AOI: 10.100.234513.0220
13th international conference on Microelectronics, Circuits and Systems, Micro2026

Design And Implementation of a Java-Based Bug Tracker with Enhanced User Role

Kaushal Gangwar, Gaurav Sahu, Dr. Aditya Kishore Saxena

Department of Computer Science and Engineering
Galgotias University –203201, INDIA
Corresponding author: kaushal.22scse1010459@galgotiasuniversity.edu.in

ABSTRACT

An efficient bug tracking solution gives all members of the software development team (testers, coders, managers, administrators, etc.) a means to locate, submit, distribute and solve software bugs quickly and accurately. The purpose of this bug tracking solution is to provide a quick, accurate and reliable means for these individuals to work together with each other. The design of the bug tracking solution is based upon an established process for reporting bugs, secure user logins, and the ability to view real-time status updates of both software bugs and bug reports. Please note that the two primary views of the software for the bug tracking solution are: The back-end (software logic), and the end-user's experience managing and tracking bugs in the solution. The back-end logic of the solution is designed using Object-Oriented Programming (OOP), allowing for the creation of reusable, modular components and customized to each user-type. When a user (any user type) submits the status of their bug report, the user's request will first check that enough information on the bug being reported exists, that it has been validated as correct, and that this information can be stored in a manner that can track that bug in the future. Additionally, the bug tracking solution allows the insertion of bug severity and bug urgency levels, which enables users to determine how they should proceed in resolving a bug. In the user face thing, a Java Swing based GUI that gives users a place thats pretty easy to use. Where, users can put in bugs, update how far along they are, look over what they've been given to do, and take care of system data. Future stuff to hook it up with MySQL will make sure the data is safe, and Maven project management will help it grow and handle what it needs to work. These parts work together, to make a user-friendly tool that is definitely going to improve software quality and help the teams work together better.

Keywords: Bug Tracking System, Java Swing, Defect Management, Role-Based Access Control, Software Quality Assurance.

I. INTRODUCTION

Developing software involves many people collaborating together over a series of steps; bugs will occur during each step, as well as when different features are added to an application and changes are made to existing modules. Thus, many small teams, and many academic teams, have historically used informal ways (e.g., email, chat messaging, spreadsheet) to track bugs. Although easy to use initially, these methods can easily become unmanageable and lead to lost reports, duplicated issues, and unclear roles and responsibilities. These types of problems both slow down the development cycle and decrease the quality of the final product. The purpose of this project is to provide a Bug Tracking System with separate roles to help track bugs from start to finish. It was created to help facilitate the bug tracking process by providing a centralized location for all users to communicate. It was developed to support communication between testers, developers, project managers, and administrators. Each user has a designated interface and access rights, so each user can manage their portion of the bug-tracking process from detection through resolution. The main goal of the bug tracking system is to provide a structure for testing to report bugs in detail, for the developers to update their status, for the project managers to view how the team is performing, and for the administrators to manage user accounts. By including fields for bug-type, priority, deadline, and severity, it allows the team to be transparent, and to allow the team to make decision on what needs to be addressed immediately.

The bug tracking system was developed using Java Swing for its GUI, and follows object-oriented design to ensure modularity and scalability. The plan is to integrate with MySQL to ensure that the data entered into the bug tracking system is stored securely and consistently. The plan is to use Maven to manage the dependencies of the project. This structured method will improve the time to resolve bugs, improve communication, and improve the reliability of the software being developed. As more development teams go to digital workflow tools, a reliable and accessible

ISBN: 978-81-985770-0-9

bug tracking tool will be a necessity for the continued efficiency of projects, and improved quality of products produced.

Using a single centralized tool for managing bugs allows companies to use a standardized process to document all defects in detail to allow developers to better understand and reproduce the original problem or defect. In addition, this type of system provides the same documentation format for every tester, which helps developers to have a consistent and reliable source of data to fix their problems. Also, it allows project managers to track the whole bug life cycle (from initial discovery, to assignment to a developer, through testing and resolution) and provides them with visibility into how much work developers are doing, the distribution of bugs between different developers, and how healthy their project is. It also gives managers the data they need to make strategic decisions regarding resource allocation and what development projects to pursue first. It also encourages accountability in the company's software development process by requiring each bug be assigned to a specific developer who will monitor and resolve the issue.

