PA2559-SOFTWARE METRICS ASSIGNMENT
Software Quality and Assessment

Serial No	Name	P- Number	E-mail
1	SRAVANI BEJAWADA	970525-9563	shiru2597@gmail.com
2	ANUSHA MOGILI	970626-8167	mogilianusha127@gamil.com
3	MANOJ KUMAR PALLAPU	970919-2018	manojkumarsep1997@gmail.com
4	GANNA ANIL	960710-7514	gannaanil96@gmail.com

I.ABSTRACT

Software metrics is the most important phase in software engineering. The main role of software metrics is to measure the quality of the software and to assess the standard of the final software by producing static analysis at every state of software development life cycle. This can be achieved with the help object-oriented design metric, where it contains a set of metrics to be measured such as system complexity.[1] In this paper, the review of different research papers related to object-oriented code metrics has been summarized. We have also discussed about metric tools suitable for extracting OO metrics.

Keywords: Object oriented, Metrics, Measures, tools, OSS.

II.INTRODUCTION

The main purpose of software quality and assessment in software engineering is to develop quality oriented software with less faults and maintain cost effective product. In order to obtain the main objective of the software product development, object oriented metrics have been implemented. Whereas, Metrics can be used to improve quality of the software, manage budget and prediction of cost estimation. The essential factors responsible for implementing object oriented code metrics are coupling, cohesion, program structure, classes, objects, defined attributes, interfaces, inheritance, polymorphism.[2] In the following sections, ten research papers are studied, with Description of the OO metrics, systems studies, tools used and visualisation methods and statistical measures used for analysis have been identified and categorized.

III.SUMMARIES

PAPER [1]: An Evolutionary Study of Fan-in and Fan-out Metrics in OSS[3]

Context: In this paper we make a study of fan-in and fan-out coupling metrics analysing with open source systems and comparing to obtain results between high incoming and low outgoing (vice versa), so that we can reduce excessive coupling.

Objective: The main objective of this paper is to analyse latest versions of five different open source system and extract fan-in, fan-out coupling metrics by using an automated tool. So, that the relationship between the well-known coupling metrics can be studied.

Main findings: In this paper, the existence of correlation between the values of FIN and FOUT has been measured. There was negative and positive correlation for two metrics depending on high FIN and low FOUT values for some packages and vice versa.

Limitations: The study was done on large packages with 10 classes in order to get better to statistical measures. Evolution of metrics have been done independently on each system with no accurate value.

PAPER [2]: Metric Pictures: the approach and applications[4]

Context: In software development process, finding a dead code is a huge problem and leads to flaws in system, even a main module of the system can also get effected. In order to reduce, metric pictures with visual approach has been proposed, which are raster images and 2D images with respect to space and are applied on metrics to a software product.

Objective: The following paper mainly focuses on producing metric pictures in order to analyse code logic and to help in dead code detection. The metrics pictures generated are from open source software eXVantage, using two different class metrics and suitable algebra the dead code classes are detected.

Main findings: Combining the two-class metrics CA and CE, index with zero columns and rows may lead to dead code classes and also zero value specifically doesn't mean as dead code.

Limitations: Adopted approach is not mainly useful for dead code detection, as metric images are hard to elaborate and analyse for specific method in a class of the whole project.

Paper [3]: Predicting Design Quality of Object-Oriented Software using UML Diagrams[5]

Context: This article is mainly assessment of the design quality of the software product in the object-oriented view with the help of the unified modelling language diagrams.

Objective: The main aim of the paper is assessing quality of design by using the developed software product. They divide the quality measures as the high, medium, low. So, we are forming the relationship between the product and the design quality of the software.

Main Findings: By adopting the following approaches, the relation between design quality attribute and product properties will give us accurate results for analysing the system based on UML diagram's and compare with design code metrics.

Limitations: In this article, the specified tools will only help in analysing the UML diagrams subsets, and cannot formulate the relation between selecting quality attributes and quality properties which leads to flaws in extracting metrics that are used to evaluate the whole system.

Paper [4]: **Empirical Analysis of Object Oriented Metrics using Dimensionality Reduction Techniques**[6]

Context: This paper discusses about the Dimensionality reduction techniques which are performed on the object-oriented metrics to remove the unwanted information in the metrics.

