IASET: Journal of Computer Science and Engineering (IASET: JCSE)
ISSN(P): Applied; ISSN(E): Applied
Vol. 1, Issue 1, Jan - Jun 2016, 5-18

International Academy of Science, Engineering and Technology
Connecting Researchers; Nurturing Innovations

VALIDATION OF SOFTWARE PRODUCTS (OBJECT-ORIENTED TECHNIQUE) USING WITH FACILITATE QUALITY ATTRIBUTES- A HOLISTIC APPROACH FORSOFTWARE QUALITATIVE DESIGN

LALJI PRASAD[1]&SARITA SINGH BHADAURIA[2]

[1]Truba College of Engineering and Technology, Department of Computer science and Engineering, Indore, TCET (RGTU), Bhopal India

[2]Department of Electronics, MITS (RGTU), Madhav Institute of Technology and Science, Gwalior, India

ABSTRACT

Based on existing work (Testing Tool: "A full featured component based architecture testing tool"); which draws a comprehensive architecture of testing method, and based on their attribute nature shows their relationship or behavior of attributes in terms of object oriented [34]. This research workincludes a cse study on "snaker game" for validatingarchitecture tool, based on object oriented testing characteristics. The tool identifies attributes and coorelate with object oriented testing method at class level architecture which provides quantification and justify for testing quality attributes based on different software matrices on each component(class).

KEYWORDS: Architectural Completeness, Architectural Quality Attribute, Architectural Metrics

1. INTRODUCTION

The deliverable produced by a quality development process is excellent software that satisfies the evolving needs of users. Comprehensive means that it includes all or nearly all features (user requirement, performance, maintainability, reusability, flexibility, reusability, simplicity 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 (component) level architecture having relationships between various testing classes. Software quality is another focus of ourarchitecture. We wish to a analyzedquality attribute of software engineering products good maintainability, reusability, flexibility and portability in the architecture of the software testing tool by validating the architecture using testingalgorithms and performing metrics calculation on each relationship existing between the different testing techniques [1, 2, 3].

This paper is organized as follows: Section 2 discussed literature review and related work. Section 3 discusses the methodology for the research work. Section 4 presented the Software Metrics use in RealizationofEmployeeManagement Systems. Section 5 presentedresult analysis and discussion and section 6 conclude research work.

2. LITERATURE REVIEW AND RELATED WORK

Some researcher work on quality of software architecture and testing for ensuring the quality of software, here discuss only prominence few literature. Bas et al., articulated importance of software architecture [12].Soni and et al.have defined, software architectures describe how a system is decomposed into components, how these components areinterconnected, and how they communicateand interact with each other's [14]. Perry and Wolf, work on Softwarearchitecture is concerned with the study of the structure of software, including its topology, properties, constituentcomponents and their

relationships and patterns of combination [21]. Gary Chastek and Robert Ferguson enlightensoftware architectural attributes and quality relation [1]. The basic rules for program testing, which provide basic principle for testing has been covered by a number of researcher [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 toprovide a single graphical interface into the test tool set. But with the limitation of emphasizing only integration tool(usability andportability).

