Validation of Software Architectural Tool for Object-Orinted Testing using with the Facilitate Quality Attributes

Lalji Prasad
Truba College of Engineering and Technology,
Department of Computer science and
Engineering (RGTU), Indore, INDIA

Sarita Singh Bhadauria
Madhav Institute of Technology and Science,
Department of Electronics Engineering (RGTU),
Gwalior, INDIA

ABSTRACT

In this research investigate, quality of software using comprehend our architecture testing model [34], with the help of object oriented characteristic relationship, using different software metrics. The objective of 'Design Architectural Testing Tool' is to facilitate a design that may contribute to the comprehensiveness of the software testing tool. In this research work first we try to draw an architecture of testing method based on their attribute nature and shows their relationship next phase will be applied testing (based on different software metrics) on each component and after testing we apply different statistical analysis for validation of our research work .

General Terms

Software Testing, Software Architecture.

Keywords

Comprehensiveness, Architectural Completeness, Architectural Quality Attribute, Architectural Metrics

1. INTRODUCTION

Different researcher work on quality of software architecture and testing for ensuring the quality of software, here discuss only prominence few literature. Bass and et al. , articulated importance of software architecture [12] .Soni and et al. " Say , Software architectures describe how a system is decomposed into components, how these components are interconnected, and how they communicate and interact with each other's" [14]. Perry and et al. Work on Software architecture is concerned with the study of the structure of software, including its topology, properties, constituent components and their relationships and patterns of combination [21]. Gary Chastek and et al., enlighten software architectural attributes and quality- related [1]. Huang and et al., describe the basic rules for program testing, which provide basic principle for testing [3,10,14,15,16,17]. Poston [26], Williams [27], and Hareton [19] shows, Integration all the data across tools and repositories, Integration of control across the tools and Integration to provide a single graphical interface into the test tool set. Limitation: emphasize only integration tool (usability & portability). Rosenberg [4] provides, the approach to software metric for object oriented based different from the standard metric sets. Some metrics, such as, line of code & cyclomatic complexity, have become accepted as standard for traditional functional / procedural programs, but for an object oriented scenario, there are many proposed object oriented metrics in the literature. Limitation: this provides the only conceptual framework for measurement .Agrawal and et al. [25] cited in this paper the importance of software measurement is increasing leading to the development of new measurement techniques. Limitation: a) It does not provide any relationship between requirement & testing attribute. b) It cannot evaluate for large data sets. Anderson and et al. [5] Emphasized the software industry has performed a significant amount of research on improving software quality using software tools & metrics will improve the software quality and reduce the overall development time. Good quality code will also be easier to write, understand, maintain and upgrade. Limitation a) it does not provide any relationship between the required testing attribute. b) Its not provide a full featured testing tool (only Complexity & cohesion measure). c) It provides the only conceptual framework for measurement. Briand and some other researchers [9,11,28,29,30,31] demonstrate aims is that empirically the relationships between most of the existing coupling & Cohesion measures for object oriented (OO) system & fault proneness of object oriented system classes can be studied. Limitation: a) Only emphasis on cohesion & coupling metric. Bitman [6] exhibit key problem in software development of changing software- development complexity and the method to reduce complexity. Limitation: a) It does provide only complexity measurement techniques. Krauskopf &Juan [32] and Harrison [8] demonstrate, Coupling is the degree of interdependence between two modules. In a good design, they are kept low. Coupling should be lower in large and complex system. No coupling is highly is desirable but practically it is not possible. The good & bad points of different types of coupling are discussed. Limitation: a) Only emphasis on cohesion & coupling metrics. Chidambaram [8] and Harrison [7] emphasized the coupling between object (CBO) metric and evaluated for five object oriented systems & compared with alternative design metric called NAS which measure the number of associations between class & its peers (Harrison R.S). NAS metric is directly collectible from design documents such as the object model. Limitation: a) it's not providing any relationship between requirement & testing attribute. b) They don't provide some basic idea for size & effort estimation. c) Measuring complexity of a class is subject to bias. Reiner R., Dumke and Achim S., Show How to manage component based software and identify related metrics. [18]

Comprehensive means that it includes all or nearly all features (maintainability, reusability, flexibility and portability) and relationships required for migrating from one testing class to another. It is designed to overcome the limitation of existing software tools by providing a final class level architecture having relationships between various testing classes. Software quality is another focus of our architecture. We wish to achieve good maintainability, reusability, flexibility and portability in the architecture of the software testing tool by validating the architecture using testing algorithms and performing metrics calculation on each relationship existing between the different testing techniques [1, 2, 3].

2. RESEARCH METHODOLOGY

- First establish a requirement specification using formal review specification. Requirement gathering from different literature (research papers, books and technical reports) for the design of comprehensive architecture for a software testing tool. [22,23,24]
- Create a software architecture testing tool architecture bases on requirement for testing through different literature [33] and identify attributes (data member and member function).
- Identify an attribute of the class's architecture and find relationships between different testing classes in the architecture.
- Based attributes and the relationship between function and component we identified different metrics which is supporting our comprehensive architecture. Descriptive Statistics Examine distribution and variance for each measure.
- Validation of our architecture and determines the quality of software products using empirical and comparative analysis of the different case studies. Principal Component Analysis PCA is the standard technique to identify the underlying dimension (class property) that explains the relations between the variation in the data set.
- Finally on the basis of the above study we determine following goals: final architecture of software for testing, determine the quality of software products and study both (Procedural and Component Based) design.

