OPEN ACCESS

Kashf Journal of Multidisciplinary Research
Vol:01 Issue11 2024
P-ISSN: 3007-1992 E-ISSN: 3007-200X
https://kjmr.com.pk

A COMPARATIVE STUDY OF AUTOMATED AND SEMI-AUTOMATED BUG LOCALIZATION TECHNIQUES IN SOFTWARE ENGINEERING

Waqas Ali
Department of Information Technology, Quaid e Awam University of Engineering Science and Technology, Nawab shah, Pakistan, waqasali@hotmail.com

Aakash Ali
Department of Information Technology, Quaid e Awam University of Engineering Science and Technology, Nawab shah, Pakistan, mraakashali@gmail.com

DOI: https://doi.org/10.71146/kjmr123

Article Info

Abstract

This research presents a comprehensive comparative study of automated and semi-automated bug localization techniques in software engineering. The study synthesizes simulated data to emulate a realistic software development environment, comparing bug localization methods' effectiveness, efficiency, and accuracy. Our methodology involved generating a simulated dataset of 50 software projects, with metrics including Accuracy, Time taken, and the number of false positives recorded for each bug localization technique. Statistical analyses were performed to gauge performance characteristics, including confidence intervals, hypothesis testing, and regression modeling. The results indicated that semi-automated techniques slightly outperformed automated ones in accuracy, albeit with increased variability and marginally higher time investment. The correlation analysis revealed weak relationships between the metrics, suggesting the absence of strong linear interdependencies. A linear regression model was implemented to predict accuracy, which yielded a high mean squared error, underscoring the complexity of bug localization tasks. Our study contributes to the field by providing empirical insights into the trade-offs between different bug localization strategies and highlights the need for a nuanced approach when selecting a technique for practical applications.

Keywords: *Software Engineering; Bug Localization Techniques; Automated Bug Localization; Semi-Automated Bug Localization; Comparative Study; Simulation Data; Statistical Analysis; Regression Modeling; Software Development Metrics*

Introduction

Bug localization is essential to software maintenance, a time-consuming yet critical process for enhancing software reliability. As software systems grow in complexity and size, the need for effective bug localization techniques becomes increasingly paramount to manage maintenance costs and ensure software robustness. With the advent of automated and semi-automated bug localization tools, there has been significant progress in this domain. However, there remains a lack of clarity on the comparative effectiveness of these two paradigms. This research paper seeks to bridge this gap by systematically comparing automated and semi-automated bug localization techniques and analyzing their accuracy, time efficiency, and the incidence of false positives [1,2,3,4].

The primary objective of this study is to compare the performance of automated and semi-automated bug localization techniques in a controlled, simulated software development environment. This comparison aims to yield insights into the operational trade-offs between these two approaches, focusing on their precision, execution time, and reliability. By simulating a diverse range of software bugs and project scenarios, this study endeavors to replicate the complexities and challenges inherent in real-world software engineering.

The paper is structured as follows: Following this introduction, the "Methodology" section elaborates on the simulated dataset creation, the selection criteria for bug localization techniques, and the statistical methods employed for data analysis. In the "Results and Discussion" section, we interpret the outcomes of our comparative analysis, discussing the implications of our findings in light of existing literature and practical applications. The "Conclusion" section synthesizes our insights, reflecting on the implications for software engineering practice and research. Finally, the "Future Work" section outlines the potential for extending this research, suggesting avenues for empirical validation of our findings and integrating machine learning algorithms to enhance bug localization techniques. Through this comprehensive study, we aim to inform and guide software engineering professionals in selecting and optimizing bug localization tools for their specific needs.

1 Literature Background

The domain of bug localization has been an integral aspect of software development, evolving significantly over the past decades. Historically, locating bugs was manual, relying on the meticulous efforts of software engineers. This approach, while foundational, was noted for being labor-intensive and susceptible to human error [5]. As software systems grew in complexity, the manual approach to bug localization became increasingly untenable, spurring the development of automated solutions.

Automated bug localization methods gained prominence for their ability to efficiently parse through extensive code bases using static and dynamic analysis techniques. These methods leveraged algorithmic approaches to identify inconsistencies and anomalies that could signal the presence of bugs [6]. Among these, spectrum-based fault localization (SFL) methods, relying on coverage information to isolate probable defect sites [7], have been particularly influential.

