (June, 2014). "An Evaluation of Software Re-Usability Using Software Metrics through Fuzzy Logic". Journal of Engineering Research and Applications www.ijera.com ISSN: 2248-9622, Vol. 4, Issue 6(Version 5), June 2014, pp.66-72.
19. Aditya Pratap Singh, Pradeep Tomar, (2014). "A Fuzzy Logic Approach to Measure the Complexity of Component-based Software". Journal of Software Engineering Tools & Technology Trends. Volume 1, Issue 2. www.stmjournals.com.
20. Yogesh Singh, Pradeep Kumar Bhatia and Omprakash Sangwan. (2011) "Software Reusability Assessment Using Soft Computing Techniques". ACM Sigsoft Software Engineering, January 2011 Volume 36 Number 1.
21. Yakimenko, O. A. (2019). Engineering Computations and Modeling in MATLAB®/Simulink®. American Institute of Aeronautics and Astronautics, Inc...
22. Roberts, E. (2016). Programming Abstractions in Java (1st ed.; Illustrated, Ed.). Pearson, 2016.
23. Hermadi, I., El-Badawi, K., & Al-Ghamdi, J. (2002). Theoretical validation of cohesion metrics in object oriented systems. In International Arab Conference on Information Technology (pp. 16-19).
24. Brito e Abreu and Carapuça. (1994). "ObjectOriented Software Engineering: Measuring and controlling the development process." 4th International Conference on Software Quality, Mc Lean, VA, USA.

25. Brito e Abreu and Melo. (1996). Evaluating the Impact of Object-Oriented Design on Software Quality. 3rd International Metric Symposium, 90–99.
26. Brito e Abreu F., O. L. and G. M. (1998). "The MOOD metrics set." INESC/ISEG Internal Report.
27. Bieman, J. M. and Kang, B-Y (1995). Cohesion and Reuse in an Object-Oriented System. In Proc. ACM Symposium on Software Reusability (SSR'95). (April 1995) 259-262.
28. Serebrenik, A. (2011). Software metrics. Dostopno na: http://www. win. tue. nl/aserebre/2IS55/2011-2012/10. pdf.
29. Chidamber, S. R., & Kemerer, C. F. (1991). "Towards a Metrics Suite for Object Oriented Design." Conference on Object-Oriented Programming: Systems, Languages and Applications (OOSPLA 91), Published in SIGPLAN, 26(11), 197–211.
30. Chidamber, S. R., & Kemerer, C. F. (1994). A metrics suite for object oriented design. Tse, 20, 476–493. https://doi.org/10.1109/32.295895
31. Omri, S., Montag, P., & Sinz, C. (2018). Static Analysis and Code Complexity Metrics as Early Indicators of Software Defects. Journal of Software Engineering and Applications, 11(04), 153.
32. Zadeh," Fuzzy Sets", Information and Control, vol-8, pages:338-353, 1965.
33. Mohammadian, M. (2020). Modelling, control and prediction using hierarchical fuzzy logic systems: design and development. In Robotic Systems: Concepts, Methodologies, Tools, and Applications (pp. 187-207). IGI Global.
34. Kreinovich, V., Kosheleva, O., & Shahbazova, S. N. (2020). Why triangular and trapezoid membership functions: A simple explanation. In Recent Developments in Fuzzy Logic and Fuzzy Sets (pp. 25-31). Springer, Cham.
35. Vargas, R. V. (2010). Using the analytic hierarchy process (AHP) to select and prioritize projects in a portfolio. Paper presented at PMI® Global Congress 2010—North America, Washington, DC. Newtown Square, PA: Project Management Institute.
36. Kassab, M., & Kilicay-Ergin, N. (2015). Applying analytical hierarchy process to system quality requirements prioritization. Innovations in Systems and Software Engineering, 11(4), 303-312.
37. Dhir, S., Kumar, D., & Singh, V. B. (2017). Requirement paradigms to implement the software projects in agile development using analytical hierarchy process. International Journal of Decision Support System Technology (IJDSST), 9(3), 28-41.

AUTHORS PROFILE

Kevin Agina Onyango is is a Graduate Assistant in the Department of Information Technology, Murang' a University of Technology. He received his BSc. in Information Technology from Murang'a University of Technology, and is currently pursuing his M S c . in Information Technology at Murang'a University of Technology. His research interests mainly include software engineering, software measurement, program analysis and soft computing.

Geoffrey Muchiri Muketha is an Associate Professor and Dean School of Computing and Information Technology, Murang' a University of Technology, Kenya. He received his BSc. in Information Science from Moi University, his MSc. in Computer Science from Periyar University, and his Ph.D. in Software Engineering from Universiti Putra Malaysia. His research interests include software and business process metrics, software quality, verification and validation, empirical methods in software engineering, and component-based software engineering. He is a Professional Member of the ACM and a member of the International Association of Engineers (IAENG).

Elyjoy Muthoni Micheni is a Senior Lecturer of Information Systems in the Department of Management Science and Technology at The Technical University of Kenya. She holds a Ph.D. in Information Technology from Masinde Muliro University of Science and Technology, MSc. in Computer Based Information Systems from the University of Sunderland, Bachelor of Education from Kenyatta University, and Post Graduate Diploma in Project Management from the Kenya Institute of Management. She has researched on and taught Management Information System courses for many years at university level. She has presented papers in scientific conferences and has many publications in refereed journals. She has also co-authored a book for Middle-level colleges entitled: "Computerized Document Processing". Her career objective is to tap computer-based knowledge as a tool to advance business activities, promote research in ICT and enhance quality service.