Objective: The main objective of the paper is to apply two-dimensional reduction techniques, i.e. Principle Component Analysis and Principal Axis Factoring to the object-oriented metrics of open source code for finding independent metrics which will give the useful data.

Main Findings: The metric with more variance (In this article greater than 0.7) are considered for the study of the metrics because if variance is more than 0.7 then we will have more useful information about metrics.

Limitations: In this article, the object-oriented metrics covered for a small set of data. So, we have fewer dimensions to analyse. When it comes to large set of data the study would become difficult to get the independent metrics.

Paper [5]: Application of object oriented metrics to C++ and JAVA: A comparative study.[7]

Context: In today's world the software measurement importance is increasing day by day, which has been a way to develop and design a new software metrics. In this paper the author studies and analyses the different programming languages in order to obtain the effective object-oriented metrics.

Objective: The main objective of this paper is to make a comparative study. As both Java and C++ show more, efficient object-oriented metrics comparatively with other programming languages. In this paper, it had been proven that C++ metrics are less effective, as it is not completely object-oriented language compared to Java programming language.

Main Findings: The results obtained from the features of object-oriented standards like data hiding, cohesion, information hiding, coupling, polymorphism and reusability are more effective in Java than C++ from comparative study.

Limitations: In this paper the counting rules are formulated based on their own requirements to extract different types of metrics. Which leads to inaccurate results in analyzed data as the counting rules can differ.

Paper [6]: Size and cohesion metrics as indicators of the long method bad smell.[8]

Context: This paper, focus on metric-based approach for refactoring and resolving the bad smell of long method. In large code resolving of errors manually is difficult so several metrics are proposed which are generic-scope for refactoring the features of urgency.

Objective: The main objective is to find which metrics are:

-To anticipate the presence of the long method smell

-The importance of the extract method for refactoring in case of urgency from software point of view in the background of open source java

Main findings: Method level cohesion metrics can be used to find out the thresholds and for the development of identifying the characteristics of an algorithm which are very important for refactoring the extract method.

Limitations: To analyze and resolve the problems in the complex code manually is time consuming and expensive. Only few tools are available for measuring metrics to get accurate values.

PAPER [7]: Re-engineering to analyse and measure object-oriented paradigms. [9]

Context: This paper explains the importance of static and dynamic analysis and gives detail description of static analysis of OO program for the beginners.

Objective: For modeling the developer and manager needs metrics to find the complexity, the two-new metrics for encapsulation are introduced in this paper

Main findings: Empirical study is necessary for development and maintenance of good quality software. During rework there is no need to work from the beginning has the concept is represented in visualized format.

Limitations: In this paper, class with more number of methods are assumed to be hard for reuse, whereas dividing the methods of a class into sub

classes makes the source code complex and space consumption occurs, links between functional code is disturbed which leads to possibility of dead code. Proposed metric approach for reengineering doesn't support, all private factors.

PAPER [8]: Assessing Traditional and New Metrics for Object-Oriented Systems [10]

Context: Software metrics are essential for software development process. In each and every phase of software development software metrics analysis should be done. By analysis of software metrics any intermediate and fine software metrics are measured. In this paper analysis has been conducted on all 111-metrics written in java and metrics derived from complex network theory and social network analysis(SNA).

Objective: The main objective is to analyse the traditional object-oriented metrics and metrics obtained from complex network and social network analysis and to analysis them each and every class level to find behaviour and relationship between them.

Main finding: Result show that all metrics have relationship between them. It also shown that many metrics have consistent co-relation between them by considering the measurements taken from analysis of metrics from class level. And it clearly shown that some software network analysis metrics are almost equivalent to each other so that one can used instead of another metrics.

Limitations: Study is only limited to java programs as many industrial code written in java and there are many free source codes available.

PAPER [9]: The Correlation between Source Code Analysis Change Recommendations and Software Metrics.[11]

Context: As the software is evolving large and complex Software quality is one of the challenging thing in the software development as it plays one of the main role in any software. Developers are many techniques to develop the software quality. In the paper we make a study about the correlation between software code analysis and software metrics.

Objective: The main objective of this paper is find how the some of the software metrics has been affected after fixing the warning detected by systematic code analysis tool(SCA) by using a fixed tool. In this we also measure the software metrics before and after modifying the defects.

Main findings: The results show that there is an impact on software metrics after fixing the warning detected by SCA tool. Care should be taken while gathering the information from SCA tool and result after SCA modification.

