Security Metric for Object Oriented Class Design-Result Analysis

Soham H. Gandhi, D. R. Anekar, Mahevash A. Shaikh, Ajinkya A. Salunkhe

Abstract--It is difficult to detect vulnerabilities in the operational stage of software, because the security concern are not addressed or known sufficiently early during software development. Accessibility (data encapsulation) and interaction (cohesion) related software metrics can be measured during the earlier phases of software development. The most importance of software measurement has led to the development of new software measure. To satisfy security requirement, it is important to protect data from unauthorized disclosure of information and alteration of information. Taking security early phase of a system development should have an impact on reducing many software vulnerabilities. A new methodology has been proposed in this paper to check accessibility and interaction of class design. These metrics allow designer of system to discover and fix the security of various alternative of class designs. We also mention the analysis of these metrics. These observations show that security design metrics can be used as early indicators of vulnerability in software.

Keywords: Class diagram, Software measurement, Vulnerability, Security Metrics, Data encapsulation, Cohesion, Model file parser.

I. INTRODUCTION

The software industry having lack of standard metrics and measurement. Software security is a major activity in the software industry. It has been reported that cost and effort spent on software security is very high, approximately between 65% to 70% of total software development and support efforts [1]. Software reengineering, recently, have been advocated as a means of reducing security costs [9]. Slightly short of software metric has multiple definition and ambiguous rules.

It is difficult to detect vulnerabilities in the operational stage of software, because the security concern are not addressed or known sufficiently early during software development. Accessibility (data encapsulation: the mechanism that binds together the code and the data it manipulates, and keeps both safe from outside interference and misuse) and interaction (cohesion: a measure of how strongly related or focused the responsibilities of a single module are) i.e. related software metrics can be measured during the earlier phases of software development. To satisfy security requirement, it is important to protect data from unauthorized disclosure of information and alteration of information.

Manuscript received on May, 2013.

Soham H. Gandhi: is currently pursuing bachelor degree program in Information Technology in Sinhgad Academy of Engineering, University of Pune, India.

D.R.Anekar: is currently assistance professor at Sinhgad Academy of Engineering, Pune, India.

Mahevash A. Shaikh: is currently pursuing bachelor degree program in Information Technology in Sinhgad Academy of Engineering, University of Pune, India.

Ajinkya A. Salunkhe: is currently pursuing bachelor degree program in Information Technology in Sinhgad Academy of Engineering, University of Pune, India.

Taking security early phase of a system development should have an impact on reducing many software vulnerabilities. Software vulnerability is an instance of a [fault] in the specification, development, or configuration of software such that its execution can violate an [implicit or explicit] security policy

[7].Software security is the ability to defend attacker's exploitation of software problems by building software to be secure throughout the whole development life cycle [4].

Object oriented class design is becoming more famous in software development and object oriented design metrics is an important part of software development environment.

This study is focuses on a set of object oriented security design metrics that can be used to measure the security and quality of an object oriented class design. The metric for object oriented class design focus on measurement that are applied to object oriented class and design characteristics. These measurement permits designers to access the software design in early phase of development, making changes that will reduce the complexity and improve the continuing capability of the design. The object oriented model closely represents the problem domain, which makes it easier to produce and understand design. It is also believed that object oriented design will encourage more re-use, i.e., new application can use existing modules more efficiently and effectively, thereby, reducing development cost and time [6].Security measurements have been defined to assess security at the level of implementation code [3]. This paper proposes a new set of metrics which are capable of assessing the security quality of OO class designs. In our case we use <<secrecy>> stereotype to identify confidential data. Once the metric's results are identified for alternative class diagrams, it is easy to find the most secure one to implement. This paper shows the study of security design metrics on the basis of testing of different classes. These metrics allow designer of system to discover and fix the security of various alternative of class designs. We also mention the analysis of these metrics. These observations show that security design metrics can be used as early indicators of vulnerability in software.

In the next session we provide a general background about metrics and software measures and existing software measurement metrics. Section 3 describes the design and implementation. Further sections describe the study of proposed architecture. Further Sections describes class design and metric results and metrics analysis.