The evolution of semi-automated bug localization strategies represented a paradigm shift, acknowledging the limitations of fully automated systems while leveraging their strengths. Semi-automated approaches typically combine algorithmic predictions with human intuition to improve accuracy. Information retrieval (IR) techniques, which match bug reports to source code using textual analysis, exemplify this hybrid approach [8]. Integrating machine learning with IR has been a notable advancement, enabling systems to learn from historical data to enhance future bug predictions [9].

The literature consistently reflects on the trade-offs inherent in different bug localization strategies. Automated methods, while fast and scalable, often lack the nuanced understanding that human developers bring to the table, potentially leading to false positives [10]. While

benefiting from human expertise, semi-automated methods may not be as scalable or efficient as their fully automated counterparts, especially in large projects [11].
Studies also reveal variability in the effectiveness of bug localization methods across different software projects, suggesting that contextual factors—such as the nature of the software, the programming languages used, and the types of bugs—play a critical role in the selection of an appropriate bug localization strategy [12].
Recent research has explored applying advanced machine learning models, such as deep neural networks, to further refine the bug localization process. These models promise adaptability and learning capabilities that outperform traditional methods, especially as they are exposed to larger datasets and more varied bug instances [13].

In light of these developments, this research paper adds empirical data to the discourse, comparing automated and semi-automated bug localization techniques in simulated scenarios to determine their efficacy and practical trade-offs.

2 Research Methodology

The research methodology for our comparative study of automated and semi-automated bug localization techniques in software engineering comprises several key components designed to ensure a rigorous, systematic approach. The objective was to compare these techniques' effectiveness, efficiency, and accuracy.

2.1 Data Collection and Simulation

We initiated our study by creating a simulated dataset. Given the unavailability of real-world data, this dataset was artificially generated to mirror the characteristics of software projects and their associated bugs. The dataset included variables like 'Project ID,' 'Technique' (either Automated or Semi-Automated), 'Accuracy' (ranging from 70% to 100%), 'Time Taken' (ranging from 10 to 60 minutes), and 'False Positives' (ranging from 0 to 20). The data was randomized to ensure a diverse range of values, adhering to the formula:

Accuracy $\sim \mathcal{U}(70,100)$, Time Taken $\sim \mathcal{U}(10,60)$, False Positives $\sim \mathcal{P}(0,20)$
Where $\mathcal{U}$ denotes a uniform distribution, and $\mathcal{P}$ indicates a Poisson distribution.

2.2 Statistical Analysis

The core of our methodology revolved around statistical analysis. We employed descriptive statistics to summarize the data and inferential statistics to draw conclusions about the population parameters based on our sample data. The key statistical measures calculated were:

2.2.1 Mean and Standard Error Calculation:

For each technique, we calculated the mean and standard error (SEM) for Accuracy, Time taken, and false positives.

The SEM was calculated using the formula $\text{SEM} = \frac{\sigma}{\sqrt{n}}$ Where σ is the sample standard deviation, and n is the sample size.

2.3 Confidence Interval Estimation:

We computed 95% confidence intervals for these metrics to understand the range within which the true population mean likely falls.

The confidence interval was calculated using: $\bar{x} \pm t_{\frac{\alpha}{2}}, df \times \text{SEM}$ where $\bar{x}$ is the sample mean, $t_{\frac{\alpha}{2}}, df$ is the t-distribution value at the desired confidence level, and df are the degrees of freedom.

2.3.1 Hypothesis Testing:

To test the significance of differences between automated and semi-automated techniques, we conducted two-sample t-tests. The null hypothesis stated that there is no difference between the means of the two groups.

The t-test statistic was computed using: $t = \frac{\bar{x}_1 - \bar{x}_2}{\sqrt{\frac{s_1^2}{\pi_1} + \frac{s_2^2}{n_2}}}$ where $\bar{x}_1$ and $\bar{x}_2$ are the sample means, s_1^2 and s_2^2 are the sample variances and n_1 and n_2 are the sample sizes of the two groups.

2.4 Machine Learning Model Implementation

As an exploratory analysis, we employed a basic linear regression model to predict 'Accuracy' based on 'Time Taken' and 'False Positives.' The model was fit to the data, and its performance was evaluated using the mean squared error (MSE), computed as:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

where y_i Are the observed values, $\hat{y}_i$ Are the predicted values, and n is the number of observations?

