International Journal On Engineering Technology and Sciences – IJETS™
ISSN(P): 2349-3968, ISSN (O): 2349-3976
Volume V, Issue XII, December - 2018

Software Defect Prediction in Object-Oriented Systems: A Systematic Review

Dr. Ashwini Sharma
Assistant Professor
Government Engineering College Jhalawar, India
ashwini.sharma.in@gmail.com

Abstract

Software defect prediction plays a crucial role in improving software quality and reliability by enabling early identification of fault-prone components. In object-oriented software systems, design-level attributes such as coupling, cohesion, inheritance, and complexity significantly influence defect occurrence. This paper presents an expanded and comprehensive survey of object-oriented software defect prediction. The study systematically reviews object-oriented metrics, traditional statistical models, machine learning-based prediction techniques, and optimization-driven hybrid approaches. Furthermore, it analyzes commonly used datasets, validation strategies, and performance evaluation measures adopted in prior studies. By synthesizing more than two decades of research, this survey highlights dominant trends, methodological limitations, and unresolved challenges. The paper aims to serve as a structured and in-depth reference for researchers and practitioners seeking to design robust and generalizable defect prediction models for object-oriented software systems.

Keywords: Software Defect Prediction, Object-Oriented Metrics, CK Metrics, Machine Learning, Hybrid Models, Software Quality

1. Introduction

Software systems are increasingly becoming large, complex, and highly distributed, making quality assurance a challenging and resource-intensive activity. Defects introduced during early development phases often propagate to later stages, significantly increasing maintenance cost and effort. Software defect prediction (SDP) addresses this challenge by identifying fault-prone modules early in the software development life cycle, thereby enabling focused testing, inspection, and refactoring activities.

Object-oriented programming (OOP) has become the dominant paradigm for modern software development due to its support for abstraction, modularity, reuse, and maintainability. However, object-oriented features such as inheritance, polymorphism, and dynamic binding introduce new forms of complexity that are not adequately captured by traditional size- or control-flow-based metrics. This limitation motivated the development of object-oriented (OO) metrics that explicitly measure design characteristics influencing software quality.

Among the proposed OO metrics, the Chidamber and Kemerer (CK) metric suite laid the foundation for empirical defect prediction research. Since its introduction, numerous studies have investigated the relationship between OO metrics and defect proneness using statistical analysis, machine

learning, and hybrid approaches. Over time, research focus has shifted from simple regression-based models to sophisticated ensemble and optimization-driven techniques. This survey aims to consolidate and critically analyze these developments, providing both historical context and future research directions.

1.1 Survey Contributions

The main contributions of this survey are summarized as follows:

- A systematic and comprehensive review of object-oriented software defect prediction studies published up to 2018.
- A structured classification of defect prediction approaches into statistical, machine learning, and hybrid models.
- A comparative analysis of object-oriented metrics, datasets, and evaluation practices used in prior research.
- An evidence-based discussion of empirical findings, limitations, and open research challenges.
- A conceptual future research framework emphasizing hybrid modeling, explainability, and industrial applicability.

2. Research Methodology

This study follows a systematic literature review (SLR) methodology to ensure transparency, reproducibility, and comprehensive coverage of object-oriented software defect prediction research. The review process was designed in accordance with established guidelines for evidence-based software engineering.

2.1 Research Questions

The survey addresses the following research questions (RQs):

- **RQ1:** Which object-oriented metrics are most frequently used for software defect prediction?
- **RQ2:** What prediction techniques (statistical, machine learning, and hybrid) dominate the literature?
- **RQ3:** Which datasets and evaluation measures are commonly adopted?
- **RQ4:** What limitations and research gaps are reported in existing studies?