Aditi et al., [4] provides,the approach to software metric for object oriented programming which is different from the standard metric sets. Some metrics, such as, line of code andcyclomatic 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 et al. [25] cited in their paper the importance of software measurement is increasing leading to the development of newmeasurement techniques. Limitation of the worka) It does not provide any relationship between requirements and 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 andmetrics will improve the software quality and reduce the overall development time. Good quality code will also be easier to write, understand, maintain and upgrade.The limitation of their work a) it's not providing any relationship between the requiredtesting attribute.b) It does not provide a full featured testing tool (only Complexity and cohesion measure). c) Here provide the only conceptual framework for measurement. Briand et al., and some other researchers [9,11,28,29,30,31] demonstrateaims are thatempirically the relationships between most of the existing coupling and Cohesion measures for object oriented (OO)system and fault proneness of object oriented system classes can be studied. Limitation of the work is only emphasis on cohesion and coupling metric. Bitman [6] exhibit key problem in software development of changingsoftware- development complexity and the method to reduce complexity. Limitation of the work it does provide only complexity measurement techniques. Krauskopf and et al. [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 andcomplex system. No coupling is highly is desirable but practically it is not possible. The good and bad points of different types of coupling are discussed. Limitation of the work is only emphasis on cohesion and coupling metrics. Chidambaram [8] and Harrison [7] emphasized the coupling between object (CBO) metric and evaluated for five object oriented systems and compared with alternative design metric called NAS which measure the number of associations between a class and its peers. NAS metric is directly collectible from design documents such as the object model. The limitation of their work: a) It does not provide any relationship between requirements and testing attribute.b) It does not provide some basic idea for size and effort estimation. c) Measuring complexity of a class is subject to bias. Reiner R. et al., Show how to manage component based software and identify related metrics [18]. Here we have taken Employee Management System fordetermining quality of software and validation of the software architecture tool, in this

3. RESEARCH METHODOLOGY/EXPERIMENT FLOW

In this research work first we establish a requirement specification for qualitative testing tool using formal review specification.Requirement gathering for Employee Management System 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).Here we take a case study for project sneaker game and design relationship classar-

chitecture .Secondly identifies an attribute of the class's architecture and find relationships between different testing classes in the architecture [34]. Based attributes and the relationship between function and component we identifieddifferent metrics which is supporting our comprehensive architecture. Descriptive Statistics Examine distribution andvariance for each measure [35]. Validation of our architecture and determines the quality of software products usingempirical 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 Component based design.

An architecture tool[34] is complete if and only if it entirely describes and specifies the system that exactlyfulfills 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 specified attributes to define and incompleteness of specification is the most difficult violation to detect [31]. According to Boehm to be consideredcomplete, the requirements document must exhibit three fundamental characteristics: a) No information is left unstated or "to be determined". b) The information does not contain any undefined objects or entities. c) 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 [22]. The third property, however, concerns the external completeness of the document [23].Architectural Completeness is defined as an architecture including all or nearly all features and relationships required for migrating from one testing class to another.

4. SOFTWARE METRICS USE IN REALIZATION OF EMPLOYEE MANAGEMENT SYSTEMS

In this section we try to identify metrics related to architecture.TheEmplayee Management System ,manageemployee related functionality in any organization here we are designing a system which was followedfunctionality(classes)Number,DataInput,Integer,Employee,Lab1,Manager,Typist, In this project we have 6 class diagram (figure1),and each class diagram related to other class diagram with some specific relationship type,all interrelated with inheritance property of object oriented system and after analysis of class architecture we find out different architecturerelated metrics.According above relationship among different testing technique/strategies, we realize the architecture of testing tool using some software metrics for determing architectural design quality and finally determine software quality of software. Chidamber et al and. [4,5,10,12,13,14] proposed twenty two metrics but, here used those metrics which are useful formeasuring the quality of the architectural diagram of research work:

1. **Size Metrics**

a) Number of Attributes (NOA)

b) Number of Methods (NOM)

c) Response for a Class (RFC)

d) Number of Children (NOC)

2. **Reuse Metrics:**

a) Reuse Ratio (U)

b) Specialization Ratio (S)

3. Inheritance Metrics:

a) Method Inheritance Factor (MIF)

b) Attribute Inheritance Factor (AIF)

c) Depth of Inheritance (DIT)

4. Polymorphism Metrics:

a)Number of methods overridden by a subclass(NMO)

b)Polymorphism Factor (PF)

5. Coupling and Cohesion Metrics:

a) Coupling Between Object (CBO)

In above metrics some of their values are very low then there impact in data analysis is negligible and others used for providing help to decide the quality of software products (details in table. 2). Quality attributes standard of architectural diagram find through metrics analysis in below graphs.

