



Bug Vision: Your Path to Seamless Debugging

Kowshika V¹, Namita R², Bhavani N³, K Tejashree ⁴, Shyji⁵

^{1, 2, 3, 4, 5} Sri Sairam College of Engineering, Bangalore, Karnataka, India.

ABSTRACT

Over the past several decades, formal mechanisms to track bugs have been embraced mostly in large software development firms although smaller teams normally sidestepped organized mechanisms completely. Rather, they used shared spreadsheets, simple lists of tasks or even an email stream to keep track of errors and bugs. This, much more often than not, turned out to be ineffective, time-consuming and full of mistakes. This gentle casualness made it possible to become complacent and let minor ones go because they rarely had the spotlight that would have made it worth getting all worked up about it.

The bug-tracking systems are capable of providing structured workflows at every stage of the bug resolution which also enhances productivity and promotes accountability and transparency by recording every action performed to resolve a given issue.

The implementation of a Bug Tracking System has numerous advantages to organizations such as enhanced efficiency of the defect resolution process and increased communication between the teams. This serves the purpose of accountability, reporting, communication within the team, and even motivating the members of the team by marking the completed tasks as satisfactory. Additionally, it brings the advantage of better management due to better historical records and report information which in turn enhances product quality as well as improves customer satisfaction by constantly improving the processes.

1 Introduction

Definition of Bug Tracking

The Need for a Bug Tracking System

As the software complexities began to grow rampantly, the need to keep track of bugs effectively heightened. A Bug Tracking System (BTS) provides:

- All Documentation Required: Each bug is recorded along with its details.
- Proper Levels Defined: Teams can make use of the levels as per their severity.
- Controlled Process: Bugs are managed right from identification to closure enhancing responsibility.
- Enhanced Reporting: Senior systems present data that helps to determine trends.

Traditional Methods and Their Flaws

In the past, most teams made use of age-old techniques such as bug lists on shared spreadsheets and management via email. This is fine for smaller projects, but such methods have major limitations:

- i) Human Error: Bugs are often missed with manual tracking.



- ii) No Clear Order: Bugs which are not enforced will likely be ignored.
- iii) Difficult to Work Together: Informal means do not facilitate teamwork in real time.
- iv) Inadequate Data Storage: Conventional approaches don't employ data analytics over a long period of time.

Current Solutions and Their Benefits

Many bug tracking systems exist today, and they all have something different to offer. They include the following:

- JIRA: Uses complex and numerous project management tools and includes a bug tracking feature.
- Bugzilla: Simple open-sourced software application.
- Azure DevOps: Bug tracking is added on top of version control and project management.
- Trello: Simple defect tracking using a well-designed and configurable graphical interface tool.

2 Literature Review

With the advent of technology in above business processes, tracking and Bug Fixing system has become an integral part of software development processes, as it allows a work group to trace and fix defects with ease. The purpose of this paper is to review the literature available in practice and in fiction concerning the tendencies, impact, problems, and recommendations on the use of the Web based Bug Tracking Systems (BTS).

Historical Background and Development Philology: Tracing the history of Bug Tracking one can pinpoint the change from simple means (like using spreadsheets, emails) to the adoption of automated approaches. First instruments study looks the present-day complacency elements where no surprise the apparatuses such as Bugzilla, JIRA, Azure DevOps were birthed to and help foster the development of software (Huang et al. 2018; Petersen & Wohlin 2019).

Evaluation of Devices: A great number of works aim to evaluate the use of different systems of bug tracking, making research literature more than ample. The review suggests that while systems such as JIRA and Bugzilla are effective in managing and tracking software bugs, JIRA is often preferred for use by big agiles teams because of its advanced integration features and ease of use when compared to Bugzilla (Smith & Johnson, 2020; Lee et al., 2019). Options for modifications had a great impact on consumers as they enabled teams to adapt systems to their processes (Chen & Wang, 2019).

Performance in Program development: Numerous studies have always demonstrated that a structured software bug tracking system more often than not brings out positive results in terms of software quality and applicative bug resolution speed. Reports show that.

Organizations who employ these systems spend less money on maintenance and enjoy high levels of user complementation because most of the bugs are resolved with no delay (Garcia et al., 2021; Patel et al., 2019). Systematic approach provides means of looking in advances,



because of that it improves the operations of a firm by contributing towards better tracking of such problems (Kumar & Singh, 2021).

Challenges and Limitations: Embracing the use of a BTS has its difficulties, for instance, team members refusing to abandon their preferred methods of informal tracking (Martin & Smith, 2020). Some systems are so complex that they tend to intimidate the users, thus causing under-use (Patel et al., 2019). Again, such systems may promote dependency which for instance can limit or do away with adverse informal communication which is very important in Agile practices (Hossain et al., 2021).

Best Practices: Bugs have annoying tendencies. To reduce these tendencies, the following best practices in the utilization of bug tracking systems should be adopted:

a. Clear Reporting – Explain the bugs in details along with the steps to reproduce the bugs (Kumar & Singh, 2021).

b. Prioritization – Based this on the severity of the bugs and try to fix the most critical ones first (Lee et al., 2019).

Ongoing Training – Train at intervals to keep users motivated (Martin & Smith, 2020).

Software Testing and Agile Methodologies: The main objective of effective bug tracking is the process of Agile and DevOps enabling the quick return of feedback to the cycle for ongoing extraneous enhancements (Hossain et al., 2021). The incorporation of BTS into CI/CD pipelines make the process of development more effective because defects can be located and fixed in less time (López et al., 2020).

