METRIC DALAM MEMPREDIKSI KESALAHAN PADA SOFTWARE: SYSTEMATIC LITERATURE REVIEW

[1]Ikhlasul Aulia Rahman, [2]Muhammad Ariq Persada, [3]Yuni Sugiarti
[1,2,3] Sistem Informasi, Fakultas Sains dan Teknologi,
UIN Syarif Hidayatullah Jakarta,
Jl. Ir. H. Djuanda No. 95, Ciputat Timur, Tangerang Selatan 15412, Banten
Email: ikhlasul.rahman21@mhs.uinjkt.ac.id, ariq.persada21@mhs.uinjkt.ac.id, sugiarti@uinjkt.ac.id

ABSTRAK

Deteksi dini kesalahan pada *software* dapat menghemat waktu, biaya, dan mengurangi kompleksitas perangkat lunak. Tujuan dari penulisan *Systematic Literature Review* (SLR) ini adalah untuk mengidentifikasi *software metric* dan penerapannya dalam memprediksi kesalahan pada *software*. Metode yang digunakan dalam penulisan ini adalah *Systematic Literature Review* (SLR) dengan menggunakan *software* Publish and Perish. Hasil dari *Systematic Literature Review* ini menunjukkan metrik *Object-oriented* digunakan lebih dari sebagian dari artikel yang dipilih dibandingkan dengan metrik ukuran, kompleksitas, dan metrik proses, dengan Chidamber dan Kemerer (CK) yang paling sering digunakan. Metrik *Object-oriented* dilaporkan lebih berhasil dalam menemukan kesalahan dibandingkan dengan metrik ukuran dan kompleksitas. Sementara untuk metrik proses tampaknya lebih baik dalam memprediksi kesalahan ketika digabungkan dengan metrik ukuran.

Kata kunci**:** Prediksi kesalahan *software*, *software metric*, *systematic literature review*

ABSTRACT

Early detection in software of errors can save time, cost, and reduce software complexity. The goal of writing this Systematic Literature Review (SLR) was to identify software metrics and their application in predicting software errors. The method used in this writing was the Systematic Literature Review (SLR) using Publish and Perish software. The results of this Systematic Literature Review showed that Object-oriented metrics were used in more than a majority of the selected articles compared to size metrics, complexity metrics, and process metrics, with Chidamber and Kemerer (CK) being the most frequently used. Object-oriented metrics were reported to be more successful in detecting errors compared to size and complexity metrics. On the other hand, process metrics appeared to be better at predicting errors when combined with size metrics.

Keywords: Software fault prediction, software metric, systematic literature review

1 INTRODUCTION

Untuk menjalankan setiap proses secara efektif, manajemen memerlukan pengukuran, kuantifikasi, dan pemodelan yang tepat. Penggunaan metrik perangkat lunak akan memberikan dasar kuantitatif yang diperlukan untuk mengembangkan serta memvalidasi model proses pengembangan perangkat lunak. Dengan memanfaatkan metrik, dapat meningkatkan produktivitas dan kualitas dari perangkat lunak yang dihasilkan [1].

Dalam pengembangan *software*, terdapat standar kualitas fungsional dan non-fungsional yang harus dipatuhi oleh para pengembang untuk memastikan kualitas *software* [2]. Kesalahan dalam program perangkat lunak adalah kegagalan atau kekeliruan yang menghalangi program berfungsi dengan semestinya, seperti menghasilkan hasil yang tidak tepat [3]. Deteksi dini kesalahan dapat menghemat waktu, biaya, dan mengurangi kompleksitas perangkat lunak karena sebanding dengan pengujian [4]. Selain itu, prediksi kesalahan yang terlambat menyebar ke tahap pengembangan selanjutnya dan memperumit seluruh proses prediksi. Kehadiran kesalahan sangat

mempengaruhi kehandalan perangkat lunak, kualitas, dan biaya pemeliharaan [5]. Oleh karena itu, diperlukan pengukuran dalam memprediksi kesalahan pada perangkat lunak.