5. RESULT ANALYSIS AND DISCUSSIONS

Realizing this model through attribute relationship and determine the quality of the model (table. 1)Usingthe measurement of metrics, and graphical representation and realizing this model

RFC:- The graph shows the relationship between RFC and simplicity factor. It increases initially but it does not affect simplicity after a certain limit and remain constant details in figure 4. **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.The graph shows in figure 2 the relationship between NOA and ***simplicity***factor which linearly increases until the number of attributes is less and later as NOA increases simplicity reduces. The graph shows in figure 5, the relationship between NOA and ***portability*** factor which linearly increases by the number of attributes is less and later as NOA increases portability reduces. The more the number of attributes the more ***requirements*** of user is satisfied,it depicts a linearrelationship in figure 7.**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. The graphfigure 3 showsthe relationship between NOM and ***simplicity*** factor. Increment in NOM reduces the ***simplicity*** of the program. The graph figure 6 shows the relationshipbetween NOM and ***portability*** factor which linearly increases by the number of attributes is less. Further NOM increases ***portability*** remains constant**.**In figure 8 shown initially the relationship between the ***user requirement*** and NOM is linear, but with further increment is the number of methods the user requirement decreases as it introduces complexity.The 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 practical 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. This metric proved to be the best indicator of the ***maintenance*** effort by indicating the class that is more***error prone***. **CBO**: - Excessive coupling limits the availability of a class for reuse, and also results in greater testing and ***maintenance efforts.*** 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 ***reuse.*****NMO: -** In figure 9 shown the overriding of methods increases the ***performance*** of theprogram. In figure 10, shown Overriding of methods decreases the reusability of the program and further increment ofoverridden methods does not affect reusability.

Result Analysis: In this section the results of PC analysis are presented in the figure 11, figure12 and table 3. The PC analysis extraction method and varimax rotation method are appliedto 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 usingtable 3 and figures show principal component and eigenvalues in the appendix along with variance (standard deviation).

6. CONCLUSIONS

In this research work, we identify implements a set of metrics for measurement of architectural testing model for the Employee Management System, used to evaluate the quality of the architectural models. Certain model characteristics are measured against quality criteria determined by users thereby allowing to check that your models meet these quality criteria and appraise the overall quality of a project conclude in table.4. Also this research work used for developingindustrial tools for larger data set, andtry to provide a template comprehensive tool for testing .Hence our architecture is useful for any testing process.

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. Lalji Prasad and Sarita Singh Bhadauria,Design Integration Based testing using Test Case generation technique, International Journal of Scientific & Engineering Research, Volume 3, Issue 5, May-2012
3. Paul Ammann, Jeff Offutt, Introduction to software testing, Cambridge University Press, ISBN. 978-0-521-88038-1,2008
4. Lalji Prasad, Aditi Nagar, Experimental Analysis of Different Metrics (Object-Oriented and Structural) of the Software, CICSYN '09 (Proceedings of the 2009 First International Conference on Computational Intelligence,

Communication Systems) IEEE Computer Society Washington, DC, USA ©2009 and Networks, ISBN: 978-0-7695-3743-6

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. 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 JurgenWust, "A unified framework for coupling measurement inobject-oriented system", *IEEEtransaction 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.
35. Lalji Prasad and SaritaSinghBhadauria,Association between different types of Testing Method uses Absolute Architecture International Journal of Engineering Research & Technology (IJERT) Vol. 1 Issue 6, August – 2012,ISSN: 2278-0181.