Future Trends: Trends suggest that the future of online BTS would be influenced by technological advancement. There has been an increased interest in using Automated Software Testing Tools that employ AI and machine learning to reduce the time spent on Bug finding and prioritization (Zhang & Zhao, 2022). Also, there is information predicting analytics that will help in most cases anticipating the patterns of bugs and help in fixing the problem ahead of time (Garcia et al., 2021).

3 Proposed Solution

- i. **Simplified and User-Friendly Interface:** Create an approach focused on user interface with simple navigation, adaptable panels and ease of use via implementation of step-by-step introduction tools.
- ii. **Smart Prioritization Using Machine Learning:** Use machine learning for efficient bug triage analysis algorithms that determine priority levels for every incoming report based on its potential severity and internal data on past occurrences availability and use crush classifier planning.
- iii. **Legacy Bug Management Module:** Provide a separate module that manages the legacy defects providing classification and problem types related to some already resolved problems and management of the process of their fixing.



- iv. **Continuous User Feedback Loop:** Integrate a feedback tab into the design so that users can report issues and make requests for changes, ensuring that system maintenance can be done in a timely manner.

Implementation

1. Bug Reporting: The software gives users a simple and easy to use interface to begin a bug tracking process by filling a report. The interface is made up of clear function and customizable windows allowing users to key in what is necessary like the type seriousness level and attachments of the bug with ease.
2. Automated Data Processing and Pre-Classification: After a report has been made, the BTS uses NLP techniques in the processing of the report to carry out keyword extraction and severity of the bug. The core processing entails the use of machine learning algorithms that reinstate smart prioritization by analyzing trends from historical data on how each report has been treated based on the intensity of appeal.
3. Duplicate Detection and Clustering: The new submission is compared to the system's library of reports in order to limit report stupidity. Related problems are solved by clustering techniques so that developers work on development of unique solutions to the problems with no or minimal overlaps.
4. Dynamic Prioritization and Assignment: This helps in assigning severity levels to the bug reports with the use of incorporated machine learning based algorithms that are used in defence thrust operations assessing the likelihood of manifestation of the bug and the internal history of previous cases. Thus they are able to pin point an appropriate developer or team in most cases depending on the kind of issues that have been raised and the burden of work that they have at that particular time.
5. Bug Resolution Workflow: Assigned developers will investigate and fix the bugs, reporting back into the simplified interface on the status. Team work enhancing tools and platforms are installed in the system thus in the course of bug fixing, team members can communicate instantly, share ideas, and give progress reports.
6. Testing and Verification: As soon as the fix is applied, the defect is passed for verification to quality assurance (QA). The optimized process allows the testers to evaluate the solution in no time and all the evaluation is incorporated in the endless user feedback system.
7. Bug Closure: In cases where a bug is validated and confirmed with the fix, the BTS systems recorded the bugs as closed meaning resolution of the bugs was successful. There's an option, however, for the users to 'reopen' the bug in question in case the problem still or again exists for better management practices.
8. Feedback and Continuous Learning: Likewise, system design includes a feedback tab, which makes it easier for users to give reports and recommendations on the system. This is so that the system can gather such intel from the users, on a constant basis, which is aimed at enriching the systems' structures and efficiency.



9. Reporting and Analytics: The BTS has an inbuilt reporting system that provides data on bug statistics, average bugs resolution time and the overall performance of the team. The analytics are then translated into visually appealing dashboards for managers enabling them to deploy resources and improve the processes accordingly.

10. System Maintenance and Updates: Administration of the system requires periodic system upgrades which ensures security from new threats and system performance optimization. The legacy bug management system solves and manages all bug fixes that have previously been resolved. The BTS system allows for continuous improvement and evolution of the system with the provision of machine learning models to users and their dynamic demands over time.

4 Conclusion

Utilizing a Bug Tracking System (BTS) is critical for improving the productivity and effectiveness of software development. Moving from a less formal approach to a systematized tracking system helps organizations ensure that they do not overlook crucial bugs that threaten the success of the project. This systematized contribution improves communication, cooperation, and responsibility of developers helping to keep everyone in the loop with what is happening on the issues at hand. Strictly dealing with and controlling every defect minor or major with the help of BTS influence product quality positively. Here, the size of the development cycles is reduced thereby increasing productivity levels. In the end, a Bug Tracking System leads to positive results of any given project since the most important quality of the software is maintained and delivered to the end users.

5 Result

As a result, product advancement is enhanced, there is an uplift in user contentment and the entire process of development is less cumbersome. The organization enjoys the reduced risks and the heightened emphasis on improvement which leads to increased output and success of projects. This approach goes beyond solving present challenges as it also sets the pace for future developmental verticals. In addition, features like reporting and analytics, which help teams to spot patterns and the recurrence of issues, which aids in improving the development goes a long way in most online BTS systems. Moreover, an online Bug Tracking System improves the quality of a product and the satisfaction of the clients and instills a quicker and more responsible approach to development activities, which, in turn, fosters the success of the entire project.

References

1. Gopal; Madhwaraj Kango; Govindaraj; Paramita Chandra; Prathiksha Shetty; Sunny Raj, Year: 2022, "Bugtrac-a new improved bug tracking system," in IEEE Delhi Section Conference (DELCON), pp. 1-7.
2. Khan; Kumar, Year: 2020, "An Expert System Framework for Bug Tracking and Management," 2020 International Conference on Information Science and Communication Technology (ICISCT), Karachi,



Pakistan, doi: 10.1109/ICISCT49550.2020.9080048, pp. 1-9.

3. Sharma; Shatakshi; Palvika; Dagur; Chaturvedi, Year: 2018, "Automated Bug Reporting System in Web Applications," 2018 2nd International Conference on Trends in Electronics and Informatics (ICOEI), Tirunelveli, India, doi:10.1109/ICOEI.2018.8553850, pp. 1484-1488.