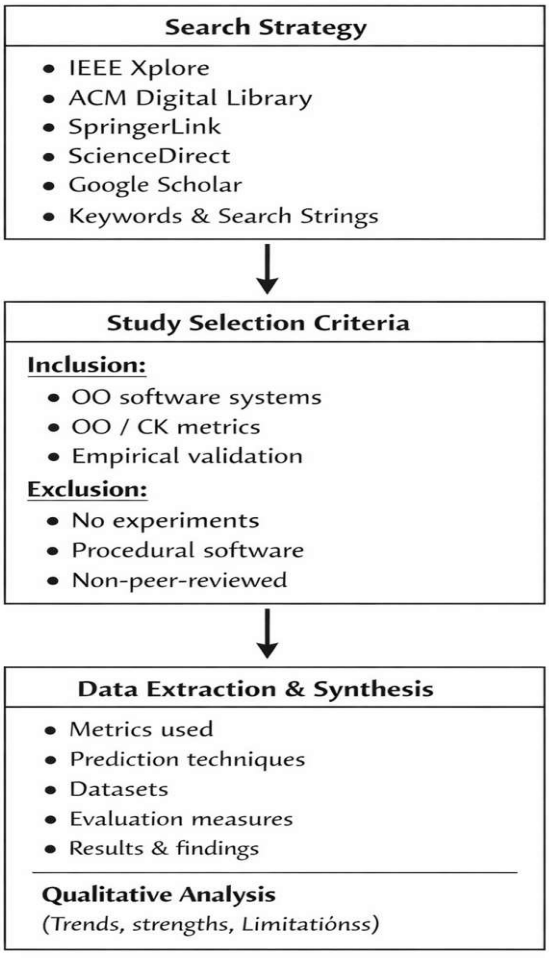

Fig. 1. Systematic Review Process

3. Object-Oriented Metrics

Object-oriented metrics quantify internal design properties of classes, packages, and subsystems that are believed to influence fault occurrence. The CK metric suite—Weighted Methods per Class (WMC), Coupling Between Objects (CBO), Response for a Class (RFC), Lack of Cohesion of Methods (LCOM), Depth of Inheritance Tree (DIT), and Number of Children (NOC)—remains the most extensively used set of OO metrics in defect prediction research [1], [2].

Empirical studies consistently report that higher values of coupling, complexity, and response size are associated with increased defect density [4], [5], [12]. In particular, CBO and RFC have been identified as strong predictors of defect proneness due to their close relationship with change propagation and testing effort [3], [14]. Cohesion-related metrics such as LCOM provide insight into design quality and maintainability, while inheritance-related metrics (DIT and NOC) show mixed results depending on application domain and system size [15].

To complement CK metrics, researchers proposed additional metric families, including size metrics, information flow metrics [22], package-level metrics [20], and design quality indices [17]. These extended metric sets aim to capture architectural complexity and modularization aspects, thereby improving prediction performance. Despite their widespread adoption, concerns regarding metric redundancy, multicollinearity, and theoretical validity continue to motivate ongoing research [6].

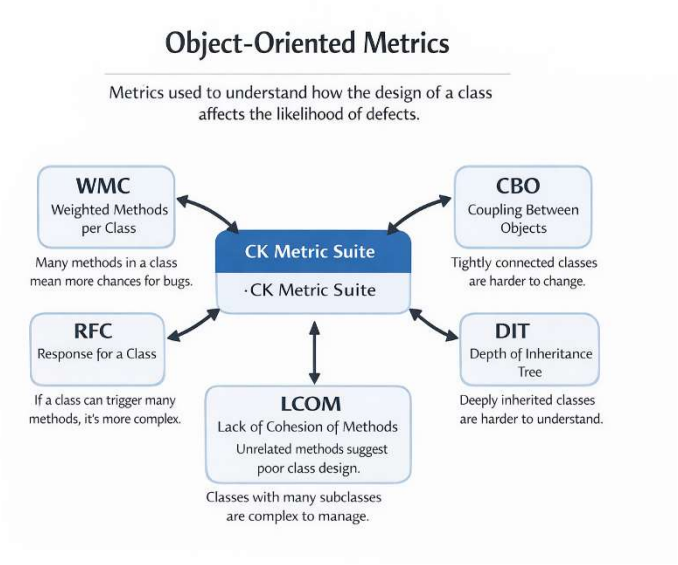

Fig. 2. Object Oriented Metrics

Table 1. Common Object-Oriented Metrics and Their Interpretation

Metric	Description	Impact on Defects
WMC	Number/complexity of methods	High WMC → higher fault risk
CBO	Degree of class coupling	Strong predictor of defects
RFC	Number of methods invoked	Increases testing complexity
LCOM	Measure of cohesion	Low cohesion → more defects
DIT	Inheritance depth	Mixed empirical evidence
NOC	Number of subclasses	Context-dependent impact

4. Empirical and Statistical Models

Early defect prediction studies primarily relied on statistical modeling techniques such as linear regression, logistic regression, and discriminant analysis [2], [15]. These models were used to establish quantitative relationships between OO metrics and defect occurrence, offering interpretability and theoretical transparency.

