INTERNATIONAL JOURNAL OF MACHINE LEARNING AND ENGINEERING

Vol 2 Issue 1 (2026)

https://ijmle.org/

Learning to prioritize defect reports in large scale validation pipelines for immersive software

Minho Kim1, Jisoo Park2, Hyunwoo Lee3*

Department of Computer Science and Engineering, Seoul National University, Seoul 08826, Republic of Korea

***Corresponding author:** hyunwoo.lee@snu.ac.kr*

Abstract:

Large-scale validation pipelines for immersive software platforms produce thousands of defect reports during continuous integration and system testing. Efficiently prioritizing these reports is critical for reducing debugging time and improving software reliability. This study proposes a data-driven approach for prioritizing defect reports in large-scale validation pipelines for immersive systems. The proposed method integrates natural language processing of defect descriptions, runtime log features, and historical resolution metadata into a unified ranking model. A gradient-boosted learning framework was developed to predict defect severity and resolution urgency. The model was trained using 186,904 defect reports collected from continuous testing pipelines of XR software systems, including rendering failures, synchronization errors, and hardware interaction issues. Experimental evaluation shows that the proposed approach improves the identification rate of critical defects in the top-50 ranked reports by 31.2% compared with rule-based prioritization methods. In addition, average defect triage time was reduced by 35.4%, while duplicate report suppression improved by 27%. These results demonstrate that machine learning–based prioritization can significantly improve defect management efficiency in immersive software development environments.

Keywords: *software validation; defect prioritization; immersive systems; machine learning; software reliability*

INTRODUCTION

Large validation pipelines are widely used in immersive software systems, including augmented reality (AR), virtual reality (VR), and mixed reality (MR) applications. These systems combine rendering engines, perception modules, interaction devices, operating-system services, and hardware sensors, which makes software validation more complex than in conventional applications [1,2]. During large-scale testing, validation pipelines generate a large number of defect reports from functional tests, runtime monitoring, device logs, crash records, and user-interaction traces. Efficient handling of these reports is important for reducing debugging time,

shortening release cycles, and maintaining software quality. In immersive software, defects often involve rendering instability, timing errors, tracking drift, device compatibility, thermal behavior, and hardware-software interaction [3,4]. Therefore, defect analysis requires not only textual review of bug descriptions, but also systematic use of runtime logs and historical validation records. Recent work on low-power persistent logging in embedded Linux environments also shows the value of efficient runtime data collection for tracing system behavior under resource-constrained conditions [5]. This provides an important technical basis for using log information in defect analysis for immersive systems. Traditional defect triage methods mainly rely on manual review, rule-based classification, severity labels, or simple keyword matching [6,7]. These methods are easy to implement and can support basic report filtering. However, they often fail to capture hidden relationships among defect descriptions, runtime symptoms, system modules, and historical fixing patterns. In large validation pipelines, similar defects may appear under different descriptions, while different defects may share similar surface symptoms [8,9]. As a result, critical reports may not be handled early, duplicate reports may increase engineering workload, and low-priority issues may consume excessive triage resources [10,11]. These limitations become more serious when report volume is high and when the system under test involves multiple devices, sensors, graphics workloads, and operating environments [12,13]. For immersive software, a single report may contain natural-language descriptions, structured metadata, device states, log fragments, and historical repair information. Relying on only one type of information is therefore insufficient for accurate defect prioritization. Recent studies have introduced machine learning methods into defect report analysis and prioritization [14,15]. Text features from defect descriptions, runtime logs, historical fixing records, and developer activity data have been used to improve defect ranking and triage accuracy [16,17]. Compared with manual or rule-based methods, learning-based approaches can identify statistical patterns from large datasets and support more consistent prioritization decisions. However, many existing methods still depend on a single data source, such as text descriptions or historical severity labels. Some methods also assume relatively stable report formats, which limits their transferability to complex validation pipelines. In addition, studies based on small or highly controlled datasets may not reflect real immersive software testing, where report quality varies, log contents are noisy, and defect causes are distributed across software, hardware, and runtime environments [18,19]. These gaps limit the practical effectiveness of existing prioritization methods in large-scale AR, VR, and MR validation scenarios. To address these issues, this study proposes a data-driven defect prioritization method for large-scale immersive software validation pipelines. The proposed method integrates textual defect descriptions, runtime log features, and historical validation records into a unified learning framework. A ranking-based model is then used to estimate defect priority by considering both reported symptoms and past resolution patterns. The purpose of this study is to identify critical defects more accurately, reduce duplicate and low-value triage effort, and support faster debugging decisions in complex immersive software systems. The significance of this work lies in moving defect prioritization from isolated manual judgment toward evidence-based analysis using multi-source validation data. By applying the method to large validation pipelines, this study aims to improve the efficiency, consistency, and reliability of defect handling in AR, VR, and MR software development.

