

Reliability Quantification Model

Aradhana Yadav

Research Scholar, Department of Computer Science, Sai Nath University, Ranchi.

Dr. Birendra Goswami

Professor, Department of Computer Science, Sai Nath University, Ranchi.

Abstract

Measuring Reliability is a crucial requirement for software program engineers at every stage of the manner, however mainly in the design stage. Designers can make improvements to their designs earlier than coding starts by evaluating Reliability early within the design method. Specialists often propose scheduling Reliability evaluation for early in the layout method. We are gifted with a Reliability Quantification version (RQOOD) because of this bankruptcy that functions within the design level. Additionally, statistical exams are run to validate the relationship between functionality, Understandability, and Reliability. The developed model was validated by way of experimentation. Ultimately, an empirical validation of the Reliability Quantification version is done.

MODEL DEVELOPMENT

The Reliability Quantification version RQMOOD was advanced the usage of the Reliability Quantification framework RQMOOD as defined underneath [1,2,3,5]

- 1) Understandability and capability had been located to be the two maximum essential aspects of Reliability.

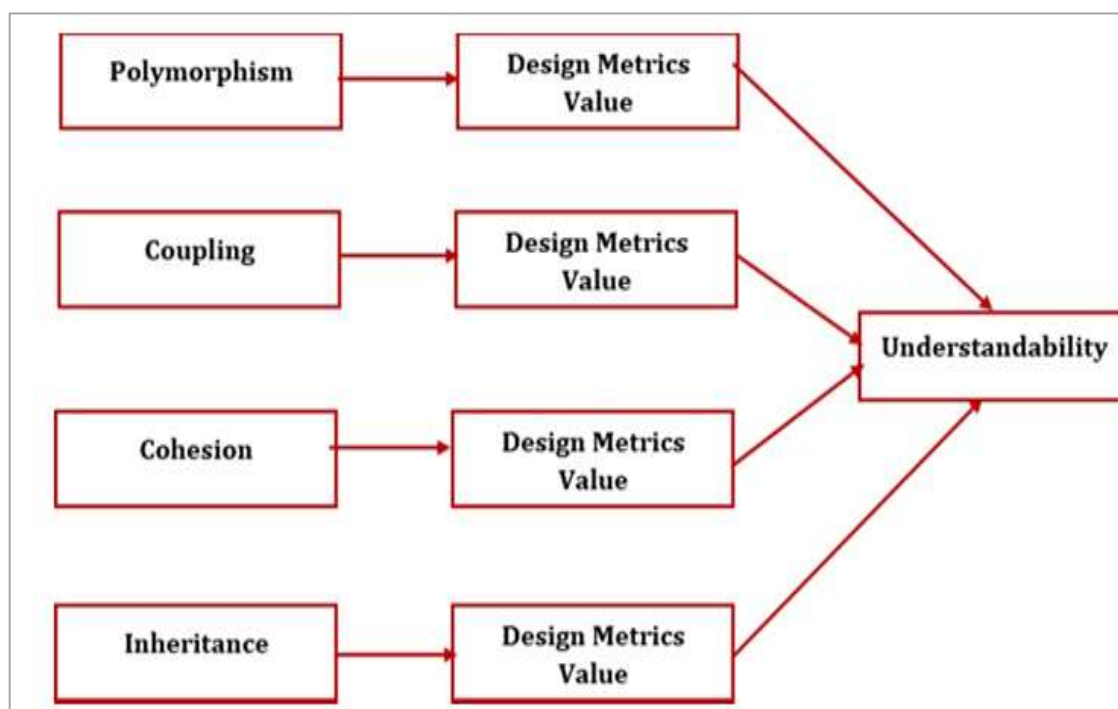


Figure. 1: Mapping among Understandability, Object Oriented Design Properties and Metrics [4]

2) Functions together with coherence, inheritance, coupling, and capability in the form of coupling, inheritance, and encapsulation were found to be associated with the Understandability of an object-oriented design. [6,7,8]

