Reverse Engineering of Object Oriented System using Hierarchical Clustering

Aman Jatain[1] and Deepti Gaur[2]
[1]Department of Computer Science, Amity University, India
[2]Department of Computer Science, North Cap University, India

Abstract: *Now a day's common problem faced by software community is to understand the legacy code. A decade ago the legacy code referred as the code written in language like Common Business Oriented Language (COBOL) or Formula Translation (FORTRAN). Today software engineers primarily use object oriented language like C++ and Java. This implies that tomorrow's legacy code is written today because object oriented programs are even more difficult and complex to understand which leads us towards making software that is vague and having insufficient design documentation. Object oriented programming produce many problems to software developers in maintenance phase. So reverse engineering methodologies can be applied to resolve it. In literature various techniques has been proposed by researchers to recover the architecture and components of legacy systems. The use of clustering algorithms has recently been discussed by many for reverse engineering and architecture recovery. Methodology: In this paper Rational Software Architect (RSA) is used to recover the design from source code during reverse engineering process and then feature selection method is applied to select the features of software system. Hierarchical clustering is used after calculating the similarity measure between classes to cluster the similar classes into one component. The proposed technique is demonstrated by a case study.*

Keywords: *Clustering, feature selection, hierarchical, reverse engineering, rational software architect*.

Received April 28, 2015; accepted November 29, 2015

1. Introduction

Today's software development is defined by continuous evolution of software products that are regularly updated during their usage. In most of the cases systems grow inevitably by adding new features or by changing the system architecture due to new technologies or business plans. This unusual growth makes the systems difficult to manage and maintain.

But in many case, these systems have been maintained for many years, these are "legacy systems" that are vital for organization, but often difficult to understand and maintain [7]. Ian Somerville defines the legacy systems as "older legacy system" that remain vital to organization.

Most of the time legacy applications are not capable to support changing business requirements and new upcoming technologies because they were developed many years ago by using traditional programming language like C, Common Business Oriented Language (COBOL) and Pascal. Legacy applications are vital for the organization, so it is not a wise decision to replace the existing application for business due to cost and other hidden issues. To address the legacy system issues, reverse engineering process has been adopted as one of the most promising technologies [25]. This paper focuses on analysing the object oriented system through the reverse engineering process. Here motive of reverse engineering is to extract design information. Here emphasis is given on class diagram because class diagram are widely used for visualizing, describing and documenting different aspect of system. Class diagram shows the structural information of the system. There are various reverse engineering tools available in literature for C and other procedural oriented language. But as focus is on object oriented programming language specially C++ and JAVA, so widely used tools are: Sniff, Together Java, Understand Cee PlusPlus, Rational Rose. Most promising tools for reverse engineering are developed by International Business Machines (IBM), which includes: Rational Rose, Rational Architect and Rational Raphsody. We have used Rational Software Architect (RSA) to extract the class diagram from the source code. After getting the class diagram, the hierarchical clustering algorithm is implemented considering various similarity measure to group classes into meaningful subsystems. The proposed methodology is implemented on payroll management system.

2. Reverse Engineering Process

Reverse engineering has always been a central point and subject of active research in software engineering [29].

Reverse engineering is the process of analysing a subject system to identify the system's components and their interrelationships and create representation of the system in another form or at a higher level of

abstraction [28]. During reverse engineering the system architecture, feature, design and structure of the legacy system are captured to obtain the design model and allows the legacy system to be easily modified without the risk and cost associated with the changes in the legacy system. As shown in Figure 1 Chikofsky and Cross [7] reverse engineering is defined as the process of analysing a system:

- To identify its components and interrelationship between them.
- To represent the system at higher level of abstraction.

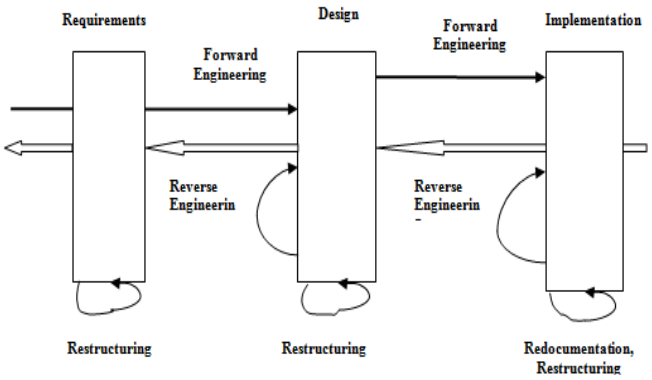

Figure 1. Reverse engineering.

Reverse engineering efforts for object oriented programming is difficult because program written in C++ and Java are highly fragmented [2, 21]. During modernization of the legacy system, software developers devote most of their effort in understanding the behaviour and structure of the system. In case of object oriented code, this might be more difficult, as multiple scattered objects assigned to the same function. The lifetime of legacy system can be increased by renovating or reengineering it [11].

Reengineering is the part of a software modernization process. But before a legacy system can be reengineered, the system must be reverse engineered [16, 24]. Reverse engineering is the prerequisite for legacy system's renovation. Reverse engineering includes various activities such as recovery of architecture, redocumentation and so on and it has strong relationship to the field of reengineering and software maintenance [12, 32]. Though the term "legacy system" is associated with systems written in older programming languages, but recent object oriented systems are having similar problems.