An architecture tool is complete if and only if it entirely describes and specifies the system that exactly fulfills all requirements and the model contains all necessary information that is needed to implement that desired model. Increasing the completeness of a requirements specification can decrease its consistency and hence affect the correctness of the final product. Conversely, improving the consistency of the requirements can reduce the completeness, thereby again diminishing correctness [20].Davis states that completeness is the most difficult of the specification attributes to define and incompleteness of specification is the most difficult violation to detect [31].According to Boehm [22], to be considered complete, the requirements document must exhibit three fundamental characteristics: (1) No information is left unstated or "to be determined", (2) The information does not contain any undefined objects or entities, (3) No information is missing from this document. The first two properties imply a closure of the existing information and are typically referred to as internal completeness. The third property, however, concerns the external completeness of the document [23]. External completeness ensures that all of the information required for problem definition is found within the specification. This definition of external completeness clearly demonstrates why it is impossible to define and measure the absolute completeness of the specification because how could analysts know with certainty what is missing from the specification when they do not even know what it is that they are looking for in the first place. Architectural Completeness is defined as an architecture including all or nearly all features and relationships required for migrating from one testing class to another.

3. SOFTWARE METRICS USE IN REALIZATION FOR COMPREHENSIVE ARCHITECTURAL TESTING TOOL

3.1 Identify Metrics

According above relationship among different testing technique/strategies, we realize the architecture of testing tool using some software metrics and finally determine software quality of software. Chidamber, Agrawal and some other researcher [4,5,10,12,13,14] proposed twenty two metrics but, here used those metrics which are useful for my research work:

1. **Size Metrics:**

a) Number of Attributes per Class (NOA)

b) Number of Methods per Class (NOM)

c) Response For a Class (RFC)

d) Weighted Methods per Class (WMC)

2. **Coupling and Cohesion Metrics:**

a) Coupling Between Objects (CBO)

3. Inheritance Metrics:

a) Depth of Inheritance (DIT)

b) Number of Children (NOC)

All of above metrics used for deciding completeness of software and provide help to measuring quality of software products.

4. RESULT ANALYSIS AND DISCUSSION

Here we summarize our work from above tables for realizing this model through attribute relationship and determine quality of the model using a different set of metrics , and finally most of the values of our architectural model are following standard values and decide the value quality of model and summarize .

WMC: -The Higher WMC values correlate with increased development, testing and maintenance efforts. As a result of inheritance, the testing and maintenance efforts for the derived classes also increase as a result of higher WMC for a base class (0-50). Above table shows for each component or relation values in between 0 to 50.DIT: - Inheritance (generalization), is a key concept in the object model. While the reuse potential goes up with the number of ancestors, so does design complexity, due to more methods and classes being involved. Studies have found that higher DIT counts correspond to greater error density and lower quality. A class situated too deeply in the inheritance tree will be relatively complex to develop, test and maintain. It is useful, therefore, to know and regulate this depth. A compromise between the high performance power provided by inheritance and the complexity which increases with the depth must be found. A value of between 0 and 4 respects this compromise. RFC: -Larger RFC counts correlate with increased testing requirements .LCOM: - A higher LCOM indicates lower cohesion. This correlates with weaker encapsulation, and is an indicator that the class is a candidate for disaggregation into subclasses. This metric measures the correlation between the

methods and the local instance variables of a class. High cohesion indicates good class subdivision. Lack of cohesion or low cohesion increases complexity. LOCM range 0 to 1 with zero representing perfect cohesion (each method accesses all attributes), however we have noticed that some values exceed 1. NOA: - A class with too many attributes may indicate the presence of coincidental cohesion and require further decomposition, in order to better manage the complexity of the model. If there are no attributes, then serious attention must be paid to the semantics of the class, if indeed there are any. A high number of attributes (> 10) probably indicate poor design, notably insufficient decomposition. A value of between 2 and 5 respects this compromise. NOC: -If Values of NOC are larger than reuse of classes also increases, and by this reason increased testing. A class from which several classes inherit is a sensitive class, to which the user must pay great attention. It should, therefore, be limited, notably for reasons of simplicity. A value of between 1 and 4 respects this compromise. NOM: - this would indicate that a class has operations, but not too many. A value greater than 7 may indicate the need for further object-oriented decomposition, or that the class does not have a coherent purpose. This information is useful when identifying a lack of primitiveness in class operations (inhibiting re-use), and in classes which are little more than data types. A value of between 3 and 7 respects this compromise. CBO: - Excessive coupling limits the availability of a class for reuse, and also results in greater testing and maintenance efforts. Use links between classes define the detailed architecture of the application, just as use links between packages define the high level architecture. These use links play a determining role in design quality, notably in the development and maintenance facilities. Value of 0 indicates that a class has no relationship to any other class in the system, and therefore should not be part of the system. A value between 1 and 4 is good, since it indicates that the class is loosely coupled. A number higher than this may indicate that the class if too tightly coupled with other classes in the model, which would complicate testing and modification, and limit the possibilities of re-use.

Architecture Diagram Result Analysis: In this section the results of PC analysis are presented. The PC analysis extraction method and varimax rotation method are applied to different class level metrics. PCA is one of the benchmarks for dimension reduction technique here first principal components extract a maximum of the variables and second they are interrelated .The First one ensures that the minimum of total information will be missed when looking at the first few principal components. The second one ensures that the extracted information will be organized in an optimal way. Numbers of dimensions captured are quite less than the total number of metrics, implying that many metrics are highly related .Here we used normalizes our variable into three dimensions. In appendix section, we discuss details result data analysis using different table and figure show principal component and eigenvalues in the appendix along with variance (standard deviation) .

5. CONCLUSION

In this research work, we identify implements a set of metrics for measurement of architectural testing model, used to evaluate the quality of the architectural models. Certain model characteristics are measured against quality criteria determined by users thereby allowing you to check that your models meet these quality criteria and appraise the overall quality of a project and find out development of different sub-systems is standard or not .This research work used for developing industrial tools for larger data set, and finally most of the values of our architectural model are following standard values .Hence our architecture is useful for any testing process.

6. Appendix

In below table shows each architectural component value (attribute, class, methods, coupling , cohesion and inheritance) , min range , maximum range, mean, median and standard deviation these all attribute are helping to decide statically calculation to determine the quality of our architecture.