Several empirical investigations demonstrated statistically significant correlations between CK metrics and fault-proneness [4], [5]. However, these models often assumed linear relationships and independence among predictors—assumptions that rarely hold in real-world software systems. Additionally, their performance degraded in the presence of noisy, imbalanced, and high-dimensional datasets [6].

Despite these limitations, statistical models continue to serve as baseline approaches and are frequently combined with modern learning techniques to enhance robustness and interpretability.

Table 2. Statistical Defect Prediction Models

Model	Strengths	Limitations
Linear Regression	Simple, interpretable	Cannot model nonlinearity
Logistic Regression	Probabilistic output	Sensitive to imbalance
Discriminant Analysis	Class separation	Strong assumptions

5. Machine Learning Approaches

Machine learning (ML) techniques have gained significant attention in defect prediction due to their ability to model complex, nonlinear relationships among software metrics [9], [10]. Commonly used classifiers include decision trees, support vector machines, k-nearest neighbors, artificial neural networks, Naïve Bayes, and random forests.

Comparative studies indicate that ensemble methods such as random forests and boosting algorithms consistently outperform single classifiers, particularly in imbalanced defect datasets [26]. ML-based models demonstrate improved predictive accuracy and recall but often suffer from reduced interpretability, which limits their adoption in industrial settings [6].

Recent research emphasizes the importance of data preprocessing techniques, including feature selection, normalization, and resampling, to address class imbalance and improve model stability [18]. Nevertheless, the lack of standardized experimental protocols remains a major challenge in comparing ML-based defect prediction studies.

Table 3. Machine Learning Techniques for Defect Prediction

Technique	Advantages	Drawbacks
Decision Tree	Explainable	Overfitting
SVM	High accuracy	Parameter sensitive
Neural Networks	Nonlinear modeling	Low interpretability
Random Forest	Robust, accurate	Complex models

6. Optimization and Hybrid Models

To overcome the limitations of standalone prediction techniques, researchers proposed hybrid frameworks that integrate statistical analysis, machine learning, and optimization algorithms. Metaheuristic optimization methods such as genetic algorithms, particle swarm optimization, and ant colony optimization are commonly used for feature selection, parameter tuning, and model calibration.

Hybrid models demonstrate improved robustness and cross-project performance by reducing noise, eliminating irrelevant metrics, and enhancing generalization. However, increased computational complexity and reduced transparency remain key concerns, particularly for large-scale industrial applications.

7. Datasets and Evaluation Practices

Empirical validation of defect prediction models predominantly relies on publicly available datasets such as NASA MDP, PROMISE, and Mozilla repositories [14], [26]. These datasets provide labeled metric-defect pairs that enable reproducible experimentation. However, heavy reliance on a limited number of datasets raises concerns regarding external validity.

Performance evaluation metrics commonly include accuracy, precision, recall, F-measure, and ROC-AUC [18]. Recent studies advocate cost-sensitive measures and cross-project validation to better reflect real-world deployment scenarios. Despite these efforts, inconsistencies in evaluation practices hinder meaningful comparison across studies.

Table 4. Common Defect Prediction Datasets

Dataset	Type	Usage
NASA MDP	Industrial	Early empirical studies
PROMISE	Open-source	Benchmarking
Mozilla	Open-source	Large-scale validation

8. Research Gaps and Future Directions

Despite extensive research on object-oriented defect prediction, several gaps remain. Most existing studies rely heavily on static design metrics, which fail to capture runtime behavior and evolution-related characteristics. Dataset dependency and imbalance issues further limit generalizability, while the black-box nature of many machine learning models hinders industrial adoption.

Future research should focus on integrating static, dynamic, and process metrics within unified prediction frameworks. Explainable artificial intelligence (XAI) techniques can enhance model transparency and practitioner trust. Additionally, large-scale industrial validation and cross-project prediction remain critical areas for further exploration.