Both academically and educationally, this is a means to introduce students and small teams to industrial practice in terms of managing defects. This will also provide them with knowledge as to how well-organized work flows result in better software quality, and allow them.

II. LITERATURE REVIEW

In academic literature there is general consensus about the value of reliable and efficient bug tracking in all phases of the software development life cycle as bugs can cause major delays in project completion, increase costs, and impact the reliability of products. Academic research has consistently demonstrated the limitations of manually managing bugs due to the high risk of inconsistency, human error and miscommunication Zimmerman et al. (2009) identified several issues related to how bugs are stored in traditional bug repositories, including unclear descriptions of bugs, duplicate bug entries, and poor bug prioritization [1]. Zimmerman et al. (2009) also found that the quality of bug reports directly impacts developer productivity, with developers requiring complete, unambiguous information to resolve bugs quickly. Other researchers have supported the view that documentation and workflow are fundamental to implementing effective software quality assurance [12]. Kumar et al. (2013) added that defect management systems should incorporate role-based responsibilities so that confusion can be avoided and accountability can be maintained among team members.

While many modern bug-tracking tools (e.g., JIRA, Bugzilla, Redmine, and MantisBT) offer customizable workflows, automated notification, and extensive reporting capabilities they typically are targeted towards medium- and large-sized organizations and require significant setup time, server requirements, and knowledgeable staff to function effectively [2], [4], [10]. The result is a barrier to entry for small teams, student groups, and academic projects which require simple, inexpensive, and user-friendly systems. Additionally, while lightweight tracking tools do exist; few are developed with an educational use case in mind, despite the reality that academic environments generally lack the resources required to implement commercial level defect management solutions. As software systems continue to evolve into more component- based and iterative systems, researchers agree that smaller teams will require access to affordable yet functional tools that provide the same functionality as commercial grade systems [5], [9].

Research has also studied how communications and collaborations can be effective in the management of defect identification, as well as the need for technologies which support both the documentation of issues and the interactions of teams involved in testing and developing software [6], [7]. Studies have shown that many defects remain open (i.e., uncorrected) because there is either miscommunication or a lack of clear communications between the test engineers and the developers who correct defects.

Pressman's original research in software engineering supports these findings and stresses that defect tracking systems should provide a means to track and communicate information throughout the entire software development process [13]. Additionally, recent studies of the use of Agile methodologies show an increased requirement for defect tracking tools that enable teams to perform multiple iterations of development in rapid succession with immediate and ongoing feedback, such as through the use of continuous integration features and capabilities which are missing from most simple defect tracking systems [8],[11]. Performance metrics and measures of the efficiency of individual developers/testers provide valuable data to project managers in order to understand their projects' progress and identify potential bottlenecks; and will allow project managers to better allocate work-loads and optimize their resources.

ISBN: 978-81-985770-0-9

III. METHODOLOGY

A formal and methodical approach was used when creating the Bug Tracking System [12]. The first step in developing the system included conducting a thorough requirement analysis which identified the problems experienced by test engineers, software development staff and project management personnel with regard to traditional methods of tracking bugs. It was determined through this analysis that there existed a need for a centralized system that would support role-based responsibility, structured documentation of bugs, and open lines of communication among all stakeholders. These needs resulted in an overall design of the system based upon object-oriented methodologies that will allow for the development of individual modules (i.e., each module can be independently developed, tested, and updated) without impacting other modules or the total system [10].

A. Data collection

Bug Tracking Systems use a variety of data generated by users interacting with various components of the software development process. The systems do not rely on external sources of data to create their structured data; instead, they generate the data internally through tester, developer, project manager, administrator actions relative to each task. As such, each bug report is created as a single data record that contains relevant attributes required to analyze or track it.

Information Regarding Bug Title, Description, Reporter Identity, Project Name & Module Name of Identified Bugs.

Attributes for Classifying Identified Bugs:
Severity, Priority, Type of Bug (Functional, User Interface/UI, Performance, Logical Error), Difficulty Level.