Metrics	Min	Max	Mean	Median	Standard deviation
Relationship between fault and scenario based testing					
WMC	3	4	3.5	3.5	0.7
NOA	0	1	0.5	0.5	0.7
NOM	3	4	3.5	3.5	0.7
RFC	0	4	2	2	2.82
Relationship between scenario and use case based testing					
NOA	0	1	0.5	0.5	0.7
NOM	3	4	3.5	3.5	0.7
WMC	0	3	1.5	1.5	2.12
CBO	0	1	0.5	0.5	0.7
Relationship between thread based, cluster based, use based and integration testing					
DIT	0	3	1.5	1.5	0.7

NOC	0	3	1.5	1.5	2.12
NOA	0	1	0.5	0.5	0.7
NOM	3	4	3.5	3.5	0.7
WMC	3	4	3.5	3.5	0.7
Relationship between state based and category based testing					
NOA	0	1	0.5	0.5	0.7
NOM	3	4	3.5	3.5	0.7
WMC	3	4	3.5	3.5	0.7
Relationship between thread based and cluster based testing					
NOA	0	1	0.5	0.5	0.7
NOM	3	4	3.5	3.5	0.7
WMC	3	4	3.5	3.5	0.7
Relationship between state based and attribute based testing					
NOA	0	1	0.5	0.5	0.7
NOM	0	3	1.5	1.5	2.12
WMC	0	3	1.5	1.5	2.12
Relationship between partition based, state based, attribute based and category based testing					
DIT	0	3	1.5	1.5	2.12
NOC	0	3	1.5	1.5	2.12
NOA	0	1	0.5	0.5	0.7
NOM	2	4	3	3	1.42
WMC	2	4	3	3	1.42
Relationship between class based ,partition based, random based testing ,random based testing					
DIT	0	2	1	1	1.41
NOC	0	2	1	1	1.41
NOA	1	2	1.5	1.5	0.7
NOM	2	4	3	3	1.42
WMC	2	4	3	3	1.42

Table: 1. Analysis of Architecture Testing tool Using Metrics Calculation and Descriptive Statics Analysis

PCA ANALYSIS: - In this section the results of PC analysis are presented. The PC analysis extraction method and varimax rotation method are applied to different class level metrics. PCA is one of the benchmarks for dimension reduction technique here first principal components extract a maximum of the variables and second they are interrelated .The First one ensures that the minimum of total information will be missed when looking at the first few principal components. The second one ensures that the extracted information will be organized in an optimal way. Numbers of dimensions captured are quite less than the total number of metrics, implying that many metrics are highly related .Here we used normalizes our variable into three dimensions.

1. **The relationship between fault and scenario based testing**

Metrics	Min	Max.	Mean	Median	S.dev.	PCA_1_Axis_1	PCA_1_Axis_2	PCA_1_Axis_3
WMC	3	4	3.5	3.5	0.7	2.32472	-0.469386	-2.20E-08
NOA	0	1	0.5	0.5	0.7	-3.63037	-1.2021	7.76E-09
NOM	3	4	3.5	3.5	0.7	2.32472	-0.469386	-2.20E-08
RFC	0	4	2	2	2.82	-1.01907	2.14087	-5.31E-08

Table: 2: PCA (Relationship between fault and scenario based testing)

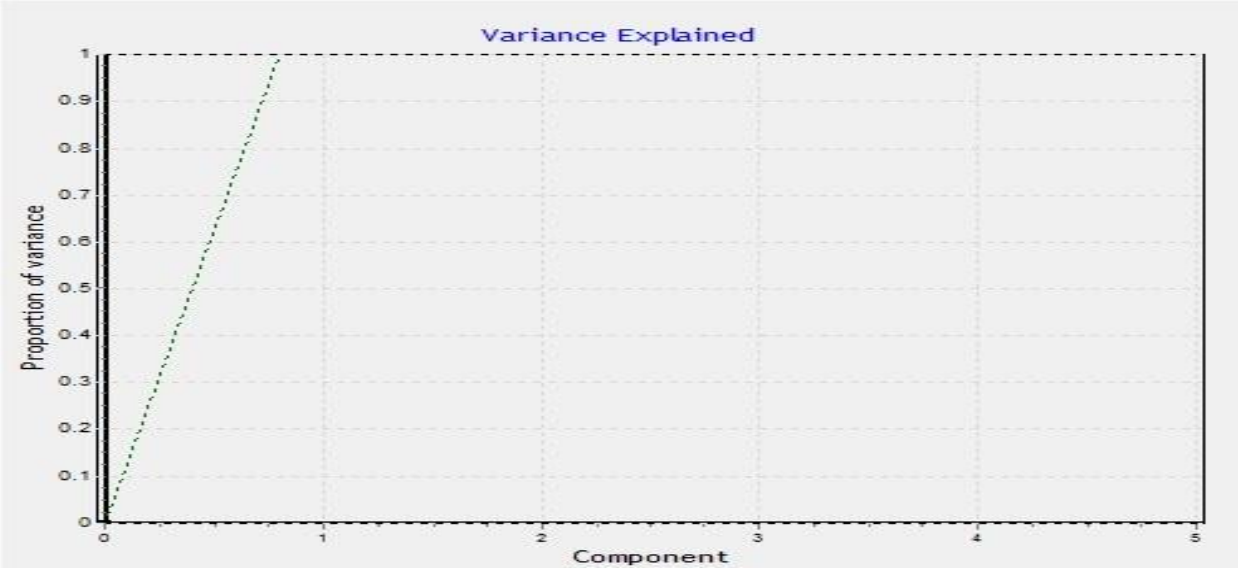

Fig: 1: Component and variance (Relationship between fault and scenario based testing)

Fig: 2: Eigenvalue with component (Relationship between fault and scenario based testing)

In the above table: 2, In first PCA the value of NOM value higher than others metrics , then its uniquely determine the characteristic, In second PCA axis RFC value higher than others metrics , then its uniquely determine the characteristics .In third PCA axis RFC value higher than others metrics , then its uniquely determine the characteristic . Fig.1 shows the relationship of the component with variance and fig. 2 eigenvalue with the component.

