Metric Approach to Anticipate Reusability of Object-Oriented (O-O) Software Systems

Bharti Bisht, Research Scholar, FCA, Manav Rachna International Institute of Research and Studies, Faridabad,onebharti@gmail.com

Corresponding author
Dr Parul Gandhi, Professor, FCA, Manav Rachna International Institute of Research and Studies, Faridabad, parul.fca@mriu.edu.in

***Abstract*:** In order to meet the fast software evolution, there is a call for the work on software development based process by reducing time as well as efforts. The aim of the development process should not only be developing software products and services but also focus on improving the quality of the particular software. Software Reusability can be considered as one of the solutions to achieve both objectives i.e. productivity as well as quality. There has been an evolution of various methods and techniques related to construction of reusable components over many years. Object-oriented approach also assures increased software reusability. It is easier to reuse object-oriented software rather than conventional software. The notion of reusability related to Object-oriented software can be achieved through inheritance which in turn contributes to development of reusable components. In this paper different metrics related to software reusability of Object-oriented software systems has been summarized and evaluated using Python. Three python-based programs are considered as datasets for this study-the first dataset depicts single-level inheritance, the second dataset depicts hierarchical inheritance whereas the third dataset depicts multilevel inheritance. This study shows more impact of multilevel inheritance on the reusability of Object-oriented software systems and also helped to understand the important role of metrics in evaluation of object-oriented systems.

Keywords: Object-Oriented software system, Software Metrics, Software Reusability, Software Complexity

1. Introduction

Software Reusability is a prominent way [2] to find out those artifacts [4] from existing components for constructing new systems. According to [5], software reusability is characteristic of a software component that illustrates software's capability of reuse [5].Software reusability also indicates the reuse capability of software component [7].It means that [7] if reusability of software component is low, then it is less reusable component.

According to [14], software reusability means the use of previously written software [14] in form of design as well as code. This method has been already observed widely during development process of most of the software projects. One of the important advantages of software reusability is that it helps significantly to reduce the number of bugs in the particular software [14].

Components can be looked upon as independent as well replaceable factor [10] of the particular application which implements clear and specific function [6]. Reusability is likely a portion of code which can be reused in order to enumerate some advanced functionalities [10].Subroutines as well as functions can be treated as elementary forms of reusability. Reusability is a way to develop larger things [4] from smaller portions and to find out similarities among these portions. Many industrial observers even suggested that up to 20% of development costs can be saved by using reuse approach [7].Software companies that entwine object technology has started using their own constructed reuse strategies by which reuse of software components is monitored as well as analyzed [9].

Object-Oriented techniques are helping out to construct software even in component form [2] and also contributing towards software reuse. There is a high demand for Object-Oriented principles that would in turn provide an effective approach to improve level of software reusability [11].The assessment of software reuse can help developers to explore various reuse levels so as to develop good quality software that can be reused easily as well as effectively. They provide quantitative view [16] of the implementation of object-oriented designs so as to contribute in improving the quality of the software.

This study attempts to use Object-Oriented (O-O) metrics as a predictor of the quality for the primitive system. This study calculates as well as analyzes object-oriented metrics on three Python-based programs. The remaining paper is structured as: Section 2 discusses Software Reusability. Section 3 discusses CK metrics and Section 4 about various literature studies related to usage of metrics for software reusability. Section 5 discusses the case study on estimating the reusability of Object-Oriented (O-O) software by making use of different metrics and Section 6 summarizes the case study analysis. Finally, Section 7 includes the conclusion part.

2. Software Reusability

Software Reusability means to alter software system [5] with the help of existing reusable components rather than constructing a brand new system. Since these reusable components have been rigorously tested and even verified [20], so they can contribute in improving the quality of particular software. Reusability contributes to increased productivity and better software quality. It not only depends upon code but also covers other software development process assets such as software components [11], test suites [11] documentations [11] and designs [11].The various crisis related to reuse are [16]:

- Ineffective Software [16]
- Poor quality Software [16]
- Incompetent Software [16]
- Uncontrollable Project [16]

Fig.1 depicting Reusability Process encompasses the following steps:

2.1 Identifying Software Components:

In this step, Software Repository is looked over to search for different software components [6].

2.2 Context Understanding [6]:

In this step, software programs are interpreted with the help of design patterns by considering existing documentation.

2.3 Software Reuse Approaches:

During this step, various software reuse approaches are implemented such as Code Reusability [9], Design Reusability [9], System Reusability [9] etc.

