Object Oriented Metrics In NASA

N.PRANITHA
SCSE, M.Tech Integrated Software Engineering,
(Student) VIT University Chennai, Tamil Nadu, India
n.pranitha2018@vitstudent.ac.in

T.SUPRIYA
SCSE, M.Tech Integrated Software Engineering,
(Student) VIT University Chennai, Tamil Nadu, India
t.supriya2018@vitstudent.ac.in

K.THANMAYEE
SCSE, M.Tech Integrated Software Engineering,
(Student) VIT University Chennai, Tamil Nadu, India
kurparthi.thanmayee2018@vitstudent.ac.in

Abstract - Object-oriented design and development is becoming common in today's software development environment. Object oriented development involves not only a different approach to design and execution, it requires a different method to software metrics.Since object oriented technology practices objects and not algorithms as its fundamental building blocks, the method to software metrics for object oriented programs must be different from the standard metrics set. Some metrics, like lines of code and cyclomatic complexity, have become recognized as "standard" for traditional functional/ procedural programs, but for object-oriented, there are many strategic object oriented metrics in the literature. The question is,"What article arranged measurements will a task use, and can the entirety of the conventional measurement salter to the item arranged climate?"

In this, the Software Assurance Technology Center (SATC) at NASA Goddard Space Flight Center contraries its way to deal with choosing measurements for an undertaking by first decision the properties related with object arranged turn of events. Within this framework, nine metrics for object oriented are designated. These metrics include 3 traditional metrics adapted for an object oriented environment, and 6 "new" metrics to evaluate the principle object oriented structures and concepts. The metrics are first well-defined, then using a very simplistic object oriented example, the metrics are useful. Interpretation guidelines are then discussed and data from NASA projects are used to establish the application of the metrics.

In the experience of the SATC, projects select the data they collect by default - if the tool they are using assembles it, the project collects it. The purpose of this paper is to support project directors choose a comprehensive set of metrics, not by defaulting, but by using a set of metrics based on qualities and features of object oriented technology.

Keywords –object oriented,Metric,Measure,Development

INTRODUCTION

Object-oriented design and development are popular ideas in today's software development environment. they're frequently heralded because the solution for explaining software difficulties, while in realism there's no silver bullet; object oriented development has showed its value for systems that essential be maintained and modified. Object oriented software development

needs a divergent approach from more traditional functional decomposition and data flow development methods. While the functional and data flow methods commence by considering the systems performance and/or data separately, object oriented analysis approaches the matter by searching for system entities that combine them. Object oriented analysis and style focuses on objects because the main agents involved during a computation; each class of information and connected operations are collected into one system entity.

This paper will early briefly discuss 9 metrics currently being applied by the SATC to NASA object oriented projects. These include 3 "traditional" metrics adapted for an object oriented environment, and 6 "new" metrics to assess the principle object oriented structures and ideas. The SATC's method to classifying a collection of object oriented metrics was to identify the first critical concepts of object oriented design and to pick metrics that evaluate those areas. The metrics attention on internal object structures that reflect the complexity of every individual entity, like methods and classes, and on external difficulty that measures the connections among entities, like coupling and inheritance. The metrics quantity computational complexity that affects the productivity of an algorithm and therefore the use of machine resources, further as psychological complexity issues that affect the flexibility of a programmer to make, comprehend, modify and maintain software.But as vital because the metrics chosen is what the metrics "tell" the developers and managers about the excellence and object oriented structure of the look and code; metrics without interpretation guidelines are of little value. Metrics for object oriented development may be are relatively new field of study, however, and haven't reached maturity. Although some thresholds are proposed by analysis developers, there's little application data to justify specific"good" and "bad" ranges.

Knowledge and knowledge of the programmers, managers, researchers and SATC staff presently assist because the basis for the interpretation guidelines of the metric analysis presented during this paper. As each metric is well-defined, guidelines for inferring the values are suggested. In many situations, however, to expand one metric means a trade-off with another.

This paper jumps with an summary of the metrics recommended by the SATC for object oriented systems. These metrics include changes of “traditional” metrics still as “new” metrics for specific object oriented structures. Since the article oriented metrics need a cursory considerate of the thing oriented ideas, Section 3 presents a pictorial image of the basic object oriented structures and expresses the key terms. In Section 5, we discuss the metricsin-depth. the look for an easy object oriented example is employed to validate the metric calculations. In Section 5, interpretation guidelines are conversed. Then in Section 6 the applications and interpretations of the metrics are conventional using NASA project data.

I.OBJECTIVE

- ✓ Choosing the object oriented metrics by the Software Assurance Technology Center (SATC) at NASA Goddard Space Flight Center.
- ✓ Understanding the various metrics at NASA to evaluate the object oriented metrics.

II.LITERATURE REVIEW

1)Author: Ramesh Ponnala

Title: Object Oriented Dynamic Metrics in Software Development

Content: Software development is the procedure of examining, designing, programming ,documenting, testing, and bug fixing elaborated in generating and preserving applications as per the client necessity. In software application development, Object Oriented Programming (OOP) Approach occupies a crucial role. Object Oriented (OO) metrics are required to evaluate belongings of object oriented software applications. Many software metrics have developed for Object Oriented paradigms such as abstraction, class, object, inheritance etc.. to determine several features like software quality, coupling, cohesion etc.. Object Oriented metrics are divided into static and dynamic. Software metrics are one of the most important tools required in Software Engineering in order to extract the quality grade of software applications. Software metric is a effortless quantitative measure extractable from any feature of the software development life cycle. Object Oriented Dynamic Metrics (OODM) is very supreme to notice the runtime behavior of Software Application. As the dynamic metric can be assessed during the runtime of the application, it is very hard to differentiate to the static metrics estimation. All though dynamic metrics have an advantage over the static metrics, they are hard to estimate. So the static metrics can be used during the estimation of dynamic metrics.

2)**Author:** Daniel Rodriguez Rachel Harrison
Title: An Overview of Object-Oriented Design Metrics
Content: The uses of metrics is inorder to control, predict and improve the quality of software product is escalating the popularity. There are many various kind of metrics that required to be used in projects. As usage of the Object Oriented paradigm had become extensively used, the arrival of certain set of metrics for these systems and has also attained popularity in the last years. As the use of the Object Oriented paradigm doesn't gets quality by itself. The objective of using Object Oriented metrics is to estimate the systems in order to acquire high quality outcome. Some traditional metrics or moderation of them can be used in Object Oriented paradigm, mainly in the method level. A set of Object Oriented design metrics has been narrated with some explanation guides as a way to assess systems inorder to acquire a robust, high quality outcome. Each metric was narrated by considering very important features such as, how to use it, analyzing guidelines, produced thresholds whenever it is possible, and evaluate its appropriateness and functionality. Some issues and suggestions are also listed such as some uncertainties in some of the definitions, utility for its purpose and its sustainability. In a way forward, lessons lettered will helps to control those problems. More work about factual validation is mandatory for proving statistical and experimental techniques inorder to increase their simplification.