3. Related Work

In today's scenario every aspect of business is driven by IT and job in the field of IT is not stable because in IT people keep switching their jobs form one industry to other. So, when people switch their jobs and leaving the ongoing project in between creates a lot of confusion for the newly joined people. So at that time reverse engineering can be better option to get insight into the project. Several past works have described how to reverse engineer an object oriented legacy system to make it suitable for future technologies. Origin of reverse engineering has been in analysis of hardware for commercial or military advantage [7, 27].

The objective is to extract design information from end products without having any additional knowledge of the methods involved in the production. The same techniques are being researched in the last few years for application in legacy software systems to modernize it. Research in reverse engineering of object oriented software has mainly started in 1999 after leading European companies, namely Daimler-Benz and Nokia launched a research project named FAMOOS [23] to investigate tools and techniques for dealing with object oriented legacy systems. Major contribution of researcher in the field of reverse engineering is summarised in Table 1. Beishu and Qian [5] discussed the design and implementation of reverse engineering tool for object oriented paradigm named as Jade Bird Object Oriented Reverse Engineering Tool (JBOORET) for C++ program.

It is developed in Microsoft Visual C++. JBOORET extract design information from C++ code through pairing and store information in database. After storing design information in database, model mapping is done to obtain the Object Oriented Development (OOD) model and OOD diagram are shown using automatic layout algorithm.

Lanza in [20] proposed an approach to reverse engineer object oriented system by combining graphs and metrics. Lanza implemented a tool called as CodeCrawler, which graphically display source code and provided an interactive environment to the user.

Lanza discussed various reasons why reverse engineering is needed and what are the different problems software industry is facing during reengineering the legacy system. First he collected various class metrics and attributes metrics and then object oriented system is visualized using graphs by using layout algorithm. Graphs are generated by a tool "CodeCrawler" implemented in Smalltalk in the Visual Works 3.0 environment and framework used by CodeCrawler to produce the graph is HotDraw. Systa [31] provided some evidence of importance of SCED, a prototype tool used for modelling the dynamic behaviour of object oriented system because both static and dynamic information need to be extracted to understand the system. During reverse engineering of existing software, a parser and debugger are used for extracting the static and dynamic behaviour. This parsed information can be seen as a graph using Rigi, a

reverse engineering environment. Both static and dynamic information is extracted to understand the different aspect of the software system. Static information is visualized using the Rigi environment and dynamic information is gathered by running the Java software in a debugger called JDebugger. This information is viewed as scenario diagram using the SCED tool.

The objective of reverse engineer is to cope with the complexity of the software system by constructing various models. A system can be visualized using multiple views. Ramasubu in [26] discussed the use of UML Model to produce the multiple views of a system in 2001. He explained the various difficulties in reverse engineering complex object oriented system and then emphasized the use of Unified Modelling Language (UML) models. In this paper an Energy Information System (EIS) is taken as a case study to formulate the experience of reverse engineering process. The EIS application is created by an energy service provider and frequently used application in real word. In reverse engineering process, the practical tools play an important role. Kollmann *et al*. [19] discussed in the importance of case tools for reverse engineering and presented state of art of reverse engineering of object oriented systems. In this paper four tools are compared form industry and research regards to reverse engineering process.

Comparison is done both manually as well as automatically. The compared tools are from industry and research regards to reverse engineering process.

The compared tools are TOGETHER and RATIONAL ROSE from industry and IDEA and UML to Java and Back Again (FUJABA) from research prototype. Reverse engineering frameworks are also discussed in literature. Ferenc *et al*. [15] in had made an effort in this direction. They presented a framework for reverse engineering process named as Columbus. This tool is capable to analyse the large C++ projects and schema.

The architecture of the tool is very flexible and is also extensible for reverse engineering process. The developed tool first analyses the subject system and then extracted information is presented in the form of schema. In this paper evaluation of the tool is also performed to check its capabilities on three case studies: IBM Jikes compiler, Leda graph library and StarOffice Writer. Reverse engineering of object oriented systems using grouping is also adopted by researchers. Grouping adds much benefit to the process of reverse engineering. Some of the benefits discussed by Talerico [32], in his master thesis are: better understanding of system and its design recovery, higher abstraction level and reduced complexity. Some of the researchers worked towards developing product tools for visualizing the software artifacts during reverse engineering process. Eshah [13] in 2003 extended a software product tool named Visualizing Object Oriented C++ files (VOO++) which was developed by Mersa in 2000 by incorporating the software metrics into the VOO++ and the new developed tool was named as Visualizing and Measuring C++ files (VMCPP). The tool was developed to visualize the software and its main objective was extraction of the software components and software attributes, and presenting them graphically to software developers. Lanza in [20] performed reverse engineering ofan object oriented system by combining two techniques:

1. Software visualization.
2. Software metrics.

By using combination of these two techniques a new approach is developed named as polymetric view. The polymetric views were used in three different reverse engineering contexts i.e.,

1. Coarse- grained software visualization.
2. Fine-grained software visualization.
3. Evolutionary software visualization.