Limitations: The study was done only on 19 projects with various sized source code.

PAPER [10]: On the Statistical Distribution of Object-Oriented System Properties[12]

Context: In past many studies have conducted on statistical distribution of properties like size, defects. And found that many of the statistical distribution follows power law. In this paper they conducted a study on chidamber and kemerer metric suite for every java file in qualitas corpus.

Objective: The main objective of this paper is to find what are the distributions followed by chidamber and kemerer suite of every java project in the corpus.

Main findings: After conducting the studies on chidamber and kemerer suite and found that metrics of high values follow power law and rest of the metrices follow lognormal distribution. But they also found that most of the ck metrics are better described by double perito distribution.

Limitations: Out of 109 projects they could only measure 69 projects due to some technical problems and all the project belong to java as mainly many industrial code are written in java.

VI. OO METRICS USED

In paper [1]:

Category: coupling metrics

Which defines a class object referencing other class objects methods and properties is known as coupling.

FIN (fan-in) or incoming coupling metric, means count of number of functions or methods calling other functions or method.

FOUT (fan-out) or outgoing coupling metric, means count of number of functions or methods called by other components.

In paper [2]:

Category: class metrics, coupling metrics

Which helps to identify the features of class.

Afferent Coupling (CA): Specific class is referenced by other class methods and properties are measures.

Inbound Intra-Package Method Dependency (IIPM): Class methods or properties, from same package is depended on specific class is measured.

Inbound Extra-Package Method Dependencies (IEPM): Class methods or properties, from different package is depended on specific class is measured.

Efferent Coupling (CE): Other class methods are referenced by specific class.

Outbound Extra-Package Feature Dependencies (OEPF): No of methods and properties of different package are depended on specific method.

Outbound Intra-Package Feature Dependencies (OIPF): No of methods and fields of same class and inherited from same package is depended on specific methods.

In paper [3]:

Category: Design Metrics

Design size in class (DC), Number of hierarchies (NH), Average count of ancestors (ACA), Data access metrics (DAM), Cohesion among methods in class (CAM), Measures of aggregation (MA), Count of polymorphic methods (CPM), Class interface size (CIS), Total number of methods (TNM), Measure of abstraction (MOA), Class Coupling (CC), Cyclomatic Complexity (CC), Lack of cohesion in methods (LCOM), Weighted methods per class (WMC), Lines of code of methods (LOCM).

In paper [5] and paper [4]:

Category: size metrics:

-Number of attributes for class (NOA), is a measure of number of fields encountered in a class or module.

- Number of methods per class (NOM), is a measure which helps to calculate the mean of all class operations per class.

-Weighted methods per class (WMC), in which the complexity is calculated by using traditional cyclometric complexity metric.

Category: complexity metrics:

-Response for a class (RFC), in which a set of class response is measured by grouping.

Category: inheritance metrics:

-Depth of inheritance tree (DIT), the class position is measured in inheritance hierarchy.

-Attribute Inheritance Factor (AIF) it attributes which measures the sum of all attributes of the class is divided by the total number of attributes.

-Internal Inheritance Factor (IIF), Number of methods overridden by a subclass (NMO), Method Inheritance Factor (MIF), Specialization Ratio (SR), Reuse Ratios, Import coupling to class attribute (IC_Attr), Import coupling to class parameter (IC_Par).

Category: coupling metrics:

-Message Passing Coupling (MPC), in a class the number of method calls.

- Coupling between object class (CBO), inheritance related to coupling of class are measured.

-Number of Children (NOC) In a project number of subclasses of each class that occur.

-Data abstraction coupling (DAC), Coupling Factor (CF).

Category: cohesion metrics:

-Lack of cohesion in methods (LCOM), which measures the different methods in a class based upon the attributes methods used in a class.

-Loose class cohesion (LCC), Tight Class Cohesion (TCC), Information based Cohesion (ICH).

Category: information hiding metrics:

-Method Hiding Factor (MHF) and Attribute Hiding Factor (AHF) metrics .

Category: polymorphism metrics:

-Polymorphism Factor (PF) .

In paper [6]:

Category: size metrics:

Lines of code (LOC), with comments the number of lines of code for each class is measured.

Category: cohesion metrics:

Lack of cohesion in methods (LCOM4) is more efficient and good compared to LCOM1 and LCOM2, LCOM1 has low cohesion to overcome these metrics LCOM2 is introduced. As these metrics are not completely object-oriented these can be applied for C++, COH, CC, LSCC, DCD and SCOM.