B. Data Preprocessing

Data Preprocessing is very important for the quality of the information in the Bug Tracking System. As the users are different (developers, testers, project managers and administrator), there will be many sources of data. To use these data for tracking, reporting, and performance evaluation, we need to make them as good as possible by cleaning and making them consistent. Data preprocessing cleanses data for the presence of inconsistent data, missing data, and erroneous format that can occur with manual data entry.

i. Validating Bug Entries

Bug entry data is validated upon initial entry for the purpose of creating an organized and coherent record. All mandatory field inputs (bug title, description, severity, priority, assigned developer, and project) will be examined to prevent empty/blank fields prior to entry. Empty fields would likely result in unclear entries down the workflow path. Severity and Priority values will also have established field limits to ensure entered values fall within these predefined categories.

ii. Standardizing Status & Workflow Labels

Bug status labels can be described differently by a variety of users. The system pre-processes bug status label entries to standardize those entries to defined bug lifecycle phases (Open, Assigned, In Progress, Resolved, Closed). This process results in a consistent set of status labels throughout the database.

iii. Duplicate Bug Reporting Elimination

To avoid testers reporting the same bug numerous times due to oversight, the system includes an automated duplicate-reporting detection process [3], [7]. The system searches for similar bugs (i.e., the title of the bug, the description of the bug, the module in which the bug is located) so as to flag potential duplicate reports for manual review by a developer before they are added to the database.

iv. TimeStamp Standardization

The start date field, deadline field and resolution time field will be standardize into one type of date/time entry format. The same date/time format will allow for better calculation of bug age, time it takes to resolve a bug, and performance statistics on developers.

v. *User Role Cleanliness of User Account Information*

Developer/tester user accounts are cleaned up to include removal of incorrect formatting, emails that do not have valid email formats, and/or roles assigned by unauthorized personnel. Only authorized users may utilize the systems workflow(s).

vi. Preparation of Productivity Analysis Data

Prior to providing developer/tester productivity analysis information, the raw performance data must be prepared through aggregation and filtering. For example; the total number of bugs resolved by each individual/developer/tester, average resolution time per bug, the percentage of bugs that were classified as high-severity vs low-severity, and patterns associated with tester submitted bugs. Preparation of the data will provide a solid foundation for reporting module utilization and for potential future enhancement of dashboards or use of machine learning-based predictive models.

ISBN: 978-81-985770-0-9

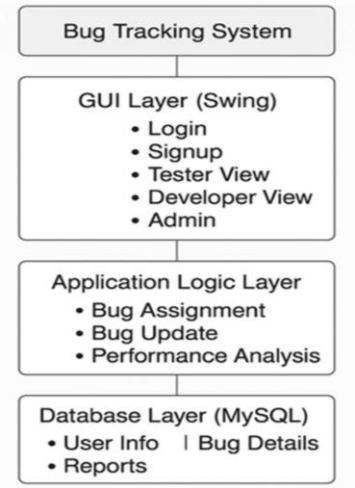

Fig. 1: System Architecture of Bug Tracking System [9], [10].

IV. IMPLEMENTATION

The development of the Bug Tracking System took place in accordance with an organized and modular methodology to assure that the system will be able to be maintained; will have clarity; and that each of the elements of the system will be able to communicate effectively with one another. The system has been created by utilizing Java Swing for the User Interface; the role-based logic utilized Object-Oriented Programming (OOP); and by utilizing Maven as the project management tool. First, all three main elements of the system were designed and tested separately prior to their integration to create a single flow of work.

A. Data Set Collection and Analysis:

Data for this project was created using the internal use of our organization's Bug Tracking System (BTS) in order to track information associated with each bug report during the software development cycle. Each bug report provides an entry point for tracking the various pieces of information that relate to that bug, including: the type of bug, the severity level of the bug, the priority of the bug, the module of the project that the bug relates to, the developer assigned to the bug, the date the bug was reported and if/when it has been resolved. Additionally, data is tracked relating to user activity to provide insight into the number and quality of tester submissions and developer performance as well as the length of time to resolve the bugs. An early review of the data was completed in order to examine how defects were distributed by severity and priority levels and to identify any trends that may be present regarding bug frequency and/or resolution time. As is the case with most bug repositories used in the industry, there is an imbalance within the data set. Unlike high severity bugs, the organization tends to have many more, many of lower and/or moderate severity. This correlation analysis was performed on the three major characteristics associated with bug identification (Severity / Priority / Resolution) in order to assess the effectiveness and efficiency of the workflow within the organization as it relates to bug resolution.