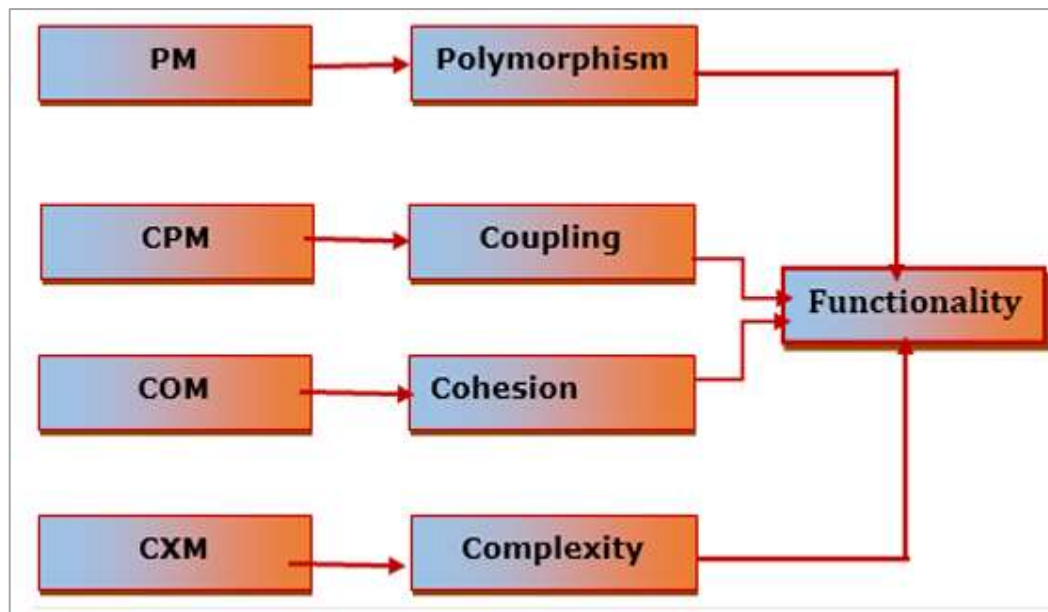


Figure. 2: Connecting Object-Oriented Design Properties and Metrics with Functionality

- 3) The design estimates for the plan homes—the most depth of the inheritance tree (Maxi), the number of strategies (NM), the associations (Nasik), and the attributes (NA)—have been mounted [9].
- 4) The statistical correlation between the Reliability factors and the diagnosed design metrics and characteristics become discussed and evidence of this correlation changed into furnished.
- 5) This bankruptcy indicates the relationship among the decided Reliability coefficients and Reliability, and how they can be used to estimate Reliability.

The relationship among Reliability and layout traits along with capability and Understandability, and design metrics. So that you can installation a version for Reliability Quantification, a multiple Linear Regression approach has been used [10,11,12].

$$\text{Reliability} = \alpha_0 + \beta_1 \times \text{Understandability} + \beta_2 \times \text{Functionality} \quad \text{Eq. (1)}$$

Equation 6.2 uses the data (P1, P2, P3, P4, and P5) from Appendix II Table II.1 to generate the Reliability model. The parameters that were employed were "MO difference and Functionality" and "encapsulation metric, inheritance metric, coupling metric, coupling metric".

Equation (6.2) illustrates how the Reliability Quantification model is put together, and the correlation coefficients were computed using SPSS.

$$\text{Reliability} = - .653 + 3.154 \times \text{Understandability} - 2.133 \times \text{Functionality} \quad \text{Eq. (2)}$$

The coefficients of the Reliability Quantification model are shown in Table 6.1. The values required to create the regression equation can be found in the "Unstandardized Coefficients" section of the output. 6.2). The standardized beta coefficient gives an indication of how each variable affects the Reliability model. Determining Reliability measures for software designs can be greatly aided by experimental Reliability evaluation.

Table 1: Reliability Quantification Model Coefficients

Coefficients ^a						
Model		Unstandardized Coefficients		Standardized Coefficients	t	Sig.
		B	Std. Error	Beta		
1	(Constant)	-.653	2.115		-.150	.726
	Understandability	3.154	.450	.525	1.097	.271
	Functionality	-2.133	.079	-.671	-2.914	.296
a. Dependent Variable: Reliability						

Table 2: Displays the Output's Descriptive Statistics. It Displays the Independent and Dependent Variable Mean and Standard Deviation

	Mean	Std. Deviation
Reliability	4.1411	.97104
Understandability	4.9423	.60112
Functionality	12.6100	2.99635

When performing multiple regression, the outline of the Reliability Quantification model given in Table 3 is very useful. The multiple correlation coefficient, marked 104 1 with a capital R, indicates the extent to which a dependent variable is related to several independent variables'-squared is very useful.

