Journal Homepage: -www.journalijar.com
INTERNATIONAL JOURNAL OF ADVANCED RESEARCH (IJAR)

Article DOI:10.21474/IJAR01/5298
DOI URL: http://dx.doi.org/10.21474/IJAR01/5298

RESEARCH ARTICLE

EMPIRICAL APPROACH TO VALIDATE HEURISTIC ALGORITHMS FOR PROCEDURAL PROGRAMMING TO OBJECT ORIENTED DESIGN MIGRATION.

Md. Samsuddoha[1], *Md. Saeed Siddik[2], Md. Selim[2] and Shah Mostafa Khaled[2].
1. Department of Computer Science Engineering, University of Barisal.
2. Institute of Information Technology, University of Dhaka.

Manuscript Info

Manuscript History

Received: 01 July 2017
Final Accepted: 03 August 2017
Published: September 2017

Key words:-
Call Graph,
Genetic Algorithm,
Local Search,
Variable Neighborhood Search,
Build Cluster Hierarchy,
Jaccard Similarity Coefficient.

Abstract

Although Object Oriented Programming is rapidly used in software industry, a wide variety of software is still used in the existing market developed in Procedural Programming languages. Sometimes, the maintenance of large software which developed in procedural languages become high costly and time consuming. To reduce this maintenance cost and make them more usable, some industries intend to transform that software from Procedural to Object Oriented Paradigm. On this purpose, researches have been focusing on automatic design migration approaches. Selecting a suitable and optimal method from the available is difficult because those approaches are not studied for any similar dataset. In this research, an empirical experiment has been conducted for identifying the optimal one among Genetic Algorithm, Local Search, Variable Neighborhood Search and Build Cluster Hierarchy algorithms. Six case studies from different real life software varying from 500 to 3700 LOCs are presented for experimental results verification. Final results are compared by a well known matrix named Jaccard Similarity Coefficient. Case studies show that Genetic Algorithm based approach outperforms other algorithms.

Introduction:-

Legacy software are still used in software industries and need regular up-gradation to keep pace with the market demands. Management of legacy software is not too easy to maintain and is very costly and time consuming. Most of this legacy software is developed in Procedural Programming (PP). On the other hand software developed in Object Oriented Programming (OOP) is very user friendly, usable and easy to maintenance. So, to reduce this maintenance cost and to make the legacy software more usable, several industries intend to transform this software from PP to OOP paradigm, which confirms some features like modularity, maintainability and manageability [1]. Sometimes these migrations are done manually, which is difficult and also time and resource consuming [2, 3]. Thus an automated migration from PP to OOP process is required.

In the recent literature, several researches have done which addressed this design migration towards OOP paradigm. Most of those are based on graph clustering. Several heuristics algorithms are proposed for migration based on Monte Carlo and Greedy [4], Genetic Algorithm (GA) [5], Local Search (LS) [6], Variable Neighborhood Search (VNS) [7] and Build Cluster Hierarchy (BCH) [8]. An empirical analysis is required for measuring the performance and accuracy of these design migration algorithms. This research intends to make an empirical evaluation of design

Corresponding Author:- Md. Saeed Siddik.
Address:-Institute of Information Technology, University of Dhaka.

migration from PP to OOP named as Genetic Algorithm (GA) [5], Local Search (LS)[6], Variable Neighborhood Search (VNS)[7] and Build Cluster Hierarchy (BCH) [8].

The existing design systems are using different algorithms for migration design. Different designs are developed using different parameters and there is a variation in the result of these design systems. Find out the similarity between the automation system and manual conversion for design migration is the main goal of this research. For evaluating those systems a renowned similarity matrix named Jaccard similarity coefficient is used. This research intends to compare the results and evaluate the existing migration system to make conduction on how far the PP to OOP migration algorithms could help this automation process in an effective and efficient way.