3)Author: Dr. Linda H. Rosenberg and Lawrence E.
Title: Software Quality Metrics for Object-Oriented Environments
Content: Object Oriented Software Development needs a incompatible procedure from traditional functional decomposition, data flow development methods. This comprises the software metrics required to estimate object

oriented software. The theory of software metrics are well confirmed, and many metrics related to product quality have developed and used. Using object oriented analysis and design methodologies the popularity is gained. Product Quality for code and design has 5 features. These are Efficiency, Complexity, Understandability, Reusability, and Testability or Maintainability. The SATC has presented 9 metrics for object oriented systems. They includes the key topics for object oriented designs: methods, classes (cohesion), coupling, and inheritance etc.. For very metric, threshold values can be assumed, depending on the applicable quality features and the application intentions. Further work will define criteria for metrics. That is, acceptable ranges for every metric has to be developed, depending on the effect of the metric on beneficial software qualities.

4**)Authors :**Priyanka Yadav , Khalid Hussain , Ashima Gambhir
Title : Analysis of Object Oriented Metrics.
Content: Object oriented design metrics is a crucial part of software environment. Object oriented measurements are required to evaluate quality of software. The metrics for object oriented design centers on measurements that are experimental to the class and design elements. These measurements allows designers to approach the software too early in process, doing modifications that will decrease the complexity and increase the ongoing potentiality of the design. Theoretical analysis of these metrics suggests that out of many various Object Oriented metrics, 6 metrics (WMC, DIT, CBO RFC, DAC, LCOM, and NOC) provides enough details for usage and other metrics are either subset of the other metrics or are by giving same information in dissimilar format. This clearly supports the culmination drawn from theoretical analysis. That is, many metrics proposed are based on comparable ideas and also provides somewhat unnecessary information.

5)Author: Seyyed Mohsen Jamali .
Content: Given the central role that software development plays in the delivery and application of information technology, managers are progressively pointing on process development in the software development area. This request has forced the supplying of a number of new improved approaches to software development, possibly the most important object orientation (OO). In addition, the focus on process development has increased the pressure for software measures, or metrics with which to control the process. The need for such metrics is especially severe when an organization is embracing a new technology for which developing practices have to be developed. This research marks these needs through the development and implementation of a suite of metrics for OO design. Metric data gives quick feedback for software designers and managers. Analyzing and getting the data can finds design quality. If used in a right manner, it leads to a consequential decrease in costs of the complete implementation and developments in quality of the end product. The upgraded quality, in turn reduced for future maintenance efforts. Using early quality measures based on objective verifiable proof is

therefore a realistic objective. According to my opinion it is influenced for the developer to get very fast and continuous feedback about the quality in design and implementation of the product They are guidelines that gives a signal of the progress that a project has made and the quality of design.

6)Author: Dr. K.P. Yadav, Ashwini Kumar
Title: Object Oriented Metrics Measurement
Content: The importance of software measurement have lead to development of new software measures. Many metrics have been implemented with respect to various constructs like class, coupling, cohesion, inheritance, information hiding and polymorphism. The main role that software development plays in the transpoting and application of information technology, managers are very much focusing on process advancement in the software development area. it is very tough for project managers and practitioners to select measures for object-oriented systems. This request has spurred the supply of a number of improved approaches to software development, with respect to the most important object-orientation (OO). Using this metric we can evaluate software overall complexity. Also there are many metrics for measuring software run time properties and would be monetary value studying. Analyzing and getting the data can forecast design quality. If used properly, it leads to a complex reduction in costs of the complete implementation and developments in quality of the end product. By using early quality indicators based on objective experimental proof is therefore for a realistic objective. According to my opinion it motivated for the developer to get very fast and continuous feedback about the quality in design and implementation of the product they develop and thus gives a possibility to increase the quality of the product as early as possible.

III OVERVIEW - OBJECT ORIENTED METRICS

During this paper, the SATC discusses its applied research of object oriented metrics. The research was done by surveying the literature on object oriented metrics and so applying the SATC experience in traditional software metrics to pick out the article oriented metrics that support the goal of determining design and quality. additionally, we required that a metric be feasible to compute and have a transparent relationship to the thing oriented structures being measured. At this point, many object oriented metric proposed within the literature lack a theoretical basis, while others haven't yet been validated. a number of these metrics are very labor intensive to gather, or are obsessed with the implementation environment. the article oriented metrics applied by the SATC are computable, may be associated with desirable software qualities, and are within the process of being validated.The SATC’s approach to identifying a collection of object oriented metrics was to specialise in the first, critical constructs of object oriented design and to pick out metrics that apply to those areas. The suggested metrics are supported by most literature and a few object oriented tools. The metrics evaluate the article oriented concepts: methods, classes, coupling, and inheritance. The metrics specialize in internal object structure that reflects the complexity of every individual entity and on external complexity that measures the interactions among entities. The metrics measure computational complexity that affects the efficiency of an algorithm and therefore the use of machine resources, likewise as psychological complexity factors

that affect the power of a programmer to form, comprehend, modify, and maintain software. We support the utilization of three traditional metrics and present six additional metrics specifically for object oriented systems. The SATC has found that there's considerable disagreement within the field about software quality metrics for object oriented systems [2, 6]. Some researchers and professionals contend traditional metrics are inappropriate for object oriented systems. There are effective reasons for applying traditional metrics, however, if it will be done. the normal metrics are widely used, they're well understood by researchers and practitioners, and their relationships to software quality attributes are validated [2, 6, 11, 12].

Table 1 presents an outline of the metrics applied by the SATC for object oriented systems. The SATC supports the frequent use of traditional metrics, but within the structures and confines of object oriented systems. the main 3 metrics in Table 1 are samples of traditional metrics applied to the thing oriented structure of methods rather than functions or procedures. the following six metrics are definitely for object oriented systems and therefore the object oriented construct applicable is indicated.

