Communicating and Managing Bugs Efficiently

By Jitu Borah
(jborah@adobe.com)
3rd September 2003

Everyone knows the fact that it is the responsibility of the tester not only to find and report bugs, but also to manage and handle bugs efficiently until those are closed. Do you think conveying a defect requires just a little effort or endeavor? How do you form an opinion about the nature of a software error and estimate the merit of that? Though all looks easy, in reality, it is not. In this article, I am going to describe few better ways to report a bug and some careful thoughts associated with it. Tester's credential is always measured in terms of quality bug reporting and makes it a valuable input for career development. Lots of effort needs to be put if you are into testing complex projects managing hundreds of bugs during the journey of software development process until the product finds a place in the market.

Efficient Bug Reporting

Many testers at the beginning of their career usually think that bug writing is an effortless and elementary job and one just needs to write whatever is being observed and convey it in some form. This casual approach produce adventitious and erratic bug reports which often lead to confrontations. It's a well-known fact that every individual has his own style of writing, but when it is of bug writing, one should be careful and conscientious, as it needs formal, accurate and organized writing skills. The statements to express a bug may vary when two persons report the same bug. But, effort should be made to make your bug report as methodical as possible, because a bug report is a written manifestation of a defect and expected not only to be viewed by the developer, but also, in many cases may be looked at by someone at the higher level or may be a review committee. Erroneous way of writing a bug report creates misinterpretation and result in unnecessary delay in the fixing process. Think of situations, where multiple teams are working at different geographical locations. In such a case, efficient reporting is necessary as the developer and the tester may not get a chance to interact frequently.

Title

Each bug needs to have a good one-line title. This is a kind of skill similar to writing a good attention-grabbing media add campaign. The nature or the symptom of the problem can be found with the one line title of the bug. Consider a realistic scenario. Look at the headlines of the newspaper and try to analyze. Can you really make out from the headline about what it is going to talk about? Ideally, a bug title should not carry more than 50-60 characters. One important aspect of a bug title is that it should not carry any subjective representations of a defect.

Summary

A brief summary along with a good title is required to describe the nature or the circumstances under which the defect actually occurred. How would you describe the bug in

approximately 2/3 sentences? A good summary quickly and uniquely identifies a bug report. Otherwise, developers cannot meaningfully query by bug summary, and will often fail to pay attention to your bug report when reviewing a long bug list. Worse, they may send the bug report back to you with a request for clarification.

Steps to Reproduce

Are you really providing the exact information how to reproduce a bug? Now, what is meant be 'exact information'? Most of the time a tester thinks that it will be easy for the developer to understand whatever he assumes and with this assumption, many important steps are sometimes skipped. Correct terminologies and appropriate naming conventions should be used while writing the steps. One should always keep it in mind about the clarity in providing the information and systematically note down the steps accordingly. Ask yourself; can this bug be reproduced by anyone who reads your bug report?

Observed Result

You need to state clearly about what is happening while performing the steps mentioned in the bug report. This has to be extremely specific and all the results how the software behaves should be mentioned without missing anything. Assume a situation when you encounter an error dialogue while doing some specific tasks. Now your job is not done just to mention about the error but you need to provide what the dialogue content is if there is any.

Expected Result

Sometimes, due to limitations of incorporating everything in the specifications, writing the expected result in a bug report becomes a difficult task. In addition, most interestingly, whatever is available in the specifications, that too changes frequently. Therefore, writing expected result of an obvious error like a crash or functionality break may not be of much difficult job, but in some complex situations, where specification is silent, a well guided decision as to 'what is expected' needs to be taken. A tester may take guidance from anyone who has knowledge or experience in the area you are testing. You can also get useful information if a previous version of the application you are testing exists or if there are any competitive industry standard applications available.

Additional Info

This is important to mention in a bug report if you have any other relevant observation associated with the problem observed, which may be of some help to the developer to establish some correlation with the actual issue. A reference to specification or a benchmark of a related scenario can be considered as an example.