In paper [7]:

Category: -Information Hiding Factor:

Attribute Hiding Factor (AHF), Method Hiding Factor (MHF)

Category: -Inheritance Factor:

Class Inheritance Factor (*CIF)* 'Method Inheritance Factor (*MIF) &* 'Polymorphism Factor' (*POF)*

Category: -Encapsulation Factor:

Public Factor (PuF), Private Factor (PrF), Weighted Class Complexity Factor(WCCF)**,** No. of Ancestor Count.

In paper [8] and paper [9]:

Category: Object oriented class metrics:

Line of code(LOC): Metrics used to count number of lines of code in program, Fan in(FIN): Count of number of functions or methods calling other functions, Fan out(FOUT): Count of number of functions or methods called by other components.

Coupling between objects(CBO): It calculates the quantity of class coupled, Response for a class(RFC): Set of all different methods and constructors applied by a class, Weighted methods per class(WMC): Measure of the all classes defined.

Lack of cohesion of methods(LCOM): Used to calculate cohesion by finding the difference between number of non-cohesion pairs and cohesion pairs, Number of children (NOC): number of subclasses occurred to each class, Depth of inheritance tree(DIT): maximum length of a class from node to the root node.

Category: Software network analysis metrics:

Size: Number of nodes in a class, Ties: Number of edges.

Brokerage (BRKG): Pairs that are not directly connected to ego network, Effective Size of Ego Network (EFFSZ): no of nodes minus average number of edges a node has, Normalized nr. Of Weak Components (NWCP): Measure of number of disjoint set with no edge connected divided by size, Reach – Efficiency (REFF): No of node in reach divide by neighbour, Closeness (CSNS): Measure of length of shortest distance between the nodes and other nodes, Information Centrality (INFCY): Mean of length of all node from starting to ending node.

In paper [10]:

Category: Coupling metrics:
Coupling between objects(CBO): It calculates the quantity of class coupled, Response for a class(RFC): Set of all different methods and constructors applied by a class, Message passing coupling(MPC): Measure of number of methods calls made by a class, Data abstraction coupling (DAC):It is a data type in which other data types are member or local variables, Information flow based coupling (ICP), Ancestor class – attribute import coupling (ACAIC), Descendant class-attribute export coupling (DCAEC), Ancestor class-method import coupling(ACMIC), Descendant class-method import coupling(DCMEC),Ancestor method-method import coupling (AMMIC),Descendant method-method import coupling(DMMEC), Number of children (NOC): Number of subclasses occurred to each class Number of attribute per class (NAC): It is the measure of average number of attributes per class Cognitive code complexity (CCC): It is measure of how hard to understand the code.

V. SYSTEMS STUDIED

In paper [1]: The five object oriented systems based on java OSS are selected with sufficient versions available and with majority no of downloads are selected to analyse.
i). Jasmin: Which is a well-known java assembler, with five versions. It is a JVM assembler, that converts java language into a byte code.
ii). SmallSQL: Is a java database management system for applications, consisting of 7 versions.
iii). DjVu: Is an applet provider and consisting totally of 8 versions.
iv). pBeans: Provides ORM functionality for Database with ten versions.
v). Asterisk: Provides interaction with Asterisk PBX sever for java applications and consists of six versions.
In paper [2]: eXVantage: It is an open source software used for as a tool suite for code coverage, system testing, debugging. Is also effective automated tool known for performance improvement.
In paper [3]: In this paper, five open source projects have been selected with the known design information and classified them into the Low, Medium and High. The five software projects are JDMO, Taming Java thread, Bonforum, Eview Applet, Student Project.
In paper [4]: In this paper, the Object Oriented Mobile Robot Model (OOMRM) which is the open source code and it is in C ++ library that used to calculate the metrics data.
In paper [5]: In the following paper the study has been done on 15 program codes in two different languages with same functionalities and implementing the main features of object-oriented paradigm
In paper [7]: For OO file a student program is added to execute the addition of neros which is parsed into a translator and from which text modeled and metrics files are the two files are founded.
In paper [8]: They have selected different system based on java to analyse the metrics on class level
1) Qualitas corpus: Used to perform large empirical studies and comparison of same artifact measurement.
2) Eclipse: It is an integrated development environment with plug in which are used to customize the environment.
3) NetBeans: It is also an integrated development environment used to develop java based applications.