1.1 Purpose of Software Metrics

The use of software metrics is for developing quality software. Software metrics measures of some piece of property of software, or software specifications.

Software metrics are measures of the attributes of the software products and processes.

Software metrics are measures that could be used to measure different characteristics of a software system.

1.2 Where The OO Metrics Applied?

If we are using metrics without OO concepts, then they involve only data flouring from process-to-process. But, if we build the metrics with interest OO concept, you will end up with an metric whose focus is centered around a set of classes and the patterns of interaction that direct how those classes work together[2].

II. BACKGROUND

Many different kinds of metrics have been developed during the past few decades, matching with the different programming paradigms like structural programming and object-oriented programming (OOP). Software metrics can be used to find out the properties of the software that we are developing and predict the needed effort and development period. "LOC (Lines of Code)" is one of the most primitive and oldest metrics. In the beginning of 1990s, Chidamber and Kemerer proposed six new object-oriented metrics to overcome the limitations of the more traditional code-based metrics. However, as software engineers' focus has shifted to the earlier stages of the life cycle, the shortcomings of OO code metrics like their predecessors have become more apparent. Therefore, a comprehensive approach to developing and applying metrics to artifacts such as designs produced at the early stages of the life cycle is needed. In the meantime, the Unified Modeling Language (UML) was adopted by the Object Management Group (OMG) in 1997 ending the so-called "OO methods war", and since then has become the de facto specification standard graphical language for specifying, constructing, visualizing, and documenting software systems, business modeling and other non-software systems. One of the earliest studies in this area was the development of software security design principles by Saltzer and Schroeder [10]. These principles were intended as guidance to help develop secure systems, mainly operating systems. Bishop's [11].Many organizations are using UML as a common language for their project artifacts and have adopted UML as their organization's standard. As the amount of UML models produced within an organization increased, a need for measuring their characteristics has arisen.

The overall aim is the developments of software metrics that can be applied to UML models. These metrics are comparable to UML itself in such a way that it plays a role as a standardized metrics suite. This study focused on the system's 'attack surface'. Similarly, a study that defined design metrics which measure certain software quality attributes was conducted by Bansiya [13].

III. DESIGN AND IMPLEMENTATION

a. Metric Experimental Test

The following experimental test describes how security design metrics are used. They can be applied on the UML class design diagram or manually entered object oriented class. In class diagram should include the UMLsec and SPARK's annotation in addition to the standard element of a UML class diagram.

The experimental tests consist of class diagram with several alternatives. The class diagram contains the some Mutator, Assessors, constructors and attributes with public, private and protected access specifier. MDL Extractor [5] i.e. The Model file parser parses the model file into tokens. Tokens are nothing but individual words and punctuation marks. The Model file parser identifies the class diagram from the model file to apply security metrics

The case study consists of alternative class design and result of software metrics. In this paper, we study the two major type cohesion and encapsulation. Cohesion is a measure of how strongly-related or focused the responsibilities of a single module are. If the methods that serve a given class tend to be similar in many aspects, then the class is said to have high cohesion. Cohesion metrics are-Classified Mutator Attribute Interactions (CMAI), Classified Accessor Attribute Interactions (CAAI), Classified Attributes Interaction Weight (CAIW) and Classified Methods Weight (CMW). Encapsulation is the mechanism that binds together the code and the data it manipulates, and keeps both safe from outside interference and misuse [5]. E.g. When a user selects a command from a menu in an application, the code used to perform the actions of that command is hidden from the user. Encapsulation metrics are- Classified Instance Data Accessibility (CIDA), Classified Class Data Accessibility (CCDA) and Classified Operation Accessibility (COA).

System take as input from the user (developer) a model file extension with (*.mdl), that is, a rational rose file which contains class diagram. Using the data library which is the input and output system we interact with the model file. In short, the data library loads or unloads the model file. After loading the model file, the Model File Parser tokenizes the words and finds out the class diagram's attributes and operations. After that, Design metrics will be applied on the loaded parsed model file, which will calculate the metrics and display them in a table. After this, a graph will be plotted which can be of any type, in our case we use Radar Graph. Finally, we can generate code for the class diagram that is most secure [5].