2.4 Integrating and evaluating [6]:

In this step, approaches such as Code cloning [6], Code Invocation [6] and COTS (Ramamoorthy C. V 1988)

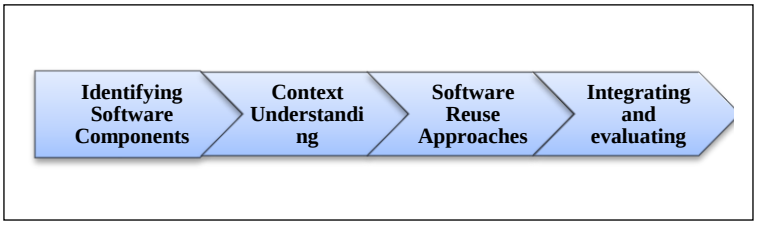


Figure 1. Reusability Process Steps (Source: Self)

Some of the advantages provided by Reusability are [16]:

- Decreases development cost
- Improves Reliability
- Market Time is decreased
- Maintenance cost reduces [16]
- Reduces Software Process Risks
- Standards Compliance [11]

3. Chidamber & Kemerer(CK) Object-Oriented Metrics

CK metrics play an important role in the reusability process. These metrics can be used as a tool to measure [10] properties of Object-Oriented systems like classes [10], message passing [10], inheritance [10], and encapsulation [10].There is a need of these metrics for software reusability because of following reasons:

- These helps to provide information [12] on quality of reused software component by finding out the defects during reusability process.
- These help in proper scheduling of ongoing software project.
- These help to present maintenance as well as cost during reusability process
- These helps to take decisions on how much improvement is required during a software reusability process.

The metrics discussed and studied in this study is as follows:

3.1 WMC (Weighted Methods per Class)

It helps to measure the complexities of a particular class. It is defined as the sum total of complexities of all methods of an individual class [15]. It is a direct predictor of time and efforts required to maintain an individual class.

$$WMC = \sum_{i=1}^{n} c_i \quad (1)$$

More methods in a class limit the probability of reuse [15].Suppose there is a Class F1, with different methods such as $m_1 ... m_k$ which are defined in the above class. Let $C_1 ... C_k$ be the complexity of the defined methods. If the complexities of various defined methods are considered to be unity, then *WMC* would be *n* i.e. the number of defined methods [15].

3.2 NOC (Number of Children)

It defines the number of children associated with an individual class [5]. Scope for reuse can be possible if NOC represents a moderate value. NOC helps to assess efficiency as well as reusability. NOC value of a particular class should reflect all the subclasses that share the properties of that particular class.

$$NOC(class) = Number\_of\_immediate\_subclasses + \sum_{i} NOC(i) \quad (2)$$

3.3 DIT (Depth of Inheritance Tree of a Class)

It represents the maximum length from root class to current class.
Deeper the class in the tree inherits more classes due to more number of methods present in that particular class. Reusability increases with deeper trees.

3.4 CBO (Coupling between Objects)

It is the count of the number of classes to which an individual class is coupled [13].Any 2 classes are coupled to each other when a method of one class [13] use other's class methods. High coupling indicates increased reusability.

CBOD = E1 + E2+ E3 + . . . + Ek (**3**)

Where [10]:
D: an object, k: number of different objects.
Ei: *1* if particular object *E* is coupled to an object *D, 0 otherwise*

3.5 RFC (Response for a Class)

It represents the methods that are executed potentially [15] in counter to the received message [14] by an object of that particular class [14].

RS = Mi U {Ri} (**4**)

Where [14]: Mi: all methods in the class
{Ri}: set of methods called by Mi

3.6 LCOM (Lack of Cohesion in Methods) [8]

It represents the methods of the particular class that refers to any given instance variable [8].

LCOM = G - H (**5**)

Where [15]: G: the number of different pairs of various methods that do not share instance variable of the particular class
H: the number of different pairs of various methods that share instance variables of that particular class

4. Related Work

Detailed literature survey related to various reusability metrics and approaches related to Object-oriented systems is reviewed in this segment.

Sandhu P., Singh H. (2005) evaluated CK metric suite and suggested some refinements to these predefined metrics in order to reflect correct results for Object-Oriented(O-O) systems.

Alvaro A., Almeida E., Meira S. (2006) discussed and proposed metrics for software reusability evaluation. These metrics helped to determine the level of reusability.

Olague H., Etzkorn H.L., Gholston S., Quattlebaum S. (2007) This paper validate three Object-Oriented metrics (CK metrics, MOOD metrics and QMOOD) for their ability[15] to measure reusability of Object-Oriented Software.

Sandhu S., Kaur H. (2008) tried to analyze CK metrics and even constructed a suite of metrics to evaluate reusability of object-oriented systems. These metrics would help to improve the quality of the software.

Bhatia P., Mann R. (2009) derived a new technique to assess the reusability of Object-Oriented (O-O) systems. In this study authors analyzed reusability using CK metrics (DIT, NOC, CBO).These metrics helped to determine the reusable component from the existing software components.

Gandhi P., Bhatia K. (2010) Object-oriented metrics plays a vital role in improving quality of the software and has been widely used in various software projects. Many researches have been done on these reusability metrics [13]. In this paper, four proposed metrics (Number of Template Children (NTC), Depth of Template Tree (DTT) Method Template Inheritance Factor (MTIF) and Attribute Template Inheritance Factor (ATIF))[13] were used in order to measure reusability of OO design.

