International Research Journal in Global Engineering and Sciences. (IRJGES)
ISSN: 2456-172X | Vol. 4, No. 3, September - November, 2019 | Pages 60-74

# Restructuring Distributed Object-Oriented Software using NeuralNetworks

S Saravanan Professor, Department of CSE, Agni College of Technology, Chennai, India

Gopinathan S, Assistant Professor, *Department of CSE, Agni College of Technology, Chennai, India*

## *Abstract*

*For the distributed software systems evolvement, ObjectOriented (OO) approach is employed by engineers with designers in the precedent era which results in Distributed Object Oriented (DOO) systems. The chief facet of DOO systems stands as the competent dispersion of software classes amongst diverse nodes. The primary design of DOO applications has no top-class distribution, hence restructuring has to be done. The DOO software restructuring is done via a proposed adaptive technique called Neural Network (NN), to intensify the performance further. Initially, Class Dependency Graph (CDG) is constructed, in which the nodes represent the classes, and also the connections betwixt the nodes represent the dependencies betwixt the classes. Now, the factors of objects, methods, variables, lines, and import linked with the classes in the CDG are extracted and given as inputs to the NN for the training process. Now, clustering of the trained features is done by which the OO system is segmented into subsystems with low coupling using Class Dependency Based Clustering (CDBC) technique. Now, the clustered classes are accumulated into cluster graphs using K-Medoid clustering technique and finally, the mapping is done with the created partitions to the fixedavailablemachines using Recursive K Means clustering in the targeted distributed architecture. Simulation outcomes exposed that the proposed work yields enhanced outcomes in an effectual way compared to the existingtechniques.*

***Keywords:*** Distributed Object Oriented systems, Class Dependency Graph, Recursive Graph Clustering, Low Coupling, Neural Network and DistributedArchitecture.

## 1. Introduction

With the developments in the technology, software applications delivered an immense ease intended for economic augmentation besides scientific advancement. In the interim, the losses on account of the malfunction of software are rising. The enhancement of the software product's quality becomes an urgent issue aimed at the software engineers [1]. By utilizing the software source code which includes integration level, unit level along with system level, Structural quality is examined. The union of structural along with functional quality expands the complete software quality. Software quality contributes a noteworthy part in the expansion of the software [2]. The constant growth of complexity, criticality, and pervasiveness in software systems forge software quality more significant in software developments. Nevertheless, to make certain a system with no faults, it required to be profoundly tested, but in mass of the instance testing is unfeasible. Consequently, development teams are created to concentrate on their testing attempt on software system [3]. With the evolution of the software on its swiftness, the system design experienced limitless changes, features get removed integrating current situation, and consequently the quality of software characteristic as well experience a sequence of limitless changes[4].

Software design along with the quality is often times become casualty of restricted budgetary. The top-notch languages, C#, Java can't swap the requirement for a straightforward modular design [8]. Normally, restructuring techniques support the source code to redeem itsqualitysubsequenttoitsmaintenanceoperations,occasionallyevenprior to the product's initial release, manufacturing its elements clearer, and more reusable beside more cohesive [5].

Refactoring mentions to the restructuring of software. A change developed in software to enhance its interior quality without varying its peripheral behavior is called refactoring. Enhancing the non-functional features of the code, say, modularity, understandability with flexibleness is referred as its key objective [6]. Refactoring together with maintenance are 2 dissimilar processes; the segment of refactoring chiefly relate to the tenability of the already present software. Refactoring stands diverse in that the maintenance together with advancement that straightforwardly modifies the actions of the software, correspondingly, to spot and clear a bug or affix or else ameliorates functionality [7]. Therefore, Software refactoring enhances the software's interior quality devoid of affecting its peripheral behavior. Consequently, it stands as an effectual and comparatively safe mode to ameliorate the software quality[9].

The software system advancement stands as a time expending continuous process. Software system typically undergoes a succession of little and large changes over an extent [10]. To find the enhanced solutions aimed at software engineering processes andsoftware products, considerable software development paradigms was presented. Amongst these examples, for numerous years OO software development is obtaining fame [11]. In OO, a program is systematized as established interacting objects; every one encompasses its individual private state instead of the established functions which share a global state. Ideas of dynamic binding, encapsulation, inheritance, along with polymorphism form the OO paradigm's base. The objects were distributed and then executed either in parallel or in sequential as the additional facet of OO application[12].

A revolutionary software paradigm termed Object-oriented programming (OOP) stands as the existent meta-model developed by classes together with objects utilizing theories of abstraction, inheritance, polymorphism et cetera Those theories assist the establishment of reusable and also maintainable programming modules [13]. Aimed at the progression of distributed system of software, OO approach is engaged by engineers with designers in the precedent era for resolving complex issues in numerous scientific areas. A major phase in modeling DOO methodology is to settle on thelocations ofobject. This phase lessens the communication requirement in unreachable locations [14]. This DOO application is generated utilizing numerous sorts of software platforms which convey portability, communication, with interoperability. The technologies explicit to these platforms work as an imperative segment of the system implementation subsequent to system design, impacting the architecture and also functionality [15]. The primary draft of the DOO application doesn't certainly possess the top notch distribution. In these issues, the solution is attained via 2 configurations: either the hardware is reconfigured to peer with the software components (hardware reconfiguration), otherwise the software is reconfigured to peer with the accessible hardware (software restructuring) [14].

