Volume 3, Issue 1, January 2013 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

An Empirical Validation of Complexity Quantification Model

Mohd Nazir
JMI, New Delhi, India
mnazir@jmi.ac.in

Abstract— It is an obvious fact that quantification of software complexity can be vital and may play a highly significant role in delivering quality software. This paper is an extension of the paper [2] that talks about complexity quantification. The proposed model seems to be more significant and of practical nature as it provides quantitative value of complexity. The paper presents empirical validation of model, which is carried out using sample tryouts of industrial projects. Moreover, model utility is illustrated with the help of statistical measures.

Keywords— Software Quality, Software Testability, Software Complexity, Testability Factor

I. INTRODUCTION

Software is playing a highly significant role in our life; multimedia information is widely used in office automation equipments or home information appliances. Software for such systems is becoming increasingly large in size and typically complex. Hence, building reliable and quality software is becoming more and more difficult as the complexity, pervasiveness and criticality of software is ceaselessly growing. Complexity is highly connected with its quality for a simple reason. Complexity measures provide detailed information about modules to help pinpoint areas of potential instability during testing and maintenance.

Complexity is realized as one of the as key quality indicator; it has the major impact on software quality attributes including efficiency, reliability and testability. Complexity is obviously relevant to the context of software testability. Testability is an external software attribute that assesses complexity and the required testing effort. Researchers and Practitioners advocated that complexity aspect of software is highly significant for developing quality software. Literature survey reveals that there are various aspects of software, including complexity factor that either directly or indirectly influence testability of software [4][5][6][7][8][9].

The organization of the paper is summarized as follows. Section 1 illustrates the significance of software complexity Section 2 briefly describes the model- $CEM^{OOD.}$ Section 3 validates authors' claims. Section 4 presents the empirical validation of the model and its significance. Section 5 concludes the paper.

II. MODEL DESCRIPTION

In most of the studies, Researchers examine the impacts OO software characteristics and successfully established their relationships with quality factors. In this paper, we examine and assessed their impact on the particular aspect/attribute i.e. Complexity. In order to establish a contextual relationship between complexity and the OO characteristics, their influence on complexity factor was examined. It was observed that each of these characteristics, either positively or negatively affect complexity of a software. To establish the model, multiple linear regression technique was used and the model for complexity quantification was formulated as given below:

$$\text{Complexity} = \alpha_0 + \beta_1 * \text{Copupling} + \beta_2 * \text{Cohesion} + \beta_3 * \text{Inheritance} \quad (1)$$

Where $\beta_1, \beta_2,$ are the coefficients of respective variables while α_0 is an intercept.

The data used for developing model is taken from [1] which consists of six industrial projects with around 10 to 20 classes. The values of variables 'Coupling Metrics (CPM), Cohesion Metrics (COM), and Inheritance Metrics (INM)' from the data set are used in MATLAB to compute the coefficients and thus, Complexity quantification model is formulated as given below.

$$\text{Complexity} = 90.8488 + 10.5849 * \text{Coupling} - 102.7527 * \text{Cohesion} + 128.0856 * \text{Inheritance} \quad (2)$$

III. VALIDATING THE CLAIMS

The applications [3] are used for validating Complexity quantification model in Eq. (2). We labelled the applications as System A, System B, System C and System D. All the systems are commercial software implemented in C++ and the

number of classes associated with them is 6, 5, 6 and 10 respectively. The correlations summary of between design constructs and the complexity is given in the table I below.

Table I
Correlation Analysis Summary

	Complexity ^ Coupling	Complexity ^ Cohesion	Complexity ^ Inheritance
System A	.77	.99	.58
System B	.87	.99	.91
System C	.98	.99	.77
System D	.39	.48	.67

Table I summarizes the results of the correlation analysis for Complexity model, and shows that for all the System, all of the design constructs are highly correlated with Complexity. It strongly indicates the higher significance of considering these constructs for estimation complexity.

In order to further justify another claim that complexity is one of the important factors of testability, Hypothesis testing is performed to test this significance (r-correlation coefficient) by using the formula as under:

$$t_r = \frac{r\sqrt{N-2}}{\sqrt{1-r^2}}$$

With *N-2* degree of freedom, a coefficient of correlation is judged as statistically significant when the *t* value equals or exceeds the *t* critical value in the *t* distribution table.

$H_{0(T\wedge C)}$: Testability and Complexity are not highly correlated.
$H_{1(T\wedge C)}$: Testability and Complexity are highly correlated.

Table II
Correlation Coefficient Test for Testability and Complexity

	System A	System B	System C	System D
Testability ^ Complexity	.99	.99	.99	.84
t_r	14.03	12.15	14.03	4.37
$t_{r\text{-Critical Value}}$	2.447	2.571	2.447	2.228
t_r>2.44	√	√	√	√
$H_{0(T\wedge C)}$	Reject	Reject	Reject	Reject

Using two-tailed test at the .05 level with different degrees of freedom, it is evident from the tables II that null hypothesis is rejected for all the Four Systems. Hence, therefore, author's claim of considering Complexity as one of the key factors of Testability is statistically justified.