IV. UML CLASS DIAGRAMS

In our experimental test, we used UML class diagram as per UMLsec and SPARK's annotation standard. These diagrams can be input to the system [5] by passing (*.mdl) Model file or manually entered. These diagrams have drawn in Rational Rose Enterprise Edition. In our experimental test, we are using six alternative class designs. We made two maintain class design profiles; these two class design profiles contain three class designs. We take the example of class design of employee management system. First class design profile contains EmployeePerson_1, Employee Person_2 and Employee Person_3. Second class design profile contains Employee Person_4, Employee Person_5, and Employee Person_6 class design.

In this, one class design made in various alternatives, Alternative class diagrams varies in access specifier and "<<secrecy>> "stereotype. EmployeePerson class attributes contains name, address, empid, empsal. EmployeePerson class operation or method responsible for getting or setting the attribute value. Details of EmployeePerson empid and empsal kept secret. For the first profile, Figure 1 has declared all attribute private as well as all operation declared as public.

Figure 2 has declared all attribute private where as in operation except setDetatils and getDetails declared private. Figure 3 has all private attribute and only two public operations. For the second profile, Figure 4 has public attribute and all public operations. Figure 5 has private attribute and all public operations. Figure 6 has all public attribute and operations. The difference in following all class designs is that, all are internal structure differently; it can be seen by these cohesion and data encapsulation metrics.