The draft arrangement of the paper is systematized as:

Section 2 surveys the associated works regarding the proposed method,
Section 3 briefly discusses the proposed technique; Section 4 estimates the investigational result Section 5 deduces this paper.

## 2.Literature Survey

Gu *et.al* [16] suggested cohesion based on complex networks (CBCN) metric, developed principally grounded upon calculating class average clustering (CAC) coefficient through graphs demonstrating connectivity motif of the several class members. Additionally, the CBCN metric was evaluated with theoretical justification bestowing with 4 possessions (cohesive modules, non- negativity along with normalization, monotonicity, maximum and null values) of class cohesion metric (CCM). The CBCN metric grounded upon data comparison with prevailing seventeen typical CCM of class cohesion system seemed supreme to all others. Application of CBCN metric to 3 open source software (OSS) to compute CAC coefficients, the outcome depicted that class required modification, understanding, and maintenance in an OSS system was less challenging compared with others. Three OSS systems had power-law dispersions for the CAC coefficient, which made probable of further comprehension of the CBCN.

Ajienka*et.al* [17] intended to associate the efficacy of gauging semantic coupling (SC) of OO software classes utilizing 1) modest identifier centered methods 2) the corpora of the complete classes in a software system. Subsequently, they investigated the interaction betwixt SC and change coupling (CC). On the relations betwixt logical and semantic dependencies, in 79 OO and OSS projects, there was not any linear relation betwixt the logical and semantic dependencies in relation to strength. Yet, they recognized a bilateral relation betwixt semantic to logical dependencies. Therefore, over seventy percentages of classes which were semantically associated will regularly co-evolve as well as classes which were changed associated would regularly share some degree of SC. The outcomes exposed that: (a) Identifier- centered techniques contained more computational competence except can't constantly use interchanging with corpora-based techniques of calculating SC classes (b) there was not any relationship betwixt SC and CC. Furthermore, they discovered that (c) there stands direct relations betwixt them, as above seventy percent of the semantic dependencies were as well connected via CC whereas not reversely possible. More comparison result was needed.

Kumar *et.al* [18] recommended a generic concept of OO framework multiple attribute quality method which scientifically regarded as quality features besides its sub-features based on diverse views and also the practice of Web applications. The key perception of this recommended framework was to inspect and enlarge the formerly established quality features in Web- centered applications. This method was selected for the impulsive and diverse nature of the Web applications. For manipulating this framework, International Organization for Standardization (ISO) and the International Electro technical Commission (IEC) 9126 framework was utilized as a locus. In this generic type method, the conventional software along with Web-based application's quality was accessed utilizing the similar framework. The hierarchical nature of aspects disturbing diverse perspectives was certainly added, modified or deleted as per requisite and Web application. Augmenting the depth and perspectives of hierarchy (by combining parameters of quality to features, sub-features, metrics, and sub-metrics) further improved the quality factor accuracy of this recommended framework. The model therefore resultant was malleable sufficient to compact with the entire software applications along with fundamentaltechnologies.

Chhabra and Parashar [19] presented unpredictable measures intended for classes specifically, Class Change-Impact Set and Change-Coupling (CC) Index. For these measures initially, the set of change committed was extracted to mine the CC amongst the classes, in addition, unpredictable measures of classes were calculated. The suggested changeability prediction frameworksupported the developer in numerous way, say, forecast of the degree of CC of a class; the architecture was validated via contrasting both CC and designed amongst classes together with correct planning of maintenance tasks via comprehending the apparent effect of class changes by change impact set. The measures proposed were empirically confirmed and specific research inquiries were replied to confirm the practicality of the change-stream. The acquired outcomes were consistent and indicated that suggested change-data stream centered changeability forecast was utmost helpful aimed at the software systems maintenance. Further specific research queries were responded to confirm the dependability of changeability measures. As of these, change data streams registered as change-history of classes were engaged to discover change-coupling in addition helped to envisage the forthcoming changeability configuration was perceived.

Nucci*et.al* [20] scrutinized the software elements changed through centered developers should as well with no inclination to error considering segments reformed by fewer absorb developers. That paper made an additional advance, by investigating the work undertaken by the developer's dispersion of bug prediction. specifically, they characterized 2 events i) by considering the quantity of code segments as structural scattering (the developer reforms segments depending on their structural spreading) ii) by regarding the obligations that was performed as semantic scattering. The characterized events were

evaluated as bug forecasters in an experiential scrutiny done on twenty-six open source frameworks. The achieved outcomes established the depth of their model and also its advanced complementarities concerning the modest methods. They as well generate and also experiment a hybridized prediction model over the 11 forecasters conquered by the 5 competitive methods. The accomplished results established that (a) the hybridized model accomplished an enhanced accuracy as for each single isolated five models;
(b) the forecasters recommended playing a notable segment in the finest performing hybridized prediction models.

Aslam and Ijaz [21] recommended a methodology called task allocation. It encompasses 2 phases: Phase 1 finds the dependencies and aspects which impacted the decision of task allocation; Phase 2 suggests a measureable technique which consigns tasks for the respective best team members. Task requisites were uttered as capabilities, catering intended for diversified aspects for instance technical, personal in addition to environment. One vital decision was allotting tasks to the associates of the team. That decision was undertaken qualitatively and not in a deterministic means, therefore involved hazards stayed aimed at successive future projects. This method considered the capability necessities aimed at every role requisite to apply in the project. Aimed at every single role, there were choices of the finest team membergrounded on the precedent working experience length, precedent appraisal intended for completing similar tasks and also relative prominence of capability necessities. Their method reside obvious to the targeted goals in contrast with the best match. Other targets are presenting quality along with price expediently; addressing numerous goals. Such targets likewise permitted quality assessment of task-member assignment amid furthermore subsequent to the project conclusion, which results in diminishing related risks.