2. **The relationship between scenario and use case based testing**

Metrics	Min	Max.	Mean	Median	S.dev.	PCA_1_Axis_1	PCA_1_Axis_2	PCA_1_Axis_3
NOA	0	1	0.5	0.5	0.7	-2	0.501024	2.34E-09
NOM	3	4	3.5	3.5	0.7	4	0.501024	-7.67E-09
WMC	0	3	1.5	1.5	2.12	1.55E-16	-1.50307	-7.43E-10
CBO	0	1	0.5	0.5	0.7	-2	0.501024	2.34E-09

Table: 3: PCA (Relationship between scenario and use case based testing)

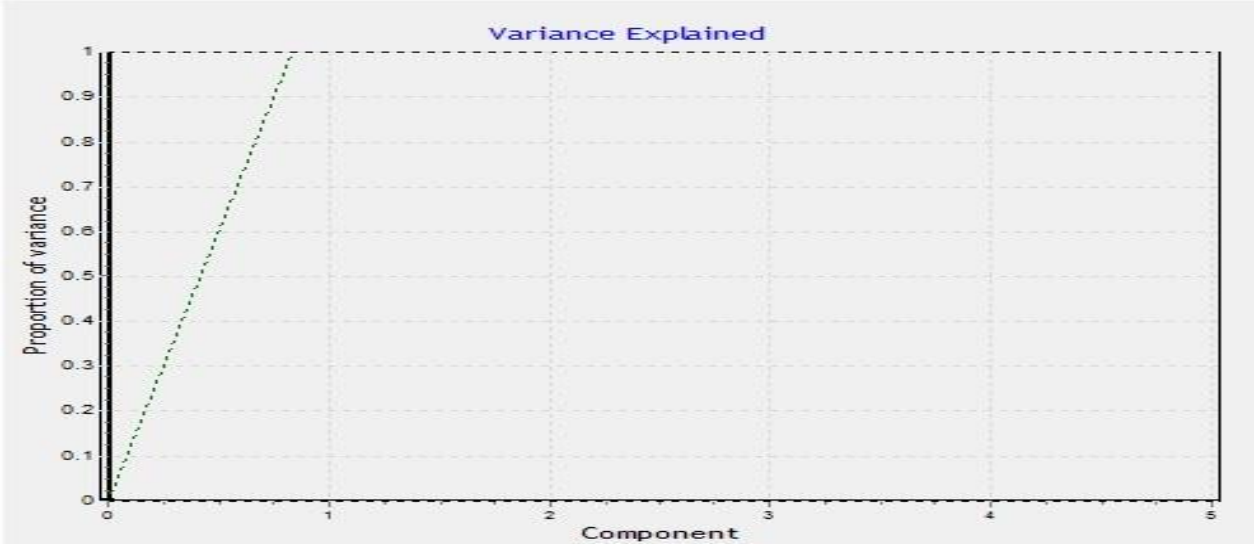

Fig: 3: Component and variance (Relationship between scenario and use case based testing)

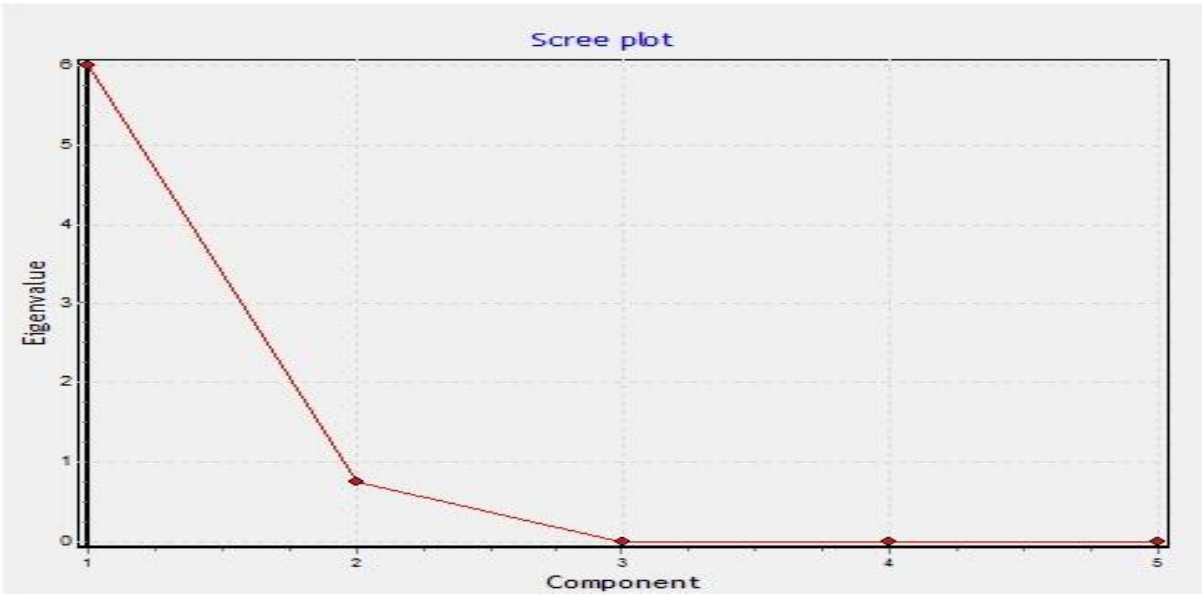

Fig: 4: eigenvalue with component (Relationship between scenario and use case based testing)

In the above table: 3, In first PCA the NOM value higher than others metrics , then its uniquely determine the characteristic, In second PCA axis NOM, CBO, NOA all have the same value that is higher than others metric's value , then its uniquely determine the characteristics .In third PCA axis CBO, NOA same value that is higher than others metrics , then its uniquely determine the characteristic and fig. 3 shows the relationship of the component with variance and fig. 4, Eigenvalue with the component.

