Debug O Bug Using Web Development

Asmita Dixit[1], Ajay Anand[2], Akash Yadav[3], Ravi Singh[4]

[1]Assistant Professor, Dept. of IT, ABES EC, UP, India
[2]Final yr Student, Dept. of IT, ABES EC, UP, India
[3]Final Yr Student, Dept. of IT, ABES EC, UP, India
[4]Final Yr Student, Dept. of IT, ABES EC, UP, India

---***---

Abstract - *Develop a web application to address and recover bugs in system and software to solve any particular problem or task. It significantly increases coding time by drastically reducing debugging time there. It will help solve the efficiency and time consumption of any automation testing technique with the help of pseudo-random generators built in many languages and with some algorithmic skills and data structures. It can also be integrated into any popular text editor to allow programmers to find bugs on the go. The software stack used for web application architecture is MERN Stack i.e., Mongo for No-sql database, Express for javascript optimization, React Javascript for user interfaces creation, and intense graphical UI and Node JS for back-end programming.*

***Key Words*: Bugs, Debugging, Software Stack, javascript, MERN Stack, etc…**

1. INTRODUCTION

If you're a developer for Word Press or not, you 're just searching for opportunities to better yourself. That is what our profession is all about. We are still looking for opportunities to make things stronger.

That being said, having developers hate debugging is not uncommon. They 're going to get a bug report and yell out in exasperation, "I want to make progress! Just correcting glitches! "But that's not the best mindset for glitches and general debugging.

That's because debugging isn't just when you fix bugs in bug reports. When writing brand new programming, we spend a lot of time debugging too. How often did you write code that wasn't working on the first try and had to debug it?

Because of this, Why debugging is such an important developmental skill. We 're spending an absurd amount of time debugging. Yet we're never thinking about how it could help us write better code too.

Debug O bug is a great web application where every competitive programmer or developer can use it to find undesirable bugs in the piece of software or program.

We need to look at the bugs themselves to understand better why debugging is such a useful skill. What's bugging? And how do they tie to the development of software?

But first, we'll start by looking at the very word "worm" itself. Why do they do that? In the manner in which we use the word "worm" is in itself fascinating. More often than not, when we think of a bug, we think of something that has appeared by itself. That we were not liable for it anyhow. But, deep down, we know that is not real.

The reality is that man makes all bugs because all of the software is made by man. (We haven't quite reached the singularity yet!) So, as much as we don't want to admit it, we 're the cause of the software bugs.

If we pursue this line of thinking, we might conclude that the mistakes we made are bugs. If bugs are mistakes that we made, it's fair to say we can also learn from those mistakes. That is an excellent excuse to find glitches and debugging as an ability to understand.

2. Proposed Method

2.1Generator Test Case

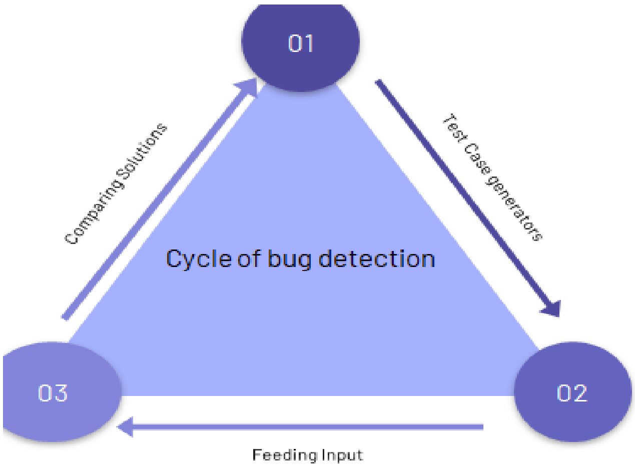
One process of our project would produce extremely likely random test case variables that are more likely to fail on programs. A potent pseudo-random generator will do this.

2.2Comparing answers

A script written in bash or js node lets users equate different solutions to the same problem with a vast number of test cases

Built-in Phase 1

2.3Internal Employment



3. Related Work

Software systems are now an essential element of our everyday life. Testing To ensure device consistency and operation The Software Creation Life Cycle (SDLC) practices were paramount. Indeed, software bugs can potentially bring dramatic consequences if the product is released without testing it to the end-user. The task of software testing is to check that the real result and the anticipated outcome are compatible and that the program is implemented without bugs. Many methods, strategies, and tools have been suggested to help ensure the device is free of defects

The field research goal was to get an understanding of professional debugging in modern tech firms. This understanding was required to construct a general debugging questionnaire in Germany.

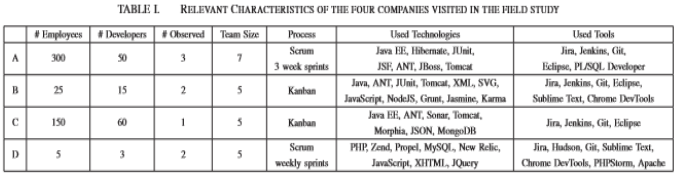


TABLE I. RELEVANT CHARACTERISTICS OF THE FOUR COMPANIES VISITED IN THE FIELD STUDY

	# Employees	# Developers	# Observed	Team Size	Process	Used Technologies	Used Tools
A	300	50	3	7	Scrum 3 week sprints	Java EE, Hibernate, JUnit, JSF, ANT, JBoss, Tomcat	Jira, Jenkins, Git, Eclipse, PL/SQL Developer
B	25	15	2	5	Kanban	Java, ANT, JUnit, Tomcat, XML, SVG, JavaScript, NodeJS, Grunt, Jasmine, Karma	Jira, Jenkins, Git, Eclipse, Sublime Text, Chrome DevTools
C	150	60	1	5	Kanban	Java EE, ANT, Sonar, Tomcat, Morphia, JSON, MongoDB	Jira, Jenkins, Git, Eclipse
D	5	3	2	5	Scrum weekly sprints	PHP, Zend, Propel, MySQL, New Relic, JavaScript, XHTML, JQuery	Jira, Hudson, Git, Sublime Text, Chrome DevTools, PHPStorm, Apache