## 2. Restructuring DOO Software Using NeuralNetwork

OO techniques form applications extensively simple to construct by contributing an advanced platform for application development. Grounded in the DOO approaches there are numerous projects to elucidate intricate problems in diverse scientific areas. The utmost central aspect of DOO systems stands as the well-organized distribution of software classes amongst diverse nodes to resolve the mismatch issue which may happen once the structure of software doesn't suit the prevailing hardware organization. The DOO software restructuring is done by via a proposed adaptive technique called NN, to intensify the performance further. This technique declines the aggregate of clusters which is initiated by the NN training and thus declining the aggregate of resources allocated. Initially, CDG is constructed, in which the nodes signify classes and also the connections betwixt the nodes represent the dependencies betwixt the classes. Now, the factors of objects, methods, variables, lines, and import linked with the classes in the CDG are extracted and given as inputs to the NN for the training process. Now, clustering is done by which the OO system is segmented into subsystems with low coupling using Class Dependency Based Clustering (CDBC) technique. Now, the clustered classes are accumulated into cluster graphs using K-Medoid clustering technique and finally, the mapping is done with the created partitions to the fixed available machines using Recursive K Means clustering in the targeted distributed architecture. The proposed methodology's framework is clarified inFig.1.

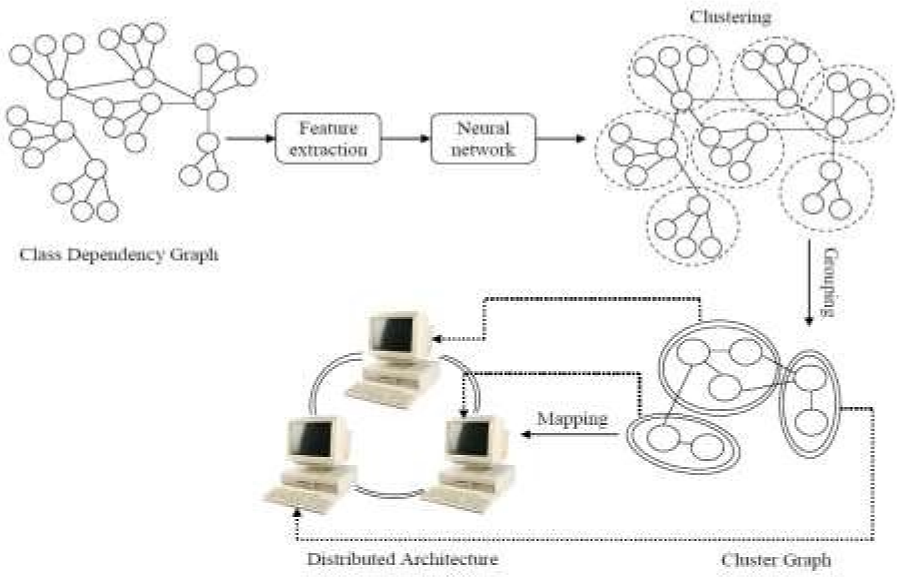


**Fig1.**The Proposed Methodology

## Class DependencyGraph

Initially, every single class is allotted to a distinct node of the distributed system to precisely assess the communication events amongst classes in a DOO system. The intended values are utilized to construct the Class Dependency Graph (CDG) of the OO system. The CDG is validated in Figure2.

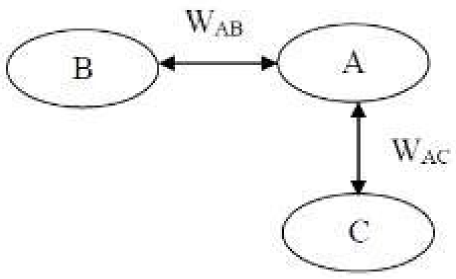


**Fig2.** Class Dependency Graph

In CDG, the vertex is denoted class and also an edge betwixt class A and B denotes a communication event which occurs betwixt these two classes on data transfer or classes' dependency. The weight of the edge$W_{AB}$ denotes the cost of the communication events betwixt class A and class B. when no relationship dependency or data communication betwixt two classes exist, no edge will join them in the CDG.

## FeatureExtraction

After the construction of CDG, features of objects, variables, methods, import and lines accompanied by the classes in the CDG are extracted.

- **Object**

Object stands as a central unit of OOP and signifies the tangible entities. A distinctive Java program builds many objects, which interrelate by appealing methods. An object entails behavior, state, and identity.

- **Variables**

In OOP with classes, Regardless of the existence of numerous class instances, in a class, a class variable is defined with a single copy and is never an instance variable. It stands as a distinct form ofclass attribute (or field, or else class property, datamember).

- **Method**

A procedure which is accompanied by an object and a message is labeled as a method in OOP. It is a programming module that comprises sequences of statements which execute a task. Methods also convey the interface which other classes utilize to modify and access an object's dataproperties.

- **Import**

Import files are utilized to pass user-defined and also built-in packages into the java source file in order that the individual class can pertain to a class of another package by directly using itsname.

- **Lines**

Lines indicate the total lines written in a program which includes both the used and unused lines.