8.1Future Research Framework

To address the identified gaps, this paper proposes a conceptual future research framework for object-oriented defect prediction. The framework integrates multiple data sources, advanced learning techniques,

and explainability mechanisms to improve robustness and practical usability.

Framework Description:

1. **Data Sources:** Static object-oriented metrics, software process metrics, and dynamic runtime metrics collected from heterogeneous repositories.
2. **Preprocessing Layer:** Feature selection, normalization, noise filtering, and class imbalance handling.
3. **Prediction Engine:** Hybrid defect prediction models combining machine learning algorithms with optimization techniques.
4. **Explainability Layer:** Explainable artificial intelligence (XAI) methods to interpret prediction outcomes and identify critical defect drivers.
5. **Feedback Loop:** Continuous learning mechanism where prediction results guide refactoring and model refinement.

The proposed framework highlights a shift toward adaptive, interpretable, and industry-oriented defect prediction systems.

Several open challenges persist in object-oriented defect prediction research. These include over-reliance on static design metrics, limited interpretability of ML models, dataset bias, and insufficient industrial validation. Future research should focus on integrating dynamic and process metrics, developing explainable prediction models, and conducting large-scale industrial case studies.

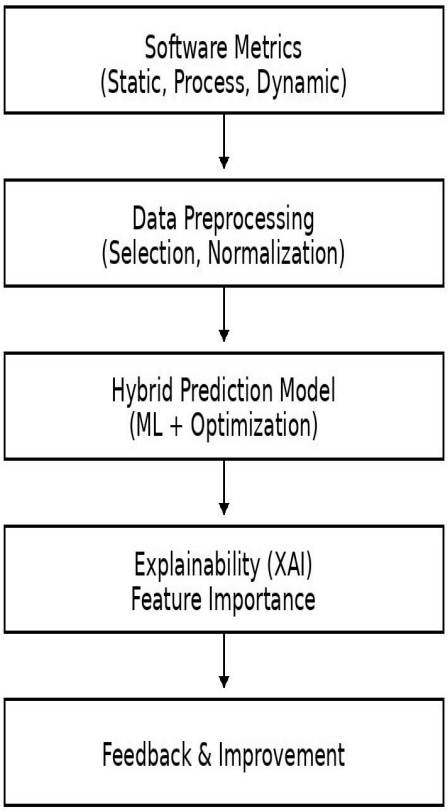

Fig. 3. Conceptual future research framework for object-oriented software defect prediction.

9. Conclusion

This expanded survey provides a comprehensive synthesis of object-oriented software defect prediction research published up to 2018. The analysis confirms the continued relevance of OO metrics while highlighting the growing dominance of machine learning and hybrid approaches. The survey also emphasizes that no single technique universally outperforms others across all datasets, underscoring the need for context-aware and adaptable prediction frameworks. Overall, this work offers a consolidated reference and a foundation for advancing future research in software defect prediction.

Table 5. Comparative Results of Defect Prediction Techniques from Literature

Study	Year	Data set	Technique	Performance Highlights
Basili et al. [2]	1996	NASA	Logistic Regression	CK metrics significantly correlated with defects
Subramanyam& Krishnan [4]	2003	Industrial	Regression	CBO and WMC strong predictors
Gyimóthy et al. [5]	2005	Mozilla	Decision Trees	High recall for fault-prone classes
Jureczko [14]	2011	PROMISE	SVM	Improved accuracy over regression
Radjenović et al. [26]	2013	Multi-project	Ensemble Models	Consistent ROC-AUC improvement
Wahono [10]	2015	PROMISE	Random Forest	Best overall prediction performance

References

[1] S. R. Chidamber and C. F. Kemerer, "A metrics suite for object-oriented design," *Proc. OOPSLA*, pp. 197–211, 1994.

[2] V. R. Basili, L. C. Briand, and W. L. Melo, "A validation of object-oriented design metrics as quality indicators," *IEEE Trans. Softw. Eng.*, vol. 22, no. 10, pp. 751–761, Oct. 1996.

[3] L. C. Briand, S. Morasca, and V. R. Basili, "Property-based software engineering measurement," *IEEE Trans. Softw. Eng.*, vol. 22, no. 1, pp. 68–86, Jan. 1996.