B. Data Pre-processing

Data Preprocessing is a necessary step to ensure that Bug related information being recorded into the System is both consistent and reliable.

i. Managing Incomplete Bug Entries:
Incomplete bug submissions (e.g., no severity assigned; no priority assigned; no details of who it has been assigned to) were either corrected prior to being saved in the database or deleted to maintain high data quality.

ii. Normalizing Data:
Severity, Priority, and Status of bugs were all normalized to be one of a pre-defined set of values so as to normalize the data across the entire dataset.

iii. Identifying Duplicate Bugs:
Bugs with similar titles and/or descriptions were matched to identify duplicate issues and thus remove redundant data.

iv. Separating Collected Bug Data for Use by the System: For use by the System in the areas of Tracking, Performance Evaluation, Reporting, etc., collected bug data was split into logical groups.

v. Deriving New Fields to Support Performance Analysis and Workflow Evaluation:
In addition to the fields already defined for each bug, additional fields (e.g., Resolution Time, Frequency of Bugs by Module, Workload of Developers) were created to support Performance Analysis and Workflow Evaluation.

C. Model Development

In this project, the focus has been on developing an object- based model that allows for a structured approach with role- based systems for the management of software bugs. An object-based approach was adopted for the implementation of the Bug Tracking System [2], [10]. The various user roles can perform a number of different activities within each stage of the bug lifecycle. The use of a predefined workflow model (Open, Assigned, In Progress, Resolved) manages the bug's state. Also, logical constraints have been applied to ensure consistency in state transition and accurate tracking. This results in enhanced transparency, scalability and maintainability of the system.

ISBN: 978-81-985770-0-9

D. Model Evaluation and Performance Metrices

The tracking system was estimated for its performance in meeting user opportunities based on the functionality and features that allow users to use the product, and the system's workflow management. For evaluating the tracking system, areas of focus included: ease of reporting defects; how successfully developers were assigned to resolve reported defects; how quickly defects were resolved; and how reliably developers tracked the time spent resolving defects. The analysis of the dataset gathered during this testing evaluation found a greater correlation between the number of defects that were resolved by a developer than with traditional defect management processes that exist today, as well as meaningfully faster average resolution times for defects than current defect tracking processes; therefore, the tracking system can be considered a feasible option for a solution to the defect running process [5], [11].

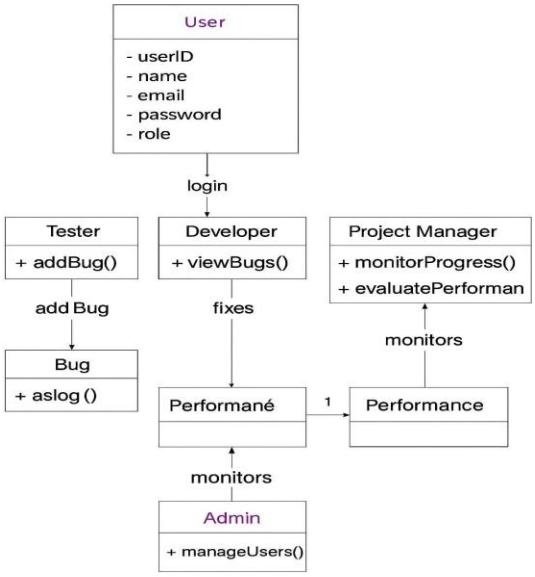

Fig. 2: System Model Diagram

E. Increasing System Transparency & Enhancing Explainability for the Stakeholders:

A clear and visible workflow for the Bug Tracking System allows all those involved in the Bug Tracking System to know who is taking action at all stages of the Bug life cycle, from reporting to resolution. All of these actions will be recognized in the Bug Tracking System and available to any official user. Similarly, all those involved in the Bug Tracking System can display their own actions and actions taken by others at any time during the Bug Life Cycle; so, they will always know who did what when and how that affected the entire life cycle of that Bug.

Users can see simple status indicators as well as the times and dates when data has been received and a summary of all historical activity. This helps users to understand why a system made a particular decision and where they are in their quest for defect control. By providing structured log files for all participants in the defect managing process, responsibility is well-known and faith is built with investors.