**NeuralNetwork**

In the previous step, the extracted features are delivered as inputs to the NN for the training process. In NN, the weights are consecutively adjusted dependent on input sets and the equivalent set of anticipated output targets. The synaptic weight adaptation consistently changing its value until it attains the anticipated behavior. Every single iteration produces two sweeps, specifically, forward stimulation to attain a solution, in addition, a backward dissemination of the errors calculated to transform the weights. The forward with backward sweeps are executed frequently until the NN solution approves with the preferred value within a previously stated tolerance. The NN structure is exposed in Figure 3.Back propagation algorithm is employed in the NN training, which is designated in the subsequent steps.

**Step 1:** Build random weights in [0, 1] interval and consign it to the output and also hidden layer neurons. Sustain a unison value weight for all input layer neurons.

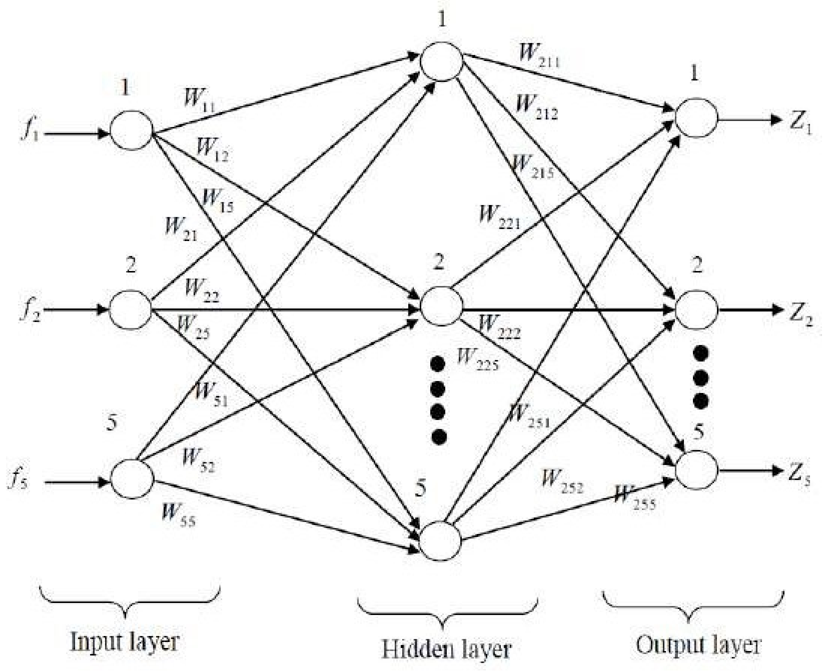


**Fig3.** Structure of NN

**Step 2:** the classifier assigns itself training dataset $G$ as input, the evaluation of BP error is

$$BP_{err} = Z_{tar} - Z_{out} \quad (1)$$

In Eq. (1), $Z_{tar}$ is the target output in addition to $Z_{out}$ is the network output that is determined as $Z = [Z^{(1)}_{out}, Z_2^{(2)} \square Z_2^{(N)}]$ $Z_2^{(1)}, Z_2^{(2)}, \square, Z_2^{(N)}$ are the network outputs. The network outputs is exhibited as

$$Z_2^{(l)} = \sum_{r=1}^{N_H} w_{2r1} Z_1(r) \quad (2)$$

where,

$$Z_1(r) = \frac{1}{1+\exp(-w_{11r} \cdot Z_{in})} \quad (3)$$

Eq. (2) and Eq. (3) signifies the activation function done in the output layer as well as hidden layer correspondingly.

**Step 3:** The entire neurons weights are accustomed $w = w + \Delta w$ to, in which, $\Delta w$ is the change in weight that is evaluated as

$$\Delta w = \gamma . X_2 . BP_{err} \quad (4)$$

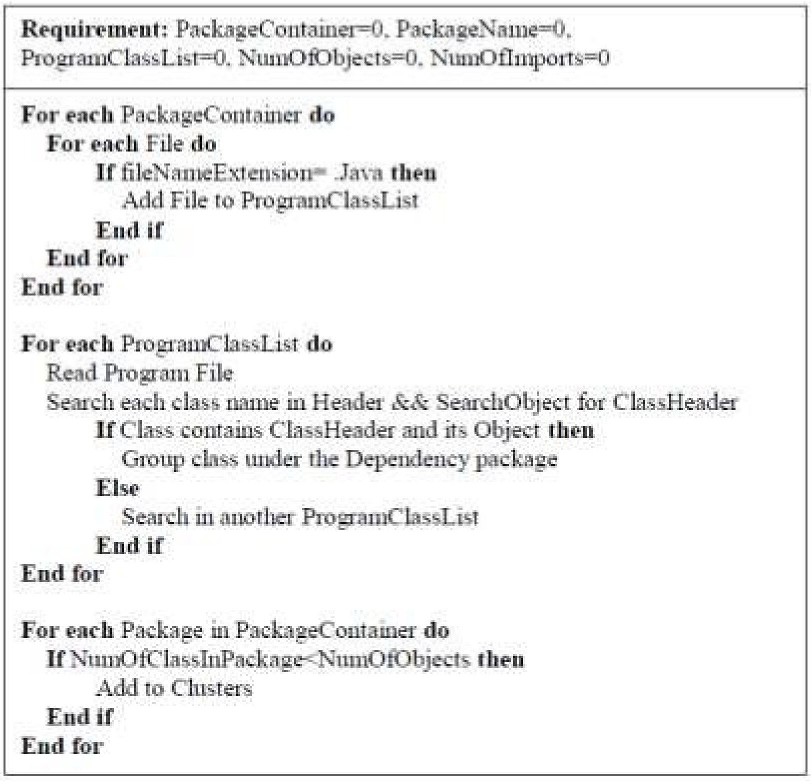
In Eq. (4), $\gamma$stands as the learning rate, commonly it extends as of 0.2 to 0.5.