Tonella in [33] discussed approaches to reverseengineer various design views from source code. He presented a framework based approach that consisted of a graph representation of a program. This approach was named as Object Flow Graph (OFG).

The proposed technique is based on tracing the flow of information of objects by allocation statements. To understand the structure and behaviour of a legacy system whose documentation is not available Lopez and S. et al proposed a metamodel in [22], for reverse engineering of C++ code into sequence diagram. In metamodel the characteristics of the system such as entities, attributes and relationship are included. UML is widely used high level object oriented language for specification.

Model driven architecture approach i.e., Model Driven Architecture (MDA) is also emerged in last few years. Favre in [14] discussed the MDA concept in his paper on reverse engineering.

He discussed this approach in reference to of both platform dependent and platform independent model of object oriented system. Objective of MDA is to increase the level of abstraction by incorporating the use of models.

Models are the primary artifacts in the software development. MDA is a model driven technical framework to enhance the portability, interoperability and reusability using separation of concerns. Now a day's distributed software system has become important for the functioning and growth of civilization. In [9] Cosma analysed the distributed object oriented systems through the process of reverse engineering in his Ph.D thesis.

The most important part of any system is having

the source code available for documentation [34].

But if the source code is not available due to some unavoidable reasons then it poses several problems in understanding the system on software engineers.

Tonella *et al*. [34] discussed various reverse engineering techniques when source code is not available.

Table 1. Techniques and tools for reverse engineering.

S. No	Year	Author	Methodology	Tool	Legacy language
1	1997	Huang *et al*.	Reverse Engineering	JBOORET	C++
2	1999	Michele Lanza	Reverse Engineering combining graph and metrics	Code Crawler	C++
3	2000	TarjaSysta	Static and Dynamic Reverse Engineering	SCED Prototype	Java
4	2001	Surendranath	UML Notation	Rational Rose	Java
5	2002	Rudolf, Tibor *et al*.	Reverse Engineering Framework	Columbus	C++
6	2003	Daniele Talerico	Reverse Engineering using graphical grouping	Code Crwaler	Smalltalk
7	2003	Nidal Bashir Eshah	Reverse Engineering by combining metrics	VMCPP	C++
8	2003	Michele Lanza	Polymetric View	Code Crwler	C++
9	2005	Polo Tonella	Object flow graph	Algorithm	Object Oriented
10	2005	Wei,Tzerpos *et al*.	Design pattern	DPVK	C++, Java
11	2006	Maximo, Azucena *et al*.	Metamodel and conversion algorithm	-	C++
12	2006	Chiara, Lazzaro *et al*.	UML Models	-	Object Oriented
13	2007	Xinyi, Godfrey *et al*.	Hybrid Model	Swagkit	Object Oriented
14	2008	Liliana Favre	MDA based Framework	-	Java
15	2009	Dan Cosma	Distributable features based approach	M SiDe	Object Oriented
16	2010	Holger, Muller *et al*.	Rigi Environment	Rigiedit	C, C++ and COBOL
17	2011	Pereira, Favre *et al*.	MDA based Approach	Algorithm	Java
18	2012	Salman, Basha *et al*.	UML (Extraction of State Transition Diagram)	-	C++
19	2012	Thakore *et al*.	Software Quality Metrics	SRET	Java
20	2013	Mrinal *et al*.	UML (Sequence Diagram)	GNU pic2plot	Java
21	2013	Abhijeet, Sagar *et al*.	UML Diagram	Framework	Java
22	2014	Hugo, Jordi *et al*.	Model Driven Reverse Engineering	Framework	Java

Rigi environment also played an important role in analysing and documenting the large software system in the late nineties. Rigi environment is a research tool that helps in reverse engineering of large software systems. Kienle and Müller in [18] in their article described Rigi‟s main components and its functionality, and asses its effectiveness on reverse engineering. Rigi is used to model C, C++ and COBOL code. Some of the researcher also emphasis the role of use case diagram while reverse engineering the software system. Claudia *et al*. [8] discussed an MDA approach to recover the use case diagram form java code. With emergence of MDE in last decade, its principle and techniques have been used as effective reverse engineering solution.

Application of MDE in reverse engineering is called as Model Driven Reverse Engineering (MDRE) and it is used to reverse engineer the legacy system in order to generate the model based views to facilitate the legacy system understanding and manipulation. Bruneliere *et al*. [6] discussed the different MDRE practices and based on their experience in legacy system modernization they introduced MoDisco: a generic, extensible and global MDRE approach and a framework implemented in eclipse plug in.

4. Proposed Methodology

Since last decade software industry and research communities felt the need of practice tools to support reverse engineering activities. Now days many of the CASE tools help support reverse engineering. To recover the architecture of software, reverse engineering plays an important role. UML is the most commonly used standard today in the industries to represent the object oriented software at higher level of abstraction [1]. Most of the existing systems do not have a software architecture which is reliable and robust [30].

Even some of the legacy systems are designed without software architecture as design phase. To migrate the object oriented systems into component based systems, reverse engineering is the first step and to transform into component based system class diagram needs to be recovered from code. By using reverse engineering tools, class diagrams can be generated as part of software architecture recovery An UML class diagram is the best suited diagram in UML to present classes and relationship between them and in turn the architecture of object oriented systems.