SOURCE	METRIC	OBJECT-ORIENTED CONSTRUCT
Traditional	Cyclomatic complexity (CC)	Method
Traditional	Lines of Code (LOC)	Method
Traditional	Comment percentage (CP)	Method
NEW Object-Oriented	Weighted methods per class (WMC)	Class/Method
NEW Object-Oriented	Response for a class (RFC)	Class/Message
NEW Object-Oriented	Lack of cohesion of methods (LCOM)	Class/Cohesion
NEW Object-Oriented	Coupling between objects (CBO)	Coupling
NEW Object-Oriented	Depth of inheritance tree (DIT)	Inheritance
NEW Object-Oriented	Number of children (NOC)	Inheritance

Table 1: SATC Metrics for Object Oriented Systems

IV.OVERVIEW - OBJECT ORIENTED STRUCTURES

A short explanation of object oriented structures is stated in this segment using the pictorial representation in Fig 1 and the definitions in Table 2.

Attribute	Defines the structural properties of classes, unique within a class, generally a noun.
Class	A set of objects that share a common structure and common behavior manifested by a set of methods; the set serves as a template from which object can be instantiated (created).
Cohesion	The degree to which the methods within a class are related to one another.
Coupling	Object X is coupled to object Y if and only if X sends a message to Y.
Inheritance	A relationship among classes, wherein an object in a class acquires characteristics from one or more other classes.
Instantiation	The process of creating an instance of the object and binding or adding the specific data.
Message	A request that an object makes of another object to perform an operation.
Method	An operation upon on object, defined as part of the declaration of a class.
Object	An instantiation of some class which is able to save a state (information) and which offers a number of operations to examine or affect this state.
Operation	An action performed by or on an object, available to all instances of class, need not be unique.

Table 2: Key Object Oriented Terms for Metrics

The latest object oriented development methods have their characteristic terminology to return the latest structural concept. Refer Fig 1, an object oriented system begins by explaining a class (Class A) that hold correlated or almost same identical features and operations (some operations are methods). The classes are used as the root for objects (Object A1). A child class inherit all of the features and operations from its parent class, in inclusion to having its unique features and operations. A child class can also be a parent class for anther classes, creating other branch in the hierarchical tree structure. When an object is formed to hold the data or information, it is an representation of the class. Classes can contact by sending messages. When a message is send in the middle of two classes, the classes are combined. These important terms are defined in Table 2 .

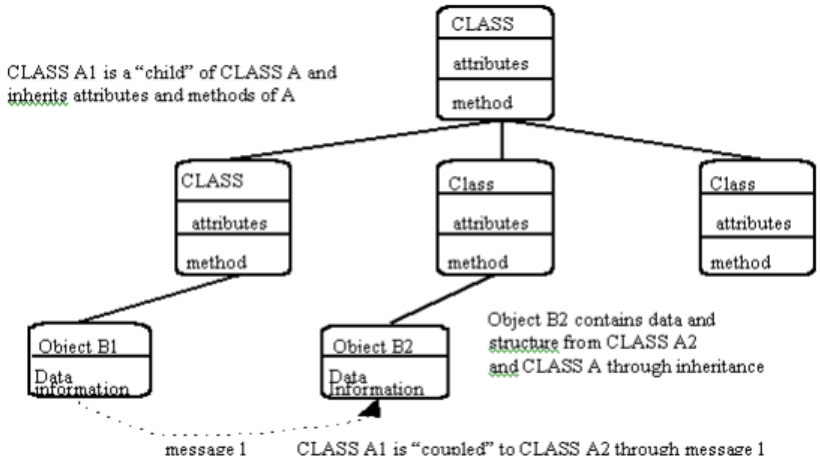


Figure 1: Pictorial Representation of Object Oriented Terms

Fig 2 is a sample application which is having 3 classes; the root or main class, Store_dept and the two child classes, Clothing and Appliances. Each department has a Manager, # Employees and Floor space; each child class inherits the features of Manager, # Employees, Floor Space from Store_dept. The Clothing class has additional features of Customer Gender, Size range and Specialty. The Appliance Departments class has the feature of Category. Definite named departments are called as objects. Objects of the Class Clothing are Toddlers Department, Men's Suits Department. In the class Appliances, the objects are sizeable Appliance Department, Small Kitchen Appliances, and Electronics Department. Methods are the operations that are performed on an object. Exemplification of what all the store departments required to make Display merchandise, Give credit, and also Exchange merchandise. The Clothing Department inherits these methods but also having Dressing rooms. The Department of Appliance will also has Delivering and Installing, Service, Part Ordering, and any other technical support.

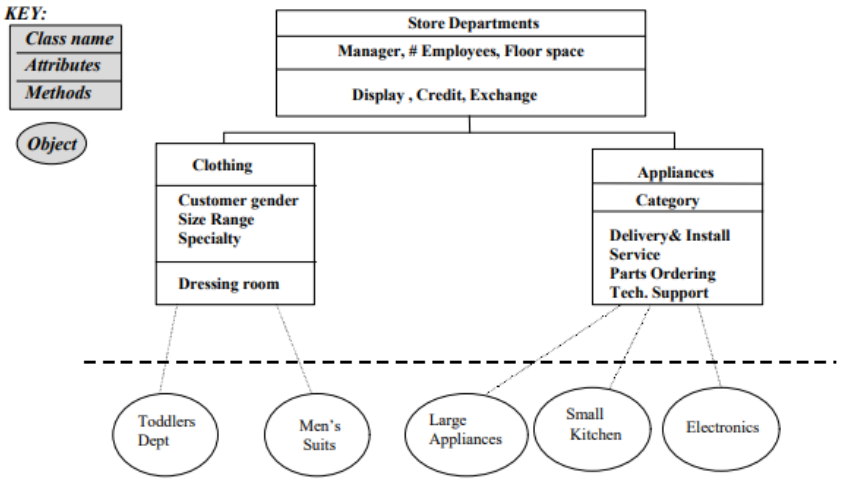


Figure 2: Example Application

V.METRICS FOR OBJECT ORIENTED SYSTEMS

A. Traditional Metrics

There are so many metrics that are used to traditional functional development. The SATC, with familiarity, has recognized three of those metrics that are appropriate for object oriented development: Complexity, Size, and Readability. To calculate the complexity, the cyclomatic complexity is utilized.

A.1 METRIC 1: Cyclomatic Complexity (CC)

Cyclomatic complexity is utilized to calculate the complexity of an algorithm in a method. It is the add up of the number of test cases

that are required to test the method successfully. The formula for calculating the cyclomatic complexity is edges – nodes + 2(edges minus nodes plus 2). For a concatenation where there is only one path, no alternative , only one test case is required. An IF loop although, has two alternatives, if the condition is true, one path is tested; if the condition is false, another path is tested. Fig 3 displays sample calculations for the cyclomatic complexity for four fundamental programming structures.