**Step 4:** the process is repeatedly done from step 2, till BP error gets curtailed to a minimum value. Basically, the criterion is to be in
$BP_{err} < 0.1$.

The trained features that were utilized for clustering declines the aggregate of clusters which is initiated by the NN training and thus declining the aggregate of resources allocated.

## Clustering SystemClasses

Presently, clustering of the trained features is finished utilizing Class Dependency Based Clustering to segment the OO system into sub- systems with low-coupling. The Pseudo code for Class Dependency Based Clustering is outlined in Figure 4.



```
Requirement: PackageContainer=0, PackageName=0,
ProgramClassList=0, NumOfObjects=0, NumOfImports=0

For each PackageContainer do
  For each File do
      If fileNameExtension= .Java then
        Add File to ProgramClassList
      End if
  End for
End for

For each ProgramClassList do
  Read Program File
  Search each class name in Header && SearchObject for ClassHeader
      If Class contains ClassHeader and its Object then
        Group class under the Dependency package
      Else
        Search in another ProgramClassList
      End if
End for

For each Package in PackageContainer do
  If NumOfClassInPackage<NumOfObjects then
      Add to Clusters
  End if
End for
```

**Fig4.** Pseudo code for Class Dependency Based Clustering

From the Java programming convention, a package is a namespace which composes a group of related as well as comparative classes and also interfaces. Theoretically, packages are like distinctive folders on thecomputer.Apackageenablesadevelopertogroupclasses(aswellas interfaces) together. These classes will all be connected somehow that they may all need to play out a particular set of tasks. As the objective is to aggregate reliance classes, clustering utilizing CDBC, portrayed in Figure 4, can without much of a stretch meet the reason. The proposed CDBC algorithm groups software into multiple clusters utilizing related and comparative OO classes. The functionality of CDBC can be sorted as OO class identification and Cluster formation.

**OO class identification**: At the start of the clustering procedure, the proposed algorithm carries out textual analysis in the software project to distinguish the entire OO classes. In this exact circumstance, the software project is constructed utilizing Java, so the textual analysis is done considering the file expansion:java.

**Cluster formation**: Subsequent to recognizing the OO classes, the CDBC discovers the group centered upon the Dependency Class by checking the condition if Header Class has formed the Object in the source code, and at that point clustering the classes rely on the dependency classes the source codes. Else if Header class doesn't have the object, move to next class. The cluster formation executes until eachclass accessible in the package was allotted to the cluster.

## Grouping of Clusters

In this approach, the clusters (sub-systems) produced at the primary stage as the fundamental candidates intended for distribution is utilized. The method depends on consolidating clusters into groups in a means which maintains the communication expenses amongst them limited. Accordingly, a cluster graph is made, in which the nodes signify clusters and also the edges will catch the conceivable communication links which might exist amongst clusters. At that point, the K-Medoid algorithm is employed to do cluster gathering such that the amount of the ensuing groups is equivalent to the number of available nodes. The outcome will be the groups of clusters which have negligible communication expenses amongst them. Ultimately, those groups are allocated to the diverse available nodes in the distributed environment.

### *K-Medoid Algorithm*

Grouping or else Clustering stands as a strategy for partitioning a group of objects into clusters with the end goal that the objects in the same cluster are majorly like one another than objects in various clusters as indicated by some characterized criteria. For the grouping of clusters, k-Medoid clustering strategy is utilized. The medoid stands as a statistic which signifies that data member from a data set whose average dissimilarity to the various individuals from the group is negligible. Along these lines, a medoid dissimilar to mean is dependably a member from the dataset. It represents the most centrally located data item in the dataset. The Pseudo code for K-Medoid clustering algorithm appears in Figure 5.

```
Input:
        $K$, Number of cluster groups
Output:
        $K$ Set of Cluster Groups

Begin
Arbitrarily choose $K$ objects as the initial representative objects
Repeat
    • Assign each remaining object to the cluster with the
      nearest representative object.
    • Randomly select a non-representative object $o_{rand}$
    • Compute total cost $S$ of swapping representative object
      $o_s$ with $o_{rand}$
  If $S < 0$ then
    • Swap $o_s$ with $o_{rand}$ to form the new set of $K$
      representative objects
Until No change
End
```

**Fig5.** Pseudo code for K-Medoid Algorithm

## Mapping Classes toNodes

In this stage, the restructuring procedure is expert by mapping the assortment of DOO application clusters to the distinctive network nodes to attain better performance. To accomplish this objective, the mapping procedure is performed by mulling over the target of limiting the effectof class reliance and also data communication. It is accepted that the objective distributed system encompasses an assortment of homogeneous processors that are completely associated by means of a communication network. In the proposed strategy, the mapping is finished with the guide of the Recursive k-means clustering system.

