International Journal for Multidisciplinary Research (IJFMR)
E-ISSN: 2582-2160 ● Website: www.ijfmr.com ● Email: editor@ijfmr.com

Intelligent Defect Triage Automation (IDTA): Leveraging AI for Efficient Defect Management

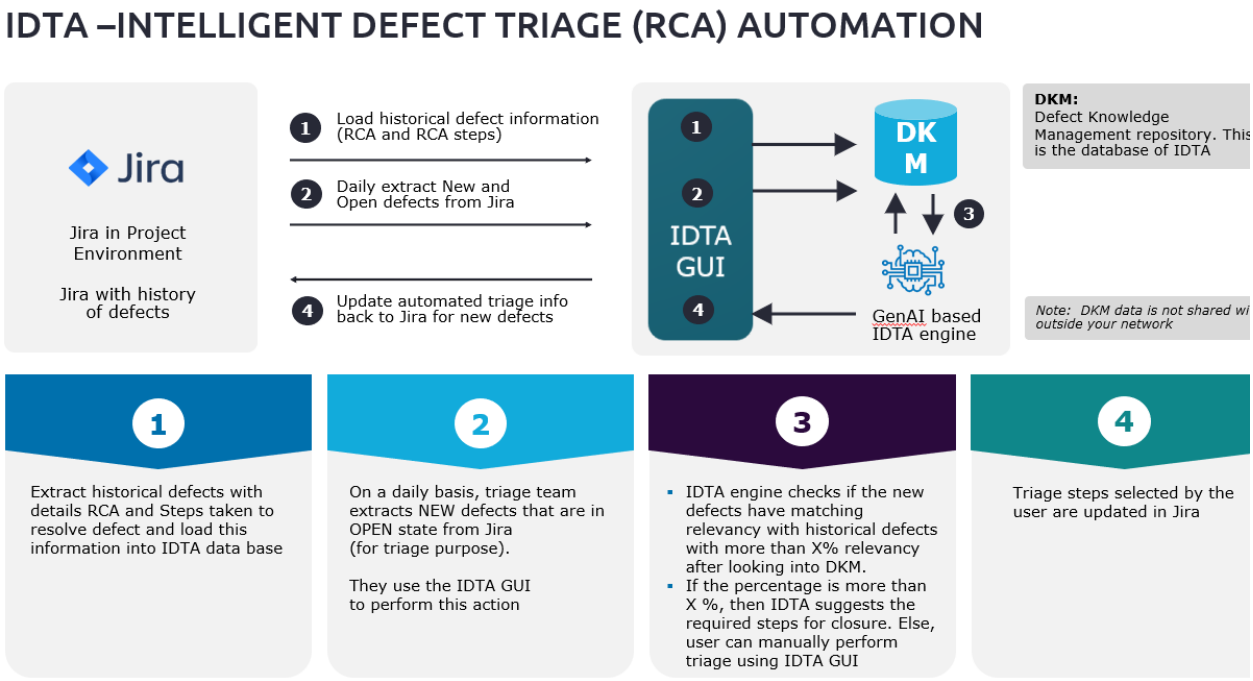
Jagan Mohan Rao Doddapaneni

Jaganmohanrao.d@gmail.com

Abstract

With the increasing complexity of software development, defect management has become a crucial aspect of ensuring high-quality releases. Traditional defect triage methods involve manual analysis, which is time-consuming and prone to human error. This paper introduces **Intelligent Defect Triage Automation (IDTA)**, an AI-driven approach leveraging historical defect knowledge to streamline the triage process. By integrating a **Defect Knowledge Management (DKM) repository** and an automated triage engine, IDTA can intelligently assess new defects, match them against historical data, and suggest resolution steps. This automation reduces the time taken for defect analysis, enhances decision-making accuracy, and improves overall software quality.

Keywords: Defect Triage, Root Cause Analysis (RCA), Intelligent Automation, Defect Knowledge Management, AI in Software Testing, Jira Integration



1. Introduction

Defect management is an essential component of software quality assurance. In large-scale projects, numerous defects are reported daily, requiring efficient triage to determine their root cause and assign appropriate resolution steps. Traditional defect triage involves manual scrutiny of historical defects and expert judgment, leading to delays and inconsistencies.