3. **The relationship between thread based, cluster based, is based and integration testing**

Metrics	Min	Max.	Mean	Median.	S.dev.	PCA_1_Axis_1	PCA_1_Axis_2	PCA_1_Axis_3
DIT	0	3	1.5	1.5	0.7	-0.830130458	-0.409093738	0.748460114
NOC	0	3	1.5	1.5	2.1199999	-1.234251022	1.90587604	-0.104535602
NOA	0	1	0.5	0.5	0.7	-2.521803856	-1.13086307	-0.408913165
NOM	3	4	3.5	3.5	0.7	2.293092728	-0.182959586	-0.117505632
WMC	3	4	3.5	3.5	0.7	2.293092728	-0.182959586	-0.117505632

Table: 4: PCA (Relationship between thread based, cluster based, use based and integration testing)

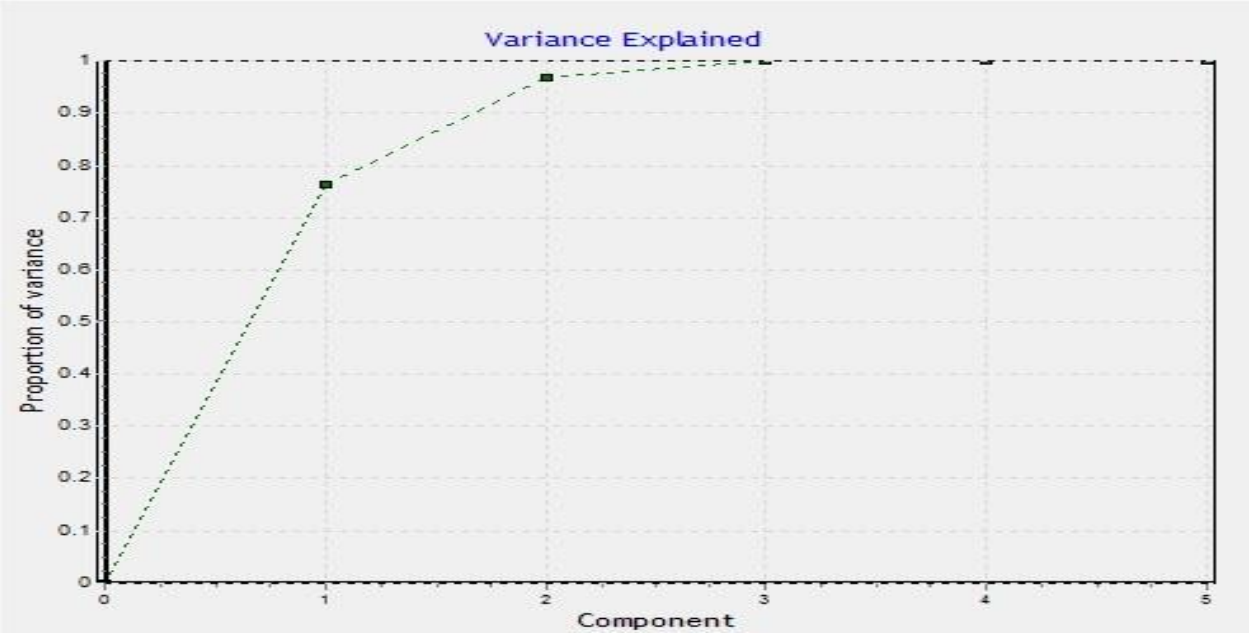

Fig: 5: Component and variance (Relationship between thread based, cluster based, use based and integration testing)

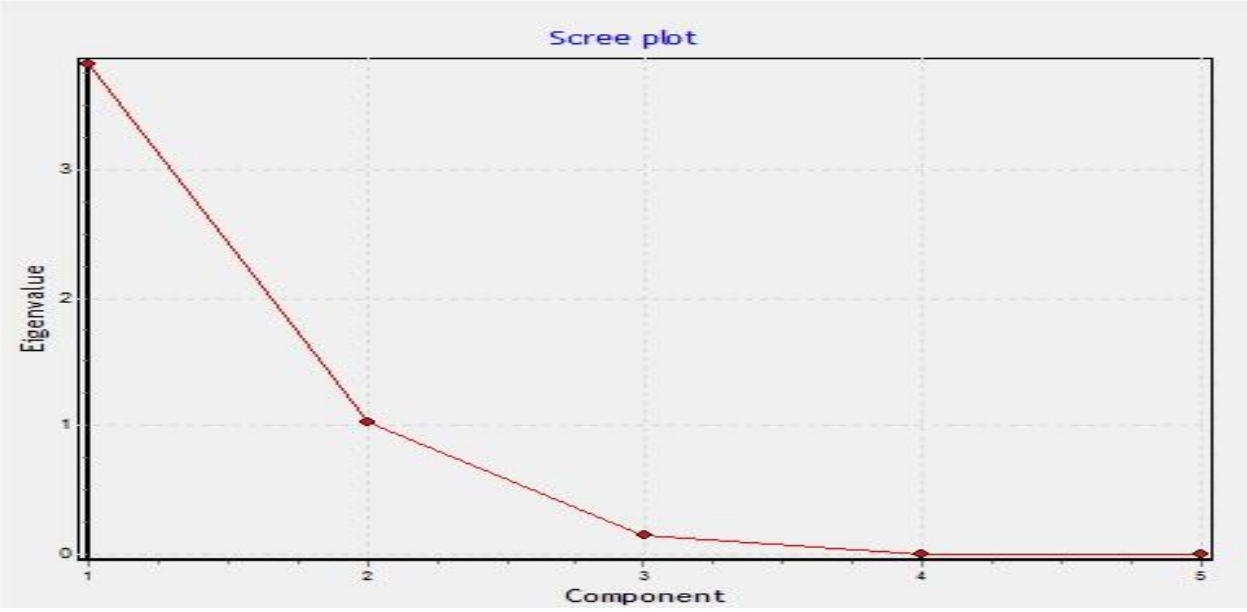

Fig: 6: Eigen-value with component (Relationship between thread based, cluster based, use based and integration testing.