### *Recursive K-Means ClusteringAlgorithm*

A novel, iterative initialization strategy aimed at the k-means algorithm which depends on a succession of recursive partitions of the cluster is proposed. The goal is to approximate the k-means solution for the complete dataset by recursively implementing a weighted form of the k-means over a developing, yet little, the number of cluster's representatives. The pseudo code for the recursive k-means clustering algorithm appears in Figure 6.

```
Input: Number of Clusters $K$, Integers $\{q_{\min}, q_{\max}\}$,
threshold $\eta$
Output: Initial set of centroid

Begin
  For $q = q_{\min} : q_{\max}$ do
    • Construct the partition $P_q$
    • Define the weight set $W_q$
    • Update the centroid set approximation
      $C_q = \{c_j^q\}_{j=1}^{K} : C_q = Weighted\_Lloyd(W_q, K, C_{q-1})$
    • Compute the centroid set displacement measure
      $d(C_{q-1}, C_q) = \max_{j=1,2,\ldots K} \|c_j^q - c_j^{q-1}\|^2$
    If $d(C_{q-1}, C_q) \le \eta$ then
      Return $C_q$
    End if
  End for
Return $C_q$
End
```

**Fig6.** Pseudo code of Recursive K-Means Clustering

In the initial step of the iterative system, the dataset is split into various disjoint subsets, called blocks that are restricted in equally sized hypercubes. Every block is then described by a representative furthermore its comparing weight. At last, a weighted version of Lloyd's algorithm is implemented over the group of representatives. Starting with one iteration after that onto the subsequent, a more refined partition is built by isolating each hypercube into 2d equally sized hypercubes. Such partition enables subsets of the past iteration to reallocate in a different cluster, in an attempt to decrease the overall error. This process is recurring to the point that a precise number of iterations are accomplished or when the approximation, in two back to back iterations, changes marginally. This algorithm is predominantly in view of a weighted variant of Lloyd's algorithm, termed weighted Lloyd's algorithm, which is connected over the group of representatives of a specified segment, mulling over the weight related to each block. At that point, the resultant clusters are mapped to the availablenodes.

## 3. Result andDiscussion

This section evaluates the existing DOO Restructuring performance without Neural Network (DOOR) along with the proposed DOO Restructuring with Neural Network (DOOR_NN) with performance metrics of the communication expense and aggregate of clusters. In MATLAB software, the proposed restructuring simulator is established. The simulator encompasses a friendly user interface which lets the user indicate the nodes in addition to the edges in the systems and after that it will produce theCDG.

**Table 1. Simulation Results for the Proposed DOOR with Neural Network and the Existing technique without NN**

| Number of Classes | Number of Clusters | | Communication Cost | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | Clustering | | Grouping | | Mapping | |
| | DOOR | ProposedDOOR_NN | RGC | Proposed CDBC | K-Partitioning | Proposed K-Medoids | Double K Clustering | Proposed Recursive K-Means |
| 51 | 6 | 4 | 2045 | 1916 | 1899 | 1714 | 1680 | 1540 |
| 62 | 7 | 6 | 2632 | 2243 | 2160 | 2095 | 2124 | 2024 |
| 79 | 9 | 7 | 2600 | 2415 | 2185 | 2060 | 2097 | 1988 |
| 107 | 11 | 9 | 3196 | 2818 | 3009 | 2698 | 2483 | 2184 |
| 153 | 15 | 13 | 4090 | 3825 | 3784 | 3515 | 3143 | 3005 |

Table 1 delineates the simulation results of the proposed Distributed Object-Oriented Restructuring with Neural Network (DOOR_NN) and the existing method without Neural Network (DOOR). The proposed DOOR_NN is contrasted with the existing strategy DOOR centered upon the number of clusters, clustering, grouping, and mapping. For clustering, the existing strategy utilizes Recursive Graph Clustering, although the proposed procedure utilizes Class Dependency Based Clustering (CDBC). For gathering, the proposed stratagem utilizes K-Medoid approach, while the existing procedure utilizes K-Partitioning approach. For mapping, the proposed stratagem utilized Recursive K- Means clustering, while the existing method utilized Double K Clustering approach. For any quantity of classes along with clusters, the proposed procedure with Neural Network demonstrates the superior performance to the existing system without Neural Network.

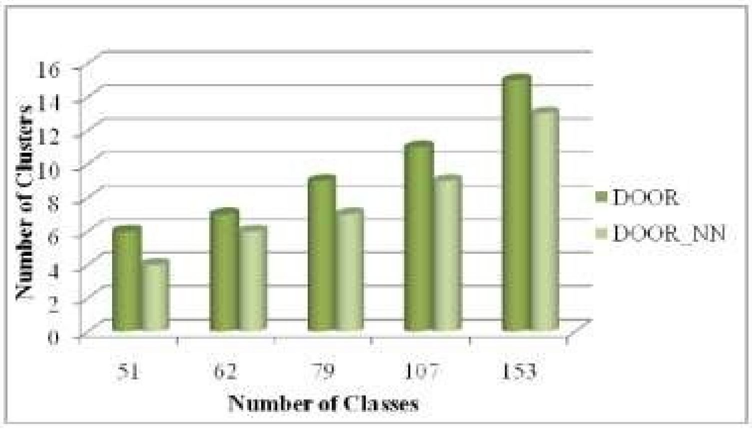


**Fig7.** Performance analysis of the proposed technique based on the number of clusters

Figure 7 analyses the proposed DOOR_NN performance and that of the existing procedure DOOR centered upon the number of clusters formed amid the clustering process. It demonstrates the results of the simulation with four nodes. It delineates the aggregate number of clusters formed for the precise quantity of classes. Here, it can be unmistakably observed that the proposed DOOR_NN produceslessnumber of clusters than the existing strategy, which clarifies the greater performance of the proposed stratagem.