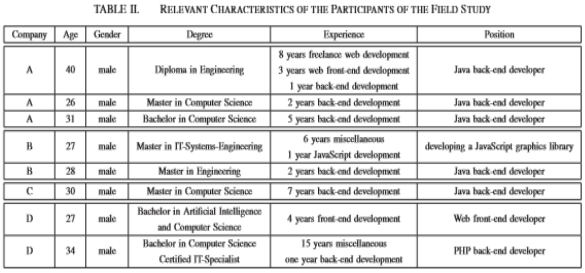


TABLE II. RELEVANT CHARACTERISTICS OF THE PARTICIPANTS OF THE FIELD STUDY

Company	Age	Gender	Degree	Experience	Position
A	40	male	Diploma in Engineering	8 years freelance web development 3 years web front-end development 1 year back-end development	Java back-end developer
A	26	male	Master in Computer Science	2 years back-end development	Java back-end developer
A	31	male	Bachelor in Computer Science	5 years back-end development	Java back-end developer
B	27	male	Master in IT-Systems-Engineering	6 years miscellaneous 1 year JavaScript development	developing a JavaScript graphics library
B	28	male	Master in Engineering	2 years back-end development	Java back-end developer
C	30	male	Master in Computer Science	7 years back-end development	Java back-end developer
D	27	male	Bachelor in Artificial Intelligence and Computer Science	4 years front-end development	Web front-end developer
D	34	male	Bachelor in Computer Science Certified IT-Specialist	15 years miscellaneous one year back-end development	PHP back-end developer

All four companies produce web applications, some are self-hosted, and some are licensed. Throughout their day, we could follow eight developers, and observe their methods. To order to get an understanding of their processes, we asked each developer to think aloud. We asked each developer at the end of each visit to explain his overall method themselves. We also asked if they understood existing devices, such as back-in-time debuggers, and found them useful.

An overview of each company's relevant features is given in Table I. It shows the number of employees for each company, the number of software developers, the number of developers we observed and the usual size of the company's teams, as well as the process of development they used, the technology they developed and the tools they used. These lists are not exhaustive but show the tools and techniques that we were able to see during our visits. Apart from that info, it's worth noting that the third company is part of a larger web-oriented sector.

Table II presents an overview of the relevant features of the individual participants. We asked for their age, highest educational degree, and software development experience. We also noted their position as regards gender and current position.

4. Result

Computer applications today are becoming an integral part of daily life. Testing activities have become primordial in the life cycle of software development (SDLC) to ensure the software's quality and operation. Indeed, software bugs can potentially bring dramatic consequences if the product is released without testing it to the end-user. The task of software testing is to check that the real result and the anticipated outcome are compatible and that the program is implemented without bugs. Many methods, strategies, and tools have been suggested to help ensure the device is free of defects

The field research goal was to get an understanding of professional debugging in modern tech firms. This perception was essential to Create a general questionnaire regarding debugging. In Germany, we have visited four software companies varying in size from five to several hundred employees. All four companies produce web applications, some are self-hosted, and some are licensed. Throughout their day, we could follow eight developers, and observe their methods. To order to get an understanding of their processes, we asked each developer to think aloud. We asked each developer at the end of each visit to explain his overall method themselves. We also questioned if they understood existing devices, such as back-in-time debuggers, and if they found them useful.

An overview of each company's relevant features is given in Table I. For every company. It displays the number of workers, the number of app engineers, the number of engineers we encountered and the normal scale of the company's staff, the method of growth they used, the technologies they used and the resources they used. These lists are not exhaustive, but show the tools and technologies that we were able to see during our visits. Apart from that info, it's worth noting that the third company is part of a larger web-oriented sector.

5. Conclusion

We presented our results on the study of professional software developer debugging behaviour. We checked the literature available, and noted a difference of 17 years after the last comparable research. We played an Explorative field study, visiting four companies in Germany and observing 8 developers in their usual working environment. They were all skilled in the use of a symbolic debugger. Although all followed a standard approach that can be seen as a simplified scientific method, none of them was aware of this or could explain his approach without resorting to proof. None of them had any formal debugging education, nor had any background knowledge in time debuggers or automatic techniques of fault localization. We created an on-line survey based on these results to consolidate and expand our results.

REFERENCES

[1] In B. W. Kernighan & P. J. Plauger: "Programming Type Elements," McGraw-Hill, vol. 1, In 1978.

[2] R. Zeller, Why programs fail: A systematic debugging guide. Morgan Kaufmann, September 2009.

[3] D. D. Gould et P. Drongowski, "An exploratory computer program debugging study," Journal of the Society of Human Factors and Ergonomics, vol. 16, No. 3, pp. 258–277.

[4] In T. Gr¨otker, H. Keding, U. Holtmann and M. Wloka, Developer's debugging guide, 2nd ed. Self-editing agency, 2012.

[5] K. C. Metzger, By Thought Debugging: A multidisciplinary approach. Digital Press, Elsevier 2004.

[6] D. J. Agans, Debugging: The 9 indispensable rules for finding even the most elusive software and hardware