In the above table: 4, In first PCA the value of NOM, WMC values higher than others metrics , then its uniquely determine the characteristic, In second PCA axis NOC value higher than others metrics , then its uniquely determine the characteristics .In third PCA axis DIT value higher than others metrics , then its uniquely determine the characteristic . Fig.5 shows the relationship of the component with variance and fig. 6 eigenvalue with the component.

4. The relationship between partition based, state based, attribute based and category based testing

Metrics	Min	Max.	Mean	Median.	S.dev.	PCA_1_Axis_1	PCA_1_Axis_2	PCA_1_Axis_3
DIT	0	3	1.5	1.5	2.1199999	0.678727567	-1.295258522	-2.54E-09
NOC	0	3	1.5	1.5	2.1199999	0.678727567	-1.295258522	-2.54E-09
NOA	0	1	0.5	0.5	0.7	2.951234579	1.437471032	-6.77E-08
NOM	2	4	3	3	1.42	-2.154345036	0.576523006	-2.32E-08
WMC	2	4	3	3	1.42	-2.154345036	0.576523006	-2.32E-08

Table: 5: PCA (Relationship between partition based, state based, attribute based and category based testing)

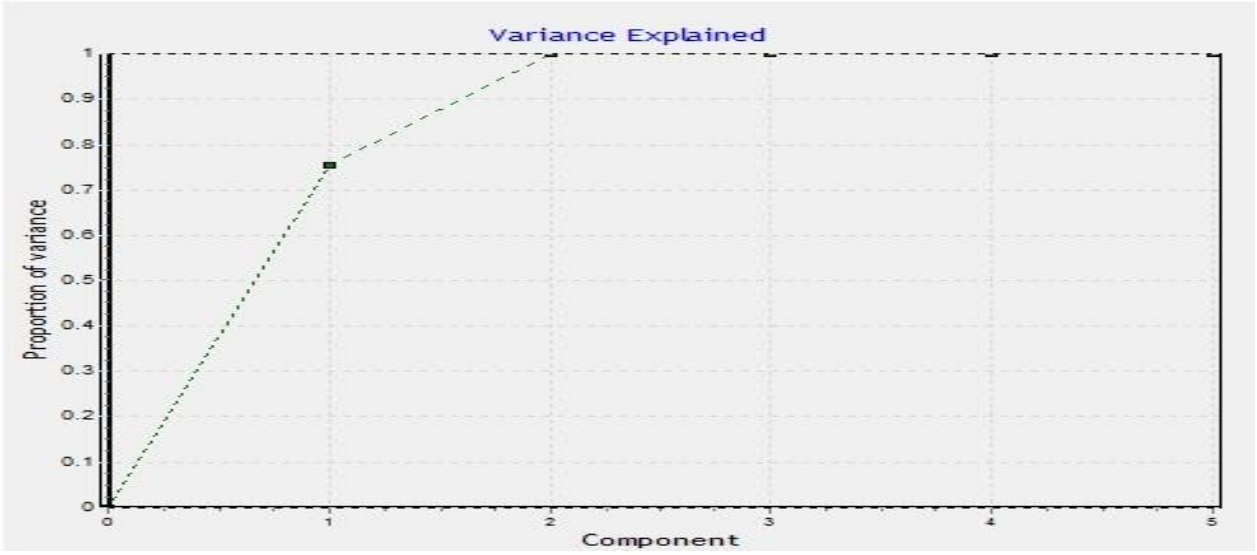

Fig: 7: Component and variance (Relationship between partition based, state based, attribute based and category based testing)

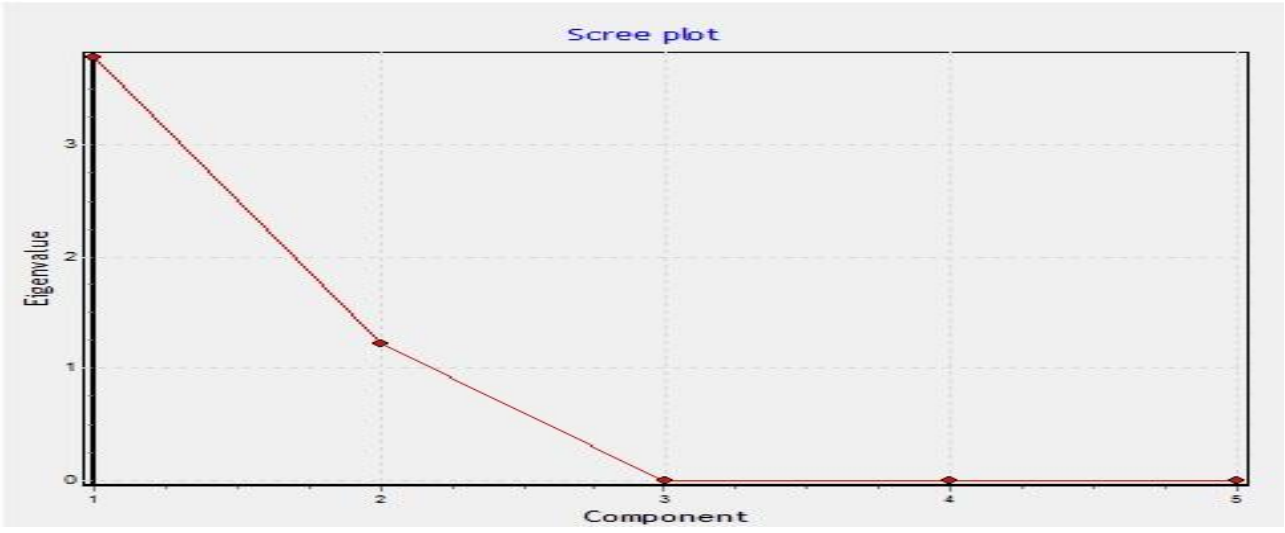

Fig: 8: eigenvalue with component (Relationship between partition based, state based, attribute based and category based testing)

In the above table: 5, In first PCA the value of NOA value higher than others metrics , then its uniquely determine the characteristic, In second PCA axis NOA value higher than others metrics , then its uniquely determine the characteristics .In third PCA axis NOM, WMC value higher than others metrics , then its uniquely determine the characteristic . Fig.7 shows relationship of the component with variance and fig. 8 eigenvalue with the component.