4) Galleon: It is a java run time environment which allows java program on different OS.
5) Fit java: Frame work for integrated testing(FIT) which help to run the customer test automatically to find how software works.
6) Junit: It is unit testing framework useful for test driven development.
In paper [9]: #C language is used in writing the source code in all the projects. Also, the project is divided into different domains
-Desktop
-Enterprise systems
-Mobile development
-Multimedia
In paper [10]: Qualitas corpus: used to perform large empirical studies and comparison of same artifact measurement. They used Quality corpus r distribution which consists of all the recent versions of 112 systems.

Eclipse sdk: It is software development kit which consist of java tools for developers.

VI. TOOLS USED

In paper [1]: JHwak tool[13]: Is a static code analysis tool widely used for quality management and cost-effective assessment. It is better known for complexity, size, relation between classes and packages.
In paper [2]: Dependency Finder tool[14]: Is used to analyse java code during compilation. Core version is available as it helps in extracting dependence graphs and useful in mining important data. It is also known as suite of tools consisting of JarJarDiff and used for computing OO software metrics.

In paper [3]: In article they used Metrics 1.3.4 and Team in a box as plugins for the Eclipse and JHwak to get the metrics for the selected software.

In paper [4]: In paper, we use two metric extraction tools they are Imagix 4D, Star UML & SD Metrics.

Imagix 4D: It is used to analyse the source code and interpret it into easily understandable form, make the speed execution of large set of data.[15]

Star UML:[16] It is the open source tool which works on the unified modelling language.

SD Metrics:[17] It is object oriented design quality measurement tool for the UML and it mainly measures the structure of the UML models.

In paper [9]: Static source code analysis tools.

Fxcop:[18] It is a static source code analysis tools released by Microsoft which analyse the compiled object code.

Coverity :[19] It is product from Synopsys which help to find defects and security burden in source code.

Moose: It an open source platform for data analysis developed in Pharo which also provide services like modelling, mining, measuring etc.

Hp fortify: It has 6 analysers like data flow, control flow, semantic flow, structural, configuration and buffer which help to find the vulnerabilities in the code.

Just code: It is developed as an add on for visual studio which boost .Net capacity by giving fast solution for code scrutiny, error checking, code navigation and refactoring. With help of cross language engine, it provides for c#, C++. Html etc.

Para soft: It is used to analyse, test, find defect and security of the application.

Code it. Right: It is a tool which join static tool analysis automatic refactoring into one application and it also consequently remedy correct mistakes and violation.

In paper [10]: Ckjm tool: It is the tool used to measure ckjm metrics by processing byte code of compiled java files.

VII. VISULIZATION METHODS

In paper [1]: Line chart with symbolic data point graphs had been used to describe the five open source systems classes and fin, fout metrics.

Jasmin: For this system two packages had been analysed to study the correlation coefficients. Jas package step line chart is used to represent and it states about the positive correlation between classes and metrics. Jasmine package consists of negative correlation in reverse step line chart.

SmallSQL: For this vertical segment line chart is used to describe the high values of FIN always. Whereas, the FOUT values increase from version to version.

DjVu: For this system the vertical, horizontal segment line chart and step line chart are drawn for three packages while comparing the FIN AND FOUT correlation values. In two packages the FIN values are high than FOUT values. But in the third package the FOUT value is higher than FIN value.

PBean: Here two packages are considered with vertical segment line graph. Where this system consists of highest FOUT values in maximum version of the whole system.

Asterisk: Three packages are considered in vertical and horizontal segment line chart to explicitly compare the high fin and FOUT values. For every package FOUT has the low correlation value.

In paper [2]: Colour representation of metric picture is used for visualisation methods, where the pictured graphs have been produced with different colours for different operations. For CA metrics obtained from eXVantage system is represented with zero columns in scattered plot graph and without is represented in vertical segment line graph highlighted with cyan primary colour band, even CE metrics is displayed by horizontal segment line graph where dead code is displayed zero on columns. Combining CA and CE metrics graph is highlighted with cyan for identifying dead code.