Terdapat banyak teknik yang dapat diterapkan untuk memproyeksikan modul *software* dari kemungkinan terjadinya kegagalan. Teknik atau metode yang paling umum digunakan adalah Bansiya dan Davis, Abreu dan Carapuca, Chidamber dan Kemerer (metrik CK), Lorenz dan Kidd, Halstead, McCabe dan teknik metrik lainnya. Dari beberapa teknik tersebut yang telah divalidasi hanya dalam sejumlah kecil studi. Terdapat perbedaan hasil yang dilaporkan dari penggunaan suatu metrik.

Tujuan dari penelitian *Systematic Literature Review* (SLR) ini adalah untuk mengidentifikasi metrik dalam memprediksi kesalahan pada *software*. Metode pengukuran yang paling umum digunakan diidentifikasi dan kemampuan untuk memprediksi kesalahan dinilai untuk menjawab pertanyaan tentang metode yang sesuai untuk memprediksi kesalahan. Hasil penelitian yang tersebar dikumpulkan untuk mencapai kesimpulan baru dan memberikan panduan bagi praktisi dan penelitian selanjutnya.

2 KAJIAN PUSTAKA

2.1 Software Metric

Software metric adalah faktor kunci untuk menentukan biaya, jadwal, dan upaya. Kesalahan dalam menentukan ukuran yang tepat dapat menyebabkan biaya yang berlebihan atau proyek yang tertunda [6]. Pengukuran dibagi menjadi dua jenis, yaitu pengukuran prediktor dan pengukuran kontrol. Pengukuran prediktor terkait dengan produk *software*, seperti panjang rata-rata identifier pada program, kompleksitas modul sistem, dan jumlah operasi serta atribut yang terkait dengan obyek pada suatu desain. Sedangkan pengukuran kontrol biasanya terkait dengan proses *software*, seperti waktu rata-rata dan usaha yang diperlukan untuk memperbaiki kesalahan yang dilaporkan [7].

2.2 Prediksi Kesalahan *Software*

Metrik prediksi cacat perangkat lunak adalah komponen paling penting dalam membangun model prediksi yang bertujuan untuk meningkatkan kualitas perangkat lunak dengan memprediksi sebanyak mungkin cacat perangkat lunak. Sebagian besar metrik prediksi cacat dapat dikategorikan menjadi metrik kode dan proses [8]. Metrik kode menunjukkan kompleksitas kode sumber, sedangkan metrik proses menunjukkan kompleksitas proses pengembangan. Karya menyediakan ulasan menyeluruh dari literatur tentang metrik prediksi cacat [9].

2.3 Systematic Literature Review

Systematic literature adalah evaluasi terencana dan menyeluruh mengenai temuan dari artikel penelitian. *Systematic literature* menerapkan pendekatan metodologis yang terstruktur dan jelas untuk mengidentifikasi, memilih, dan mengevaluasi secara kritis temuan penelitian yang sedang ditinjau. Dengan menggunakan pendekatan metodologis ini, *systematic literature* dapat menghasilkan data yang dapat direplikasi dan memberikan jawaban atas pertanyaan penelitian yang spesifik [10].

3 METODOLOGI

3.1 Obyek Penelitian

Obyek penelitian ini adalah *software metric* dalam memprediksi kesalahan pada *software*. Bagian kunci dari pengembangan perangkat lunak adalah memastikan bahwa perangkat lunak yang dibuat memiliki mutu yang lebih tinggi. Deteksi dini kesalahan dapat menghemat waktu, biaya, dan mengurangi kompleksitas perangkat lunak karena sebanding dengan pengujian. Sehingga perlu adanya pemahaman yang memadai tentang *metric* dalam memprediksi kesalahan pada *software*.

3.2 System Usability Scale (SUS) method

Tahapan *Systematic Literature Review* adalah klasifikasi, identifikasi, pengumpulan dan analisis *software metric* dalam memprediksi kesalahan pada *software*.