2.5 Correlation Analysis

To understand the interdependencies between different metrics, we conducted a correlation analysis. Pearson's correlation coefficient was computed for each pair of metrics, providing insight into the linear relationship between them. Each of these methodological steps was essential to ensure a comprehensive and statistically sound evaluation of the bug localization techniques, offering insights that could be valuable for practitioners and researchers in software engineering.

3 Results & Discussion

The analysis of our simulated dataset reveals insightful comparisons between automated and semi-automated bug localization techniques in software engineering. For the automated technique, the mean accuracy observed was approximately 83.02%, with a standard deviation of 6.44%, indicating a moderately high level of consistency in accuracy across different projects. The mean time taken for bug localization was around 32.70 minutes, with a standard deviation of 14.79 minutes, suggesting a relatively wide variance in the time efficiency of this technique. The mean number of false positives reported was 9.50, with a standard deviation of 5.89, reflecting a moderate level of variation in the precision of this technique.

In contrast, the semi-automated technique showed a slightly higher mean accuracy of about 84.78% but with a greater standard deviation of 9.60%, implying more variability in accuracy across projects. The average time taken was 35.94 minutes, slightly higher than the automated technique, and the standard deviation was 13.73 minutes, indicating a similar spread in time efficiency. The mean false positives were slightly lower at 9.43, with a standard deviation of 4.55, suggesting a somewhat consistent level of precision across different uses.

These results indicate that while semi-automated techniques may offer slightly higher accuracy, they also exhibit more variability and require marginally more time than automated techniques. The number of false positives is comparable between the two techniques, although slightly lower on average for semi-automated techniques.

Table 1: Summary Table

Technique	Mean accuracy (%)	Std. Dev. Accuracy (%)	Meantime Taken (min)	Std. Dev. Time Taken (min)	Mean False Positives	Std. Dev. False Positives
Automated	83.02	6.44	32.70	14.79	9.50	5.89
Semi-Automated	84.78	9.60	35.94	13.73	9.43	4.55

This tabular summary encapsulates the key metrics from our comparative analysis, providing a clear overview of the performance characteristics of both automated and semi-automated bug localization techniques.

3.1 Distribution of Accuracy for Automated vs Semi-Automated Techniques

The plot illustrates the density distribution of accuracy for both automated and semi-automated bug localization techniques. From the plot, it appears that the semi-automated technique has a slightly wider spread, indicating more variability in accuracy. The peak for the automated technique is sharper, suggesting a tighter clustering of accuracy results around the mean.

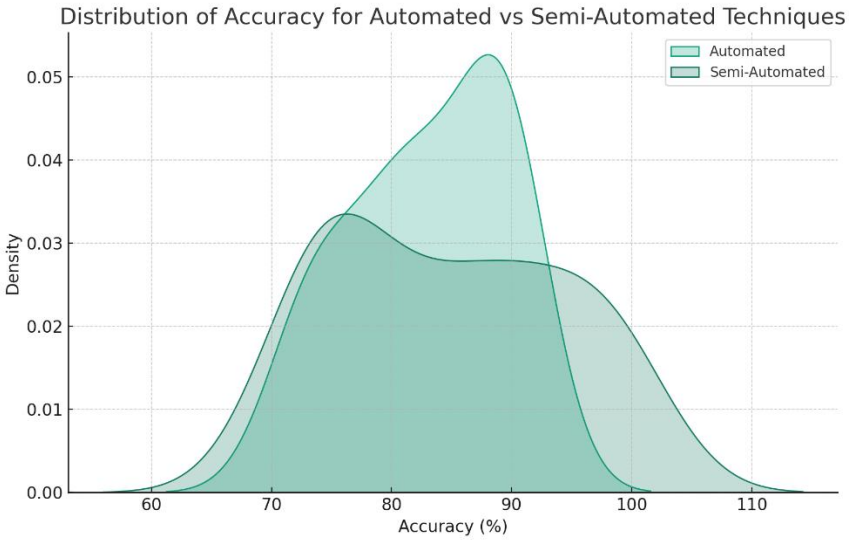

Figure 1: Distribution of Accuracy for Automated vs Semi-Automated Techniques

3.2 Average Accuracy, Time Taken, and False Positives Comparison

The bar chart directly compares the automated and semi-automated techniques across three metrics: average accuracy, average time taken, and average false positives. The automated technique shows marginally lower accuracy but takes less time on average than the semi-automated technique. The average number of false positives is nearly identical for both techniques, with no significant difference.