Manhas S., Neeru N. (2010) In this paper, proposed metrics (McCabe's Measure, Regularity Metric, Cyclomatic Metric, Reuse Frequency Metric, and Halstead Software Science metric)[17] were used to calculate reusability value of OO software design.

Amin F., Mahmood A., Oxley A. (2011) This paper proposes a model that uses metrics such as Size Metrics, Coupling Metrics, Cohesion Metrics and Variability Metrics[11] as input to estimate reusability of software.

Gandhi P., Bhatia P. K. (2011) Inheritance as well as templates are very important concepts related to Object-oriented (O-O) software system as well as play important role in achieving software reusability. The paper uses two metrics GRr (Generic Reusability Ratio) and ERr (Effort Ratio) [14].

Kumar A. (2012) The paper proposes a flexible model using proposed metrics (Regularity Metric, Cyclomatic Metric, Reuse Frequency Metric, Coupling metric) which identifies reusable software modules.

Budhija N., Singh B., Ahuja S. (2013) developed a model to find out reusable components from source code by using proposed reconfigurable automated metrics. These metrics helped to predict the reusability aspects of Object-Oriented Software.

Nyasente S., Mwangi P., Kimani S. (2014) In this paper, authors have proposed reusability framework [18] for Object-Oriented software which uses metrics such as CBO,NOC,GC,LCOM as input to determine OO components reusability.

Vispute J. N., Sahu K. D., Rajput A. (2015) In this paper authors analyzed the impact of CK metrics on object-oriented systems. They found that DIT and NOC metrics play an important role in evaluation of reusability of object-oriented systems.

Hudaib A., Huneiti A. (2016) Reusability helps to cut down cost, time as well as efforts [11] which in turn improves the software quality. Reusability of classes can be calculated by its metrics values [11].In this paper, authors have used CK metrics to find out the reusability level of Object-Oriented systems.

Rathee A., Chhabra J. (2017) Current software development process is widely incorporating Object-Oriented principles. Cohesion plays an important role in measuring the quality of Object-Oriented systems. This paper analyses the software to find out those classes consisting of unrelated member functions [16] and even covers the code problem concept.

Padhy N., Singh R., Satapathy (2018) developed a model that employs object-oriented metrics such as CK metrics to predict the level of reusability. They analyzed that Object Oriented (OO) metrics [22] especially complexity as well as coupling metrics [22] were much useful for examining software reusability.

Maheswari G., Chitra K. (2019) compared proposed metrics Metric 1, Metric 2 and Metric 3 which were a combination of various object-oriented metrics and analyzed them to find out reusability level of software components.

5. Case Study

5.1 Data Set[12]

Programs constructed using Python are taken into consideration for this case study. First Dataset depicts "Single-level inheritance " means that [4] when a particular class is allowed to inherit properties from a single class only [3].Second Dataset depicts "hierarchical inheritance" means that [4] properties of one class are inherited by multiple classes [4] whereas third dataset depicts "multilevel inheritance" means that [4] a particular class has been derived from other derived class [5].All these programs are analyzed for their utilization of different object-oriented features during design stage [7].

5.2 Metrics Used

In this study class level metrics are used. The metrics used in this study are as follows:

- WMC (Weighted Methods per Class)
- NOC (Number of Children)
- DIT (Depth of Inheritance Tree of a Class)[10]
- CBO (Coupling between Objects)[10]
- RFC (Response for a Class)
- LCOM (Lack of Cohesion in Methods)[8]

5.3 Data Selection Method

Semi-automated strategy was used to calculate metrics for this particular study. Proper details of selected classes with their attributes as well as the methods were used as input and on this input basis, various metrics were calculated.

In Object-oriented approach, early effort during software project life cycle is needed in order to find out classes and related objects, encapsulation [8], polymorphism [8] as well as inheritance [8].Much effort would be saved instead of rewriting code [9] that would in turn helps in producing high quality software [9].

5.4 Object-Oriented Design [10]

An object-oriented system [11] begins with class definition containing related attributes as well as methods [11].Objects can be created from a template known as class. These objects share common design and behavior demonstrated by various methods. The operation that is defined inside class declaration scope is defined as method. Three imaginary object-oriented designs [10] for python-based programs are depicted in appendix

a) Dataset 1 represents Object-Oriented(O-O) design for Single-Level Inheritance
b) Dataset 2 represents Object-Oriented(O-O) design for Hierarchical Inheritance
c) Dataset 3 represents Object-Oriented design(O-O) for Multilevel Inheritance

6. Analysis And Discussion

In this section the dataset used for the particular study has been discussed. The results are analyzed [11] as well as evaluated with the help of graphs. To implement the study three programs written in Python has been used.

Dataset 1: Event Program has been used for depicting Single-Level inheritance that consists of Parent class Event and a Single Child class Manager .Class Event has theme and venue as its attributes. The methods used by event class are def_init_

() and print_event ().Class Manager has ID and Name as its attributes. The Manager class inherited print_event () method from its parent class Event and even contains its own methods such as def_init_Manager () and print_aud ().