APPENDICES

Appendix(Case Study): Employee Details (Object Oriented System)

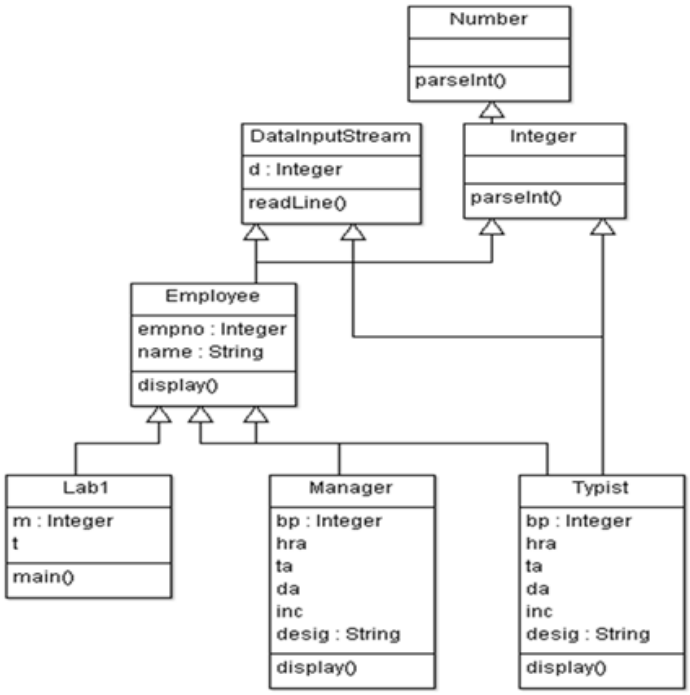

Figure1: Class Diagram for Employee Details

Metrics Calculation Table: The table depicted below gives the metric values for the classes in the employee details case study.

Table 1: Metrics Calculation Table for Employee Details

Classes / Metrics	NOA	NOM	RFC	DIT	NOC	CBO
Number Class	0	1	1	0	1	1
IntegerClass	0	1		1	2	0
Lab1Class	2	1	6	3	0	3
TypistClass	6	1	4	3	0	3
DataInput Stream Class	1	1		1	2	0
Employee Class	2	1	1	2	3	0
Manager Class	6	1	4	3	0	3

Range table: The range table evaluates the minimum and maximum ranges for the metrics calculated in previous table.

Table 2: Ranges for Metrics for Employee Details

Ranges for Metrics for Employee Details					
Metrics	**Minimum**	**Maximum**	**Mean**	**Median**	**St. Deviation**
NOA	0	6	2.42	2	2.57
NOM		1	1	1	0
RFC	1	6	2.28	1	2.36
DIT	0	3	1.85	2	1.21
NOC	0	3	1.14	1	1.21
CBO	0	3	1.42	1	1.51

Table2 shows the value of architectural tool, its shows the mean and standard deviation which ishelp us for deciding our architecture validation.

Inheritance Metrics

a) MIF- Method Inheritance Factor

$$MIF = \frac{\sum_{i=1}^{TC} M_i(C_i)}{\sum_{i=1}^{TC} M_a(C_i)}$$

Where $M_a(C_i)=M_i(C_i)+M_d(C_i)$

And TC=7

MIF=3/10

b) AIF-Attribute Inheritance Factor

$$AIF=\frac{\sum_{i=1}^{TC} A_d(C)}{\sum_{i=1}^{TC} A_a(C_i)}$$

AIF=1

Reuse Metrics

a) Reuse Ratio(U)

U= Number of super classes/Total number of classes

U=4/7

b) Specialization Ratio(S)

S= Number of subclasses/Number of super classes

S=5/4

Polymorphism Metrics

a) NMO-Number of methods overridden by a sub class

NMO DataInputStream=0

NMO Integer= 1

NMO Employee= 0

NMO Lab1= 0

NMO Manager= 1

NMO Typist= 1

a) Polymorphism Factor(PF)

$$PF = \frac{\sum_{i=1}^{TC} M_o(C_j)}{\sum_{i=1}^{TC} [M_n(C_i) \times DC(C_i)]}$$

Where $M_n(C_i)$= number of new methods

$M_o(C_i)$ = number of overriding methods

DC (C_i) =Descendant count

PF= ¼

Size metrics affecting simplicity

Number of attributes (NOA): The graph shows the relationship between NOA and simplicity factor which linearly increases until the number of attributes is less and later as NOA increases simplicity reduces.