1. Research Question
   Dalam mencapai tujuan penelitian, terdapat 3 pertanyaan penelitian. Pertanyaan penelitian ini membantu mengumpulkan semua informasi yang diperlukan untuk menganalisis berbagai artikel jurnal. Pertanyaan penelitian antara lain:
   a) Apa *sofware metric* yang digunakan untuk memprediksi kesalahan?
   b) Apa validasi *sofware metric* yang digunakan untuk memprediksi kesalahan?
   c) Apakah *software metric* berguna untuk prediksi kesalahan?
2. Proses Pencarian
   Proses pencarian data yang dibutuhkan dalam penelitian ini telah ditentukan kata kunci sebelumnya yang digunakan untuk mencari *database.* Dalam proses pencarian, *database* akan dianalisis secara cermat untuk menemukan informasi yang berkaitan dengan topik penelitian. Hasil pencarian yang ditemukan kemudian akan diekstrak dan digunakan sebagai bahan untuk studi ini. Berikut *database* yang digunakan untuk untuk memudahkan pencarian jurnal diperlukan.
   a) IEEE
   b) Google Scholar
   c) Science Direct
   d) Springer Link
3. Kriteria Inklusi and Eksklusi
   Kriteria dan batasan *input* untuk menentukan apakah data layak atau tidak untuk digunakan dalam penelitian ini. Kriteria berikut memenuhi syarat dalam penelitian ini:
   a) Data yang digunakan hingga tahun 2023.
   b) Memilih hanya publikasi yang berfokus pada *software metric*.
   c) Hanya memilih artikel jurnal yang berfokus pada *software metric* dalam memprediksi kesalahan pada *software*.
   d) Data diperoleh dari *database* yang digunakan pencarian data dalam penelitian ini.
4. Ekstaksi Data
   Mengacu pada pertanyaan penelitian, informasi diambil untuk mengidentifikasi *software metric* dalam memprediksi kesalahan pada *software*. Setelah itu, setiap kandidat artikel jurnal dievaluasi dengan kriteria inklusi dan eksklusi untuk memastikan relevansinya. Kualitas penelitian yang dilaporkan juga dinilai dengan memeriksa keakuratan deskripsi teknis yang terdapat dalam setiap publikasi. Setelah tahap ini, terdapat 21 publikasi yang telah terpilih.

Tabel 1. Jumlah total artikel illmiah yang dipilih

IEEE	Google Scholar	Science Direct	Springer Link	Total artikel ilmiah
9	6	4	2	21

4 HASIL DAN DISKUSI

4.1 Hasil Proses Penelusuran dan Kriteria Inklusi dan Eksklusi

Hasil Proses Pencarian dan Kriteria Inklusi dan Eksklusi, terdapat 21 artikel yang telah melalui proses seleksi dengan kriteria data hingga tahun 2023 dan data yang digunakan hanya jurnal mengenai *software metric*. Pada tabel 2 dibawah ini merupakan artikel yang telah dipilih.

Tabel 2. Daftar judul artikel ilmiah

No.	Artikel Jurnal	Tahun
1.	Vovel Metrics—Novel Coupling Metrics for Improved Software Fault Prediction	2021

No.	Artikel Jurnal	Tahun
2.	Comparative Analysis of Classification Methods for Prediction Software Fault Proneness Using Process Metrics	2021
3.	A Comparison of Software Defect Prediction Metrics Using Data Mining Algorithms	2020
4.	Penerapan Teknik PSO Over Sampling Dan Adaboost J48 untuk Memprediksi Cacat Software	2020
5.	Penerapan PSO Over Sampling dan Adaboost Random Forest untuk Memprediksi Cacat Software	2020
6.	Mining Software Defects: Should We Consider Affected Releases?	2019
7.	Experimental Validation of Inheritance Metrics' Impact on Software Fault Prediction	2019
8.	A Study on Software Fault Prediction Techniques	2019
9.	Taxonomy of Machine Learning Algorithms in Software Fault Prediction Using Object Oriented Metrics	2018
10.	An Empirical Analysis of the Effectiveness of Software Metrics and Fault Prediction Model for Identifying Faulty Classes	2017
11.	Design of Software Fault Prediction Model Using BR Technique	2015
12.	Using Complexity, Coupling, and Cohesion Metrics as Early Indicators of Vulnerabilities	2011
13.	An Investigation of the Relationships Between Lines of Code and Defects	2009
14.	Empirical Analysis for Investigating the Effect ff Object-Oriented Metrics on Fault Proneness: A Replicated Case Study	2009
15.	A Replicated Quantitative Analysis of Fault Distributions in Complex Software Systems	2007
16.	Empirical Analysis of Software Fault Content and Fault Proneness Using Bayesian Methods	2007
17.	How good is your blind spot sampling policy	2004
18.	Metrics that Matter	2002
19.	Replicated Case Studies for Investigating Quality Factors in Object-Oriented Designs	2001
20.	Quantitative Analysis of Faults and Failures in a Complex Software System	2000
21.	Code Churn: A Measure for Estimating the Impact of Code Change	1998