Rest of the paper is organized as follows: The related work available in the existing literature illustrated in section 2. Existing methods those were tested/evaluated in this paper are presented in the Section 3. Section 4 presents the evaluation process. The experimental results with numerical analysis and justification are presented in the section 5 and finally the section 6 concludes the paper with some comments and future work direction.

Literatures Survey:-
Object oriented software paradigm is a growing interest which results the need of migrating procedural legacy software [2]. Though automatic procedural programming to object oriented design migration has been rarely addressed as a direct research problem in the existing literature, object identification from procedural code was started at early nineties [9]. Most of the experiments were done on COBOL language [10]. However, procedural systems greatly differ from one another. Existing approaches cannot be applied to all procedural languages. This research project explores few notable design migration works in this field including like object oriented designs measurement matrices. These design migration techniques use different types of hierarchical, non-hierarchical clustering algorithms and heuristics.

Maqbool et al. reviewed hierarchical clustering research in the context of software architecture recovery and modularization [10]. The paper presented a detailed analysis of the behavior of different similarity and distance measures for clustering. Moreover, in this research different types of hierarchical clustering algorithms were studied on software for architecture recovery; mainly modularity analysis, feature extraction and architecture recovery. This approach was experimented on COBOL programs which cannot solve the migration problem of other procedural programs. They found that jaccard distance and Euclidean measure works well for design architectural recovery.
Sneed et al. was the first to introduce the concept of code to design migration [11]. They proposed a method of finding object oriented design from a legacy network workstation program written in COBOL language. The method converts a COBOL program to an object oriented design document by capturing and documenting the sequence of operations executed in a COBOL system. Components are considered as initial objects of the system. Relationships among the objects are identified by using data access mode and parameter list. Collaboration graphs were used to represent relationship between the objects. The work identified different types of objects like user interface objects, information objects, le objects, view objects etc. This work is only for COBOL and does not generalize for all procedural languages.

Object identification technique is centralized on persistent data stores like files, database and data structures in a program [12]. Their proposed method consists of two sections. The first section identifies potential objects from the analysis of data structure and last section assigns the methods in the corresponding objects. In the object identification section files that contain similar type of information are kept together and heterogeneous files are kept separate. The last section assigns method to objects by maintaining high coupling and low cohesion. Methods which access data of two objects are decomposed into smaller components and are assigned into suitable objects. This object identification process is not fully automated. It needs a good amount of manual interpretation for data field analysis inside source code.

Dineshkumar et al. presented an empirical approach to migrate from structured program to object oriented design [13]. Their work introduced a new technique for code to design migration which creates agglomerative cluster using Jaccard distance matrices. They initialized the relationship between variables and functions of the structured program using Jaccard distance measure that leads to grouping or clustering. Jaccard distance is used to identify distance between global programming elements, i.e., variables and functions. Agglomerative clustering technique starts with considering each element as a cluster. The closest pair is then merged iteratively until all elements are in one single cluster. Based on a threshold, one of the intermediate clusters set is chosen to be the class design.

Coupling and cohesion of the resulted system is evaluated using Ck metric. They represented the result in Unified Modeling Language (UML) and collected industrial survey to evaluate the accuracy of their proposal. Their use of Jaccard distance as a measure of distance between clusters considers all types of relation-ships to be equally contributing to the distance between the clusters. For example, consider a situation where a function f1 calls another function f2, and f1 reads a data d. With other relationships remaining same, f2 and d are equally likely to be in the same class with f1. The proposed clustering technique was applied on a simple rectangle and triangle area calculator system. Hierarchical clustering provides advantage over non-hierarchical by establishing relationship between different clusters. However, determine the cut-off point is a limitation of using agglomerative clustering algorithm.