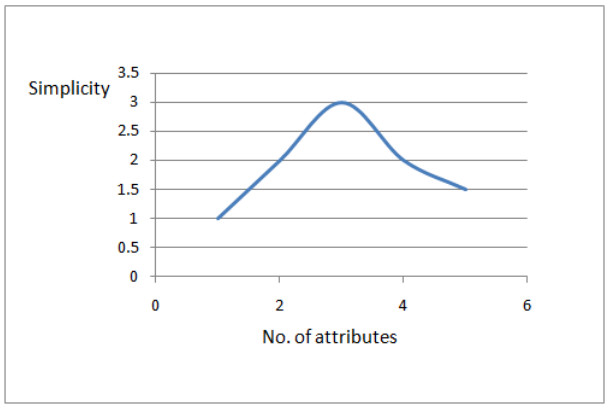

Figure2: Graph Between Simplicity and NOA

Number of Methods (NOM): The graph shows the relationship between NOM and simplicity factor. Increment in NOM reduces the simplicity of the program

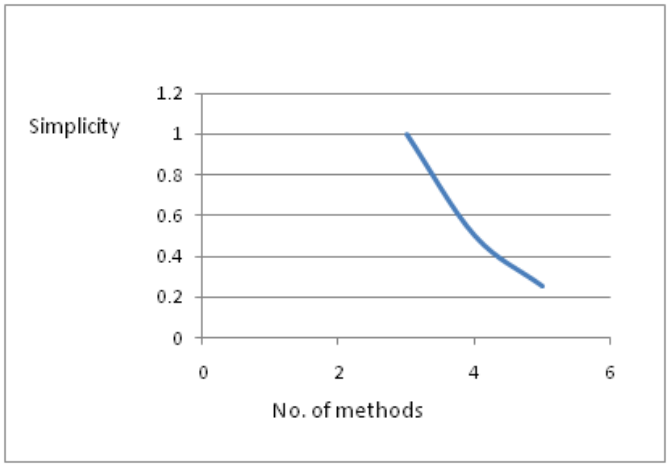

Figure3: Graph between Simplicity and NOM

Response for a class (RFC): The graph shows the relationship between RFC and simplicity factor. It increases initially but it does not affect simplicity after a certain limit and remain constant.

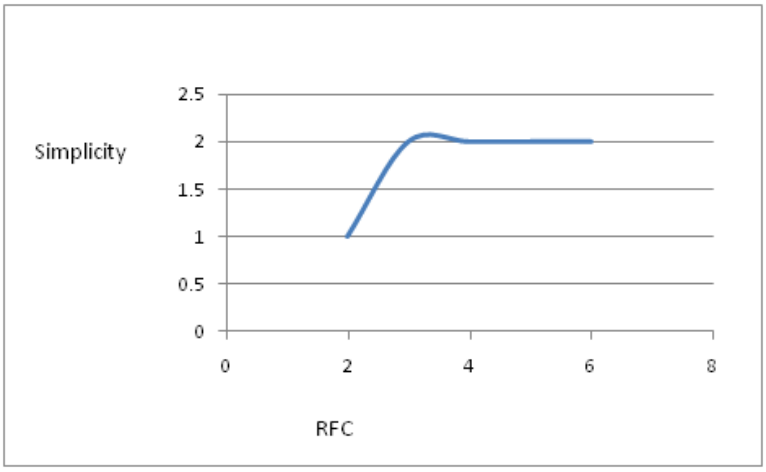

Figure4: Graph between Simplicity and RFC

Size Metrics Affecting Portability

Number of attributes (NOA): The graph shows the relationship between NOA and portability factor which linearly increases until the number of attributes is less and later as NOA increases portability reduces.