Dataset 2: Tax_Details Program has been used for depicting Hierarchical inheritance that consists of Parent class Tax and Two Child classes DirectTax and IndirectTax.Class Tax has Status as its attribute. The methods used by Tax class are def_init () and print_tax (). Class DirectTax has Amount, Rate and Exemption as its attributes. DirectTax class inherited print_tax () method from its parent class Tax and even consists of its own methods such as def_init _amount (), def_init _rate (), calc_total ().Class IndirectTax has Income, InterestRate and Deduction as its attributes. The IndirectTax class inherited print_tax () method from its parent class Tax and even contains its own methods such as def_init_idt (), total_inc (), print_ir () and calc_total ().

Dataset 3: Dictionary Program has been used for depicting multilevel inheritance that consists of Parent class Dictionary, Child class WordCount and Grand Child class Vocab. Class Dictionary has Terms as its attribute. The methods used by Dictionary class are def_init_ter () and print_dic (). Class WordCount has Words and Word_mean as its attributes. WordCount class inherited print_dic () method from base class Dictionary and even consists of its own methods such as def_init _wc (), print_wc () and len ().Class Vocab has Text, Text_wc and Text_mean as its attributes. The Vocab class inherited len () method from child class WordCount and even contains its own methods such as def_init_vb (), set (), print_wc () and print_vocab ().

The metrics used in above case study gives insight of how different classes are dependent on each other [15].High degree inheritance [3] adversely affect the health of software system whereas low degree inheritance do not support object-oriented concepts [6].It is much easier to reuse more independent class in another application [8].

6.1 Evaluation Results:

Table 1 provides the values for different such as metrics WMC, NOC, DIT, RFC, CBO and LCOM [13] for Event program considered for the study [11].

Table 1. Metrics values (Event program)

Class	WMC	NOC	DIT	RFC	CBO	LCOM
Event	2	1	0	2	1	0
Manager	3	0	1	5	1	1

In the above Table 1. :
DIT= Depth of Inheritance Tree [13], NOC=Number of Children [13], CBO=Coupling between Objects [13], LCOM=Lack of Cohesion in Methods [13], WMC=Weighted Methods per Class [13], RFC=Response for a Class [13].Fig.2 represents graph for metrics used in Event Program

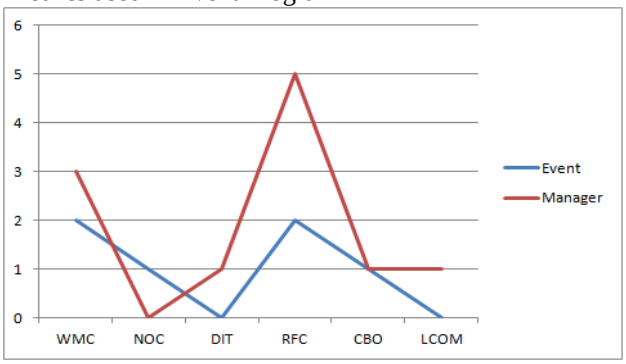


Figure 2. Graph for metrics-Event program (Source: Self)

Table 2 provides the values for different metrics such as WMC, NOC, DIT, RFC, CBO and LCOM [13] for Tax_Details program considered for the study [11].

Table 2. Metrics values (Tax_Details program)

Class	WMC	NOC	DIT	RFC	CBO	LCOM
Tax	2	2	0	2	1	0
DirectTax	4	0	1	6	1	1
IndirectTax	5	0	1	7	1	2

In the above Table 2. :
DIT= Depth of Inheritance Tree [13], NOC=Number of Children [13], CBO=Coupling between Objects [13], LCOM=Lack of Cohesion in Methods [13], WMC=Weighted Methods per Class [13], RFC=Response for a Class [13]. Fig.3 represents graph for metrics used in Tax_Details Program

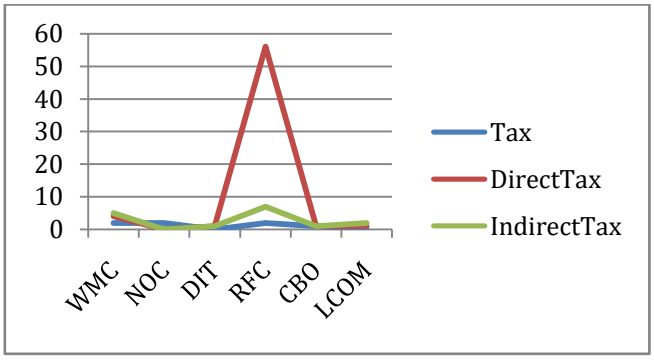


Figure 3. Graph for metrics-Tax_Details program (Source: Self)

Table 3 provides the values for different metrics such as WMC, NOC, DIT, RFC, CBO and LCOM [13] for Dictionary program considered for the study [11].