IV. EMPIRICAL VALIDATION

This section assesses how well the model- CEM^{OOD} is able to quantify the complexity of the software using design constructs, hence the quality of an object oriented software design. The internal characteristic of a design varies significantly with scope or boundary of domain and its context. These characteristics positively or negatively influence software complexity. As a result, the overall quality gets affected. Complexity estimated by the model-CEM^{OOD} is validated using tryout data, followed by statistical analysis and interpretation. In order to validate the model, data set [3] is used; the known testability rating for the given projects (P_1-P_{10}) is shown in table III.

Table III
Given Complexity Rating

	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	P_9	P_{10}
Complexity Rating	2	3	4	1	5	7	10	8	6	9

Using the same set of data for the given projects (P_1-P_{10}), Complexity was computed using the proposed Complexity Model-CEM^{OOD}, and Computed Complexity using the model is then ranked on the basis of the results that are shown in table IV.

Table IV
Computed Testability Rating

	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	P_9	P_{10}
Complexity Rating	1	3	5	6	7	9	8	10	2	4

Table IV summarizes the Known Testability Ranking and Computed Testability Ranking using the model-CEM^{OOD} and their rank correlations. Speraman's Rank Correlation coefficient r_s is used to test the significance of correlation between calculated values of complexity using the model-CEM^{OOD} and known Complexity ranking. r_s is computed using the formula given as under:

$$r_s = 1 - \frac{6\sum d^2}{n(n^2-1)} \qquad -1.0 \le r_s \le +1.0$$

Where'd' is the difference between the ranks of calculated values of Complexity using model and the known value, and n is the number of Projects (n=10) used in the experiment.

Table IV
Computed Ranking, Known Ranking and their Correlations

Projects → / Complexity Ranking	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	P_9	P_{10}
Computed Ranking	2	3	4	1	5	7	10	8	6	9
Known Ranking	1	3	5	6	7	9	8	10	2	4
$\sum d^2$	1	0	1	25	4	4	4	4	16	25
r_s	.99	1	.99	.84	.97	.97	.97	.97	.89	.84
r_s >.781	√	√	√	√	√	√	√	√	√	√

The correlation values between Complexity using Model CEM^{OOD} and Known Ranking are shown in table IV. Pairs of these values with correlation values r_s above [±.781] are checked in the table above. The correlation is acceptable with high degree of confidence, i.e. at the 99%. Therefore, we can conclude without any loss of generality that Complexity Model CEM^{OOD} estimates are reliable and valid in the context. However, the study needs to be standardized with a larger experimental tryout for better acceptability and utility.

V. CONCLUSIONS

The paper highlighted the importance of software complexity. Complexity is obviously relevant to the context of software testability and plays a highly significant role for delivering quality software. It has been shown that model-CEM^{OOD} is able to quantify the complexity of the software design using these constructs. Hence, therefore, the model has been validated theoretically as well as empirically using experimental try-out. However, the model is validated on a small data set and it is to be done further on live industrial projects for better acceptability and utility.

REFERENCES

[1] Khan, R.A., Mustafa, K.: Metric based Testability Model for Object Oriented Design (MTMOOD). SIGSOFT Software Engineering Notes 34(2), March 2009.

[2] Nazir M., Khan R. A., and Mustafa K. "Complexity Quantification Model: A Metric-Based Approach", 4th IEEE International Conference on Advanced Computing & Communication Technologies, Panipat, October 30, 2010.

[3] Genero M., J. Olivas, M. Piattini and F. Romero, "A Controlled Experiment for Corroborating the Usefulness of Class Diagram Metrics at the early phases of Object Oriented Developments", Proceedings of ADIS 2001, Workshop on decision support in Software Engineering, 2001.

[4] S. Mouchawrab, L. C. Briand, and Y. Labiche, "A measurement framework for object-oriented software testability", Information and Software Technology, Volume 47, Issue 15, pp. 979-997, Dec. 2005.

[5] Bruce W.N.Lo and Haifeng Shi, "A preliminary testability model for object-oriented software", in Proc. International Conf. on Software Engineering, Education, Practice, IEEE 1998, pp. 330-337.

[6] E. Mulo, "Design for Testability in Software Systems", Department of Software Technology, Faculty EEMCS, Delft University of Technology, Netherlands, 2007.

[7] J. Gao, S. Ming-Chih, "A Component Testability Model for Verification and Measurement", In Proceedings of the 29th Annual International Computer Software and Applications Conference, pages 211–218. IEEE Computer Society (2005).

[8] G. Jimenez, S. Taj, and J. Weaver, "Design for Testability" in the Proceedings of the 9th Annual NCIIA Conference, 2005.
S. Jungmayr, "Testability during Design", Softwaretechnik-Trends, Proceedings of the GI Working Group Test, Analysis and Verification of Software, Potsdam, pp. 10-11, 2002.