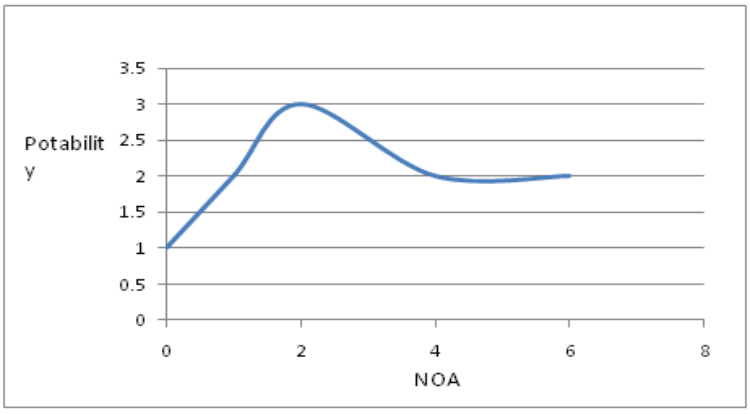

Figure5: Graph between Portability and NOA

Number of methods (NOM): The graph shows the relationship between NOM and portability factor which linearly increases by the number of attributes is less and later as NOM increases portability remain constant.

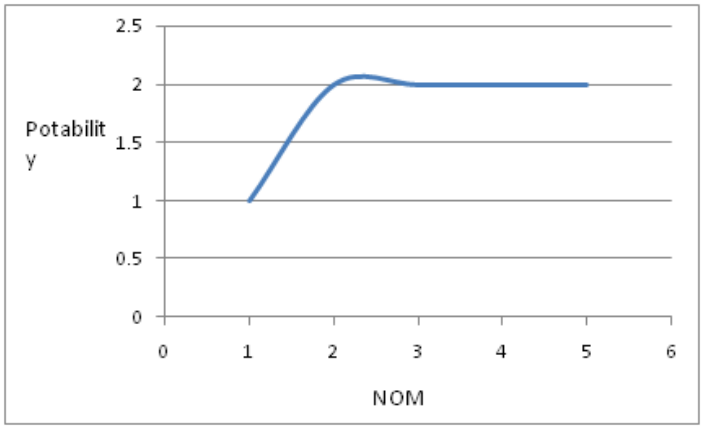

Figure6: Graph between Portability and NOM

Size Metrics Affecting User Requirements

Number of attributes (NOA): The more the number of attributes the more requirements of user is satisfied. Hence it depicts a liner relationship.

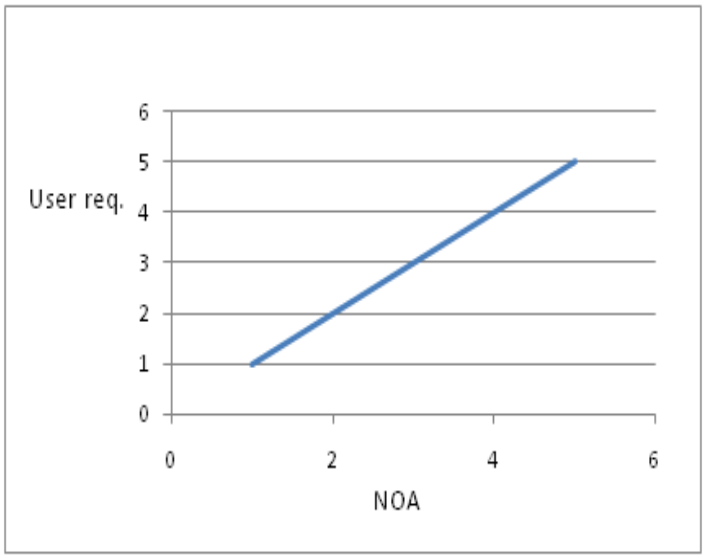

Figure7: Graph between Portability and NOM

Number of methods (NOM): Initially the relationship between the user requirement and NOM is linear, but with further increment is the number of methods the user requirement decreases as it introduces complexity.