Cyclomatic Complexity

Number of Independent Test Paths => edges - nodes + 2

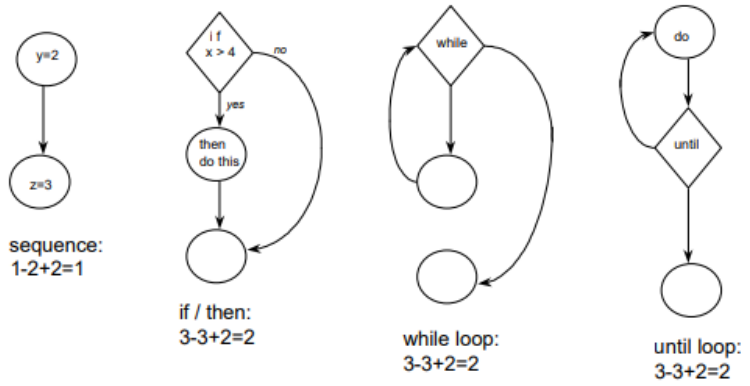


Figure 3 : Sample Calculations Cyclomatic Complexity

A procedure with a low cyclomatic complexity is usually best. This may indicate reduced testing and increases the comprehensibility or that decisions are delayed into and out of message passing, not that the method is not complex. Cyclomatic complexity cannot be used to calculate the complexity of a class for the reason that inheritance, but the cyclomatic complexity of single methods can be integrated with other measures to assess the complexity of the class. Even though this metric is specifically significant to the estimation of Complexity, it is also connected to all of the other features.

A.2 METRIC 2: Size

Size of a class is used to estimate the effortlessness of comprehension of code by organizers and maintainers. Size can be computed in different number of ways. These comprise total all physical lines of code, the number of statements, the number of blank lines, and the number of comment lines. Lines of Code(LOC) enumerate all lines. Non-comment Non-blank (NCNB) is at times mentioned to as Source Lines of Code and counts all the lines that are not comments and not blanks. Executable Statements (EXEC) are the total count of executable statements for all that of number of physical lines of code. For instance, in FORTRAN and IF statement it can be noted as:

```
IF X = 3
        Then
                Y = 10
```

Here, there are 3 LOC, 3 NCNB, and 1 EXEC.

Executable statements are less affected by programmer or language style. As a result, since NASA programs are habitually written by using numerous languages, the SATC uses executable statements to assess the project size. The main part for estimating the meaning of size measures varying based on the coding language used and the complexity of the method.

Despite that, size influences ease of interpretation by the organizers and maintainers, classes and methods of very large size will every time cause a high threat.

A.3 METRIC 3: Comment Percentage

The line counts are made to evaluate the Size metric, can be lengthen to comprise a count of the number of comments, both on-line (with code) and stand-alone. The comment percentage is estimated by the total number of comments divided by the total lines of code less the number of blank lines. Since comments assist organizers and maintainers, higher comment percentages increases comprehensibility and maintainability.

B. Object-Oriented Specific Metrics

As discussed, many changed metrics have been planned for object oriented systems. The object oriented metrics to were select by the SATC measure standard structure that, if offensively intended, negatively engage the design and code quality attribute. The certain object oriented metrics are mostly useful to the concept of classes, coupling, and legacy. former each metric, a brief description of the object oriented structure is given. For some of the object-oriented metrics discuss here, multiple definition are given; researchers and practitioners have not reached a common definition or counting methodology. In some bags, the including method for a metric is gritty by the software analysis package being used to gather the metrics. Recall, a class is a template from which objects can be created. This set of matter shares a common construction and a frequent behavior manifest by the set of methods. A method is an procedure upon an object and is patent in the class statement. A message is a demand that an object makes of another object to perform an operation. The operation execute as a result of delivery a message is called a method. Cohesion is the degree to which method inside a class are related to one a new and work jointly to give well-bounded behavior. Effective object oriented designs make the most of cohesion because cohesion promote encapsulation. Coupling is a assess of the power of group established by a connection from one entity to another. Classes (objects) are joined when a note is accepted among objects; when methods stated in one class use method or attribute of another class. Inheritance is the hierarchical connection among course that enables programmers to reuse previously defined objects with variables and operators. [2, 3, 5, 8]

Figure 2 is duplicate here as Figure 4 to use as an example application to display how these metrics would be planned.

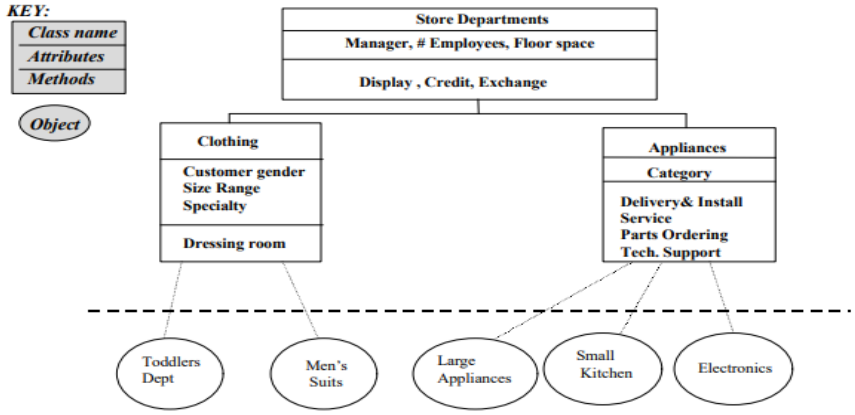


Fig 4:Object Oriented Application Example

B.1 METRIC 4: Weighted Methods per Class (WMC)

The WMC is a tally of the techniques executed inside a class or the amount of the in difficulties of the strategies (strategy intricacy is estimated by cyclomatic complexity). The subsequent estimation is hard to execute since not all strategies are assessable inside the class order because of legacy. The quantity of strategies and the difficulty of the techniques included is an indicator of how long and exertion is needed to create and keep up the class. The bigger the quantity of techniques in a class, the more noteworthy the possible effect on kids; youngsters acquire the entirety of the strategies characterized in the parent class. Classes with huge quantities of strategies are probably going to be more application explicit, restricting the chance of reuse. [2, 6, 7, 8]

Referring to Figure 4, WMC is determined by including the quantity of strategies in each class, hence:

WMC for Clothing_dept = 1

WMC for Appliance_dept = 4

B.2 METRIC 5: Response for a Class (RFC)

The RFC is the check of the arrangement of all strategies that can be conjured in reply of a message to an object of the class or by some technique in the class. This incorporates all strategies open inside the class pecking order. This measurement takes a gander at the mix of the intricacy of a class through the quantity of strategies and the measure of correspondence with different classes. The bigger the quantity of strategies that can be conjured from a class through messages, the more noteworthy the intricacy of the class. In the event that countless techniques can be conjured in Response of a Store Departments.