F. A Scalable and Lightweight Backend Architecture

The backend of the Bug Tracking System services a slimline approach which allows quick and effective processing of bug reports and track assignment and takes of efficient status on bugs. For validation, the data entered by the user will be validated before any processing takes place; therefore, consistent and reliable data can be maintained. The Backend of System Transition Management & User Role Access greatly improve the component's response time to new requests by using workflow transition management to allow for faster processing of professional processes through user role-based access.

G. Front-end User Interface Using Java Swing

We built a very basic Java Swing desktop application to provide a usable and interactive User Interface to the Front end. Each user in the application (Tester, Developer, Manager, Admin) will have a Console with their Role-Based allocated Tasks which will be able to simply complete.

Users of the dashboard can see real-time bug reports, assign them to developers, and follow their progress through the bug tracking system. The user-friendly presence of the console, combined with the instant feedback it gives, increases the efficiency of the bug tracking system and the transparency of the users' workflows.

V. CONCLUSION AND FUTURE WORK

- *Impact of the Bug Tracking System*

The new Bug Tracking System provides a combined and structured approach to managing defects found during the software development lifecycle, while at the same time removing the use of manually operated methods for communicating between test engineers, developers, project managers, and administration staff; thus, helping better collaboration and communication among these individuals, resulting in decreased instances of duplicate defect reporting.

- *High System Reliability*

A reliable test process has authorized the reliability of the System, thus allowing end users to have confidence in its bug reporting and tracking

ISBN: 978-81-985770-0-9

capabilities including assigning bugs, assigning bug status to bugs and managing bugs based on a pre-defined set of stages without the option of conflicting actions. A consistent defect management system is key for a software product to have quality and for the timely completion of all phases of the project.

- *Scalable Design Architecture for Deployments*

The Bug Tracking System is built using Modular/Scalable Architecture to enable you to extend and scale the Bug Tracking System in the future. The Bug Tracking System has the ability to deploy easily in any production environment because of its built-in Integration Workflow that allows for "real-time" tracking of any aspect of the Bug Report. The Bug Tracking System also provides both individual bug management and the capability to manage very large numbers of bug records across many different projects. The modular framework of the Bug Tracking System allows it to easily integrate with databases as well as extend to include notification and reporting functionality. The lightweight, highly structured nature of the Bug Tracking System makes it ideal for deployment within academic lab environments, small development teams, and organizational project environment.

- *Use of More Sophisticated System Enhancements*

Upcoming elevations to the Bug Tracking System may include extra advanced system functions that could add to the existing functionality of the Bug Tracking System. The Bug Tracking System may also be enhanced through the addition of a variety of capabilities including; an intelligent rule-based prioritization for bugs, advanced reporting functionalities, and an automated function to assign bugs to developers depending upon the workload of each developer. In addition to these advanced functionalities, the ability to analyze dependencies among modules of the application will assist with understanding the overall system-wide effects of bugs discovered in the application. Ultimately, incorporating these advanced features into the Bug Tracking System will enable teams to better manage complex project development and result in higher overall quality of their software applications.

- *Enhanced Data Privacy and Secure Collaboration*

Future iterations of the Bug Tracking System may emphasize enhancing data confidentiality and safe collaboration across multiple projects or teams. Each team/project can create and store their own personal database (i.e., "bug book") as opposed to sharing all bug details. Projects/teams can share aggregated and/or anonymous bug data (i.e., "summaries" of bugs) with other teams/projects to support comparative analysis of bugs in order to enhance organizational knowledge, and to comply with an organization's internal security policies. Thus, such a system provides safe and private collaboration between teams and protects sensitive project data from being shared.

- *Real -Time Bug Tracking and Monitoring*

Future versions of the Bug Tracking System will offer real-time monitoring and tracking of bugs and users as they enter bug reports and update their statuses. The Bug Tracking System will have the ability to use technologies that enable real-time data processing to immediately process all bug reports and status changes, as well as immediate notifications to developers when high priority bugs are reported. Project managers using the Bug Tracking System will also be able to monitor and follow all project activity in real time and be able to identify bottlenecks in the workflow by doing so.