Siddik et al. modeled structured to object oriented design migration as an optimal graph clustering problem [4]. The work presents several heuristic algorithms based on Monte Carlo and Greedy approaches. Object oriented design migration was considered as a graph clustering problem in this work. A call graph is generated from the procedural source code, where vertices represent functions and edges represent function calls. Three matrices namely Characteristics Path Length (CPL), Clustering Coefficient (CC) and Kal index (κ) were used for measuring cluster quality. Main objective of this work is to maximize the number of intra-cluster edge and minimize inter-cluster edge for finding the optimal result. Among the approaches, greedy was found to work better than Monte Carlo. However, this work only covered the encapsulation part of object oriented programming. Two others OOP principles Inheritance and polymorphism did not considered in this research.

A study of using variables and their nature in object identification of legacy source code is presented in [14]. Relational database is used to store program information and its dependencies. Object is treated as a triple of function, variable and type. The aim of this research is to study three existing object identification algorithms, namely Global Based Object Identifier (gboi), Type Based Object Identifier (tboi) and Receiver Based Object Identifier (rboi). gboi defines an object simply as a pair of functions and variables, tboi defines an object based on type of former parameter list with return type and rboi defines an object based on the variable that is accessed on routines. These three algorithms were experimented on a 3000 lines student project called RESTRUCT written in C. The experiment reported that rboi algorithm is performing better for identifying classes. However, if legacy system has many unwanted data type, there is a probability of generating faulty object design.

Existing Algorithms for Design Migration:-

In this section four selected algorithms are described, named Genetic Algorithm (GA) [5], Local Search (LS) [6], Variable Neighborhood Search (VNS) [7] and Build Cluster Hierarchy (BCH) [8] for an empirical analysis. All four algorithms addressed different solutions for the same problem. Each of the algorithms used different parameters and criteria for measuring the performance.

A. Genetic Algorithm for Design Migration [5]:-

Genetic Algorithm (GA) is a meta-heuristic search algorithm that is used to find near optimal solution. Processes like inheritance, mutation, selection and crossover are used. In [7], GA has been used to find a solution to the graph clustering problem. Previously, in our study, we had presented a GA based meta-heuristic approach that takes a procedural code and uses the underlying undirected graph G(V,E) as input and produces a scheme for clustering the graph to form k number of clusters. These clusters can be the basis of potential classes and/or interfaces for the object oriented design.

Selim et al. proposed a Genetic Algorithm for design migration [5]. Objective of this Genetic Algorithm is to find optimal clustering by maximizing the number of intra-cluster edge and minimizing inter-cluster edge. GA requires an initial seed which is generated by greedy heuristic methods that we presented previously in [5]. The seed is the initial candidate solution. The solution is then iteratively improved by the GA as it searches for a better solution. The algorithm is programmed to stop when the current best solution cannot be improved any further for a consecutive t number of times [7]. The followings tasks are iteratively performed:

i. Measure the fitness of every cluster in the solution
ii. Create pair of clusters based on the fitness
iii. Perform cross-over between the cluster pairs by exchanging member vertices of the clusters
iv. Perform mutation by changing the order of vertices within a randomly picked cluster

Compare the KAL index κ of the candidate solution in hand with the best solution found so far. If κ for candidate solution is better than current solution, change the current solution to the candidate solution. The scheme was implemented and tested against a set of real and synthetic data. The experimental results show that GA outperforms the research work [4] based on Greedy and Monte Carlo approaches by 40% and 49.5% respectively.