We have examined various reverse engineering tools to generate the class diagrams form object oriented code and studied tools are: ArgoUML, Astah, Enterprise Architect, Rational Software Architect and Eclipse. Based on some model properties and language support we have examined during reverse engineering process, we found Rational Software Architect as best suited for our

requirements. It is a modelling and development environment that provides aid to UML. It helps in designing architecture of system developed in C++ and Java. It is built on the Eclipse framework. Rational Software Architect is well recognized tool to depict no. of classes and dependency information between them. It takes source code file as input and produce corresponding UML diagram. In our work we have generated the class diagram, to understand the dependency information. Figure 2 depicts the research framework.

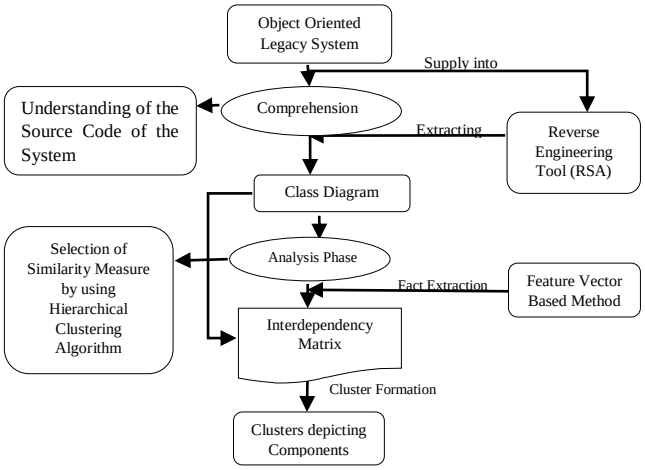

Figure 2. Framework for reverse engineering of object oriented system.

4.1. Agglomerative Hierarchical Clustering

In this section, we have provided an overview of hierarchical clustering algorithm and illustrate howit is utilized to cluster the similar classes into one cluster.

The generated cluster can be seen as a component having similar set of classes [24]. In agglomerative clustering algorithm set of individual entities are grouped into one cluster [3]. So, first step is to identify the entities to be clustered. In this paper our aim is to reverse engineer the object oriented system to identify classes and cluster similar classes into one with purpose of generating the components. Classes are considered as the entities to be clustered because they are the fundamental units of an object oriented system.

Agglomerative hierarchical clustering takes the set of classes as input and then a similarity measureis used to calculate the similarity between classes. In this paper the following notations are used to describe a software system:

- Assume ΔC is a set of classes in a software system X and ΔCOM is the set of components in a software system.
- Let U_SET(com) is the set of classes in a component com. Classes are denoted by $clas_i$ and $clas_j$. Classes are grouped into components.
- Rel (com_i,com_j) depict the set of dependency relation between classes com_i and com_j.
- Con *(*$clas_i$, $clas_j$.), indicate the connection strength between classes.

In agglomerative hierarchical clustering, we start with the single entity i.e., class and as we move forward at each step similar entities are grouped together into one. During clustering process to calculate the similarity there are various linkage methodologies available of hierarchical algorithm. To depict the working of linkage method, consider com_i and com_j denotes two different components and com_n is newly formed component after combining com_i, and com_j. sim (com_i,com_j) indicates the similarity between com_i, and com_j.

- Single Linkage
  Sim(com_i, com_n)=Max(sim(com_i, com_j), sim(com_i, com_n))
- Complete linkage
  Sim(com_i, com_n)=Min(sim(com_i, com_j), sim(com_i, com_n))
- Average Linkage
  a. Weighted Average Linkage
     Sim(com_i, com_n)=1/2(sim(com_i, com_j), 1/2sim(com_i, com_n))
  b. Unweighted Average Linkage
     Sim (com_i,com_n)= (sim (com_i,com_j)*size (com_j)+sim(com_i, com_n)*size (com_n)) /(size (com_j)+ (size(com_n))

Algorithm 1: Agglomerative Hierarchical Clustering

1. *Find the similarity between classes in ΔC.*
2. *Initially a set of component is created having a single class in each component.*
3. *Repeat*
   a. *Most similar component com and com are grouped into a new component.*
   b. *U_SET (com) is updated with newly formed component.*
   c. *Recompile the similarity between newly formed component com and other component till |U_SET(com)| =1*
4. *Return U_SET(com)*

In literature researcher discussed that, for software clustering complete linkage provides the most appropriate clusters [4, 17]. In this paper we have adopted the complete linkage method during hierarchical clustering.

4.2. Selection of Similarity Measure and Weighting Scheme

To quantitatively determine the similarity between classes, similarity measures play an important role.