In paper [5]: Multiple bar chart is used for the comparison of all the OO metrics at class and system level. From bar chart the metrics can be visualized diagrammatically in the form of bars.in figure2, the size and complexity metrics for C++ and java are calculated and represented in the form of bar graph which shows approximately nearer values. Whereas, when we observe figure5 and figure6 the cohesion and coupling features are not so prominent in C++ there is huge difference between the metrics for C++ and java.

In paper [6]: From the dataset, Scatter plot are used to explain the correlation between cohesion and size metrics. From the statistical analysis logistic regression is chosen for long method and spearman correlation is chosen for extract method urgency.

In paper [7]: Lognormal distribution in JMeter: it is java application designed to calculate functional behaviour and performance.

Pareto distribution in Exportal system:

Gamma distribution in Tomcat system:

In paper [9]: Histogram has been used as method to visualize the extracted results. In these histograms they compared the mean of metrics after and before the SCA recommended changes.

In paper [10]: In this line chart has been used which is created by connecting the point marked to display the information. Using this line chart, they displayed the cumulative distribution of functions

of the metrics in which both axes are marked with logarithmic scale.

VIII. STATISTICAL MEASURES

In paper [1]: The correlation values for every is calculated with the help of Pearson's, Kendall's, spearman's values. The correlation value states the positive and negative relation between the two metrics. The correlation value is based on packages of the system and the range of these values are from 0.01 to 0.05. To know the highest or minimal values of FIN and FOUT metrics, the mean and median had been calculated and analysed with the help of different line graphs for each package.

In paper [2]: The metric picture allows to form a set of attributes such as unary operator. It is classified into negative operator, contrast stretch and δ-thresholding are defined for metric picture given value X[i,j], limit {1…n}, classes are used to draw a graph between CA and CE metrics calculated from algebra formula.

In paper [3]: In the paper, they used the mean and standard deviation as the statistical measures to get the product quality metrics after the correlation analysis.

In paper [4]: In this paper, they used the variance percent as the statistical measure to calculate the interpretations of principle component and factors using this dimensional technique we get the independent metrics.

In paper [5]: From the experimental results, in table2 the size and complexity metrics are calculated the total number of classes and average is taken for calculating java and C++ languages. In table3&4, inheritance metrics for class and system level are calculated in separate tables to get appropriate results. Similarly, for coupling and cohesion also the procedure is followed. From the results obtained from all the metrics proves that java language is more prominent.

In paper [6]: The metrics score is calculated from the prediction model and prediction power. In prediction power the measures like accuracy, precision and recall are taken and in prediction model the measures like beta values and significance are taken for cohesion metrics.

In paper [7]: AHF and MHF are calculated to know whether the code is good for programming. Maximum value of AHF and MHF shows that the code is good. By calculating the value of class inheritance factor using ancestor count helps to remove the problems in class level. By using public factor and private factor metrics the anomalies are removed by measuring both unity and integrity parameters.

In paper [8]:

Mean: Sum of the object divided by total number of object.

Median: Separation of higher data sample from lower half of data sample.

Variation coefficient: It is value of standard deviation divided by mean.

Normalized 90% quantile: 90% quantile divided by mean.

Using this mean, median, variation coefficient and normalized 90% quantile they calculated the co-relation between the metrics and also the behaviour of them. Using normalized 90% quantile they calculated the leptokurtotic distribution and they also created a correlation matrix of each metrics in system.

In paper [9]: Mean is used as a statistical measure to calculate the impact of SCA recommended changes on the metrics and they also calculated the mean of the metrics before changes to compare the metrics before and after the changes.

In paper [10]: Cumulative distribution function: it is a probability that the random variable evaluated at argument of the function can less or equal value of argument of the function.

Part-II: Reflections on the suitability of the selected tools

In this paper, we have studied and analysed about different object-oriented metrics. We have selected two open source metric tools which are suitable for extracting various code metrics.

Metrics Reloaded for IntelliJ IDEA[20]: Is automated metrics tool which is used for measure of LOC, module, complexity metrics and all languages are supported.

STAN (Structure Analysis for Java) for Eclipse IDE[21]:Is an eclipse metrics plugin which is used for measuring quality, estimating flaws in design.

TOOLS CONFIGURATION

The two-metrics tool are easy to install and run on open source system to extract code metrics

Metrics Reloaded: Metrics reloaded is available plugin for IntelliJ IDEA and need to install and restart the IDE. While implementing on a OSS we need to analyse the code by executing it. A common calculate metrics window is displayed, we need to select the type/category of metrics we need to extract and the results are displayed.