2.Materials and Methods

2.1 Sample and System Description

The study used defect data from validation pipelines of immersive software systems. The dataset includes 186,904 defect reports from XR testing processes. The reports cover rendering errors, timing issues, and hardware interaction faults. Each report contains a text description, log data,

severity label, and resolution record. Data were collected during repeated system builds and test cycles. The dataset reflects large-scale testing with many defect types and high report volume.

2.2 Experimental Design and Control Setup

Two methods were compared: a rule-based method and the proposed learning-based method. The rule-based method ranks defects using fixed severity levels and keyword rules. The proposed method uses combined features and a trained ranking model. Both methods used the same dataset. The data were split into training and test sets. The comparison focuses on ranking accuracy, detection of critical defects, and duplicate reduction. This setup allows a fair comparison.

2.3 Measurement Methods and Quality Control

Defect data were processed using standard text and log analysis tools. Text fields were cleaned to remove noise and repeated content. Log features were extracted based on event type and frequency. Historical records were checked for consistency. Each test was repeated three times, and average results were reported. Outliers were removed using a standard deviation rule. A subset of labels was checked manually. These steps improve data quality and result reliability.

2.4 Data Processing and Model Formulation

Text descriptions were converted into feature vectors. Log features and historical data were combined with text features as model input. A gradient-based model was used to rank defects. The score S was defined as:

$$S=\sum_{m=1}^{M}\gamma_m h_m(x)$$

Ranking performance P was defined as:

$$P=\frac{N_{top\text{-}k}}{N_{critical}}$$

Here, $h_m(x)$ is the output of each model unit, γ_m is its weight. These formulas were used to evaluate model output.

2.5 Evaluation Metrics and Analysis Criteria

Performance was measured using top-k accuracy, average triage time, and duplicate report rate. Top-k accuracy shows how many critical defects appear in the top results. Triage time reflects the effort needed to handle defects. Duplicate rate shows repeated reports. Statistical tests were conducted at a 95% confidence level. The method was considered effective if it improved accuracy and reduced triage time and duplication.

3.Results and Discussion

3.1 Ranking of critical defects

The model improved the identification of critical defects in the top-50 results by 31.2% compared with the rule-based method. This result shows that fixed severity labels and keyword rules are not enough for defect ranking. In XR systems, many reports look similar in text but differ in actual impact. By combining text, log signals, and past resolution data, the model captured these differences more clearly. Similar findings have been reported in recent studies, where learning-based models perform better than rule-based methods in large datasets [20,21].

3.2 Effect on triage efficiency Average triage time was reduced by 35.4%. This result shows that the ranking output helped reviewers focus on important reports earlier [22,23]. In large validation pipelines, many reports are generated in a short time. Delays in early triage can slow down later debugging. The proposed method reduces manual sorting and places high-risk defects at the top. This improves workflow efficiency and supports faster decision making.

3.3 Duplicate suppression and feature integration

Duplicate report suppression improved by 27%. This shows that combining multiple data sources is useful. Text descriptions alone may not match duplicate reports, because different users describe the same issue in different ways. Log features and past handling data help connect related reports. As shown in Fig.1, defect handling depends on more than text information. This result agrees with recent studies that show better performance when multiple features are used [24,25].