B. Local Search for Design Migration [6]:-

Siddik et al. proposed a Local Search algorithm [6] which extends their previous work Monte Carlo [4] and Greedy approach [5]. They presented the design migration as a graph clustering problem where the function of procedural languages are defined as vertex, and the call between two functions are defined as an edge of the graph [4, 5]. The underlying undirected graph of this call graph is used for searching the optimal cluster. Two variations of heuristic algorithms has been presented in this paper for finding optimal clusters by maximizing intra-cluster edges and minimizing inter-cluster edges for finding an optimal number of clusters. The proposed two variations of local search based heuristic that outperforms the previous work by 24.71% and 5.66% for Greedy approach and Genetic approaches proposed in [4, 5]. One of the variation of local search algorithm is presented in Algorithm 1.

```
Algorithm 1 Local Search
Input: An initial solution $C_{init}$
Output: Local Optimal Solution $C_{LocOpt}$ better or equal to $C_{init}$
1:   Begin
2:   $C \leftarrow C_{init}$
3:   repeat
     $C_{best} \leftarrow C_{init}$
4:     for each Cluster $C_i \in C$ in decreasing order of inter cluster edge degree do
5:         $v_{chosen} \leftarrow$ vertex with lowest edge connection
6:   $C_{LocOpt} \leftarrow$ UpdateSolution(C, $v_{chosen}$)
7:           if $\kappa(C_{LocOpt}) \geq \kappa(C)$ then
8:                 $C \leftarrow C_{LocOpt}$
9:           else
10:                no change
11:          end If
12:      end for
13:  until no update made over x consecutive rounds of loop
14:  $C_{LocOpt} \leftarrow C$
15:  return $C_{LocOpt}$
16:  End
```

C. Variable Neighborhood Search for Design Migration [7]:-

Selim et al. [7] presented a Variable Neighborhood Search (VNS) approach. The method provides a set of clusters that gives a clue for possible structure of the object oriented architecture. This approach is based on the objective to minimize the coupling and maximize the cohesion within the clusters. The proposed algorithm was implemented and its performance was compared with state of the art techniques. It is observed that the proposed method produced 37.15%and 12.02% better results in contrast to genetic algorithm [5] and local search [6] heuristics. Algorithm 2 presents the variable neighborhood search algorithm, where local search algorithm (presented in Algorithm 1) has been used for improve function.

```
Algorithm 2 Variable Neighborhood Search
Input: Call Graph G = (V;E)
Output: Best Local optimal solution $C_{best}$
1:   Begin
2:   Start with an initial solution $C_{init}$
3:   $C_{best} \leftarrow C_{init}$
4:   $C \leftarrow C_{init}$
5:   repeat
6:   $C \leftarrow$ ImproveFunction(C)
7:   if $\kappa(C) \geq \kappa(C_{best})$ then
8:           $C_{best} \leftarrow C$
```

9:	**else**
10:	no change
11:	**end If**
12:	C ⇠ Generate-Neighbor(C)
13:	**until** no update on C_{best} over x consecutive rounds of loop
14:	**return** C_{best}
15:	**End**

D. Build Cluster Hierarchy for Design Migration [8]:-

Islam et al. proposed a new heuristic algorithm for procedural to object oriented design migration using agglomerative clustering on the underlying dependency structure of procedural codes [8]. This dependency structure involves functions, parameters and global data structures of the procedural code. They represented the problem as a weighted graph clustering problem. They used their own similarity measure called weighted similarity coefficient to compute similarity between two nodes of a weighted graph. Their proposed algorithm has been tested against a database of procedural codes. The obtained results have been empirically validated using several similarity metrics.

The review shows that only a few of the works analyzed the proposed methods using some synthetic datasets but most of the work didn't validate their system by analyzing any real life dataset. In this paper some works are described those were analyzed using different case studies on real life datasets.

Evolution Process:-

Main task of this research is to empirically analysis some heuristic algorithms those transform PP to OOP paradigm. After some initial study we have collected some legacy software those were developed in procedural languages (e.g. C, FORTAN, etc.) for research data. Source code of these programs is collected from different open-source sites (e.g. github, codeincodeblocks, etc.). Using call graph generator call graph were generated each of the source code those are the input of each algorithms. Actual results were generated from each of the design migration algorithms. Expected results were collected from software engineers. Finally, we conducted a comparison to validate these systems by finding out the similarity between the AR and ER. Comparison was done by Jaccard Similarity Coefficient. The working procedure is shown in the Figure 1.