STAN (Structure Analysis for Java): Is an Eclipse metrics plugin found in eclipse marketplace. After installation process we need to configure the execution based on our requirement such as, whether the extracted metrics is for whole class or a specific method and run the OSS project on structured metrics to obtain the visualization design of the whole project can be extracted.

OBJECT ORIENTED METRICS

Metrics Reloaded tool [20] covers most of the object-oriented metrics and are mainly classified into package, module, project, class, method levels:

- Chidamber-kemrer metrics: CK metrics further categorized into class and module level, where CBO, DIT, LCOM, NOC, RFC, WMC metrics will be calculated with accurate values.

- Class count metrics: are divided into three types such as package, module, project levels. Number of classes(C), number of product classes (Cp), number of test classes(Ct) metrics values are calculated.

- Complexity metrics: are divided into five types such as package, module, project, class, method levels. Further classified in essential cyclomatic complexity(ev(G)), design complexity(iv(G)), cyclomatic complexity(v(G)).

- Other types of metrics in Metrics Reloaded are: dependency metrics, Junit and Javadoc metrics, line of code metrics, MOOD metrics, number of files metrics, Martin packaging metrics can be measured. So, we draw a conclusion that most of the object-oriented metrics can be covered in this tool

STAN: Is most widely used metrics plugin to extract wide range of metrics, mainly for quality purpose. STAN is ahead of mostly used metrics stool in order to extract all types of object oriented metrics. Some of them are listed below:

STAN has an incredible feature that is, it can display extracted metrics into visualization design format. This is classified into five types such as composition, coupling, pollution, sandbox, and distance where graphical representation is shown.

Stan can show the accurate statistical measures for OO metrics. They are classified into the following categories:

- **Count metrics**: Where the number of libraries, packages, units, methods, fields, ELOC are measured.

- **Complexity**: Here, the size, CC (average cyclomatic complexity), Fat-libraries, packages, units, Tangled, (ACD)average component dependency modules will be calculated.

- **Robert C. Martin metrics**: The average distance and absolute distance between the instability and abstractness is presented.

Same as metrics reloaded the STAN metrics plugin also measure Chidamber & Kemer metrics accurately.

Therefore, both the tools satisfy most of the object-oriented metrics.

TYPES OF REPORTS

Types of reports generated while extracting code metrics on OSS are:

STAN[22], generated report are easily understandable, customized and can capture efficient structural quality while recording the extracted metrics based on brief explanations consists of visualization design format, graphs, statistical measures represented in a tabular forms. The following list are the reported generated by STAN tool:

- **Library dependency**: Comes under visualization design format, where a graphical representation is used to elaborate the depend ices between the libraries.

- **Metrics summary**: Where different types of metrics have been listed and provides accurate statistical measures. Represented in a tabular form.

- Top violations: In STAN there is an adorable feature known as Rating where it is represented with graphical traffic light partitioning into the extracted value ranges from high to low as red subranges, amber, green. Using these colours the artifacts are classified into metrics and accurate values are shown in desired colour, to show the acceptable and unacceptable metric values.

- Pollution charts: Is represented in visualization design format where a pie chart is shown in different colours to describe the metrics contributions of artifacts to violations in order to control them.

- Queries: Is used to trace the violated metric values of artifact threshold or rating types.

- **Map view**: Is a visualization hierarchical design used for mapping artifacts according to their ratings. STAN uses a tree maps tool to present it.

Metrics Reloaded tool also consists of tabular form, but less visualization design formats compared to STAN tool.

Metrics reloaded: Reports general consists of types of metrics, level of modules and accurate values of different metrics in tabular form with average and total count at bottom of the table.

Metrics reloaded tool consists of visualization methods such as distribution, histogram, pie chart graphs for each type of metrics to understand the overall view of the extracted metrics.

Compared to STAN, metrics reloaded may be less effective for types of reports generated such as graphical representation or diagrammatical view but it a best tool in calculating the accurate values of different OO metrics same as STAN.

LEVELS OF METRICS GENERATED

Metrics can be Extracted at package or class level and at the same time, both the package and class levels even at method level. This is proven by implementing on a OSS in eclipse and Intellij IDEA, IDE platform. The results obtained are, metrics can be generated at both package and class levels. At package level the metrics calculate the

dependences between the classes and the main class of the whole project is analysed and at class level, the metrics measure is constrained to its methods and functions.