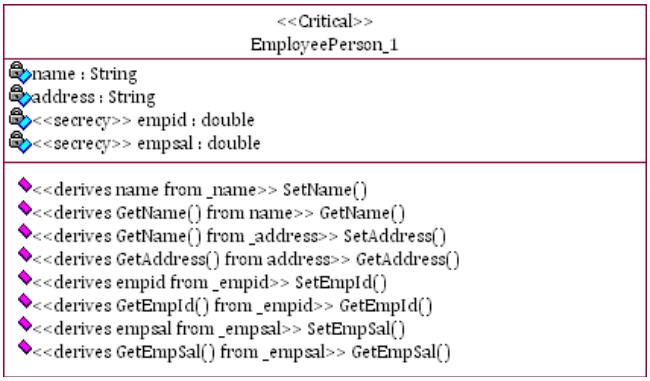

Figure 1: EmployeePerson_1 Class Design

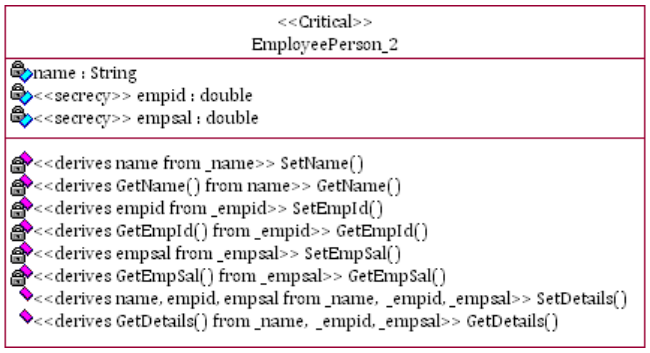

Figure 2: EmployeePerson_2 Class Design

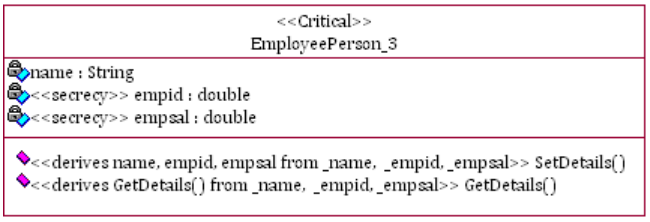

Figure 3: EmployeePerson_3 Class Design

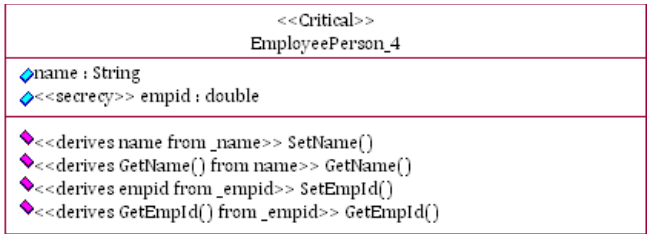

Figure 4: EnployeePerson_4 Class Design

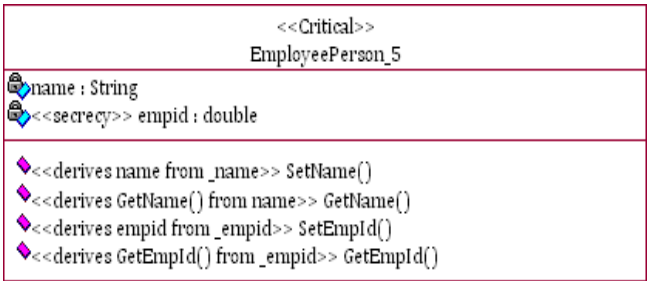

Figure 5: EmployeePerson_5 Class Design

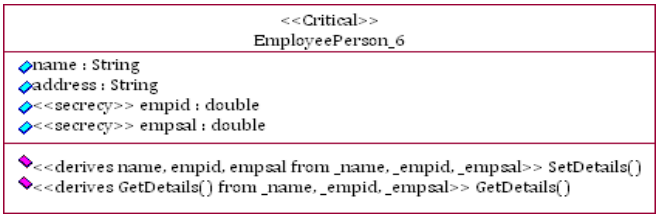

Figure 6: EmployeePerson_6 Class Design

V. SECURITY DESIGN METRICS SYSTEM

This paper is the experimental study analysis of our previous paper "Finding Accessibility and Interaction vulnerability of Rational Rose Class Design Using Design Metrics" [5]. We design the system according to proposed the system architecture mentioned in our previous paper [5]. Following screen shots describes the actual implemented system.

i. Main Menu

System having some menus shown in Figure 1. System having two ways input one is "Add Classes Manually" and second is "Import From UML".

The input and output performed by "Load CLASS LIST", "Save CLASS LIST As", "Clear CLASS LIST Profile". Remaining menus relates with the Metrics, Graph and Code generation with compare metrics and average compare metrics.

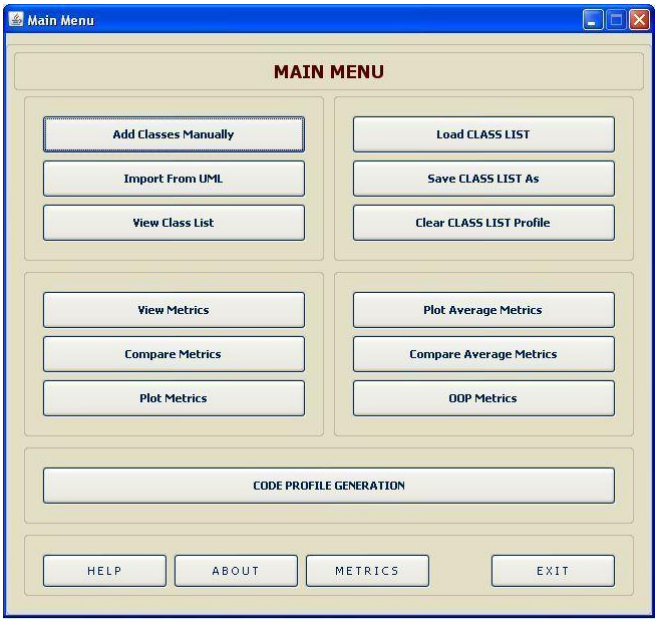

Figure 1: System Main Menu

ii. Manual Class Entry

This module will be provided where in user will be able to add classes to profile manually. User can enter class name, attribute name, attribute stereotype, attribute type, and attribute access specifier, Operation name, operation parameters and their return type, operation return type and access specifier.