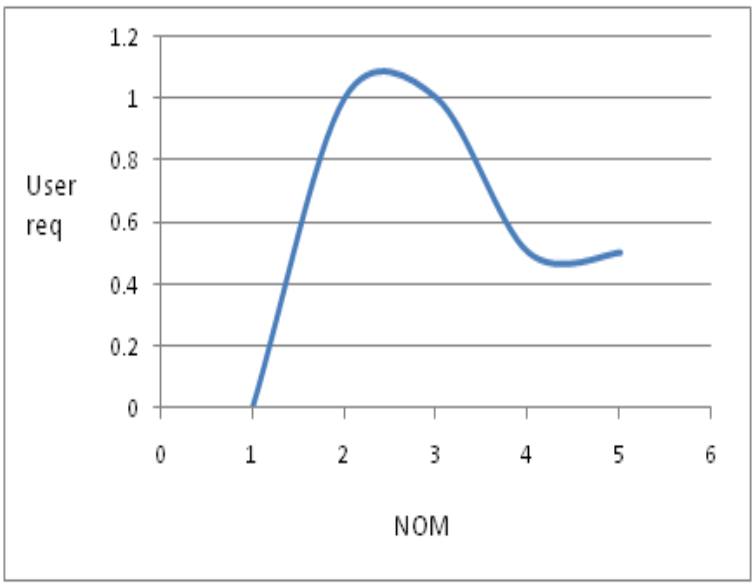

Figure8: Graph between User Requirements and NOM

Polymorphism Metrics Affecting High Performance

Number of Methods Overridden by a Subclass (NMO): Overriding of methods increases the performance of the program and hence depicts a linear relationship.

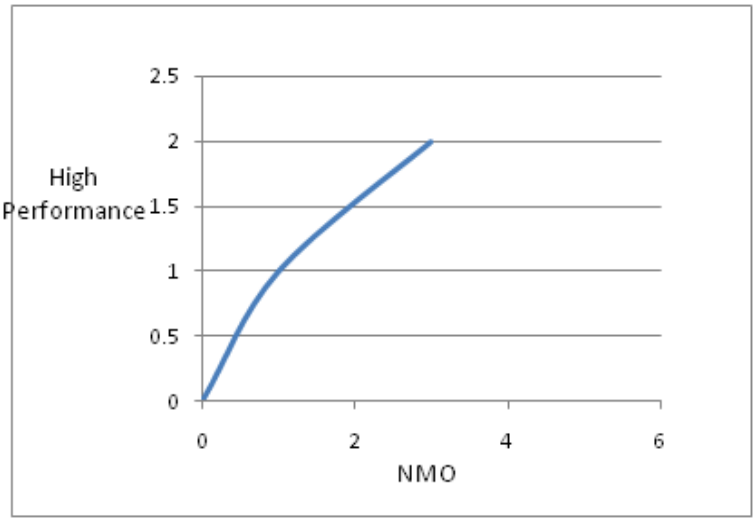

Figure9: Graph between High Performance and NMO

Polymorphism Metrics Affecting Reusability

Number of Methods Overridden by a Subclass (NMO): Overriding of methods decreases the reusability of the program and further increment of overridden methods does not affect reusability.