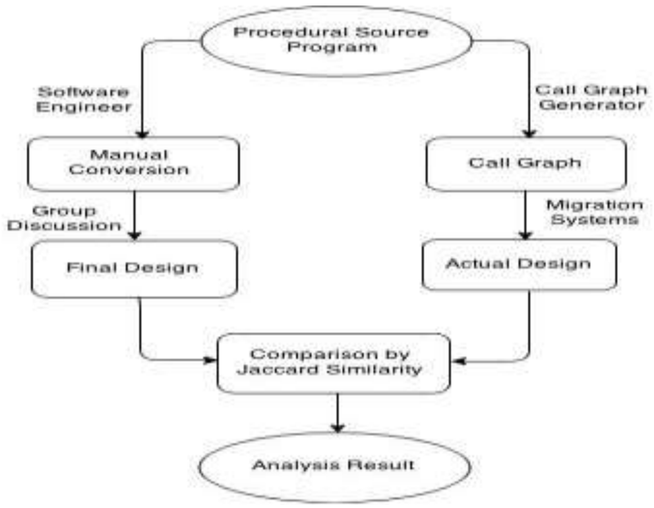

Figure 1:- Flow Diagram of Evaluating Process

Expected Result:-

The results we collected from manual conversion those were done by some software experts defined as Expected Result (ER). We talked with some expert software engineers to migrate our legacy code manually to OOP design. Each of these suggests different design and all of the designs were almost same. Then, we made a discussion to select final design for analysis.

Actual Result:-

The results got from the design migration systems are Actual Result (AR). We have collected result from different migration system using call graph of each of source code. Before that we generate call graph using call graph generator of our selected source codes. Then we generated actual result from the heuristic algorithms of these design systems. From these result sheets we have collected the best possible result to proceed our next part.

Evaluation by Jaccard Similarity Coefficient:-

For each dataset, we have calculated similarity between actual result from the design technique and expected result from engineer using Jaccard Similarity Coefficient. For similarity calculation we find best matching cluster pairs between the two designs. If a good matching is not found, we pair that cluster with an empty cluster.

We calculate the similarity between each pair and take mean of the similarity values. This mean is the degree of correctness of the output. In addition to direct Jaccard similarity, we calculate two more similarities: similarity-A and similarity-E. Similarity-A is the weighted similarity based on actual output and similarity-E is the weighted similarity based on expected output. For a pair p and set of programming elements E, which are defined in Equation (1) to (5).

$$SimilarityofPairP, \qquad S(p) = \frac{|P_a \cap P_e|}{|P_a \cup P_e|} \tag{1}$$

$$Similarity - ACoefficient, \qquad JC_a(p) = \frac{|P_a|}{|E|} \tag{2}$$

$$Similarity - ECoefficient, \qquad JC_e(p) = \frac{|P_a|}{|E|} \tag{3}$$

$$PairSimilarity - A, \qquad JS_a(p) = JC_a(p)\,.S(p) \tag{4}$$

$$PairSimilarity - E, \qquad JS_e(p) = JC_e(p)\,.S(p) \tag{5}$$

Finally, we calculate three type of similarity of all pairs P of a dataset D:

$$Similarity, \qquad JS(D) = \frac{\sum_p(p)}{|P|} \vee_{p\in P} \tag{6}$$

$$Similarity - A, \qquad JS_a(D) = \sum_p JS_a(p) \vee_{p\in P} \tag{7}$$

$$Similarity - E, \qquad JS_e(D) = \sum_p JS_e(p) \vee_{p\in P} \tag{8}$$

By using the above equations Jaccard similarity is calculated in this study. This evaluation process is iterated over the selected design migration approaches by different dataset.

Experimental Setup and Result Analysis:-

The selected migration approaches have been implemented using C++ and Java programming language. Torun these code we used Eclipse IDE on Linux Mint15 Operating System (OS), the Platform is32bit, Processor Model: Intel Core-i3, Processor Speed: 2.20 GHz, Cache Memory: 6 MB, and RAM: 4 GB.