All information can be store in class profile. In future one can load or unload the class information.

Figure 2 shown the how to put input to the System?

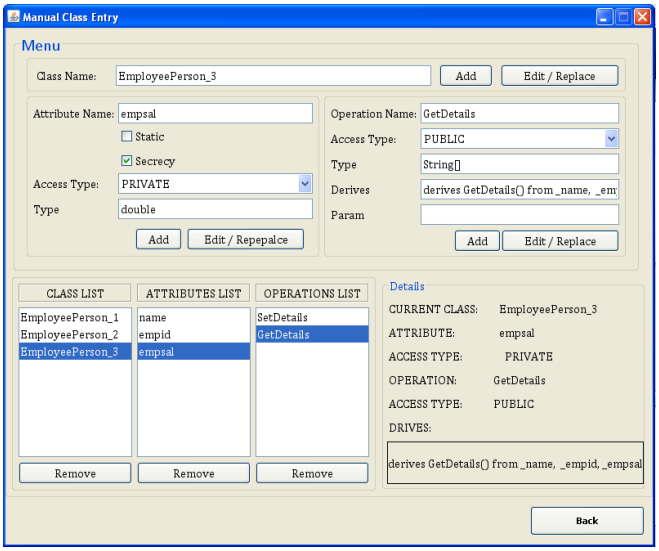

Figure 2: Manual Class Entry

iii. Import Class From Uml

This module that will tokenized a UML diagram was designed using Rational Rose software. The Module file tokenization will basically be a parser that will extract all required information from a model file and add all acquired classes information to the current profile.

Figure 3 shown the input taken from (*.mdl) model file then extracted information placed in respective list boxes. The current extracted information can be added to class list which will use in further operations.

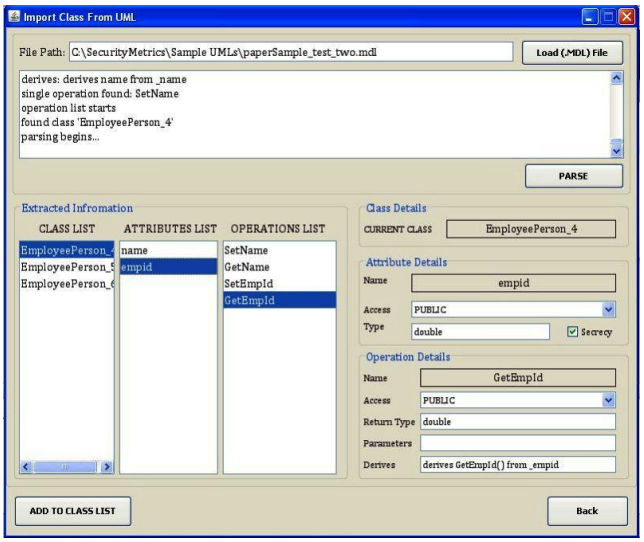

Figure 3: Import Class From UML

iv. View Security Metrics

Following figure 4 shows the calculated metrics value.This module will calculate security metrics values for the current profile and display it in lists boxes. Bandar Alshammari [12] states some seven security matrices. The metrics have been scaled to all fit with the range 0 to 1. A low value is desired for each. These metrics at this stage are concerned with the properties of individual object- oriented classes.

3.3.4.1 Accessibility Metrics

These Metrics used for measuring the accessibility level of attributes and operations or methods in a particular class from access modifiers perspective, like public denoted as '+', private denoted as '-', protected denoted as '#'.

This category is divided into three kinds of accessibility metrics-CIDA, CCDA, and COA.

3.3.4.2 Interaction Metrics

These metrics used to measure the impact of class interaction between operations and attribute on the security of that class.

This category divided into four kinds of interactions, like classified setters or constructors, getters and unclassified methods.