The **Intelligent Defect Triage Automation (IDTA)** system addresses these challenges by leveraging historical defect resolution data stored in the **Defect Knowledge Management (DKM) repository**. IDTA extracts new and open defects from Jira, evaluates their similarity against past defects, and automatically suggests resolution steps when relevancy exceeds a predefined threshold. This system enhances efficiency by reducing manual intervention, improving accuracy, and accelerating defect resolution cycles.

2. Key Concepts of IDTA

2.1 Defect Knowledge Management (DKM) Repository

- Centralized database storing historical defect information, including RCA steps.
- Continuously updated with resolved defects and associated resolution methodologies.
- Enables pattern recognition and AI-driven defect analysis.

2.2 Automated Defect Extraction from Jira

- IDTA extracts newly reported defects from Jira daily.
- Filters defects that remain in an **open state** for triage processing.
- Automates data ingestion, reducing the need for manual tracking.

2.3 Intelligent Triage and RCA Matching

- IDTA evaluates new defects against historical defects stored in the DKM.
- Uses AI-driven similarity analysis to determine if a new defect has **X% or higher** relevancy to past defects.
- If a match is found, IDTA automatically suggests resolution steps.
- If no match is found, triage is performed manually via the IDTA GUI.

2.4 Integration with Jira

- Once triage is completed, IDTA updates Jira with automated or manually determined RCA steps.
- Ensures real-time synchronization of defect information across systems.
- Facilitates better collaboration between development, QA, and support teams.

3. Challenges in Traditional Defect Triage

Despite the structured nature of defect management, traditional triage processes face several challenges:

- **Time-Consuming Analysis:** Manual triage requires extensive review of historical defects, slowing down defect resolution.
- **Inconsistent RCA Decisions:** Different team members may approach the same defect differently, leading to inconsistent resolutions.
- **Knowledge Retention Issues:** Expertise is often siloed within teams, making it difficult for new members to leverage past defect resolutions.
- **Scalability Constraints:** As projects grow, the volume of defects increases, making manual triage inefficient.

By addressing these challenges, **IDTA enhances defect management efficiency, reduces costs, and improves software reliability.**

4. Benefits of IDTA

- **Accelerated Defect Resolution:** Automating RCA recommendations significantly reduces the time taken to analyze and resolve defects.
- **Enhanced Decision Accuracy:** AI-driven pattern matching minimizes human errors and inconsistencies in defect triage.
- **Improved Knowledge Retention:** The DKM repository ensures that historical defect resolutions are always accessible.
- **Seamless Jira Integration:** Real-time updates enable better collaboration and traceability across teams.
- **Scalability:** IDTA efficiently handles large defect volumes, making it ideal for enterprise-scale projects.

5. Conclusion

The Intelligent Defect Triage Automation (IDTA) system revolutionizes defect management by leveraging AI-driven analysis and historical knowledge to enhance the efficiency of defect resolution. By integrating seamlessly with Jira and utilizing the DKM repository, IDTA ensures faster, more accurate defect triage with minimal manual intervention. As software projects continue to scale, adopting such intelligent automation solutions will be crucial for maintaining high software quality and operational efficiency. Future enhancements may include **predictive defect analytics, deeper AI-driven RCA insights, and expanded integrations with additional defect tracking tools.**

6. References

1. IEEE Std 829-2008, 2008, "IEEE Standard for Software and System Test Documentation,"
2. IEEE Transactions on Software Engineering, 2017, Menzies, T., Williams, L., & Zimmermann, T., "Automated Defect Prediction: A Systematic Review and Future Research Directions,".
3. 2003 Bach, J., "Exploratory Testing Explained," Software Testing and Quality Engineering Magazine.
4. 2020 Choudhary, R., & Kumar, P., "AI in Software Testing: The Future of Quality Assurance," ACM Computing Surveys.
5. Zhu, H., Hall, P., & May, J., "Software Unit Test Coverage and Adequacy," ACM Computing Surveys, 2018.
6. Kim, S., Zimmermann, T., & Whitehead, E. J., "Predicting Faults from Cached History," IEEE Transactions on Software Engineering, 2019.
7. Munir, H., & Sarro, F., "Defect Prediction using Machine Learning Techniques: A Systematic Review," ACM Transactions on Software Engineering and Methodology, 2021.
8. Hindle, A., & Godfrey, M. W., "Mining Software Repositories for Defect Prediction: A Case Study," Journal of Software Evolution and Process, 2022.
9. Hassan, A. E., "The Road Ahead for Mining Software Repositories," IEEE Software, 2023.