## Clustering

Figure 8 analyses the performance of clustering for the proposed procedure Class Dependency Based Clustering (CDBC) and the existing method Recursive Graph Clustering (RGC) regarding communicationcost.

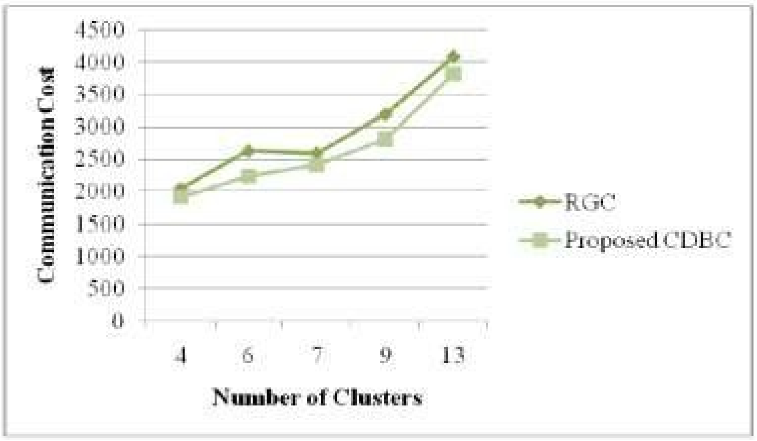


**Fig8.** Performance analysis of Clustering in terms of Communication Cost for the existing RGC and the proposed CDBC

In Figure 8, X-axis signifies the number of clusters produced and Y- axis signifies the communication cost in units of time which deliberate betwixt classes situated in various nodes. Here, for 4 clusters, the communication expense is roughly same for the RGC procedure and the proposed CDBC method. Be that as it may, as the amount of clusters expands, the communication expense additionally increments and an adequate distinction in cost is seen for the existing and the proposed strategies. Out of them, the proposed CDBCmethod demonstrates the most minimal cost as far as any number ofclusters.

**Grouping**Figure 9 analyses the grouping performance for the proposed K-Medoid Clustering method and the existing system K- Partitioning Clustering strategy regarding communicationcost.

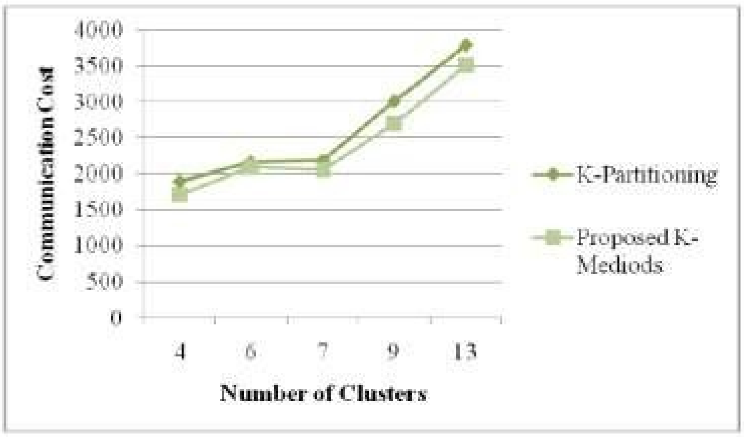


**Fig9.**Performance analysis of Grouping in terms of Communication Cost for the existing K- Partitioning and the proposed K-Medoids

For Grouping, the proposed K-Medoids clustering method is a cluster grouping approach. For 6 and 7 quantity of clusters, the communication expense is relatively same for both the existing K-Partitioning and the proposed K-Medoids method. Viewing at the communication cost of both the procedures, the proposed K-Medoids system has the most reduced communication cost intended for any number of clusters.

## Mapping

Figure 10 examines the performance of mapping for the proposed Recursive K-Means Clustering system and the existing Double K- Clustering technique regarding the communication expense.

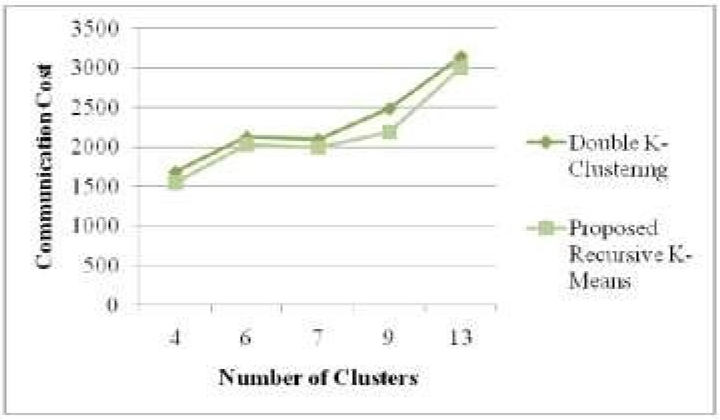


**Fig10.** Performance analysis of Mapping in terms of Communication Cost for the existing Double K Clustering and the proposed Recursive K-Means

For Mapping, the proposed Recursive K-Means Clustering strategy is utilized. Here, the difference in the communication cost intended for the proposed and the existing system is remarkable for 4, 6 and 7 quantity of clusters. A substantial contrast among the entire number of clusters is seen for 9 quantities of clusters.