The choice of a similarity measure has put more impact on the results rather than clustering algorithm [17]. The relation between classes is weighted by using connection strength of classes. It is likely that higher the number of connection between classes, more similar would be classes; this methodology is called as direct link method. A class

can be more accurately measured by features it is providing to the system. The method used to quantify the classes using feature is called as feature vector method. In software clustering feature vector method is more appealing than direct link method [4]. For object oriented system, the value of feature vector can be easily calculated by weighted schemes. In literature there are three weighted scheme:

1. Binary weighting scheme.
2. Absolute weighting scheme.
3. Relative weighting scheme.

In this paper we have applied binary weighting scheme to avoid additional amount of time wasted in computation of weights and due to this it has been largely used in software clustering [10, 17]. Binary weighting scheme weight is determined in between classes by using the connection strength. This scheme is defined as follows:

$$W_B\left(clas_i\ clas_j\right)=\begin{cases}1, if\ con\left(clas_i\ clas_j\right)|>0\\ or\ |con\left(clas_i\ clas_j\right)|>0\ or\ i=j\\ 0, otherwise\end{cases} \quad (1)$$

Equation (1) depicts that there is a connection between classes and 0 denotes, there is no connection. A class is connected to itself, so there is one when *i=j*. By assigning weight to classes based on the connection strength and the binary weighting scheme, an interdependency matrix is prepared. For example a class can be modelled by the interdependency matrix as (1, 0, 1, 0), which denotes that clas1 is having connections with clas1, clas3.

5. Case Study-Payroll Management System

In this section proposed framework is implemented on the case 'Payroll Management System (PMS)'. PMS is a window based system for a small company. Mainly it includes three modules: salary management, attendance management and employee management. The system is implemented in Java to support object oriented concepts. The aim of the PMS is to help the administrative staff in getting all the information required by the employer. The source code of the project is available at the link: http://www.enggroom.com/java.aspx. In our case study we have retrieved the classes in the form of class diagram from the source code of the project. Extracted class diagram is shown in Figure 3. and classes are shown by Rectangle boxes. Each class is consisting of operations and methods inside it. PMS is having 10 classes. After getting the class diagram next step is to implement the hierarchical clustering algorithm to cluster the similar classes into one component as discussed below stepwise.

- *Step 1.* First step is to generate the interdependency matrix using the feature vector method. In interdependency matrix classes are characterized by features and are applied to feature vector method to calculate the similarity between classes.

Table 2. Interdependency matrix of PMS.

	Cls1	**Cls2**	**Cls3**	**Cls4**	**Cls5**	**Cls6**	**Cls7**	**Cls8**	**Cls9**	**Cls10**
Cls1	1	0	0	0	0	0	1	1	1	0
Cls2	0	1	0	0	0	0	1	1	1	0
Cls3	0	0	1	0	0	0	1	1	1	0
Cls4	0	0	0	1	0	0	1	1	1	0
Cls5	0	0	0	0	1	0	1	1	1	0
Cls6	0	0	0	0	0	1	0	0	1	0
Cls7	1	1	1	1	1	0	1	1	0	0
Cls8	1	1	1	1	1	0	1	1	0	0
Cls9	1	1	1	1	1	1	0	0	1	0
Cls10	0	0	0	0	0	0	0	0	0	1

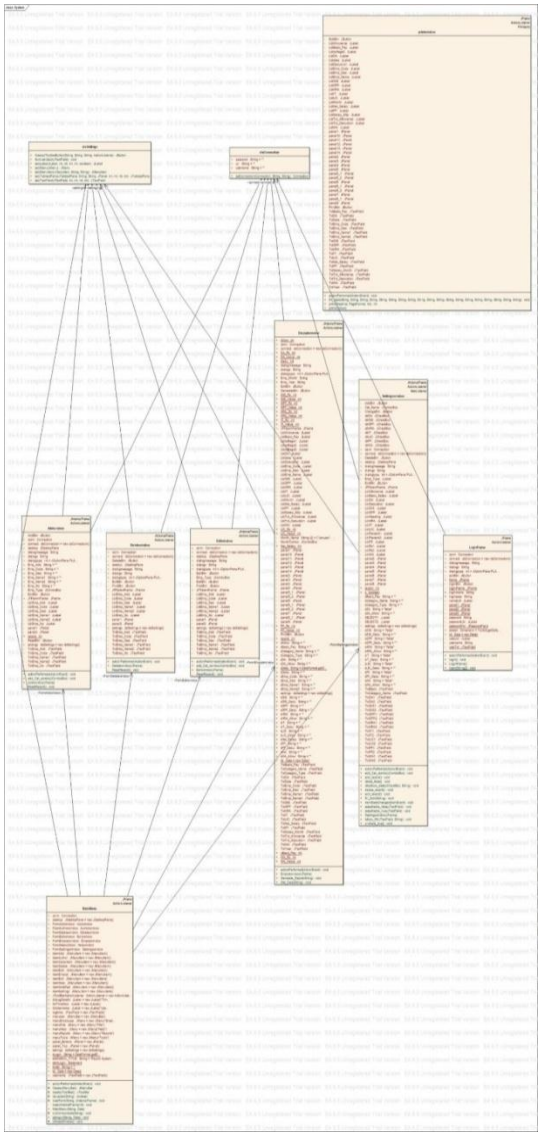

Figure 3. Extracted class diagram of PMS.