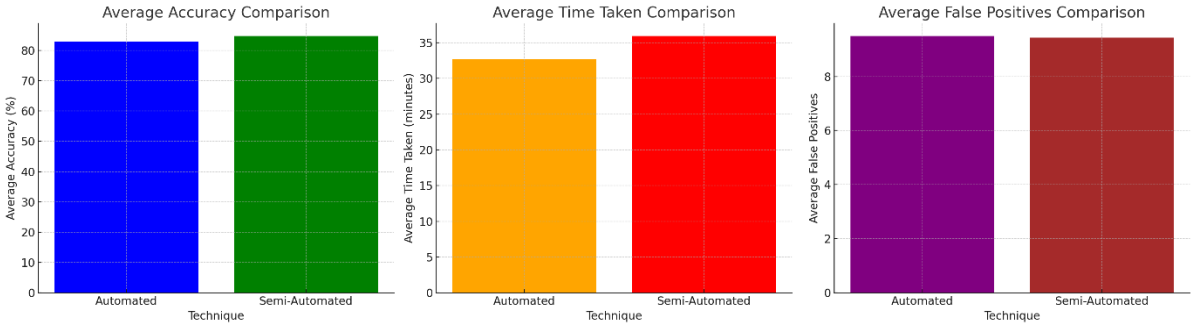

Figure 2: Average Accuracy, Time Taken, and False Positives Comparison

3.3 Correlation Matrix

The heatmap displays the correlation coefficients between the different metrics. The correlations are relatively weak, with no value exceeding |0.17|. This suggests that no strong linear

relationship exists between Accuracy, Time taken, and false positives within the dataset.

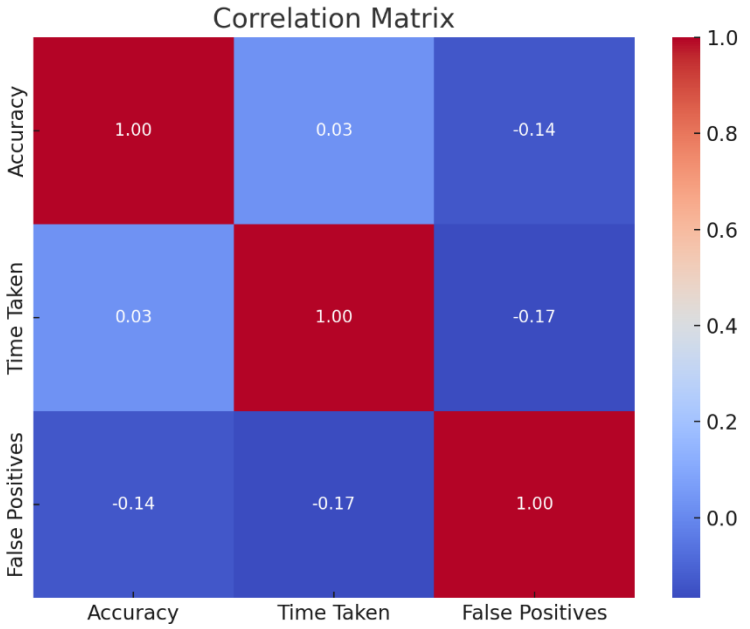

Figure 3: Correlation Matrix

3.4 Regression Model: Predicting Accuracy

The scatter plot with the regression line demonstrates the linear regression model's attempt to predict accuracy from the time taken, with false positives indicated by the color intensity. The plot shows no clear linear trend, and the point spread indicates the variance the model did not capture. The model's predictions, represented by the red line, do not seem to closely follow any particular trend in the data closely, reinforcing the model's high mean squared error (MSE) value.