Six different datasets have been used for experiment. The datasets are of different in the size like number of functions and line of codes (LOC), which cover every performance analysis of those algorithms in versatile projects. The number of programming elements varies from 15 to 100 respectively. Table 1 present the details of the dataset.

Table 1:- Experimental Datasets

No	Name of application	Number of functions	Lines of code
1	Phone	15	880
2	Snake Game	18	550
3	Department Store	21	680
4	Bank Manag. System	29	2250
5	Medical Store	45	2700
6	Silver Searcher	71	3500

Table 2 presents the number of classes those were designed by the software engineer is Expected Result (ER) and the number of clusters generated by GA, LS, VNS and BCH algorithms. These results show different design for the same dataset. Each of the class or cluster forms with function of the procedural code.

Table 2:- Proposed and Expected Class Numbers in OOP

Application /Algorithm	**Expected Result (ER)**	**GA [5]**	**LS [6]**	**VNS [7]**	**BCH [8]**
Phone	9	5	3	3	15
Snake Game	6	4	3	2	18
Department Store	6	5	4	1	8
Bank Manage System	15	5	4	2	27
Medical Store	11	8	4	3	14
Silver Searcher	13	9	4	3	63

Table 3 presents the result of similarity between the manual conversion design and the generated design for each of the four algorithms. For the four different algorithms similarity were calculated in percentage. However, for each dataset Genetic algorithm performs better than the other algorithms in terms of similarity.

Table 3:- Percentage of Accuracy in Experimented Datasets

Application	**GA [5]**	**LS [6]**	**VNS [7]**	**BCH [8]**
Phone	39.99	33.33	40	42.22
Snake Game	56.67	32.22	27.78	15
Department Store	39.05	36.90	28.57	46.30
Bank Manag. System	51.72	37.93	31.03	43.11
Medical Store	51.25	32.75	26.67	26.05
Silver Searcher	42.52	28.28	17.13	13.58
Average	46.87	33.56	28.53	31.04

Figure 2 presents the comparison among the four algorithms for migrating functional programming to object oriented programming. It is found that Genetic Algorithm (GA) performs best accurate migration result for OOP.

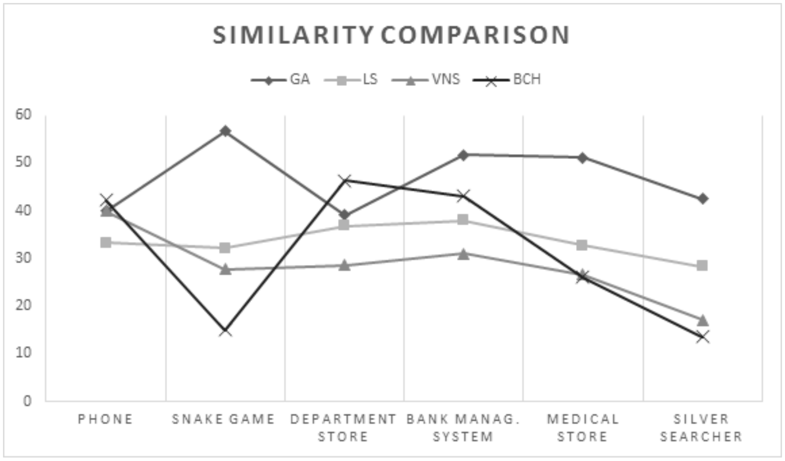

Figure 2:- Comparison among Implemented Algorithms

Conclusion and Future Work:-

This research addressed an empirical approach of design migration problem from Procedural Programming to Object Orient Paradigm. Experiment was performed on six C programs. The results of the implemented algorithms are compared with designs provided by two professional software engineers. Similarity between engineer's design and the designs from studied algorithms are calculated using Jaccard Similarity Coefficient. In the result section details of the analysis are described. All studied algorithm worked on the same problem design migration. GA, LS and VNS algorithms only consider the functions to be potential member of class ignoring variables and constants which are also potential member of object oriented programming. BCH works with functions as well as variables and constants. Polymorphism and inheritance of classes do not address any of those algorithms. Call graph and WDCG have used in these algorithms have not considered run time call scenario, i.e. how many times a function calls the other. However, BCH provides better result for the small dataset that contains few numbers of functions and variables. LS and VNS shows average result for all size of dataset but it depends on the number of classes those designed by software engineer. GA shows better result for all size of dataset than the other algorithms. From the result analysis we can conclude that Genetic Algorithm performs better over Build Cluster Hierarchy, Local Search and Variable Neighborhood Search algorithms.