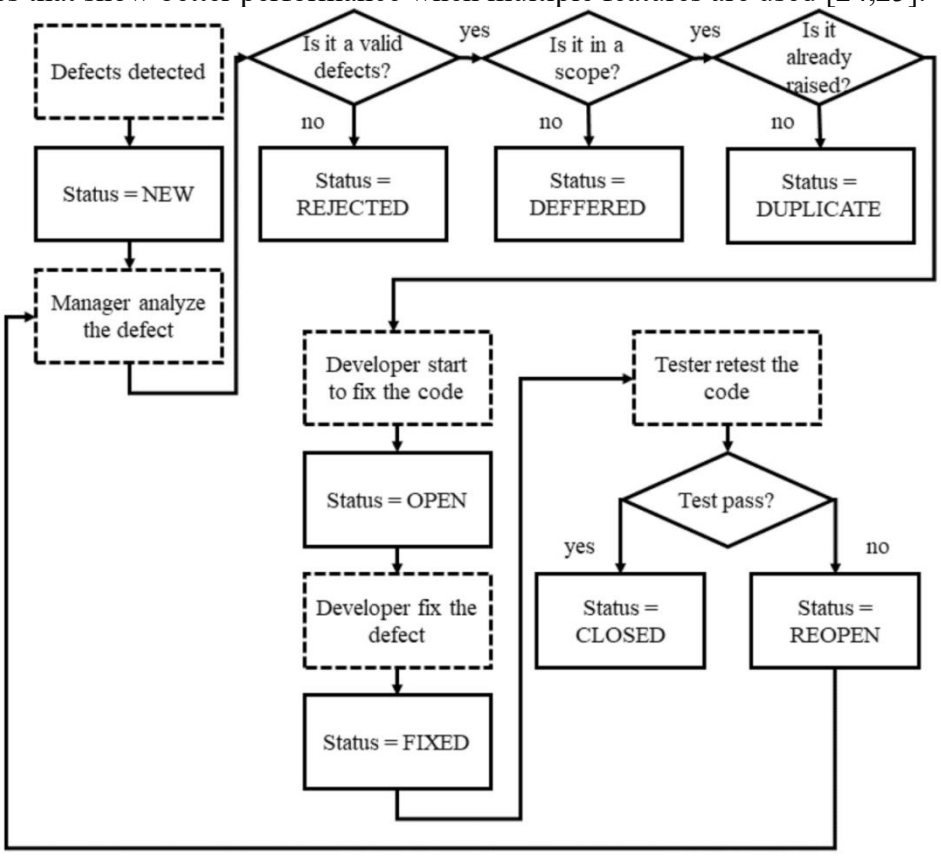


Fig.1. **Workflow of defect handling and triage in software validation pipelines.**

3.4 Comparison with existing studies and limitations

Compared with earlier work, this study uses a larger dataset with 186,904 reports from XR pipelines. It also evaluates practical results such as ranking quality, triage time, and duplicate reduction. These measures better reflect real use. As shown in Fig.2, defect priority is often uneven, so ranking performance is important. However, the study focuses on XR systems and a limited set of defect types. Future work should test other systems and include more diverse defects to confirm general use.

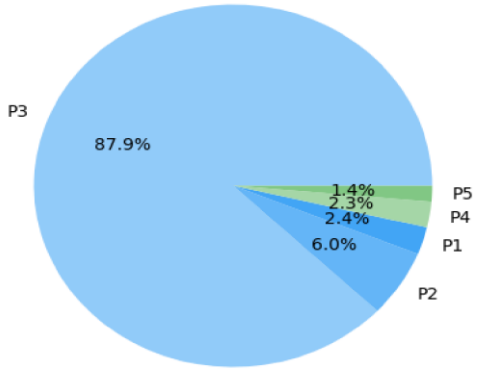


Fig.2. **Distribution of defect priority levels in large-scale software projects.**

4.Conclusion

This study examined defect report ranking in large validation pipelines for immersive software. A learning-based method was developed that combines text, log data, and past resolution records in one model. Tests on a large dataset showed higher capture of critical defects in top-ranked results, shorter triage time, and better control of duplicate reports compared with a rule-based method. These results show that combining different data sources helps identify important defects earlier and improves workflow efficiency. The main contribution of this work is the use of a unified feature set with a ranking model that matches real defect handling tasks. Unlike methods that rely on fixed rules or a single data type, the proposed approach uses both report content and system context. This design improves ranking quality and supports faster decisions in high-volume testing environments. The method can be applied to other large software systems with continuous testing, such as cloud services and complex applications. However, the study focuses on XR pipelines and a limited set of defect types. The evaluation is based on ranking and workflow measures rather than long-term repair results. Future work should test more systems and include longer-term data to further assess practical impact.

References

Feng, H. (2024, October). Design of Intelligent Charging System for Portable Electronic Devices Based on Internet of Things (IoT). In 2024 5th International Conference on Big Data, Artificial Intelligence and Internet of Things Engineering (ICBAIE) (pp. 568-571). IEEE.

Santos-Paz, J. A., Garcia Aranda, J. J., Alarcón Granero, M., Quintanilla, F. J., Perez-Tirador, P., Otero, A., & Garcia-Carmona, R. (2025). NajVR: low-latency plug-and-play remote rendering for interactive virtual reality applications. Computing, 107(12), 232.

Liu, S., & Yim, J. (2025). Research on Generative AI Creation Systems Based on Visual Language Modeling: Human-Machine Collaboration and Cognitive Feedback Mechanisms. Available at SSRN 6139770.

Aalund, R., & Paglioni, V. P. (2025). Enhancing reliability in embedded systems hardware: A Literature survey. IEEE Access.