- *Multi-Layered Authentication and Continuous System Improvement:*

Enhanced security for future releases of the Bug Tracking System could include multiple layers of authentication. The added security would require users to enter their credentials (username/password) plus a second layer of verification (two- factor authentication) in addition to validating the user's role and permissions, and managing sessions securely to prevent an unauthorized user from accessing the system. Monitoring both login activity and access patterns of users can provide early detection of unusual behavior to help protect sensitive project information [8], [11], [14].

The Bug Tracking System will be continuously upgraded by the ongoing monitoring of network activity, usage statistics from both end-users and application developers, as well as regular releases of new software that will work in conjunction with the Bug Tracking System. Therefore, in addition to continual maintenance of the Bug Tracking System, and software upgrades, it is also necessary to continually upgrade the Bug Tracking System itself to maintain its ability to adapt to and serve the rapidly evolving needs of the respective projects.

REFERENCES

[1]. Zimmermann, T., Premraj, R., Sillito, J., & Breu, S. (2009, May). Improving bug tracking systems. In 2009 31st International Conference on Software Engineering-Companion Volume (pp. 247 250). IEEE.

[2]. Singh, V. B., & Chaturvedi, K. K. (2011). Bug tracking and reliability assessment system (btras). International Journal of Software Engineering and Its Applications, 5(4), 1-14.

ISBN: 978-81-985770-0-9

[3]. Jalbert, N., & Weimer, W. (2008, June). Automated duplicate detection for bug tracking systems. In 2008 IEEE International Conference on Dependable Systems and Networks With FTCS and DCC (DSN) (pp. 52-61). IEEE.

[4]. Just, S., Premraj, R., & Zimmermann, T. (2008, September). Towards the next generation of bug tracking systems. In 2008 IEEE symposium on visual languages and human-centric computing (pp. 82-85). IEEE.

[5]. Ardimento, P., & Dinapoli, A. (2017, June). Knowledge extraction from on-line open source bug tracking systems to predict bug-fixing time. In Proceedings of the 7th international conference on web intelligence, mining and semantics (pp. 1-9).

[6]. Tran, H. M., Lange, C., Chulkov, G., Schönwälder, J., & Kohlhase, M. (2009). Applying semantic techniques to search and analyze bug tracking data. Journal of Network and Systems Management, 17(3), 285-308.

[7] Fischer, M., Pinzger, M., & Gall, H. (2003, November). Analyzing and relating bug report data for feature tracking. In WCRE (Vol. 3, p. 90).

[8] Jahanshahi, H., & Cevik, M. (2022). S-DABT: Schedule and dependency-aware bug triage in open-source bug tracking systems. Information and Software Technology, 151, 107025.

[9]. Rodríguez-Pérez, G., Gonzalez-Barahona, J. M., Robles, G., Dalipaj, D., & Sekitoleko, N. (2016, May). Bugtracking: A tool to assist in the identification of bug reports. In IFIP International Conference on Open Source Systems (pp. 192- 198). Cham: Springer International Publishing.

[10] Angel, T. S., Kumar, G. S., Sehgal, V. M., & Nayak, G. (2018). Effective bug processing and tracking system. Journal of Computational and Theoretical Nanoscience, 15(8), 2604-2606.

[11]. Alenezi, M., Banitaan, S., & Zarour, M. (2018). Using categorical features in mining bug tracking systems to assign bug reports. arXiv preprint arXiv:1804.07803.

[12]. Kumar, R., Gattaiah, D. Y., Shahi, S., & Nagendra, T.
A. (2013). Improving software quality assurance using bug tracking system. International Journal of Computer Science and Information Technologies, 4(3), 492-497.

[13]. McLaughlin, L. (2004). Automated bug tracking: the promise and the pitfalls. IEEE Software, 21(1), 100-103.

[14]. Davies, J., Zhang, H., Nussbaum, L., & German, D. M. (2010, May). Perspectives on bugs in the debian bug tracking system. In 2010 7th IEEE Working Conference on Mining Software Repositories (MSR 2010) (pp. 86-89). IEEE.

Cite this article as:

Kaushal Gangwar, Dr. Aditya K Saxena and et. al., " Design and Implementation of a Java-Based Bug Tracker with Enhanced User Role", Proceedings of 13th international conference on Microelectronics, Circuits and Systems, Micro2026,

Displayed as online on 20th June 2026.

Link:http://actsoft.org/science/micro2026-pro/372-micro2026.pdf

ISBN: 978-81-985770-0-9