Table 3: Reliability Quantification Model Summary

Model Summary				
Model	R	R Square	Adjusted R Square	Std. Error of the Estimate
1	.967 ^a	.736	.569	.46211
a. Predictors: (Constant), Functionality, Understandability				

SUMMARY

This paper considers the impact of key software Reliability elements on measuring and enhancing Reliability throughout the design segment. The two most critical elements influencing the Reliability of item-oriented designs have been considered: Understandability and capability. Quantification model for item-oriented designs (RQMOOD) evolved thinking A about both key factors and 153 conclusions and tested to have Reliability statistical a higher degree of recognition. The evaluation demonstrates that the recommended model, RQMOOD, has a more potent association with the Reliability ratings provided by is capable of reliably calculating Reliability. Through experimental checking out, the Reliability Quantification version has been Reliability version's application validation empirically demonstrated. demonstrates how reliable, ideal, and relatively consistent the counseled version is the following bankruptcy discusses this end in addition to associated work.

REFERENCES

1. M. A. Alshayeb *et al.*, “Data-oriented QMOOD model for quality assessment of multi-client software applications,” *Engineering Science and Technology, an International Journal*, vol. 51, p. 101660, 2024.
DOI: <https://doi.org/10.1016/j.jestch.2024.101660>
2. M. Singh and J. K. Chhabra, “Machine learning based improved cross-project software defect prediction using new structural features in object oriented software,” *Applied Soft Computing*, vol. 165, p. 112082, 2024.
DOI: <https://doi.org/10.1016/j.asoc.2024.112082>
3. B. Battulga, L. Tsodol, E. Dovdon, N. Bold, and O.-E. Namsrai, “Metric-based defect prediction from class diagram,” *Array*, vol. 27, p. 100438, 2025.
DOI: <https://doi.org/10.1016/j.array.2025.100438>
4. S. Rochimah, T. R. Hadiningrum, B. D. Mardiana, D. O. Siahaan, R. J. Akbar, and A. M. Shiddiqi, “A Maintainability Framework to Ensure the Software Quality in Object-Oriented Programming,” *IEEE Access*, vol. 13, pp. 195796–195821, 2025.
DOI: <https://doi.org/10.1109/ACCESS.2025.3633265>
5. G. Yenduri and N. Veeranjanyulu, “An empirical analysis of software maintainability metrics: object-oriented approach versus traditional,” *International Journal of Advanced Intelligence Paradigms*, vol. 30, no. 5, pp. 419–432, 2025.
DOI: <https://doi.org/10.1504/IJAIP.2025.149746>
6. N. A. A. Khleel and K. Nehéz, “Improving accuracy of code smells detection using machine learning with data balancing techniques,” *The Journal of Supercomputing*, vol. 80, pp. 21048–21093, 2024.
DOI: <https://doi.org/10.1007/s11227-024-06265-9>
7. L. Rumbutis, A. Slotkienė, and B. Pliuskuvienė, “Identification of software quality attributes from code defect prediction: A systematic literature review,” *New Trends in Computer Sciences*, vol. 2, no. 1, pp. 57–68, 2024.
DOI: <https://doi.org/10.3846/ntcs.2024.21305>

8. Y. Zhou, X. Liu, Z. Guo, Y. Zhou, C. Zhang, and J. Qian, “Deep learning or classical machine learning? An empirical study on line-level software defect prediction,” *Journal of Software: Evolution and Process*, vol. 36, no. 10, p. e2696, 2024.
DOI: <https://doi.org/10.1002/smr.2696>
9. O. P. Omondiagbe, S. A. Licorish, and S. G. MacDonell, “Improving transfer learning for software cross-project defect prediction,” *Applied Intelligence*, vol. 54, pp. 5593–5616, 2024.
DOI: <https://doi.org/10.1007/s10489-024-05459-1>
10. A. Ali, S. S. H. Rizvi, and S. H. Adil, “Enhancing Software Quality with AI: A Transformer-Based Approach for Code Smell Detection,” *Applied Sciences*, vol. 15, no. 8, p. 4559, 2025.
DOI: <https://doi.org/10.3390/app15084559>
11. M. Borg, M. Ezzouhri, and A. Tornhill, “Ghost Echoes Revealed: Benchmarking Maintainability Metrics and Machine Learning Predictions Against Human Assessments,” in *Proc. IEEE Int. Conf. Software Maintenance and Evolution (ICSME)*, 2024.
DOI: <https://doi.org/10.1109/ICSME58944.2024.00072>
12. “An Empirical Study on Maintainability Index of Software Design Patterns,” in *Proc. IEEE/ACIS 23rd Int. Conf. Software Engineering Research, Management and Applications (SERA)*, 2025.
DOI: <https://doi.org/10.1109/SERA65747.2025.11154571>