So, by using binary weighting scheme weight is assigned to each feature and interdependency

matrix is prepared as shown in Table 2. Here, cls1 can be represented by the vector <1, 0, 0, 0, 0, 0, 1, 1, 1, and 0>, which denotes that cls1 has calling connection with the classes cls1, cls7, cls8 and cls9 and so on.

- *Step 2.* Next step is to cluster the most similar classes. For this distance between each pair of class which is represented by feature in interdependency matrix is calculated. Here distance measure is Euclidean distance and the resulting matrix is shown in Figure 4.

	Clas1	Clas2	Clas3	Clas4	Clas5	Clas6	Clas7	Clas8	Clas9	Clas10
Clas1	0	1.4142	1.4142	1.4142	1.4142	2.0000	2.2361	2.2361	2.6458	2.2361
Clas2	1.4142	0	1.4142	1.4142	1.4142	2.0000	2.2361	2.2361	2.6458	2.2361
Clas3	1.4142	1.4142	0	1.4142	1.4142	2.0000	2.2361	2.2361	2.6458	2.2361
Clas4	1.4142	1.4142	1.4142	0	1.4142	2.0000	2.2361	2.2361	2.6458	2.2361
Clas5	1.4142	1.4142	1.4142	1.4142	0	2.0000	2.2361	2.2361	2.6458	2.2361
Clas6	2.0000	2.0000	2.0000	2.0000	2.0000	0	3.0000	3.0000	2.2361	1.7321
Clas7	2.2361	2.2361	2.2361	2.2361	2.2361	3.0000	0	0	2.0000	2.8284
Clas8	2.2361	2.2361	2.2361	2.2361	2.2361	3.0000	0	0	2.0000	2.8284
Clas9	2.6458	2.6458	2.6458	2.6458	2.6458	2.2361	2.0000	2.0000	0	2.8284
Clas10	2.2361	2.2361	2.2361	2.2361	2.2361	1.7321	2.8284	2.8284	2.8284	0

Figure 4. Euclidean distance between classes.

- *Step 3.* In step 2 distances between various classes is calculated. Now in this step grouping of those classes takes place, those are in close proximity. Two individual classes or newly formed clusters are grouped together into one individual cluster using the linkage function or similarity measure as shown in Figure 5.

7.0000	8.0000	0
4.0000	5.0000	1.4142
1.0000	12.0000	1.4142
3.0000	13.0000	1.4142
2.0000	14.0000	1.4142
6.0000	10.0000	1.7321
9.0000	11.0000	2.0000
15.0000	16.0000	2.0000
17.0000	18.0000	2.2361

Figure 5. Linkage matrix.

In Figure 5 each row identifies a link between clustered classes. The first two columns denote the classes that have been clustered. The third column denotes the distance between these classes. Here the linkage function starts by grouping clas7 and clas8, which have close proximity and distance value=0 and continue grouping clas4 and clas5. The third row denotes that the linkage function clubbed together clas1 and clas12. Question here is that if in the PMS there are only 10 classes, what are the clas12, clas13, clas14, clas15, clas16, clas17 and clas18 in the Figure 6. Clas12 is the newly formed cluster by grouping the clas4 and clas5 and similarly clas13 is the newly formed cluster by groping class4 and clas5, clas14 is the newly formed cluster by combining clas1 and cluster 12 and so on.

- *Step 4.* Final output is shown in the form of dendrogram is shown in Figure 6. All the steps are implemented in matlab.

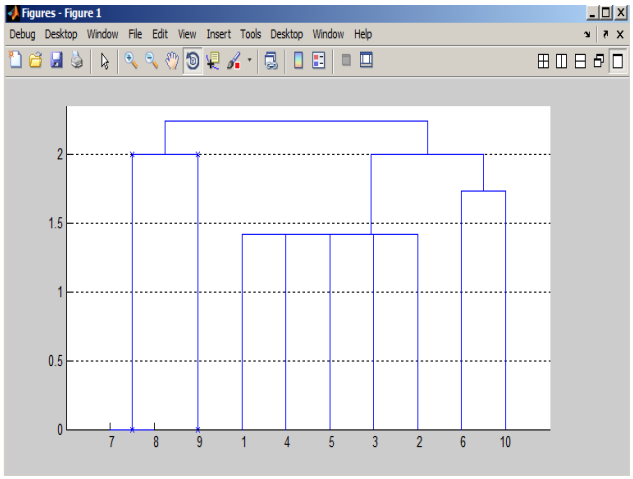

Figure 6. Dendrogram.

- *Step 5.* Now the clusters from the dendrogram are extracted using inconsistency matrix. Inconsistency matrix is shown in Figure 7.

Mean	Standard Deviation	Number of links	Inconsistency Coefficient
1.4141	0	1.0000	0
1.4141	0	2.0000	0
1.4141	0	2.0000	0
1.4141	0	2.0000	0
1.7321	0	1.0000	0
1.0000	1.4142	2.0000	0.7071
1.7154	0.2932	3.0000	0.9704
2.0787	0.1363	3.0000	1.1547

Figure 7. Inconsistency matrix.