REFERENCES:

[1] S. K. Dubey, A. Sharma, and D. A. Rana, "Comparison Study and Review on Object-Oriented Metrics," p. 11, 2012.

[2] "Object-Oriented Metrics," in *The IT Measurement Compendium*, Springer, Berlin, Heidelberg, 2008, pp. 241–255.

[3] A. Mubarak, S. Counsell, and R. M. Hierons, "An evolutionary study of fan-in and fan-out metrics in OSS," in *2010 Fourth International Conference on Research Challenges in Information Science (RCIS)*, 2010, pp. 473–482.

[4] R. Francese, S. Murad, I. Passero, and G. Tortora, "Metric pictures: The approach and applications," in *The 2010 International Conference on Computer Engineering Systems*, 2010, pp. 320–325.

[5] V. Yadav and R. Singh, "Predicting Design Quality of Object-Oriented Software using UML diagrams," in *2013 3rd IEEE International Advance Computing Conference (IACC)*, 2013, pp. 1462–1467.

[6] R. Sharma, S. Sabharwal, and S. Nagpal, "Empirical analysis of object oriented metrics using dimensionality reduction techniques," in *International Conference on Recent Advances and Innovations in Engineering (ICRAIE-2014)*, 2014, pp. 1–5.

[7] U. Kumari and S. Bhasin, "Application of Object-Oriented Metrics To C++ and Java: A Comparative Study," *ACM SIGSOFT Softw. Eng. Notes*, vol. 36, no. 2, p. 1, Mar. 2011.

[8] S. Charalampidou, A. Ampatzoglou, and P. Avgeriou, "Size and cohesion metrics as indicators of the long method bad smell: An empirical study," 2015, pp. 1–10.

[9] S. Singh, K. S. Kahlon, and P. S. Sandhu, "Re-engineering to analyze and measure object oriented paradigms," in *2010 2nd IEEE International Conference on Information Management and Engineering*, 2010, pp. 472–478.

[10] G. Concas, M. Marchesi, A. Murgia, S. Pinna, and R. Tonelli, "Assessing traditional and new metrics for object-oriented systems," 2010, pp. 24–31.

[11] A. Abuasad and I. M. Alsmadi, "The correlation between source code analysis change recommendations and software metrics," 2012, pp. 1–5.

[12] I. Herraiz, D. Rodriguez, and R. Harrison, "On the statistical distribution of object-oriented system properties," in *2012 3rd International Workshop on Emerging Trends in Software Metrics (WETSoM)*, 2012, pp. 56–62.

[13] "JHawk - the Java metrics tool - Product Overview." [Online]. Available: http://www.virtualmachinery.com/jhawkprod.htm. [Accessed: 20-Apr-2018].

[14] "Dependency Finder." [Online]. Available: http://depfind.sourceforge.net/. [Accessed: 20-Apr-2018].

[15] "Imagix 4D User Guide - Information Displays." [Online]. Available: https://www.imagix.com/user_guide/information-displays.html. [Accessed: 20-Apr-2018].

[16] "StarUML." [Online]. Available: http://staruml.io/. [Accessed: 20-Apr-2018].

[17] "SDMetrics - the design quality metrics tool for UML models." [Online]. Available: https://www.sdmetrics.com/. [Accessed: 20-Apr-2018].

[18] "FxCop." [Online]. Available: https://msdn.microsoft.com/en-us/library/bb429476(v=vs.80).aspx. [Accessed: 20-Apr-2018].

[19] "Synopsys Static Analysis Tool (Coverity) | Synopsys." [Online]. Available: https://www.synopsys.com/software-integrity/resources/datasheets/coverity.html. [Accessed: 20-Apr-2018].

[20] "MetricsReloaded :," *JetBrains Plugin Repository*. [Online]. Available: https://plugins.jetbrains.com/plugin/93-metricsreloaded. [Accessed: 20-Apr-2018].

[21] "STAN - Structure Analysis for Java," *Eclipse Plugins, Bundles and Products - Eclipse Marketplace*. [Online]. Available: https://marketplace.eclipse.org/content/stan-structure-analysis-java. [Accessed: 20-Apr-2018].

[22] B. I. C. UG, "STAN." [Online]. Available: http://stan4j.com/reports/. [Accessed: 20-Apr-2018].