4.2 Obyek Penelitian

Pada tahap ini data dianalisis dan hasilnya akan menjawab *Research Question* (RQ) yang telah ditentukan dan akan dibahas *metric* dalam memprediksi kesalahan pada *software*.

1. Hasil RQ1: *Sofware metric* yang digunakan untuk memprediksi kesalahan
   Metrik yang paling mudah dipahami, mudah diekstrak, dan umum digunakan, yaitu LOC (*Lines of Code*). Studi paling sederhana telah mengklasifikasikan modul sesuai dengan ukurannya untuk mengetahui apakah beberapa modul besar bertanggung jawab atas sebagian besar kesalahan. Metrik ini digunakan pada 6 [11] [12] [13] [14] [15] [16]. Sedangkan metrik kompleksitas yang populer, seperti kompleksitas siklomatik McCabe [17] [18] [19] dan metrik Halstead [20] [21], telah digunakan dalam beberapa penelitian. Metrik Chidamber-Kemerer adalah yang paling banyak digunakan dan paling sukses di antara metrik *Object-oriented* 9 [3] [22] [23] [24] [25] [26] [27] [28] [29]. Namun, kinerja setiap metrik Chidamber-Kemerer tidak selalu sama baiknya. Selain itu, studi dari [30], penggunaan metrik proses juga dapat memprediksi kesalahan dan memberikan hasil yang efektif.
2. Hasil RQ2: Validasi *software metric* yang digunakan untuk memprediksi kesalahan.

Bagian ini menyajikan hasil evaluasi metrik yang ditemukan yang ditemukan dalam studi yang dipilih. Studi-studi tersebut dikelompokkan menurut metrik perangkat lunak ke dalam ukuran, kompleksitas, OO, dan metrik proses. Keseluruhan efektivitas metrik dan efektivitas dalam hal ukuran dan bahasa pemrograman. Dalam efektivitas keseluruhan, semua studi diambil diperhitungkan untuk mendapatkan penilaian umum. Untuk menyelidiki yang mana metrik mana yang efektif dalam lingkungan tertentu, kami menilai bagaimana efektivitas metrik dipengaruhi oleh sebelum dan sesudah rilis, set data kecil dan besar, dan prosedural. Untuk kategori ukuran, set data berukuran sedang dan besar digabungkan. Dengan cara ini, perbedaan yang jelas antara studi yang menggunakan set data kecil dan besar dibuat.