## 4. Conclusion

OO techniques form applications extensively simple to construct by contributing an advanced platform for application development. The DOO software restructuring is done by via a proposed adaptive technique called NN, to intensify the performance further. This technique declines the aggregate of clusters which is initiated by the NN training and thus declining the aggregate of resources allocated. The existing DOO Restructuring without Neural Network (DOOR) performance and that of the proposed DOO Restructuring with Neural Network (DOOR_NN) are evaluated with performance metrics, for instance, the number of clusters and communication cost. Training the features with NN greatly lessens the quantity of clusters generated during clustering and thus the communication cost. Simulation outcomes illustrated that the proposed work yields enhanced outcomes in a proficient manner compared to the existingtechniques.

## References

[1] Wang J, Ai J, Yang Y, Su W Identifying key classes of object- oriented software based on software complex network. In System Reliability and Safety (ICSRS), 2ndInternational Conference on, IEEE, (2017) pp.444-449.

[2] Tarwani S, Chug A Prioritization of code restructuring for severely affected classes under release time constraints. In Information Processing (IICIP), 1st India International Conference on, IEEE,(2016) pp.1-6.

[3] Boucher A, Badri M Predicting Fault-Prone Classes in Object- Oriented Software: An Adaptation of an Unsupervised Hybrid SOM Algorithm. In Software Quality, Reliability and Security (QRS),IEEE International Conference on, IEEE, (2017) pp. 306- 317.

[4] Rajdev U, Kaur A Automatic detection of bad smells from excel sheets and refactor for performance improvement. In Inventive Computation Technologies (ICICT), International Conference on, Vol. 2, IEEE, (2016) pp.1-8.

[5] Kaya M, Fawcett JW A new cohesion metric and restructuring technique for object oriented paradigm. In Computer Software and Applications Conference Workshops (COMPSACW), IEEE 36th Annual, IEEE,(2012) pp.296-301.

[6] Mourad B, Badri L, Hachemane O, Ouellet A Exploring the Impact of Clone Refactoring on Test Code Size in Object-Oriented Software. In Machine Learning and Applications (ICMLA), 16th IEEE International Conference on, IEEE, (2017) pp.586-592.

[7] Amirat A, Bouchouk A, Yeslem MO, GasmallahN RefactorSoftware architecture using graph transformation approach. In Innovative Computing Technology (INTECH), Second International Conference on, IEEE, (2012) pp.117-122.

[8] Bhatti MU, Ducasse S, Huchard M Reconsidering classes in procedural object-oriented code. In Reverse Engineering, WCRE'08, 15thWorking Conference on, IEEE, (2008) pp.257-266.

[9] Liu H, Li G, Ma ZY, Shao WZ Conflict-aware schedule of software refactoring. IET software, (2008) 2(5):446-460.

[10] Nongpong K Feature envy factor: A metric for automatic feature envy detection. In Knowledge and Smart Technology (KST), 7th International Conference on, IEEE, (2015) pp. 7-12.

[11] Tomyim J, PohthongA Requirements change management based on object-oriented software engineering with unified modeling language. In Software Engineering and Service Science (ICSESS), 7th IEEE International Conference on, IEEE, (2016) pp. 7-10.

[12] Hamad, SH, Ammar, RA, Khalifa, ME, Fergany, T Randomized Algorithms for Mapping Clustered Object-Oriented Software onto Distributed Architectures. In Signal Processing and Information Technology, ISSPIT, IEEE International Symposium on,IEEE,(2018) pp.426-431.

[13] Sugandhi R, Srivastava P, Srivastav P, Sanyasi A, Awasthi LM, Parmar V, Makadia K, Patel I, Shah S Implementation of object oriented software engineering on LabVIEW graphical design framework for data acquisition in large volume plasma device. In Cloud Computing, Data Science & Engineering-Confluence, 7th International Conference on,IEEE,

(2017) pp.798-803.
[14] Faheem MT, Ammar RA, Sarhan AM, Ragab HAM A hybrid algorithm for restructuring distributed Object-oriented software. In Signal Processing and Information Technology (ISSPIT), IEEE International Symposium on, IEEE,(2010) pp. 202-208.
[15] Cosma DC Reverse engineering object-oriented distributed systems. In Software Maintenance (ICSM), IEEE International Conference on, IEEE, (2010) pp.1-6.
[16] Gu A, Zhou X, Li Z, Li Q, Li L Measuring Object-Oriented Class Cohesion Based on Complex Networks. Arabian Journal for Science and Engineering, (2017) 42(8):3551-3561.
[17] Ajienka N, Capiluppi A, Counsell S An empirical study on the interplay between semantic coupling and co-change of software classes. Empirical Software Engineering, (2017)pp.1-35.
[18] Kumar N, Dadhich R, Shastri A MAQM: a generic object-oriented framework to build quality models for Web-based applications. International Journal of System Assurance Engineering and Management, (2017) 8(2):716-729.
[19] Parashar A, Chhabra JK Mining software change data stream to predict changeability of classes of object-oriented software system. Evolving Systems, (2016) 7(2):117-128.
[20] Nucci DD, Palomba F, Rosa GD, Bavota G, Oliveto R, Lucia AD A developer centered bug prediction model", IEEE Transactions on Software Engineering,(2018) 44(1):5-24.
[21] Aslam W, Ijaz F A Quantitative Framework for Task Allocation in Distributed Agile Software Development, IEEE Access(2018).