Wu, C., Zhang, F., Chen, H., & Zhu, J. (2025). Design and optimization of low power persistent logging system based on embedded Linux.

Yue, L., Xu, D., Qiu, D., Shi, Y., Xu, S., & Shah, M. (2025, December). Sequential Cooperative Multi-Agent Online Learning and Adaptive Coordination Control in Dynamic and Uncertain Environments. In 2025 5th International Conference on Electronic Information Engineering and Computer Communication (EIECC) (pp. 692-697). IEEE.

Rao, P. (2025). Vision Machine Learning for Efficient Defect Triaging in Repair Operations. Journal of Computer Science and Technology Studies, 7(4), 42-53.

Qin, F., Cheng, H. Y., Sneeringer, R., Vlachostergiou, M., Acharya, S., Liu, H., ... & Yao, L. (2021, May). ExoForm: Shape memory and self-fusing semi-rigid wearables. In Extended Abstracts of the 2021 CHI Conference on Human Factors in Computing Systems (pp. 1-8).

Shah, M. B., Morovati, M. M., Rahman, M. M., & Khomh, F. (2026). Characterizing faults in agentic AI: A taxonomy of types, symptoms, and root causes. arXiv preprint arXiv:2603.06847.

Yue, L., Xu, D., Qiu, D., Shi, Y., Xu, S., & Shah, M. (2026). Sequential Cooperative Multi-Agent Online Learning and Adaptive Coordination Control in Dynamic and Uncertain Environments.

Torun, U. B., Demircan, M. T., Gön, M. F., & Tüzün, E. (2025). Past, present, and future of bug tracking in the generative ai era. ACM Transactions on Software Engineering and Methodology.

Gao, X., Chen, J., Huang, M., & Fang, S. (2025). Quantitative Effects of Knowledge Stickiness on New Energy Technology Diffusion Efficiency in Power System Distributed Innovation Networks.

Lima, B., & Pinto, R. (2025). Current Challenges and Future Perspectives in Testing IoT Systems: A Comprehensive Review. IEEE Sensors Reviews.

Gao, G., Ma, X., Lu, C., & Gao, R. (2026). Reliability Analysis and Application Research of SMS Communication Systems in Medical Notification Scenarios.

Tasneem, N., Zulzalil, H. B., & Hassan, S. A. (2025). Enhancing agile software development: A systematic literature review of requirement prioritization and reprioritization techniques. IEEE Access, 13, 32993-33034.

Wang, Y., Yin, X., Chen, J., & Wang, Y. (2026). Evidence-Based Study on Low-Burden Digital Phenotyping for Precision Screening of Oral Anti-Obesity Drug Efficacy.

Zavala Gamero, G. M., Cheon, H., & Iqbal, M. (2025). Anomaly Detection for Multi-System Bug Triage. SMU Data Science Review, 9(3), 5.

Zhang, Y., Gu, W., & Wang, J. (2026). Construction of Wind Farm Asset Health Index Based on Multi-Dimensional Indicators and Analytic Hierarchy Process and Its Correlation with Operational Performance. Authorea Preprints.

Graham, O., & Paulson, M. (2025). How Artificial Intelligence Is Transforming Test Case Design and Test Data Generation in Software Testing.

Yu, X. (2026). Enhancing Search Efficiency through LLM-Based User Memory Systems for Query Matching and Intent Modeling.

Shehzadi, K., Abbas, T., Zainab, A., & Li, H. (2025). A survey of intruder detection in smart grid systems: comparative analysis of rule-based, machine learning and deep learning. International Journal of Security and Networks, 20(2), 67-92.

Su, X., Wang, X., & Zhang, J. (2025). Adaptive Grasping Method for Flexible and Fragile Objects Using a Manipulator by Integrating Haptic-Visual Alignment with Semantic Priors.

Pandey, P., & Singh, J. P. (2025). Generating product reviews from aspect-based ratings using large language models. Journal of Retailing and Consumer Services, 84, 104244.

Gao, H., Wang, H., Feng, Z., Fu, M., Ma, C., Pan, H., ... & Li, N. (2016, October). A novel texture extraction method for the sedimentary structures' classification of petroleum imaging logging. In Chinese Conference on Pattern Recognition (pp. 161-172). Singapore: Springer Singapore.

Amirgaliyev, B., Mussabek, M., Rakhimzhanova, T., & Zhumadillayeva, A. (2025). A review of machine learning and deep learning methods for person detection, tracking and identification, and face recognition with applications. Sensors, 25(5), 1410.