5. The relationship between class based , partition based, random based testing , random based testing

Metrics	Min	Max.	Mean	Median.	S.dev.	PCA_1_Axis_1	PCA_1_Axis_2	PCA_1_Axis_3
DIT	0	2	1	1	1.41	1.85126543	-0.806734622	-1.60E-08
NOC	0	2	1	1	1.41	1.85126543	-0.806734622	-1.60E-08
NOA	1	2	1.5	1.5	0.7	1.144376278	1.933143616	1.81E-09
NOM	2	4	3	3	1.42	-2.423453569	-0.159837201	-2.22E-08
WMC	2	4	3	3	1.42	-2.423453569	-0.159837201	-2.22E-08

Table: 6: PCA (Relationship between class Based, partition based, random based Testing, random based testing)

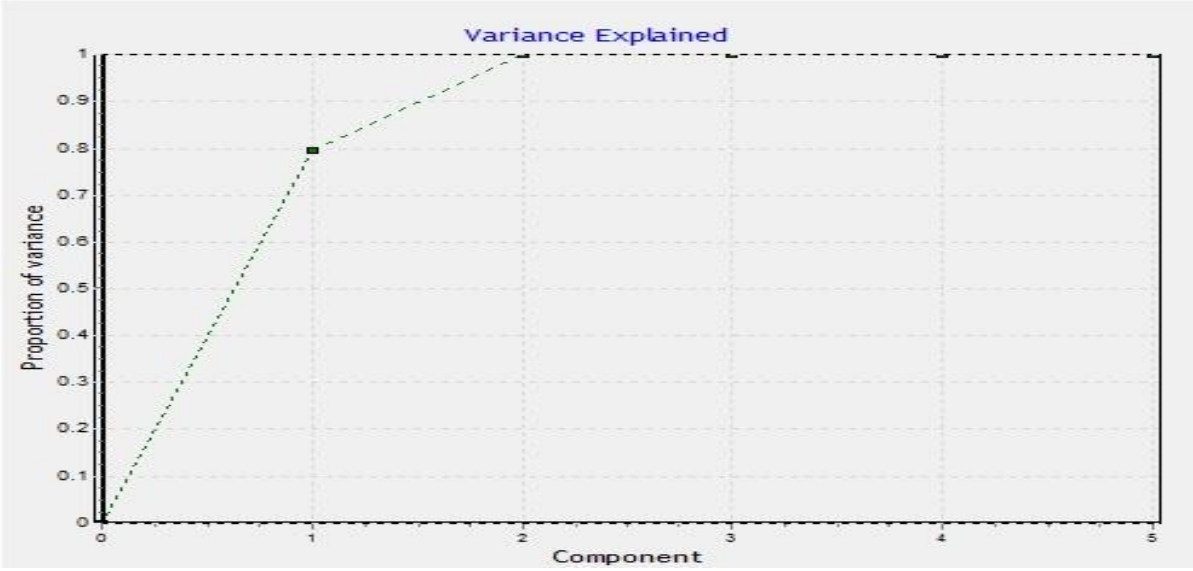

Fig: 9: Component and variance (Relationship between class based, partition based, random based testing , random based testing)

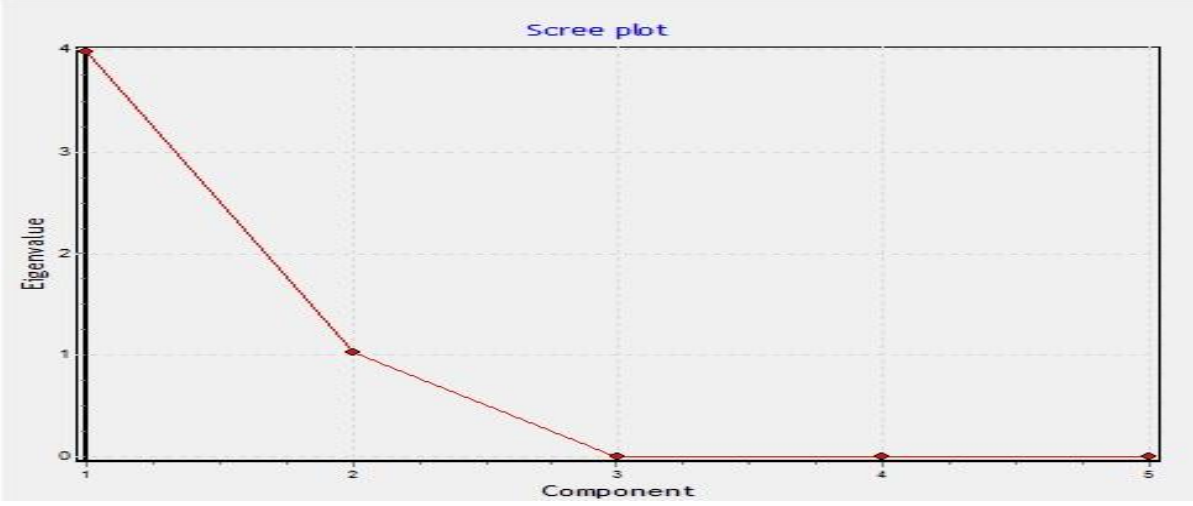

Fig: 10: eigenvalue with component (Relationship between class Based, partition based, random based testing , random based testing)

In the above table: 6, In first PCA the value of DIT, NOC value higher than others metrics , then its uniquely determine the characteristic, In second PCA axis NOA value higher than others metrics , then its uniquely determine the characteristics .In third PCA axis NOA value higher than others metrics , then its uniquely determine the characteristic . Fig.9 shows relationship of the component with variance and fig. 10 eigenvalue with the component.