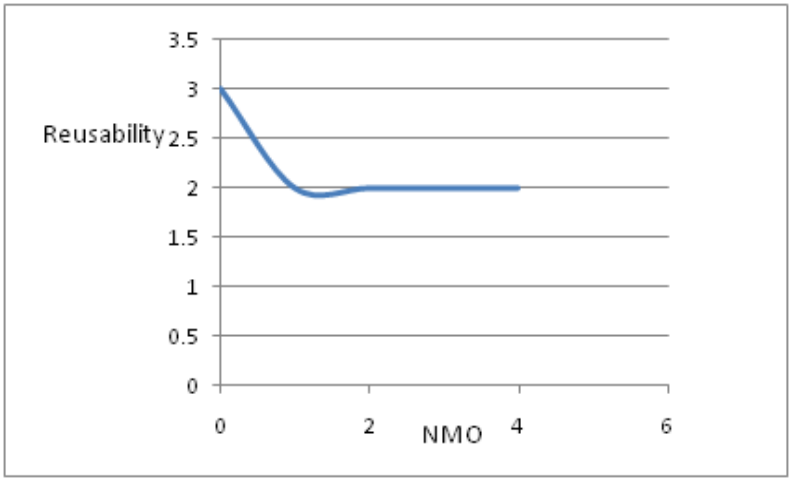

Figure 10: Graph between Reusability and NMO

3. Employee Details

Table: 3: PCA (Employee Details)

Metrics	Min	Max.	Mean	Mdn.	S.dev.	PCA_1_Axis_1	PCA_1_Axis_2	PCA_1_Axis_3
NOA	0	6	2.42000008	2	2.56999993	2.032869339	-1.465382457	-0.597680569
RFC	1	6	2.27999997	1	2.3599999	2.294239044	1.627095461	0.400834352
DIT	0	3	1.85000002	2	1.21000004	-0.937280059	-1.308396816	0.876112819
NOC	0	3	1.13999999	1	1.21000004	-1.982560277	0.637340605	-0.279248655
CBO	0	3	1.41999996	1	1.50999999	-1.407268524	0.509343445	-0.400017887

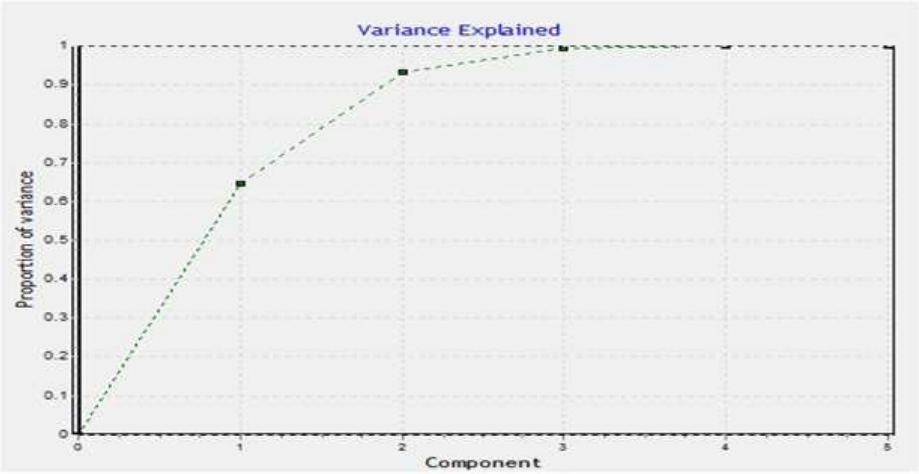

Figure11: Component and Variance (Employee Details)

Figure12: Eigenvalue with Component (Employee Details)

In above table.3, In first PCAthe NOM valuehigher than others metrics, then its uniquely determine the characteristic, In second PCA axisRFCvalue ithigher than others metric's value, then its uniquely determines the characteristics .In third PCA axis CBO ishigher than others metrics, then its uniquely determine the characteristic andfirure11 shows the relationship of the component with variance and figure12, Eigenvalue with the component.In table3 show conclusion quality attributes with related software metrics for design architectural testing tool.

Table 4: Summary of Quality Attributes with Related Software Metrics for Design Architectural Testing Tool

Quality Attributes (Comprehensive Attributes)	Object –OrientedSystem	Mapping Related Metrics
Reusability	High	NOC,CBO, WMC
Simplicity	Low	RFC, NOC,
Portability	High	RFC
High performance	High	NOC
Costeffectiveness	High	DIT,
Testability	High	CBO,RFC, DIT
Maintainability	High	CBO, RFC
Usability	High	CBO, RFC, CBO, WMC
Fault Tolerance	High	RFC, CBO, NOC
Reliability	High	RFC, LCOM