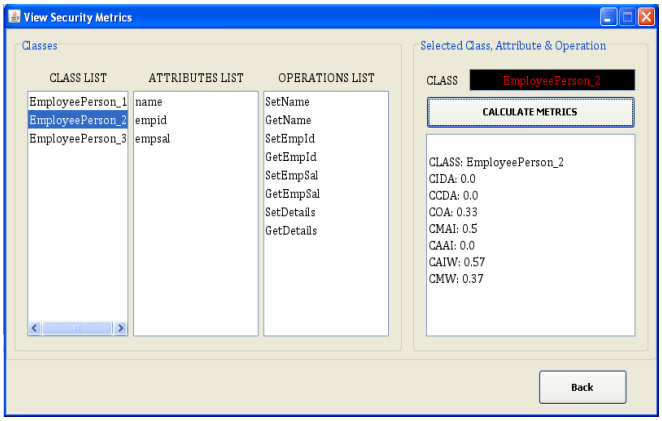

Figure 4: View Security Metrics

v. Plot Metric / Compare Metrics

This module will read values from the from current profile and display it in graphs (Radar or Web Graphs). We used Radar graph because lines closer to the center mean that a specific class diagram is more secure than the lines away from the center.

It easily concludes the result, meaning which diagram is more secure among the alternatives. The Radar graphs have been scaled to fit the range 0 to 1. Lower value indicates more secure class design and higher value indicates less secure class design.

We can plot one graph for comparison of one or more class designs, or different graphs. Following figure 4 shows the calculated metrics value. We use the different colors to identify respective class design.

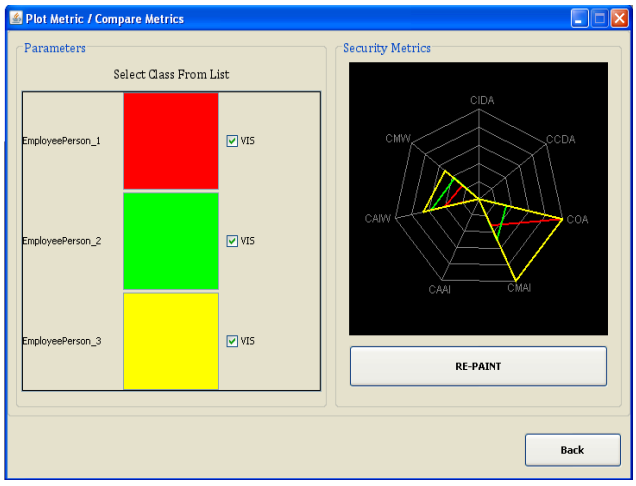

Figure 5: Plot Metric / Compare Metrics

b. EXPERIMENTAL TEST RESULTS AND ANALYSIS

i. Test Results

In section 3.4 described the input for the experimental test to security metrics system. These designs shown in various alternatives.

Table 1 shows the result analysis of class designed described in section 3.4. For easy to understand compare these result analysis. We also have shown the output of system in radar graph.We used two class profiles one profile contains the three class designs and second profile contain another three class designs.

After getting values, we found that EmployeePerson_4 and EmployeePerson_6 are most insecure design. EmployeePerson_3 and EmployeePerson_5 result values more than EmployeePerson_2, hence EmployeePerson_2 is most secure design among these six class design.

Table 2 shows the average security metrics results as well as it shows the compare average security metrics.

Observing table 2, we conclude that average metrics values of second profile containing EmployeePerson_4, EmployeePerson_5and EmployeePerson_6 class designs has more than first profile containing EmployeePerson_1, EmployeePerson_2,and EmployeePerson_3 class designs.

TABLE 1: Security Metrics Result

	CIDA	CCDA	COA	CMAI	CAAI	CAIW	CMW
EnployeePerson_1	0	0	1	0.33	0	0.4	0.25
EnployeePerson_2	0	0	0.33	0.5	0	0.57	0.37
EnployeePerson_3	0	0	1	1	0	0.66	0.5
EnployeePerson_4	1	0	1	0.5	0	0.33	0.25
EnployeePerson_5	0	0	1	0.5	0	0.33	0.25
EnployeePerson_6	1	0	1	1	0	0.66	0.5

TABLE 2: Average Metrics Result

	CIDA	CCDA	COA	CMAI	CAAI	CAIW	CMW
EnployeePerson_1							
EnployeePerson_2							
EnployeePerson_3	0	0	0.77	0.61	0	0.54	0.37
EnployeePerson_4							
EnployeePerson_5							
EnployeePerson_6	0.66	0	1	0.66	0	0.44	0.33

ii. Test Analysis

The simplest way of comparing class design is look at the radar graph of their design to identify which one design is most secure. The design with the lowest values is most secure.