In Figure 7 column1 gives mean of the length of all the links, column2 gives the standard deviation of all the links till its level. Column3 denotes the number of links and column4 indicates the inconsistency coefficient. Lower the value of inconsistency coefficient, more similar the classes within a cluster. Figure 7 shows that clas1, clas2, clas3, clas4, clas5 are more similar as they have zero inconsistency coefficients and validated our results where we have placed these classes into one cluster. Same way, clas10 with clas6 is having less similarity value and clas9 with clas7 and clas8 are least similar.

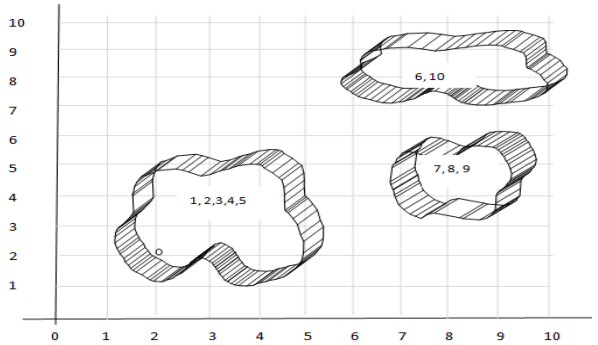

Figure 8. Clusters of classes.

We have presented the final result in Figure 8 in the form of cluster. Here each cluster represents a component consisting of number of classes inside it.

6. Conclusions and Future Work

In this paper we have proposed a novel approach for reverse engineering object oriented legacy systems.

We have provided a through literature in the field of reverse engineering and summarize the review in the form of a table. Reverse engineering is the powerful technique to understand a legacy system working. We have considered the source code as an important artefact in the proposed methodologies and class diagram is extracted using Rational Software Architect (RSA) tool. After getting the class diagram hierarchical clustering algorithm is implemented. The hierarchical clustering is a very appropriate algorithm to apply to any software system because the nature of this is clear and demonstrative. Classes are represented in the form of feature and weight is assigned to each feature using binary weighting scheme. Similarity measures are used to calculate the similarity between classes so that similar classes can be placed into a cluster that represent component. To validate the cluster formation inconsistency coefficient is applied. The proposed methodology is implemented on a case study PMS. Future work will focus on finding a cut-off position to get accurate clustering results and identification of suitable components from legacy object oriented system by clustering. We will also study procedure for implementation of validation rules for better output.

References

[1] Abbaneo G., Flammini F., Lazzaro A., and Marmo P., "UML Based Reverse Engineering for the Verification of Railway Control Logics," *in Proceedings of the International Conference on Dependability of Computer Systems*, Szklarska Poreba, pp. 3-10, 2006.

[2] AbdulMoiz S. and Basha J., "Extraction of State Transition Diagrams from Legacy C++ Application," *Procedia Technology*, vol. 4, pp. 543-547, 2012.

[3] Abu Abbas O., "Comparison between Data Clustering Algorithm," *The International Arab Journal of Information Technology*, vol. 5, no. 3, pp. 320-325, 2008.

[4] Anquetil N. and Lethbridge T., "Experiments with Clustering as Software Remodularization Method," *in Proceedings of 6th Working Conference On Reverse Engineering*, Atlanta, pp. 235-255, 1999.

[5] Beishu H. and Quian B., "JBOORET: An Object Oriented Reverse Engineering Tool," *in Proceedings of 25th International Conferenceon Computer and Application*, China, pp. 71-76, 1999.

[6] Bruneliere H., Cabot J., Dupe G., and Madiot F., "MoDisco: A Model Driven Reverse Engineering Framework," *Journal of Information and Software Technology*, vol. 56, no. 8, pp. 1012-1032, 2014.

[7] Chikofsky E. and Cross J., "Reverse Engineering and Design Recovery: A Taxonomy," *IEEE Software*, vol. 7, no. 1, pp. 13-17, 1990.

[8] Claudia P., Liliana M., and Liliana F., "Recovering Use Case Diagram form Object Oriented Code: an MDA Based Approach," *in Proceedings of the 8th International Conference on Information Technology: New Generations*, Las Vegas, pp. 737-742, 2011.

[9] Cosma D., Reverse Engineering of Distributed Object Oriented System, Ph.D Thesis, Politehnica, University of Timisoara, 2009.

[10] Davey J. and Burd E., "Evaluating the Suitability of Data Clustering For Software Remodularization," *in Proceedings of the 7th Working Conference on Reverse Engineering*, Brisbane, pp. 268, 2000.

[11] Demeyer S., Ducasse S., and Nierstrasz O., *Object Oriented Reengineering Patterns*, Morgan Kaufann, 2002.

[12] Dong X. Godfrey M., "A Hybrid Program Model for Object-Oriented Reverse Engineering," *in Proceedings of 17th International Conference on Program Comprehension*, Canada, pp. 1-10, 2007.

[13] Eshah N., Incorporating Object-Oriented Metrics into a Reverse Engineering Tool, Thesis, University of Putra, 2003.

[14] Favre L., "Formalizing MDA-Based Reverse Engineering Processes," *in Proceedings of the 6th International Conference on Software Engineering Research, Management and Applications*, Prague, pp. 153-160, 2008.

[15] Ferenc R., Beszedes A., Tarkiainen M., and Gyimothy T., "Columbus- Reverse Engineering Tool and Schema for C++," *in Proceedings of the International Conference on Software Maintenance*, Montreal, pp. 172 - 181, 2002.