Table 3. Metrics values (Dictionary program)

Class	WMC	NOC	DIT	RFC	CBO	LCOM
Dictionary	2	2	0	2	1	0
WordCount	4	1	1	6	2	1
Vocab	5	0	2	11	1	2

In the above Table 3. :
DIT= Depth of Inheritance Tree [13], NOC=Number of Children [13], CBO=Coupling between Objects [13], LCOM=Lack of Cohesion in Methods [13], WMC=Weighted Methods per Class [13], RFC=Response for a Class [13]. Fig.4 represents graph for metrics used in Dictionary Program

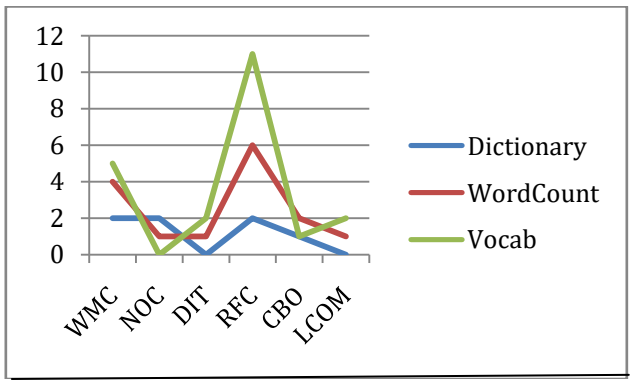


Figure 4. Metric Graph for Dictionary program (Source: Self)

Table 4 provides the values for different metrics such as WMC, NOC, DIT, RFC, CBO and LCOM [13] for average of all the classes of three programs (Event Program, Tax_Details program and Dictionary Program) considered in this above study.

Table 4. Metrics values of average of classes

Metric	WMC	NOC	DIT	RFC	CBO	LCOM
Event Program	3.7	0.7	0.7	5.3	1.2	1
Tax_Details Program	3.7	0.7	0.7	5	1.3	1
Dictionary Program	3.7	1	1	6.3	1.3	1

In the above Table 4. :
DIT= Depth of Inheritance Tree [13], NOC=Number of Children [13], CBO=Coupling between Objects [13], LCOM=Lack of Cohesion in Methods [13], WMC=Weighted Methods per Class [13], RFC=Response for a Class [13]. Fig.5 represents graph for Average of classes of different programs used in this case study.

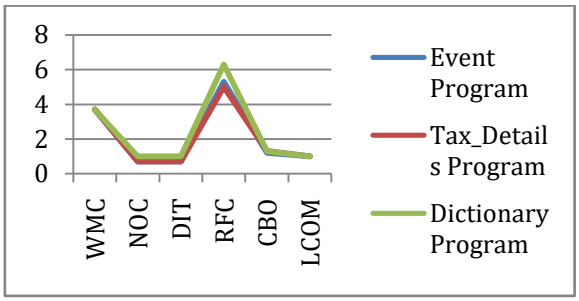


Figure 5. Graph for Average of classes (Source: Self)

The value of metrics used above in all programs indicates that in all programs, there is a moderate use of inheritance. Figure 5 depicts that WMC, NOC, DIT, CBO and RFC has larger values for Dictionary Program than Event program and Tax_Details program. So, Dictionary Program depicting multilevel inheritance [9] is more reusable than other two programs (Event Program and Tax_Details Program).

7. Conclusion And Future Work

In this paper, metrics used are being analyzed and measured for three Python programs (Event, Tax_Details and Dictionary). These metrics helped to figure out the features related to object-oriented software system. One of the result of this study was that different datasets that were used for the above study showed good utilization of object-oriented (O-O) features [11].It was also concluded that reusability is more impacted by multilevel inheritance. This study [11] also helps to understand that metrics plays an important role in evaluation of object-oriented software systems.

The future study would propose more metrics and would also analyze their impact on object-oriented systems.

ACKNOWLEDGMENT

I would like to acknowledge Dr Parul Gandhi, Professor and Ph.D. Coordinator in Faculty of Computer Applications, MRIIRS, Faridabad for her continuous support and helpful guidance during the preparation time of this article.