In our next paper we try to analysis of large data that are cover maximum characteristics of any software products.

7. ACKNOWLEDGMENTS

We extend our thanks to Dr. Abhay Kothari Director of Sanghvi Institute of Management and Science, Indore (India) for their valuable support and discussion on the testing classification of testing.

8. REFERENCES

[1] Gary Chastek and Robert Ferguson, "Toward Measures for Software Architectures (Software Engineering Measurement and Analysis)," Software Engineering Institute, Carnegie Mellon University, CMU/SEI-2006-TN-013, March 2006.

[2] Howden W. E., "Functional Testing and Design Abstractions," *System and Software (Elsevier)*, vol. 1, pp. 307-313, 1980.

[3] J. Huang C., "An Approach to Program Testing," *ACM Computing Surveys*, pp. 113-128, September 1975.

[4] Rosenberg Linda H., "Applying & interpreting object oriented Metrics," Software Assurance Technology Center (SATC) and NASA Goddard Space Flight Center , Utah, Software Technology Conference April 1998.

[5] Anderson John L. Jr., "How to Produce Better Quality Test Software," *IEEE Instrumentation & Measurement Magazine* , vol. 8, no. 3 ISSN : 1094-6969, August 2005.

[6] Bitman William R, *Balancing software composition & inheritance to improve reusability cost, and error rate*.: Johns Hopkins APL Technical Digest Vol. 18(4) , 485–500., November 1997.

[7] Harrison R., Counsell S., and Nithi R., "Coupling metrics for object oriented design," in *Software metrics, symposium*, MD, USA, November 1998, pp. 150-157.

[8] Chidamber S. and Kemerer C., "A metrics suite for object oriented design," *IEEE Trans. Software Eng.*, vol. 20, pp. 476-493, 1994.

[9] Agarwal k. K., Sinha Y., Kaur A., and Malhotra R., "Exploring Relationships among coupling metrics in object oriented systems," *CSI*, vol. 37 (1), March 2007.

[10] Glenford J. Myers, *The Art of Software Testing*, 2nd Ed.: John Wiley & Sons, 2004.

[11] Dr. Linda Rosenberg, Ted Hammer, and Jack Shaw, "Software Metrics and Reliability," Software Assurance Technology Center (SATC), NASA, 1998.

[12] Bass L., Clements P., and Kazman R., *Software Architecture in Practice*, 2nd Ed. Boston: MA: Addison-Wesley, 2003.

[13] Nick Jenkins, "A Software Testing Primer," 2008.

[14] Soni D., Nord R., and Hofmeister C., "Software Architecture in Industrial Applications," in *Proceedings of the 17th International Conference on Software Engineering. Seattle NY: ACM Press*, Washington, New York, April 23-30, 1995.

[15] Hetzel William C., *The Complete Guide to Software Testing*, 2nd Ed.: Wellesley, Mass.: ED Information Sciences ISBN:0894352423. Physical description: ix, 280 p.: ill; 24cm, 1988.

[16] Jiantao Pan, *Software Testing 18-849b Dependable Embedded Systems Spring*., 1999.

[17] Edward Miller, "Introduction to software testing technology. In Tutorial: Software Testing & Validation Techniques," *IEEE Computer Society Press*, pp. 4-16, 1981.

[18] Reiner R. Dumke and Achim S. Winkler, "Managing the component- Based Software Engineering with Metrics," *0-8186-7940-9/97 IEEE*, 1997.

[19] Hareton K.N. Leung, "Test Tools for the Year 2000 Challenges,".

[20] Williams C. T, "The STCL test tools architecture," vol. 41, no. 1

[21] Perry D. E. and Wolf A. L., "Foundations for the study of . software architecture," *SIGSOFT Soft. Engg*., 17 (4), 1992.

[22] Boehm BW, "Verifying and validating software requirements and design specifications," *IEEE Software*, vol. 1, no. 1, pp. 75-88, 1984.

[23] Cordes DW and Carver DL., "Evaluation methods for user requirements documents," *Information and system Technology*, vol. 31, no. 4, pp. 181-188, 1989.

[24] Davis AM, *Software Requirements: Analysis and Specification*, 2nd Ed.: Prentice Hall, 1993.

[25] K. K. Agarwal, Yogesh Sinha, Arvinder Kaur, Ruchika Malhotra " Exploring Relationships among coupling metrics in object oriented systems. Journal of CSI vol. 37, no. 1, January March 2007

[26] Robert M. Poston, "Testing tool combine best of new and old," *IEEE Software*. March 2005.

[27] Williams et. Al., "The STCL Test Tool Architecture," *IBM Systems Journal*, Vol 41, No.1, 2002.

[28] Lionel C. Briand, John W. Daly, and Jurgen Wust, "A unified framework for coupling measurement in object-oriented system", *IEEE transaction on software engineering*, 1996.

[29] Lionel C. Briand, John Daly " A Comprehensive Empirical Validation of Design Measures for Object-Oriented Systems", Fraunhfer IESE, 1999.

[30] Lionel C. Briand, "Investigating Quality control in object oriented design: an industrial case study" *ACM-1999.*

[31] Birand, W. Daly and J. Wust "Exploring the relationship between design measures and software quality, .*Journal of systems and software*, 5 (2000) 245-273.

[32] Juan Carlos Esteva, "Learning to Recognize" (Krauskopf, 1990) Jan Krauskopf, "The cohesive highs and the coupling lows of good software design*", IEEE*, 1990.

[33] Sun Chong-ai ,Leu Chao, "Architecture Framework for object-oriented Design," *IEEE Transaction on Software Engineering*, 2004.

[34] Lalji Prasad and Sarita Singh Bhadauria, A full featured component based architecture testing tool, *International Journals of Computer Science Issues*, Vol. 8, Issue 4, 2011.