Six case studies are conducted only on C programs. The empirical analysis addressed as procedural programming to object oriented paradigm but the analysis was done only on C programs. Different languages have different styles and conversion which may affect the structure of design paradigm. We have measured the similarity using only one matrix. Further research on the problem may be directed towards considering the run time call graph scenario and empirical analysis can be done using other procedural languages and more matrices.

References:-

1. H. Schildt, Java 2: The complete reference. McGraw-Hill Professional, 2000.
2. K. Chisolm and J. Lisonbee, “The use of computer language compilers in legacy code migration," in AUTOTESTCON'99. Systems Readiness Technology Conference. pp. 137-145, 1999. IEEE
3. Zou, Ying, and Kostas Kontogiannis. "A framework for migrating procedural code to object-oriented platforms." Eighth Asia-Pacific Software Engineering Conference, pp. 390-399, 2001, IEEE
4. M. Siddik, A. U. Gias, and S. M. Khaled, “Optimizing software design migration from structured programming to object oriented paradigm," in Proceedings of the 16th International Conference on Computer and Information Technology, pp. 1-6, , December 2013. IEEE
5. M. Selim, S. Siddik, A. U. Gias, M. Abdullah-Al-Wadud, and S. M. Khaled, “A genetic algorithm for software design migration from structured to object oriented paradigm," 8th International Conference on Computer Engineering and Applications, Spain. pp187–192. Jul 23, 2014,
6. M. Siddik, A. U. Gias, M. Selim, S. M. Khaled, K. Sakib, “A direction of migrating procedural paradigm to object based architecture by forming cluster of functions using local search heuristics," in Informatics, Electronics & Vision (ICIEV), 2014 International Conference on, pp. 1-6, , 2014. IEEE
7. M. Selim, S. Siddik, A. U. Gias, R. Tajkia, and S. M. Khaled, “Approximating object based architecture for legacy software written in procedural languages using variable neighborhood search," in SKIMA, 2014, IEEE.
8. M. M. Islam, “Paradigm migration from structured to object oriented by clustering data call graph," in MS Thesis, Institute of Information Technology, University of Dhaka, 2014.
9. R. E. Sward and T. C. Hartrum, “Extracting objects from legacy impera-tive code," in 2011 26th IEEE/ACM International Conference on Automated Software Engineering (ASE 2011), pp. 98-98, IEEE Computer Society
10. O. Maqbool and H. A. Babri, “Hierarchical clustering for software architecture recovery," IEEE Transactions on Software Engineering, vol. 33, no. 11, pp. 759-780, 2007.
11. H. M. Sneed and E. Nyary, “Extracting object-oriented specification from procedurally oriented programs," in, Proceedings of 2nd Working Conference on Reverse Engineering, pp. 217-226, 1995 IEEE,.
12. A. Cimitile, A. De Lucia, G. Antonio Di Lucca, and A. Rita Fasolino, “Identifying objects in legacy systems using design metrics," Journal of Systems and Software, vol. 44, no. 3, pp. 199-211, 1999.
13. J. D. V. Dineshkumar, “Code to design migration from structured to object oriented paradigm," in International Journal of Information and Communication Technology Research, vol. 1, 2011.
14. S. Pidaparthi and G. Cysewski, “Case study in migration to object-oriented system structure using design transformation methods," First Euromicro Conference on Software Maintenance and Reengineering., pp. 128-135,1997. IEEE