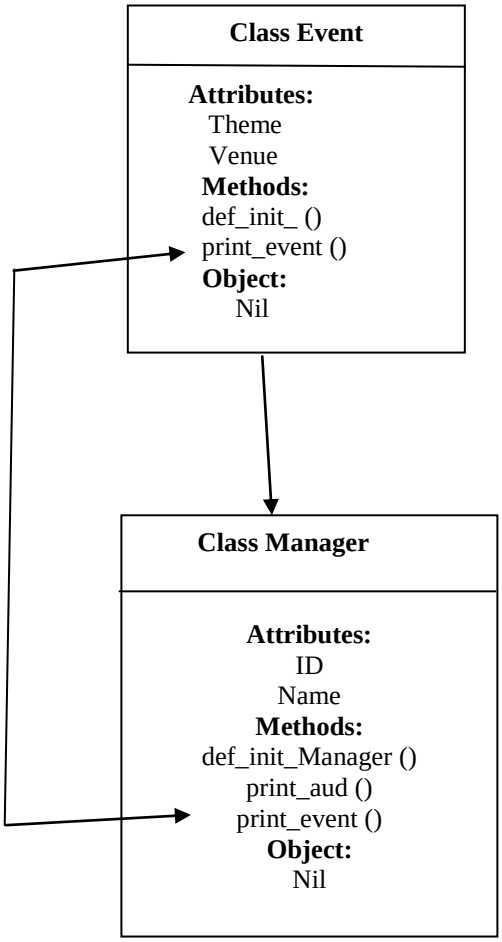
References

1. Abreu B., Carapuca R., 1994, “Candidate Metrics for Object-Oriented Software within a Taxonomy Framework”, Journal of systems software, Vol. 26, pp. 87–96.