The testing and troubleshooting of the class gets muddled since it requires a more noteworthy level of comprehension with respect to the analyzer. A most pessimistic scenario an incentive for potential reactions will aid the suitable distribution of testing time. [2, 6, 7, 8]

The RFC for Store_dept in Figure 4 is the quantity of strategies that can be conjured in reaction to messages without help from anyone else (Store_dept), by Clothing_dept, and by Appliance_dept.

The RFC for Store_dept = 3 (self) + 1 (Clothing_dept) + 4 (Appliance_dept) = 8

B.3 Metric 6 – Lack of Cohesion (LCOM)

Lack of Cohesion (LCOM) measures the disparity of techniques in a class by case variable or qualities. An exceptionally strong module should remain solitary; high attachment demonstrates great class development. Lack of cohesion or low attachment expands intricacy, subsequently improving the probability of mistakes during the improvement cycle. High attachment suggests effortlessness and high reusability. High attachment demonstrates great class development. Absence of attachment or low cohesion builds intricacy, accordingly improving the probability of mistakes during the improvement interaction. Classes with low union could presumably be partitioned into at least two subclasses with expanded attachment. [2, 3, 6; 7, 8] Figure 5 is an elective program plan for the section in Figure 4 and valuable in showing attachment.

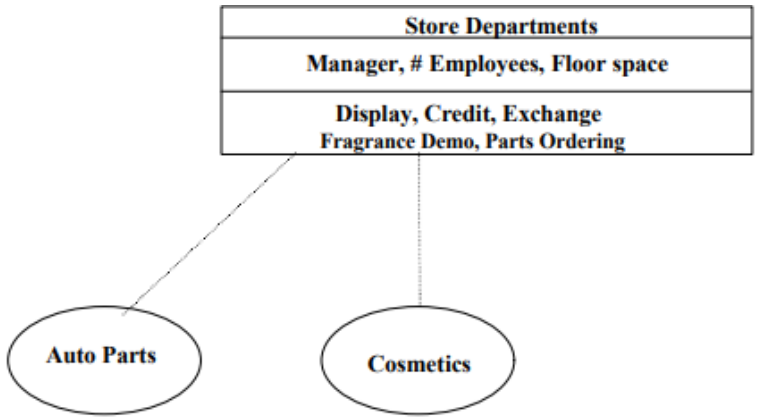


Figure 5: Alternative Design

In Figure 5, the two child classes of Clothing and Appliances have been removed, the properties and techniques joined into one class Store_dept. Figure 5 shows a plan with high absence of attachment on the grounds that there are generally not many regular credits and strategies among the objects. Auto_Parts needs the technique Parts_Ordering however not Fragrence_Demonstrations. Beautifying agents needs Fragrence_Demonstrations yet not Parts_Ordering. Since the items have not many techniques in like manner, there is a high absence of attachment. This suggests further deliberation is required – comparable articles should be gathered by making child classes for them.

B.4 METRIC 7: Coupling Between Object Classes (CBO)

Coupling Between Object Classes (CBO) is a check of the quantity of different classes to which a class is coupled. It is estimated by checking the quantity of unmistakable non-legacy related class chains of command on which a class depends. Extreme coupling is hindering to secluded Show, Credit, Exchange Store Departments Chief, # Employees, Floor space ,Vehicle Parts Cosmetics ,Scent Demo, Parts Ordering ,plan and forestalls reuse. The more free a class is, the simpler it is reuse in another application. The bigger the quantity of couples, the higher the affectability to changes in different parts of the plan and subsequently upkeep is more troublesome. Solid coupling muddles a framework since a class is more enthusiastically to get, change or right without anyone else on the off chance that it is interrelated with other classes. Intricacy can be diminished by planning frameworks with the most fragile conceivable coupling between classes. This improves particularity and advances exemplification. [2, 3, 5, 6, 7, 8]

Figure 6 is a misrepresented illustration of high coupling between objects. Here two divisions are distinguished as Jackets and Trousers. The two of them have similar traits and the same techniques. This suggests that the plan in Figure 6 is likely not the most proficient plan also, these divisions ought to be joined into one class.

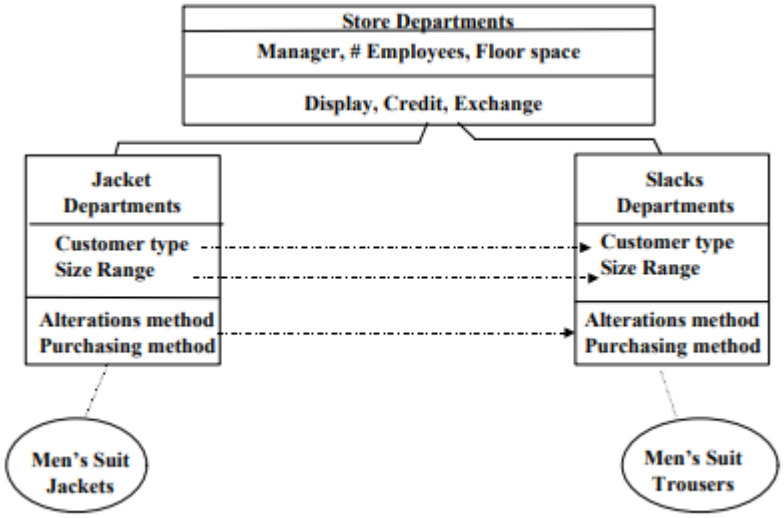


Figure 6: Example of Excessive Coupling

B.5 METRIC 8: Depth of Inheritance Tree (DIT)

The depth of a class inside the legacy order is the most extreme number of steps from the class hub to the base of the tree and is estimated by the quantity of progenitor classes. The more profound a class is inside the order, the more prominent the number strategies it is probably going to acquire making it more unpredictable to foresee its conduct. More profound trees comprise more noteworthy plan intricacy, since more strategies and classes are included, however the more noteworthy the potential for reuse of acquired strategies. A help metric for DIT is the quantity of strategies acquired (NMI). [2, 3, 6, 7, 8] .