[16] Gade A., Patil S., Patil S., and Pore D., "Reverse Engineering of Object Oriented System," *International Journal of Scientific and Research Publications*, vol. 3, no. 4, pp. 1-7, 2013.

[17] Jackson D., Somers K., and Harvey H., "Similarity Coefficient: Measures of Occurrence and Association or Simply Measure of Occurrence," *The American Naturalist*, vol. 133, no. 3, pp. 436-453, 1989.

[18] Kienle H. and Muller H., "Rigi: an Environment for Reverse Engineering, Exploration, Visualization and Redocumentation," *Science of Computer Programming*, vol. 75, no. 4, pp. 247-263, 2010.

[19] Kollmann R., Selonen P., Stroulia E., Systa T., and Zndorf A., "Study on the Current State of the Art in Tool- Supported UML-Based Static Reverse Engineering," *in Proceedings of the 8th Working Conference on Reverse Engineering*, Richmond, pp. 22-32, 2002.

[20] Lanza M., Object-Oriented Reverse Engineering Coarse-grained, Fine-grained, and Evolutionary Software Visualization, Ph.D Thesis, Universitat Bern, 2003.

[21] Lejter M., Meyers S., and Reiss S., "Support for Maintaining Object-Oriented Programs," *IEEE Transactions on Software Engineering*, vol. 18, no.12, pp. 1045-1052, 1992.

[22] López M., Alfonzo G., Pérez J., González S., and Montes R., "A Metamodel to Carry out Reverse Engineering of C++ Code into UML Sequence Diagrams," *in Proceedings of the Electronics, Robotics and Automotive Mechanics Conference*, Cuernavaca, pp. 1-6, 2006.

[23] Markus H. and Oliver C., "The FAMOOS Object-Oriented Reengineering Handbook," *ESPRIT Program Project no. 21975 (FAMOOS)*, Swiss Government under Project no .NFS-2000-46947.96 and BBW-96.0015, 1999.

[24] Meng F., Zhan D., and Xu F., "Business Component Identification of Enterprise Information System: A Hierarchical Clustering Method," *in Proceedings of the International Conference on E-Business Engineering*, Beijing, pp. 473-480, 2005.

[25] Muller H., Story M., Jahnke J., Smith D., Tilley S., and Wong K., "Reverse Engineering: A Roadmap," *in Proceedings of the Conference on The Future of Software Engineering*, Limerick, pp. 47-60, 2000.

[26] Ramasubbu S., Reverse Software Engineering Large Object Oriented Software Systems using the UML Notation, Master Thesis, Virginia Polytechnic Institute and State University, 2001.

[27] Rosenberg L., *Software Re-Engineering*, Software Re-engineering," *Lawrence E. Hyatt Manager, Software Assurance Technology Centre System Reliability and Safety Office Goddard Space Flight Centre*, (NASA), Report No. -301-286-7475, 1997.

[28] Salton G., "Development in Automatic Text Retrieval," *Science*, vol. 253, no. 5023, pp. 974-980, 1991.

[29] Sarkar M. and Chaterjee T., "Reverse Engineering: An Analysis of Dynamic Behavior of Object Oriented Programs by Extracting UML Interaction Diagram," *International Journal of Computer Technology and Applications*, vol. 4, no. 3, pp. 378-383, 2013.

[30] Sinha A. and Jain H., "Ease of Reuse: an Empirical Comparison of Components and Objects," *IEEE Software*, vol. 30, no. 5, pp. 70-75, 2013.

[31] Systa T., Dynamic Modeling In Forward And Reverse Engineering Of Object Oriented, Software Systems, Ph.D Thesis, University of Tampere, 1999.

[32] Talerico D., Grouping in Object-Oriented Reverse Engineering, M.Tech Thesis, University of Bern, 2003.

[33] Tonella P., "Reverse Engineering of Object Oriented Code," *in Proceedings of the 27th International Conference on Software Engineering*, Saint Louis, pp. 724-725, 2005.

[34] Tonella P., Torchiano M., Bois B., and Systa T., Empirical Studies in Reverse Engineering: State of The Art and Future Trends,"*Journal of Empirical Software Engineering*, vol. 12, no. 5, pp. 551-571, 2007.

Aman Jatain is currently working with Amity University, Gurgaon in the department of computer science. She is pursuing her doctoral degree in software engineering. She has 6 years of teaching and research experience. She has done her master's from Thapar University, Patiala and B.tech from MaharshiDayanand University, Rohtak. She has published more than 27 research papers in peer-reviewed international journals and international conferences.

Deepti Gaur received M.Tech in CSE degree from BIT Mesra Ranch, India and Ph. D. from Banashali University, Banasthali India. Dr. Gaur is presently working Associate professor in NorthCap University, formerly ITM University Gurgaon, Haryana, India. She has 17 years of teaching and research experience. She had successfully completed a sponsored project of AICTE Govt. of India. She has published more than 25 research papers in peer reviewed international journals and international conference of repute. She was convener of IEEE International Conference (IACC) 2014, Gurgaon, India.