[4] R. Subramanyam and M. S. Krishnan, "Empirical analysis of CK metrics for object-oriented design complexity," *IEEE Trans. Softw. Eng.*, vol. 29, no. 4, pp. 297–310, Apr. 2003.

[5] T. Gyimóthy, R. Ferenc, and I. Siket, "Empirical validation of object-oriented metrics on open source software for fault prediction," *IEEE Trans. Softw. Eng.*, vol. 31, no. 10, pp. 897–910, Oct. 2005.

[6] N. E. Fenton and M. Neil, "A critique of software defect prediction models," *IEEE Trans. Softw. Eng.*, vol. 25, no. 5, pp. 675–689, Sept. 1999.

[7] N. Nagappan and B. Ball, "Using software dependencies and churn metrics to predict field failures," *Proc. ICSE*, pp. 364–373, 2005.

[8] International Software Testing Qualifications Board, "Software fault prediction metrics: A review," Tech. Rep., 2013.

[9] B. Isong and O. Ekabua, "State-of-the-art software defect prediction: A survey," *Int. J. Comput. Appl.*, vol. 145, no. 6, pp. 1–10, 2016.

[10] R. S. Wahono, "A systematic literature review of software defect prediction," *J. Softw. Eng.*, vol. 1, no. 1, pp. 1–16, 2015.

[11] P. U. Pooja and P. K. N. Banu, "Object-oriented metrics for defect prediction: An empirical study," *Int. J. Adv. Comput. Sci.*, vol. 9, no. 3, pp. 210–218, 2018.

[12] S. Singh, S. Kaur, and R. Malhotra, "Empirical analysis of design metrics for fault proneness," *ACM SIGSOFT Notes*, vol. 32, no. 1, pp. 1–6, 2007.

[13] "Analytical study of object-oriented metrics," *Int. J. Eng. Trends Technol.*, vol. 4, no. 3, pp. 897–902, 2013.

[14] M. Jureczko, "Significance of software metrics in defect prediction," *Proc. PROMISE*, pp. 1–10, 2011.

[15] M. H. Tang, M. H. Kao, and M. H. Chen, "An empirical study on object-oriented metrics," *IEEE Trans. Softw. Eng.*, vol. 25, no. 2, pp. 1–14, 1999.

[16] W. W. Agresti, W. M. Evanco, and J. M. Dey, "Distance-based software measurement," *IEEE Softw.*, vol. 17, no. 2, pp. 1–7, 2000.

[17] A. Bansiya and C. G. Davis, "A hierarchical model for object-oriented design quality assessment," *IEEE Trans. Softw. Eng.*, vol. 28, no. 1, pp. 4–17, Jan. 2002.

[18] R. Manjula and S. Florence, "A survey on defect prediction in software engineering," *Proc. Int. Conf. Comput. Commun.*, pp. 1–6, 2016.

[19] A. Kaur and D. Chopra, "Entropy-based metrics for defect prediction," *Entropy*, vol. 20, no. 2, pp. 1–18, 2018.

[20] Y. Zhao, H. Liu, and J. Li, "Package-level modularization metrics for fault prediction," *J. Syst. Softw.*, vol. 102, pp. 1–14, 2015.

[21] T. J. McCabe, "A complexity measure," *IEEE Trans. Softw. Eng.*, vol. SE-2, no. 4, pp. 308–320, Dec. 1976.

[22] S. Henry and D. Kafura, "Information flow metrics for software development," *IEEE Trans. Softw. Eng.*, vol. SE-7, no. 5, pp. 510–518, Sept. 1981.

[23] M. Hitz and B. Montazeri, "Measuring coupling and cohesion in object-oriented systems," *Proc. Int. Symp. Appl. Corp. Comput.*, pp. 25–34, 1995.

[24] M. Lorenz and J. Kidd, *Object-Oriented Software Metrics*, Englewood Cliffs, NJ, USA: Prentice-Hall, 1994.

[25] N. E. Fenton and S. L. Pfleeger, *Software Metrics: A Rigorous and Practical Approach*, 2nd ed. Boston, MA, USA: PWS Publishing, 1997.

[26] D. Radjenović, M. Heričko, R. Torkar, and A. Živkovič, "Software fault prediction metrics: A systematic literature review," *Inf. Softw. Technol.*, vol. 55, no. 8, pp. 1397–1418, 2013.