In Figure 4, Store_Dept is the root and has a DIT of 0. The DIT for Clothing is 1.

B.6 METRIC 9: Number of Children (NOC)

The Number of children is the quantity of quick subclasses subordinate to a class in the chain of importance. It is a pointer of the potential impact a class can have on the plan and on the framework. The more prominent the quantity of youngsters, the more noteworthy the probability of ill-advised deliberation of the parent and might be an instance of abuse of subclassing. In any case, the more prominent the number Store Departments classes, the more prominent the reuse since legacy is a type of reuse. On the off chance that a class has a huge number of child classes, it might require more testing of the techniques for that class, in this way increment the testing time. [2, 6, 7, 8]

In Figure 4, Store_Dept has a NOC of 2. NOC for Clothing is 0 since it is an ending or then again leaf hub in the tree structure.

VI. INTERPRETATION GUIDELINES

While it is interesting to propose a bunch of metrics for object arranged framework, the worth of the measurements is in their application to programs – how might they assist engineers with improving the nature of the projects? While there are numerous rules with respect to how to decipher the measurements, there is inadequate factual information to demonstrate that a worth of 8 for one measurement is twice as unpredictable or on the other hand twice as "awful" as a worth of 4. The SATC accordingly, proposes translation rules based on a correlation of the qualities, taking a gander at the anomalies to decide why they are unique in relation to

different modules of code. This isn't a sign of "disagreeableness" however a pointer of contrast that should be explored. Table 3 is a synopsis of the destinations for the qualities recommended above in the description of the measurements.

METRIC	OBJECTIVE
Cyclomatic Complexity	Low
Lines of Code/Executable Statements	Low
Comment Percentage	~ 20 – 30 %
Weighted Methods per Class	Low
Response for a Class	Low
Lack of Cohesion of Methods Cohesion of Methods	Low High
Coupling Between Objects	Low
Depth of Inheritance	Low (trade-off)
Number of Children	Low (trade-off)

Table 3 : Interpretation Guidelines

Nonetheless, as demonstrated in the last two measurements, there is a compromise with large numbers of the measurements. A high Depth in Tree will expand practicality intricacy yet additionally shows expanded reuse. A high number of child will build testing endeavors yet will likewise go with expanded the degree of reuse effectiveness. A designer should know about the connections of the constructions also, that adjusting the size of one measurement can affect regions like testing, understandability, practicality, advancement exertion and reuse. [10]

VII. APPLICATION

For a some of the measurements, a straightforward histogram shows overarching and outrageous qualities. For a portion of the measurements, a straightforward histogram exhibits overarching and outrageous qualities. For the this undertaking in Figure 7, a histogram of Weighted Methods per Class (WMC) uncovers that, while most classes have a WMC of under 20, there are a couple of classes with WMC more prominent than 100. Those couple of classes with the most noteworthy WMC are possibility for review as well as modification.

This histogram is likewise helpful for observing intricacy after some time. the this task in Figure 7, a histogram of Weighted Methods per Class (WMC) uncovers that,

while most classes have a WMC of under 20, there are a couple of classes with WMC more noteworthy than 100. Those couple of classes with the most noteworthy WMC are possibility for examination and additionally update.

This histogram is additionally valuable for checking intricacy after some time.

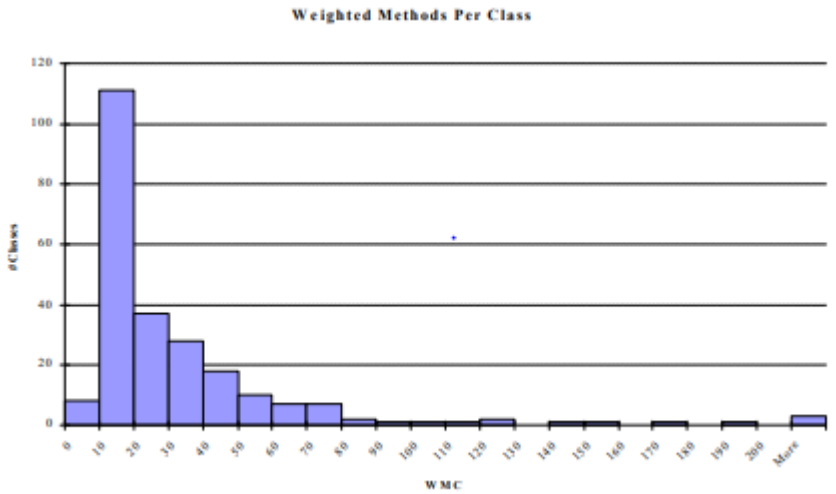


Figure 7: Weighted Methods Per Class

There are a couple of classes in the task appeared in Figure 8 that are equipped for conjuring in excess of 200 strategies. Classes with huge RFC have a more prominent complexity and diminished understandability. Testing and investigating are more convoluted. This data is helpful when observed after some time moreover.

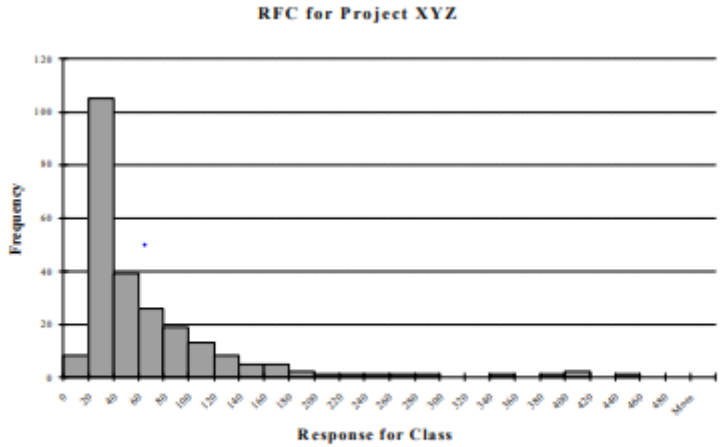


Figure 8: Response for a Class

Figure 9 analyzes the Coupling Between Objects (CBO). Of the 240 classes in this project, in excess of a third are independent. Higher CBO shows classes that might be hard to see, more outlandish for reuse and more hard to keep up.