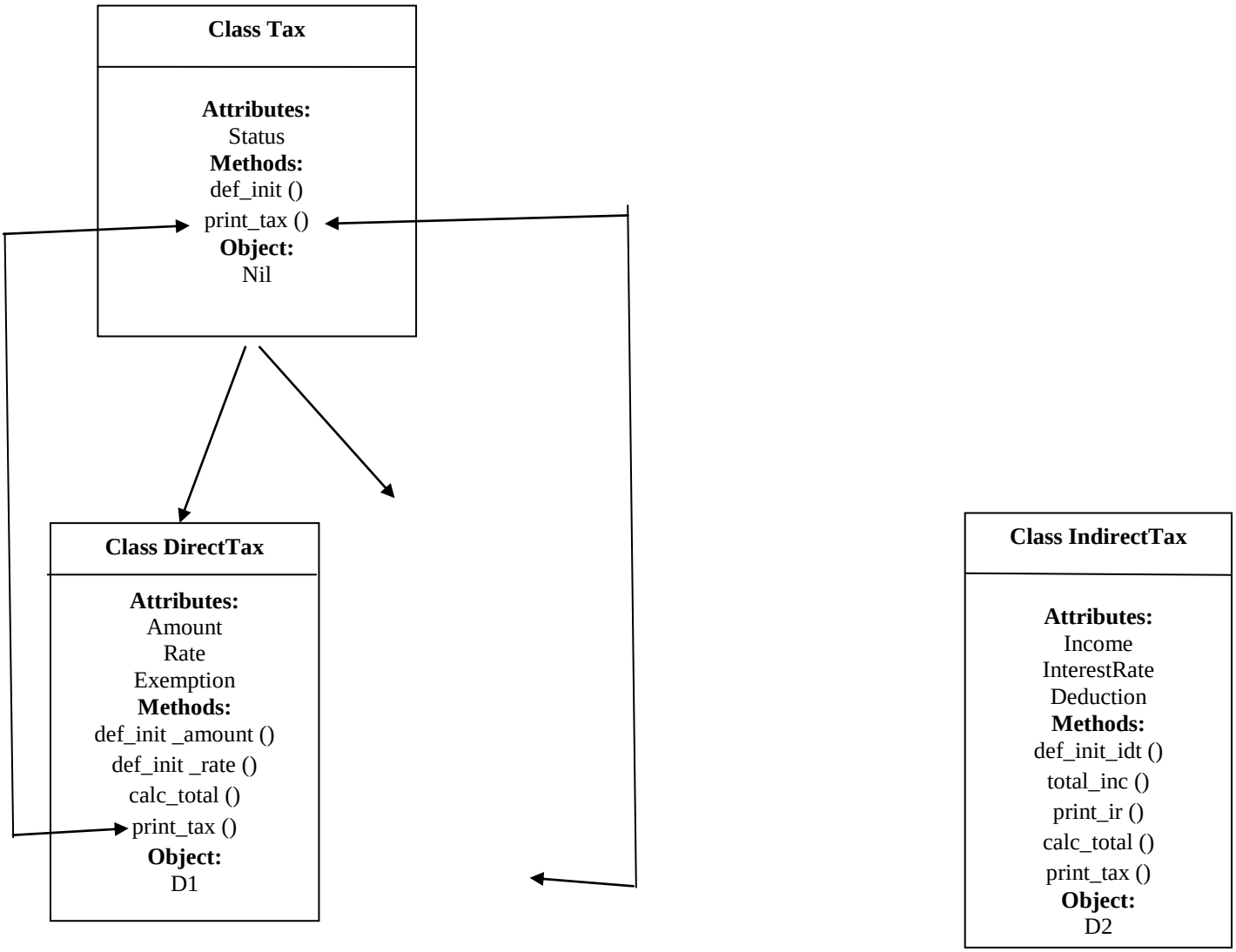
2. Aggarwal K., Singh Y., Kaur A., Malhotra R.,2006,"Empirical Study of Object-Oriented Metrics", Journal of Object Technology ,Vol. 5, no. 8,pp. 110-115
3. Arifa B., Mohamed N., Archana K., 2013," Study of Software Reusability in Software Components", *IJET*, Vol. 5, pp. 2455-2460.
4. Amin F., Mahmood A., Oxley A.,2011, "Reusability Assessment of Open Source Components for Software Product Lines", International Journal on New Computer Architectures and Their Applications (IJNCAA), 1(3), pp. 519-533
5. Alexandre A., Almeida E., Meira S.,2006, "A Software Component Quality Model: A Preliminary Evaluation", IEEE, Vol. 4,pp. 1-8
6. Barnard J , 1998, "A New Reusability Metric for Object-Oriented Software", Software Quality Journal, Vol. 7 no. 1, pp. 20-25
7. Basili J., B. V.R., Melo W., 1996, "How reuse influences productivity in object-oriented systems", *Communications of the ACM*, Oct. 1996, vol. 30, no.10, pp.104-114.
8. Bhatia P. ,Mann R.,2009,"An approach to measure Software Reusability of OO Design",COIT-2009,pp. 26-30
9. Budhija N., Singh B., Ahuja S., 2013, "Detection of Reusable Components in object Oriented Programming Using Quality Metrics", International Journal of Advanced Research in Computer Science and Software Engineering, Vol. 3, Iss.1
10. Chahal K., Singh H., 2008,"A Metrics Based Approach to Evaluate Design of Software Components", IEEE, pp. 269- 272
11. Chidamber R., Kemerer F., 1994, "A Metrics Suite for Object-Oriented Design", IEEE Transactions on Software Engineering, Vol. 20 no. 6, pp. 476-492.
12. Deeba K., Amutha B., 2016, "Classification Algorithms of Data Mining*",* Indian Journal of Science and Technology, vol. 9, pp. 2-5.
13. Gandhi P., Bhatia K., 2010, "Reusability metrics for object-oriented system: an alternative approach", IJSE, Vol. 1(4), pp. 63–72
14. Gandhi P., Bhatia P. K., 2011, "Estimation of generic reusability for object-oriented software: an empirical approach", ACM, Vol. 36(3), pp. 1-4
15. Gill S. N., 2006, "Importance of Software Component Characterization for Better Software Reusability", *In ACM SIGSOFT Software Engineering Notes*, vol. 31, pp. 1-3.
16. Goel B., Bhatia P.,2013," Analysis of Reusability of Object-Oriented Systems using Object-Oriented Metrics", ACM, Vol. 38 ,pp. 1-5
17. Hudaib A., Huneiti A., 2016, "Software Reusability classification and prediction using Self-Organizing Map (SOM*)",* in Communications and network, vol. 8, pp. 179-192.
18. J. Bhagwan, A. Oberoi, 2011, "Software Modules Clustering: An Effective Approach for Reusability", Journal of Information Engineering and Applications, Vol. 1, No.4.
19. Kakkar P., Sharma M., Sandu P.,2012, "Modeling of Reusability of Procedure based Software Components using Naïve Bayes Classifier Approach", International Journal of Computer Applications, Vol. 55 , pp. 12-17.
20. Kayarvizhy N., Kanmani S., 2011, "Analysis of Quality of Object-Oriented Systems using Object-Oriented Metrics", ICECT, Vol. 2, pp. 25-32
21. Maheswari G., Chitra K., 2019, "Enhancing Reusability and Measuring Performance Merits of Software Component using CK metrics", International Journal of Innovative Technology and Exploring Engineering, Vol.8, pp.1577-1583.
22. Manhas S., Vashisht R., Sandhu P., Neeru N.,2010, " Reusability Evaluation Model for Procedure Based Software Systems", International Journal of Computer and Electrical Engineering, Vol.2, pp. 1107-1110
23. Mateen A., Kausar S., Sattar A,2017,"A Software Reuse Approach and Its Effect On Software Quality, An Empirical Study for The Software Industry", International Journal of Management, IT & Engineering ,Vol. 7 Issue 2, pp. 266-279.
24. Morisio, Ezran M., Tully C., 2002, "Success and Failure Factors in Software Reuse", IEEE Transactions on Software Engineering, vol. 28, pp. 340–357.
25. Nyasente S., Mwangi P., Kimani S.,2014," A Metrics-based Framework for Measuring the Reusability of Object-Oriented Software Components", Journal of Information Engineering and Applications, Vol.4,pp. 71-84
26. Olague H., Etzkorn H.L., Gholston S. ,Quattlebaum S.,2007 , "Empirical Validation of Three Software Metrics Suites to Predict Fault-Proneness of Object-Oriented Classes Developed Using Highly Iterative or Agile Software Development Processes," , IEEE Transactions on Software Engineering, Vol. 33, pp. 402-419
27. Padhy N., Singh R., Satapathy, 2018," Software reusability metrics estimation: Algorithms, models and optimization techniques",Elsevier,Vol.69,pp. 653-668
28. Sandhu P., Singh H., 2006, "A Reusability Evaluation Model for OO-Based Software Components", International Journal of Computer Science, vol. 1, no. 4, pp. 259-264.
29. Sharma, A., Grover, P.S. and Kumar, R., 2009," Reusability Assessment for Software Components", *ACM* SIGSOFT Software Engineering Notes, Vol. 34, pp. 1-6.
30. Shatnawi A., Abdelhak D. S., 2013, "Mining reusable software components from object-oriented source code of a set of similar software, Information Reuse and Integration", *2013* IEEE 14th International Conference on IEEE, pp.2-18.
31. Subramanyam R., Krishnan S., 2003, "Empirical analysis of CK metrics for object-oriented design complexity: implications for software defects Software Engineering", IEEE Transactions, Vol. 29, pp. 297- 310.
32. Swathy V., Niranjan R.,2012, "A Resolved Retrieval Technique for software Components", *IJARCET*, Vol. 1, pp. 245-252.