3. Hasil RQ3: *Software metric* berguna untuk prediksi kesalahan.
   a) Metrik Ukuran
      Dengan batasan tertentu, terdapat korelasi antara ukuran dan jumlah kesalahan [13] [14] [15]. Namun bukti yang kuat tidak menunjukkan bahwa metrik ukuran seperti *Lines of Code*, merupakan penunjuk kesalahan yang baik [12] [11] [16]. Hubungan yang kuat terlihat antara metrik ukuran *Lines of Code* dan risiko kesalahan pada studi pra-rilis dan berskala kecil, sementara pada studi pasca-rilis dan berskala besar, hanya hubungan yang lemah yang ditemukan. Hal ini bahwa studi yang memiliki validitas lebih rendah (data yang lebih kecil) menunjukkan signifikansi yang lebih besar pada metrik *Lines of Code* daripada studi yang memiliki validitas lebih tinggi (data yang lebih besar). Oleh karena itu, keandalan metrik ini dalam penilaian kualitas harus diperhitungkan.
   b) Metrik Kompleksitas
      Prediktor yang baik dari kerentanan kesalahan perangkat lunak di [17] [18] [19] adalah kompleksitas siklomatik McCabe. Dalam lingkungan pasca-rilis besar yang menggunakan bahasa *Object-oriented*, kompleksitas siklomatik cukup efektif. Namun tidak demikian halnya dalam lingkungan pra-rilis kecil dengan bahasa prosedural. Oleh karena itu, kompleksitas siklomatik mungkin lebih efektif dalam lingkungan besar dan *Object-oriented*. Dalam laporan menyatakan hasil yang kurang memuaskan bagi metrik Halstead [20] [21]. Berdasarkan bukti yang dikumpulkan, kompleksitas siklomatik McCabe atau *Lines of Code* terbukti lebih baik daripada metrik Halstead [19]. Meskipun demikian, dalam studi pra-rilis dan berskala kecil, metrik tersebut masih terbukti cukup efektif, meskipun menghasilkan hasil validitas yang rendah. Namun, dalam kategori lain, metrik Halstead tidak terlalu efektif jika dibandingkan dengan metrik-metrik lainnya.
   c) Metrik Object-oriented
      Menurut beberapa penulis [22] [23] [24] [25] [26] [27] [28] [3], *Response for a Class* (RFC), *Weighted Methods per Class* (WMC), dan *Coupling Between Objects* (CBO) dianggap sebagai metrik terbaik dari rangkaian metrik Chidamber-Kemerer yang efektif di semua kelompok. Bahkan dalam penelitian [21] yang menggunakan metode Halstead menyarankan untuk memakai metrik *Object-oriented.* Meskipun *Lack of Cohesion in Methods* terbukti efektif dalam studi pra-rilis kecil dan dianggap sebagai metrik oleh beberapa penulis [24], ketika dibandingkan dengan metrik Chidamber-Kemerer lainnya, *Lack of Cohesion in Methods* tidak terlalu berhasil dalam menemukan kesalahan. Selain itu dalam studi [29] hal positif juga dihasilkan pada metrik *coupling.* Dalam studi ini, mengeksplorasi efektivitas metrik *coupling* dalam prediksi kesalahan *software*. Digambarkan bahwa metrik *coupling* berguna dalam memprediksi kesalahan pada *software.*
   d) Metrik Proses
      Bansal [30] menganlisis dan membandingkan untuk mengklasifikasikan metrik proses dengan metrik ukuran dalam memprediksi kesalahan *software.* Dalam laporannya membuktikan bahwa penggunaan metrik proses untuk memprediksi kesalahan memberikan hasil yang efektif. Kinerja prediksi yang paling terbaik jika digabungkan antara metrik proses dan metrik ukuran.

5 KESIMPULAN

Penelitian ini dilakukan untuk mengidentifikasi metrik dalam memprediksi kesalahan pada *software*. Metode pengukuran yang paling umum digunakan diidentifikasi dan kemampuan untuk memprediksi kesalahan dinilai untuk menjawab pertanyaan tentang metode yang sesuai untuk memprediksi kesalahan. Pencarian data hingga tahun 2023 terdapat 21 artikel yang dipilih dan dianalisis berdasarkan sejumlah pertanyaan penelitian. Berdasarkan identifikasi artikel yang dipilih adalah sebagai berikut:

1. *Metric* yang digunakan untuk memprediksi kesalahan adalah metrik *Lines of Code* dari metrik ukuran, metrik kompleksitas siklomatis CC dari McCabe dan Halstead, dan metrik Chidamber dan Kemerer (DIT, CBO, LCOM, RFC, NOC,dan WMC) dipilih sebagai metrik berorientasi objek.
2. Pengukuran perangkat lunak seperti *Lines of Code* (LOC) dapat dipakai untuk meramalkan kegagalan. Walaupun demikian, hasilnya tidak konsisten dalam meramalkan kegagalan sesudah pengaplikasian. Di samping itu,
3. Pengukuran kompleksitas seperti kompleksitas siklomatik McCabe dan pengukuran Halstead terbukti efektif dalam lingkungan pengaplikasian yang besar dan menggunakan bahasa pemrograman *object-oriented*.
4. Pengukuran yang object-oriented seperti RFC, WMC, dan CBO, juga efektif dalam meramalkan kegagalan perangkat lunak. Artikel yang menggunakan metrik object-oriented, Chidamber dan Kemerer (CK) paling sering digunakan. Menurut artikel yang dipilih metrik *object-oriented* dilaporkan lebih berhasil dalam menemukan kesalahan dibandingkan dengan metrik ukuran dan kompleksitas.
5. Penggunaan pengukuran proses seperti jumlah pengembang, jumlah kegagalan masa lalu, jumlah perubahan, usia modul, dan ukuran set perubahan juga memberikan hasil yang efektif dalam meramalkan kegagalan.

Namun, perlu dipertimbangkan bahwa keandalan dan validitas pengukuran-pengukuran tersebut dalam penilaian kualitas perangkat lunak harus dipertimbangkan. Terdapat perbedaan dalam hasil penelitian yang dilakukan dengan data yang lebih kecil atau lebih besar, serta perbedaan dalam lingkungan pengembangan perangkat lunak sebelum dan sesudah pengaplikasian. Oleh karena itu, pemilihan pengukuran yang sesuai dan mempertimbangkan konteks pengembangan perangkat lunak akan menjadi penting dalam memprediksi kegagalan dengan tingkat ketepatan yang lebih baik.

REFERENSI

[1] E. E. Mills, "Software Metrics," SEI Curriculum Module SEI-CM12-1.1, Carnegie Mellon University, Pittsburgh, PA, 1988.

[2] K. Wiegers dan B. J., Software Reqirements, Washington: Pearson Education, 2013.

[3] S. R. Aziz, T. Khan dan A. Nadeem, "Experimental Validation of Inheritance Metrics' Impact on Software Fault Prediction," *IEEE Access,* vol. 7, pp. 85262-85275, 2019.

[4] R. Jayanthi dan L. Florence, "Software Defect Prediction Techniques Using Metrics Based on Neural Network Classifier," *Cluster Computing,* vol. 22, pp. 77-88, 2019.

[5] H. B. Yadav dan D. K. Yadav, "A Fuzzy Logic Based Approach for Phase-wise Software Defects Prediction Using Software Metrics," *Information and Software Technology,* vol. 63, pp. 44-57, 2015.

[6] A. Sethi dan S. K. Sharma, "Comparative Evaluation of Aproaches and Mechanisms for Software Metrics," *International Refereed Journal of Reviews and Research ,* vol. 5, no. 5, 2017.

[7] I. Sommerville, Software Engineering (Rekayasa Perangkat Lunak), Jakarta: Erlangga, 2011.

[8] F. Rahman dan P. Devanbu, "How, and Why, Process Metrics are Better," dalam *2013 35th International Conference on Software Engineering (ICSE) (pp. 432-441). IEEE.*, California, 2013.

[9] Z. Li, X. Y. Jing dan X. Zhu, "Progress on Approaches to Software Defect Prediction," *IET Software,* vol. 12, no. 3, pp. 161-175, 2018.

[10] S. Hadi, H. K. Tjahjono dan M. Palupi, SYSTEMATIC REVIEW : META SINTESIS untuk Riset Perilaku Organisasional, Yogyakarta: Viva Victory Abadi, 2020.

[11] C. Andersson dan P. Runeson, "A Replicated Quantitative Analysis of Fault Distributions in Complex Software Systems," *IEEE Transactions on Software Engineering,* vol. 33, no. 5, pp. 273-286, 2007.

[12] N. E. Fenton dan N. Ohlsson, "Quantitative Analysis of Faults and Failures in a Complex Software System," *IEEE Transactions on Software Engineering,* vol. 26, no. 8, pp. 797-814, 2000.

[13] H. Zhang, "An Investigation of the Relationships Between Lines of Code and Defects," dalam *2009 IEEE international conference on software maintenance*, Beijing, 2009.