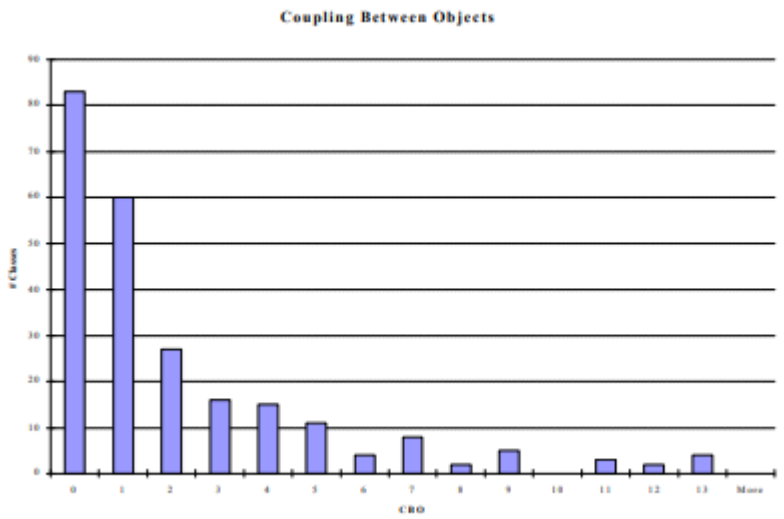


Figure 9: Coupling Between Objects

The measurements for the progressive design, Number of Children (NOC) and Depth in Tree (DIT) can likewise be graphically portrayed as demonstrated in Figure 10. A class with DIT = 0 is the "root" of a progressive system. Assuming it is likewise a "leaf", NOC = 0, it is independent code that doesn't profit from legacy or reuse. Practically 66% of this current task's classes are beneath different classes in the tree, which demonstrates a moderate degree of reuse. Higher rates for DIT's of 2 and 3 would show a more serious level of reuse, however expanded complexity .

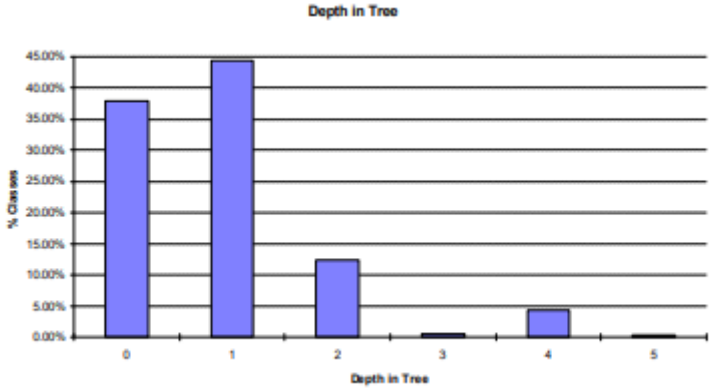
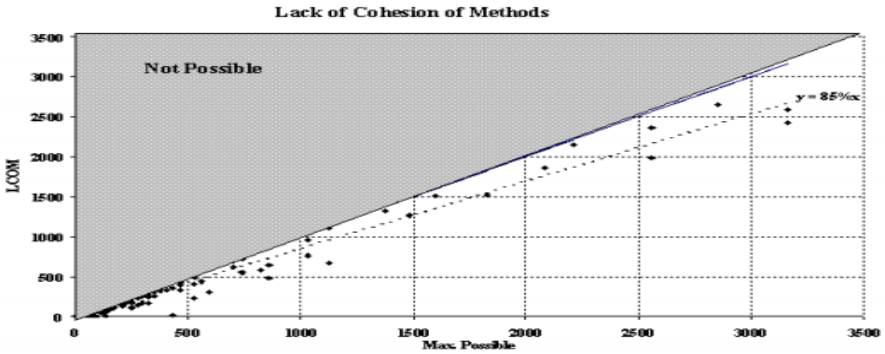
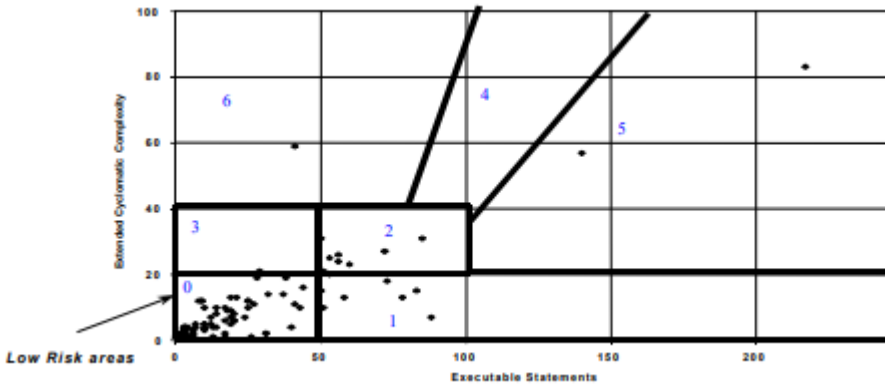


Figure 10: Depth in Tree (DIT)

The rate of Lack of Cohesion of Methods (LCOM) depends on the amount of method, so around is a most value likely. Figure 11 is a plot of careful LCOM compare to possible maximum. While to hand is small skill with the LCOM metric, insight says that the smaller real LCOM is compared to its likely maximum, the better. In Figure 11 we seem combination among Classes at LCOM to see the values nearby to the line. The SATC also uses the trend line exposed in the chart to make comparison among project and between language.



For large numbers of the metrics, it is more powerful to dissect the modules utilizing two measurements. In Figure 12 the techniques are plotted dependent on size and difficulty.. The SATC has done broad applied exploration to distinguish the favored qualities. The "hazard locales" shown demonstrate where strategies have the potential for low quality that will impact viability, reusability and meaningfulness. (These districts of hazard were created for non article arranged code and are normal to diminish in size with additional exploration.) The table underneath the chart sums up the outline.



	Risk 0	Risk 1	Risk 2	Risk 3	Risk 4	Risk 5	Risk 6	Total
Count	85	7	9	1	0	2	1	105
Percent	81.0%	6.7%	8.6%	1.0%	0.0%	1.9%	1.0%	100.0%

Higher risk due to size and complexity

Figure 12: Size to Complexity Comparison

In Figure 13, Response for a Class is plotted against the number of methods. Focuses on or close the "conceivable" line address classes that don't invoke outside techniques. This demonstrates to engineers that there are a few classes with in excess of 40 techniques that additionally influence numerous articles in different classes. These are prime possibility for walk-throughs and testing.