In terms of accessibility, the class designs of EmployeePerson 4 and 6 declared all of their classified methods as public which result in most insecure design. EmployeePerson 2 is most secure design, because it has lowest value in all other class designs.

We conclude that the metrics on the top of the chats: CMW, CIDA, CCDA, and COA are the ones which mostly contribute to the reducing the attack surface. In other side, metrics CMAI, CAAI and CAIW are the mostly associated with the principal of least privilege.

VI. CONCLUSION

In our experimental test, we conclude that, these security designs metrics easy to calculated and easy to apply using UMLsec and SPARK's annotations. The system described in this paper can prevent all this and also improve efficiency-the software developer need not worry about which of his class diagrams is most secure; the system will assess that for him. These metrics allow system designer to find and fix security vulnerabilities at an early phase of software development, by comparing the security of various class diagrams. Thus, the programmer can focus more on the other quality aspects of coding, while still ensuring security.

Future work will be a some metrics that are used to reducing the complexity of design as well as code implementation. New software metrics will combine the future of this paper and new metrics.

ACKNOWLEDGMENT

We would also like to thank to our friends for listening to our ideas, asking questions and providing feedback and suggestions for improving our ideas last but not the least,we would like to thank Mrs. Devata R. Anekar for their valuable guidance and our Institution and other faculty members, without whom this Paper would have been a distant reality.

REFERENCE

1. Krishan K Aggarwal, Yogesh Singh, Jitender Kumar Chhabra. An Integrated Measure of SoftwareManitainability. Proceedings Annual Reliability and Security Symposium. 2002; 235-241.
2. Steve Counsell, Stephen Swift, Jason Crampton "The interpretation and Utility of Three Cohesion Metrics for Object – Oriented Design" (ACM Transactions on SE & Methodology, Vol. 15, No. 2, April 2006)
3. I. Chowdhury, B. Chan, and M. Zulkernine, "Security metrics for source code structures," in Proceedings of the Fourth International Workshop on Software Engineering for Secure Systems Leipzig, Germany: ACM, 2008.
4. Hoglund, G. and McGraw, G., Exploiting Software: How to Break Code. Boston: Addison-Wesley, 2004.
5. Finding Accessibility and Interaction vulnerability of Rational Rose Class Design Using Design Metrics Soham H. Gandhi, D. R. Anekar, Mahevash A. Shaikh, Ajinkya A. Salunkhe
6. Pankaj Jalote. An Integrated to Software Engineering. 2nd Edition. Narosa Publication Home. 2002.
7. Krsul, I. V., Software Vulnerability Analysis, PhD Thesis, West Lafayette, Purdue University, 1998.
8. M HarrySneed, Agnes Kapose. A Study on the Effect of Reengineering Upon Software Security.IEEE Press. 1990; 91-99.

9. J. H. Saltzer and M. D. Schroeder, "The protection of information in operating systems," in Proceedings of the IEEE, 1975, pp. 1278-1308.
10. M. Bishop, Computer Security: Art and Science. Boston: Addison-Wesley, 2003.
11. Alshammari, Bandar and Fidge, Colin J. and Corney, Diane (2009) "Security metrics for object-oriented class designs". In: QSIC 2009 Proceedings of: Ninth International Conference on Quality Software, August 24-25, 2009, Jeju, Korea. (In Press).
12. J. Bansiya, "A Hierarchical Model for Quality Assessment of Object-Oriented Designs," Ph.D. Thesis, University of Alabama in Huntsville, 1997.