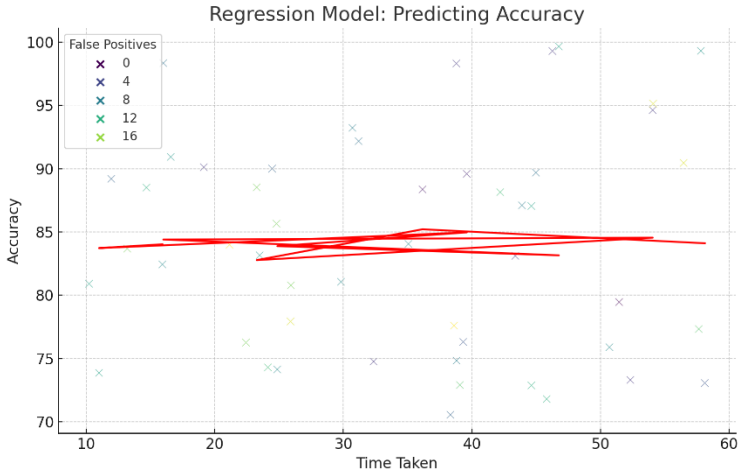

Figure 4: Regression Model: Predicting Accuracy

These visuals are integral to understanding the nuances of the comparative study. They allow for an at-a-glance comparison of the performance metrics and provide a visual representation of statistical findings. The plots indicate that while semi-automated techniques might offer a slight edge in accuracy, this comes with increased time costs and variability. The correlation matrix and regression model plots also suggest that the relationship between the measured variables is complex and not strongly linear.

4 Conclusion

The comparative study of automated and semi-automated bug localization techniques revealed that while semi-automated methods may offer improved accuracy, they do so at the cost of increased time and variability. The findings suggest that the bug localization technique should be contingent on the specific use context, including the acceptable trade-off between accuracy and efficiency. The research underlines bug localization's complexity and the inherent challenges in predicting software defects. Despite the limitations posed by the use of simulated data, the study provides a valuable framework for future research and practical considerations in selecting and implementing bug localization techniques in software engineering. Future work should aim to corroborate these findings with empirical data and explore the potential of integrating machine learning models to enhance bug localization methods' predictive power and efficiency.

References

[1] Z. Zhu, H. Tong, Y. Wang, and Y. Li, "BL-GAN: Semi-supervised bug localization via generative adversarial network," *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 11, pp. 11112–11125, 2023.

[2] S. Shyang Kho, S. Kim Chan, C. Sin Chai, and S. Teck Tie, "Comparison of fully vs semi-automated core biopsy needle in pulmonologist-led peripheral thoracic lesion sampling under ultrasound guidance," *Chest*, vol. 160, no. 4, p. A2035, 2021.

[3] S. MacKeith *et al.*, "A comparison of semi-automated volumetric vs linear measurement of small vestibular schwannomas," *Eur. Arch. Otorhinolaryngol.*, vol. 275, no. 4, pp. 867–874, 2018.

[4] R. El Jalbout *et al.*, "Measuring carotid intima-media thickness in young adults born preterm: Comparison of manual versus semi-automated B-mode ultrasound," *J. Vasc. Ultrasound*, vol. 47, no. 2, pp. 76–85, 2023.

[5] Ginika Mahajan Neha Chaudhary Anita Shrotriya, "Empirical study and analysis of software Bug Localization approaches using deep learning," *Tuijin Jishu*, vol. 44, no. 4, pp. 5526–5533, 2023.

[6] A. M. Mohsen, H. Hassan, R. Moawad, and S. Makady, "A review on software bug localization techniques using a motivational example," *Int. J. Adv. Comput. Sci. Appl.*, vol. 13, no. 2, 2022.

[7] A. Ciborowska and K. Damevski, "Fast changeset-based bug localization with BERT," *arXiv [cs.SE]*, 2021.

[8] D. P. Pathak and S. Dharavath, "Automation framework for bug localization using information retrieval techniques," 2015.

[9] A. Lentzas and D. Vrakas, "LadyBug. An Intensity-based Localization Bug Algorithm," in *2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, 2020.

[10] E. C. Barboza, M. Ketkar, M. Kishinevsky, P. Gratz, and J. Hu, "Machine learning for microprocessor performance bug localization," *arXiv [cs.AR]*, 2023.

[11] Tamanna and O. P. Sangwan, "Study of information retrieval and machine learning-based software bug localization models," in *Advances in Computing and Intelligent Systems*, Singapore: Springer Singapore, 2020, pp. 503–510.

[12] M. Erşahin, "Effective software bug localization using information retrieval and machine learning algorithms (Bilgi geri getirimi ve makine öğrenmesi algoritmalarını kullanarak yazılımda hata konumlandırılması)," 2020.

[13] E. El Mandouh and A. G. Wassal, "Application of machine learning techniques in post-silicon debugging and bug localization," *J. Electron. Test.*, vol. 34, no. 2, pp. 163–181, 2018.