Appendix:

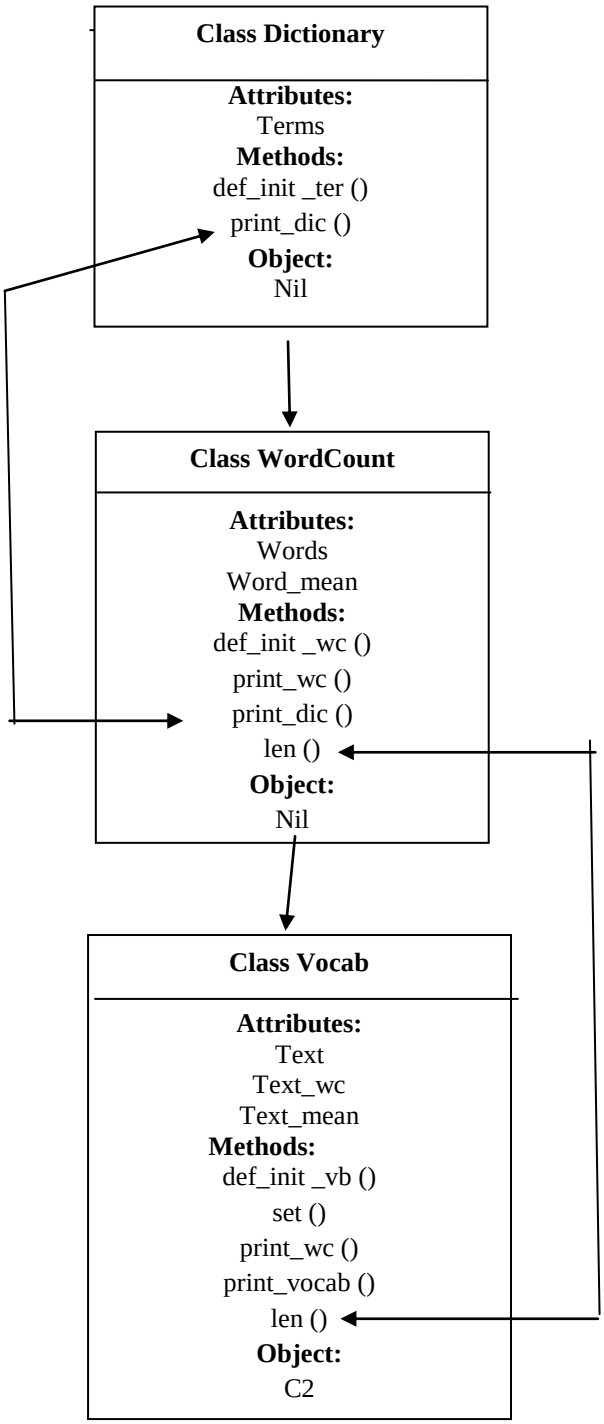
Dataset 1: Object-Oriented Design for Event Program (Single-Level Inheritance)



DATASET 2: Object-Oriented Design for Tax_Details Program (Hierarchical Inheritance)



DATASET 3: OBJECT-ORIENTED DESIGN FOR DICTIONARY PROGRAM (MULTILEVEL INHERITANCE)



AUTHORS PROFILE

Bharti Bisht, is doing her P.hD in Computer Science under the guidance of Dr Parul Gandhi in Manav Rachna International Institute of Research and Studies (MRIIRS), Faridabad. She has published paper in IEEE Digital Library and even presented her papers in International Conferences. She is active participant of various workshops and seminars conducted by educational institutions. Her research area is Software Reusability using Data Mining.

Dr Parul Gandhi, is working as Professor, Manav Rachna International Institute of Research and Studies (MRIIRS), Faridabad and also as Associate Coordinator of Non-Engineering Ph.D. Programs. A Doctorate in the subject of Computer Science with the study area in Software Engineering from Guru Jambheshwar University, Hisar. She is also a Gold Medalist in M.Sc. Computer Science. She is having a strong inclination towards academics and research. She has 13 yrs. of academic, research and administrative experiences. She has published many research papers in reputed International/ National journal and conferences. Her current research interests include software reusability, soft computing, and Software metrics and Component Based Software Development etc.