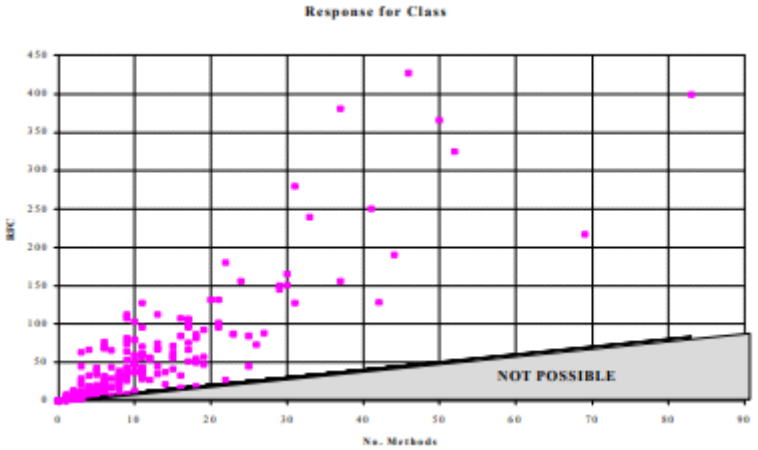


Figure 13: Number of Methods by Response for class

As examined, there is a compromise while deciding the suitable number of child classes furthermore, the profundity of the tree. Higher DIT's show a compromise between expanded intricacy and expanded reuse. Higher NOC's likewise show reuse, yet may require really testing. Figure 14 exhibits how the two-route perspective on the information recognizes a fascinating class – one that is three ventures down structure the root and has 40 children.

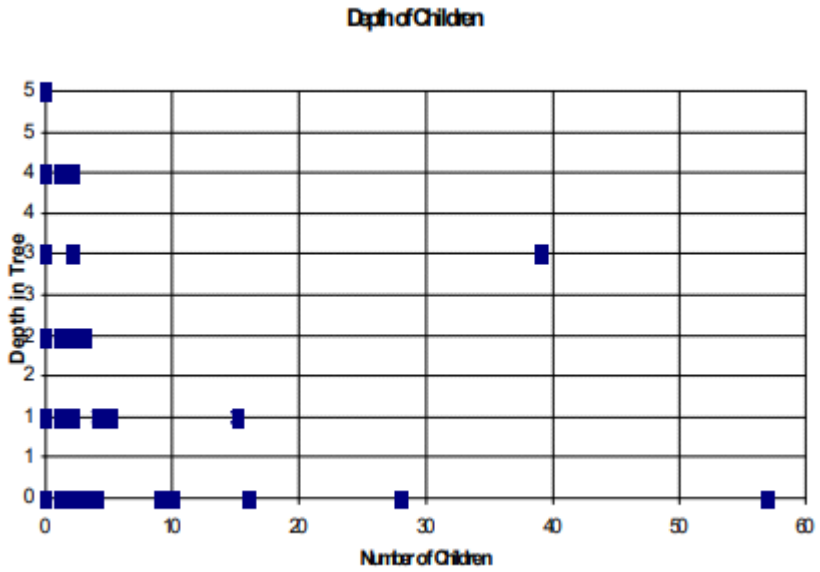


Figure 14: Hierarchical Evaluation

VII. SUMMARY

Object oriented metrics help assess the growth and testing pains needed, the understandability, maintainability and reusability. This in order is summarized in Table 4.

Metrics	Objective	Testing Efforts	Understan -dability	Maintain -ability	Develop Effort	Reuse
Complexity	↓	↓	↑	↑		
Size (LOC)	↓	↓	↑	↑		
Comment %	↑	↓	↑	↑	↓	
WMC	↓			↑	↓	↑
RFC	↓	↓				↑
LCOM	↓		↑	↑	↓	↑
CBO	↓	↓	↑	↑		↑

Table 4 : Object Oriented Metrics Effects

IX.CONCLUSION

Object oriented metrics exist and do give significant data to protest arranged developers and project managers. The SATC has tracked down that a blend of "traditional" measurements and measurements that action structures remarkable to protest situated advancement is most Powerful. This permits engineers to keep on applying measurements that they know about, for example, complexity and lines of code to another improvement climate. In any case, presently that new ideas and designs are being applied, such legacy, coupling, attachment, techniques and classes, measurements are expected to assess the adequacy of their application. Measurements, for example, Weighted Methods per Class, Response for a Class, and Lack of Cohesion are applied to these regions. The use of a progressive design likewise should be assessed through measurements such as Depth in Tree and Number of Children. Right now there are no unmistakable translation rules for these measurements despite the fact that there are rules dependent on good judgment and experience.

9. REFERENCES

1. Booch, Grady, Object Oriented Analysis and Design with Applications, The Benjamin/Cummings Publishing Company, Inc., 1994.

2. Chidamber, Shyam and Kemerer, Chris, “A Metrics Suite for Object Oriented Design”, IEEE Transactions on Software Engineering, June, 1994, pp. 476-492.

3. Hudli, R., Hoskins, C., Hudli, A., “Software Metrics for Object Oriented Designs”, IEEE, 1994.

4. Jacobson, Ivar, Object Oriented Software Engineering, A Use Case Driven Approach, Addison-Wesley Publishing Company, 1993.

5. Lee, Y., Liang, B., Wang, F., “Some Complexity Metrics for Object Oriented Programs Based on Information Flow”, Proceedings: CompEuro, March, 1993, pp. 302-310. Metrics Objective Testing Efforts Understan -dability Maintain -ability Develop Effort Reuse Complexity ¯ ¯ Size (LOC) ¯ ¯ Comment % ¯ ¯ WMC ¯ ¯ RFC ¯ ¯ LCOM ¯ ¯ CBO ¯ ¯

6. Lorenz, Mark and Kidd, Jeff, Object Oriented Software Metrics, Prentice Hall Publishing, 1994.

7. McCabe & Associates, McCabe Object Oriented Tool User’s Instructions, 1994.

8. Rosenberg, Linda H., “Metrics for Object Oriented Environments”, EFAITP/AIE Third Annual Software Metrics Conference, December, 97.

9. Sommerville, Ian, Software Engineering, Addison-Wesley Publishing Company, 1992.

10. Sharble, Robert, and Cohen, Samuel, “The Object Oriented Brewery: A Comparison of Two object oriented Development Methods”, Software Engineering Notes, Vol 18, No 2., April 1993, pp 60 -73.

11. Tegarden, D., Sheetz, S., Monarchi, D., “Effectiveness of Traditional Software Metrics for Object Oriented Systems”, Proceedings: 25th Hawaii International Conference on System Sciences, January, 1992, pp. 359-368.

12. Williams, John D., “Metrics for Object Oriented Projects”, Proceedings: ObjectExpoEuro Conference, July, 1993, pp. 13-18.