[14] A. Taufik dan R. F. Amir, "Penerapan Penerapan Teknik PSO Over Sampling Dan Adaboost J48 untuk Memprediksi Cacat Software," *Jurnal Responsif: Riset Sains dan Informatika,* vol. 2, no. 2, pp. 198-203, 2020.

[15] S. Yatish, J. Jiarpakdee, P. Thongtanunam dan C. Tantithamthavorn, "Mining Software Defects: Should We Consider Affected Releases?," dalam *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE) (pp. 654-665)*, 2019.

[16] R. F. Amir, I. A. Sobari dan R. Rousyati, "Penerapan PSO Over Sampling dan Adaboost Random Forest untuk Memprediksi Cacat Software," *Indonesian Journal on Software Engineering (IJSE),* vol. 6, no. 2, pp. 230-239, 2020.

[17] I. Chowdhury dan M. Zulkernine, "Using Complexity, Coupling, and Cohesion Metrics as Early Indicators of Vulnerabilities," *Journal of Systems Architecture,* vol. 57, no. 3, pp. 294-313, 2011.

[18] T. Menzies, J. S. Di Stefano, M. Chapman dan K. McGill, "Metrics that Matter," dalam *27th Annual NASA Goddard/IEEE Software Engineering Workshop, 2002*, Morgantown, 2002.

[19] T. &. D. S. J. S. Menzies, "How good is your blind spot sampling policy," dalam *Eighth IEEE International Symposium on High Assurance Systems Engineering*, Fairmont, 2004.

[20] J. C. Munson dan S. G. Elbaum, "Code Churn: A Measure for Estimating the Impact of Code Change," dalam *Proceedings. International Conference on Software Maintenance (Cat. No. 98CB36272)*, Moscow, 1998.

[21] Z. B. G. Aydin dan R. Şamli, "A Comparison of Software Defect Prediction Metrics Using Data Mining Algorithms," *Journal of Innovative Science and Engineering (JISE),* vol. 4, no. 1, pp. 11-21, 2020.

[22] K. K. Aggarwal, Y. Singh, A. Kaur dan R. Malhotra, "Empirical Analysis for Investigating the Effect ff Object-Oriented Metrics on Fault Proneness: A Replicated Case Study," *Software Process: Improvement and Practice,* vol. 14, no. 1, pp. 39-62, 2009.

[23] L. C. Briand, J. Wüst dan H. Lounis, "Replicated Case Studies for Investigating Quality Factors in Object-Oriented Designs," *Empirical Software Engineering,* vol. 6, pp. 11-58, 2001.

[24] G. J. Pai dan J. B. Dugan, "Empirical Analysis of Software Fault Content and Fault Proneness Using Bayesian Methods," *IEEE Transactions on Software Engineering,* vol. 33, no. 10, pp. 675-689, 2007.

[25] S. S. Rathore dan S. Kumar, "A Study on Software Fault Prediction Techniques," *Artificial Intelligence Review,* vol. 51, pp. 255-327, 2019.

[26] A. Singh, R. Bhatia dan A. Singhrova, "Taxonomy of Machine Learning Algorithms in Software Fault Prediction Using Object Oriented Metrics," *Procedia Computer Science,* vol. 132, pp. 993-1001, 2018.

[27] R. Mahajan, S. K. Gupta dan R. K. Bedi, "Design of Software Fault Prediction Model Using BR Technique," *Procedia Computer Science,* vol. 46, pp. 849-858, 2015.

[28] L. Kumar, S. Misra dan S. K. Rath, "An Empirical Analysis of the Effectiveness of Software Metrics and Fault Prediction Model for Identifying Faulty Classes," *Computer standards & interfaces,* vol. 53, pp. 1-32, 2017.

[29] R. Muhammad, A. Nadeem dan M. A. Sindhu, "Vovel Metrics—Novel Coupling Metrics for Improved Software Fault Prediction," *PeerJ Computer Science,* vol. 7, pp. 1-27, 2021.

[30] A. Bansal, "Comparative Analysis of Classification Methods for Prediction Software Fault Proneness Using Process Metrics," 2021.