Comments/Discussion History

A tester needs to be careful while making his comments as all these comments or the discussion history is important. Imagine a situation where you are asked to do testing for someone who has left the organization. You have to rely totally on the discussion history to really understand how to proceed from that stage. Also, when a fix is ready from development and now it's your job to verify and close, don't forget to write comment even when closing the issue.

Attachments

Sometimes to have a better understanding of any situations, apart from simply writing down the error, a test file or a snapshot should be provided along with the bug report. This proves to be very effective as it can be treated as evidence of existence of a problem. It also saves lot of time as the developer don't have to waste his time in recreating the error, which on the other way

can be detected and observed in the test case provided by the tester. Care should be taken to provide the most basic test case that reproduces the error or where the bug can be viewed.

Focus on Milestone

Logging bugs at any stage of the testing cycle is the primary job but more than what is needed is the nature of the bugs with respect to the specific stage of the product. Usually the testing milestones are divided in three basic phases, which are Alpha, Beta and Release. Finding a bug should be much more focused during each phase, for e.g., say during alpha stage, it is always a good practice that findings are more towards feature implementation with respect to specifications etc. Since every milestone is based on certain specific criteria or conditions, defect findings are also expected to follow the same.

Analyzing Severity and Priority

There are many debates on whether bug priorities and severities are subjective or objective standards or a set of rules on which a judgment or decision can be based. Objective severity and priority assignment is theoretically possible but in practice, most of the time, it is seen that these are considered based on one's subjective perception. Criteria can be made to define severity comparatively, increasing proportionately the accuracy of severity assignment. However, hundreds of potential circumstances, actions and omissions may not match mechanically the criteria, then one must make a well guided, but subjective, decision. Although there is no universal definition that can be provided for severity levels but mature companies define the severity levels for their organization relevant to the nature of specific activity. These companies have guidelines for assigning the severity to the defect, these guidelines are provided to the tester, and these set of rules or guidelines are tailored as the project requirement. At the end of the day, what is being aimed is that the defects are classified by severity in an agreeable and constant manner by all the testers in the project.

Some broad categories can be defined for the severity levels. It is not imperative to strictly follow the terminologies involved, which may vary depending on the nature of job and different culture followed by different organizations.

- Crash/Hang
- Major Functionality Break
- Data Loss
- Minor Functionality Break
- Cosmetic

Priorities can also be set accordingly. All bug-tracking tools define and provide different priority levels, which can be treated as a standard procedure. Following are some of the examples

- Fix ASAP
- Higher
- Normal
- Lower

Severities and priorities fields are generally customized depending on the nature of testing activities done in an Organization.

Analyzing Quality Reporting

Concise Bug Report

Have you ever come across people who always exaggerate any small problem they come across? A more elaborative and story-like explanation should never be written while reporting a bug. There should not be unnecessary or irrelevant information that may confuse and lose focus on the actual problem. A precise and balanced explanation about an issue helps the developer to uniquely concentrate on the specific problem reported. On the other hand, a report should not also be too short which may miss certain observation or sequence of steps, which may be important from the development perspective.

Bug Isolation

Tester's credibility is measured in terms of bug isolation skills, which shows his abilities to do correct analysis and investigation. Testers should be able to 'zoom in' on the underlying fact or cause that provides logical sense for occurrence of a defect. Lets take an example, think of a situation where you encounter a crash while opening a 10-page document containing text with images in your application. Now you wonder why this has happened and many questions will come in your mind. Consider the following reasons

- Is it because the document contains 10 pages?
- Is it because the document contains text with images?
- Is it because of a specific page creating the problem?
- Is it because of a specific line of text or a specific image?
- Is it because of some combination of factors?

Sometimes, bug isolation is extremely hard to accomplish or comprehend, as there may be innumerable number of factors involved, which may cause the program to behave in an unexpected manner. Bug isolation needs a deep thought and understanding about the application that you are testing. But, while isolating one should not start falling into any loop or extreme level of depth, which may finally leads to debugging and consuming a significant amount of time.

Single Bug vs. Blanket bug

A crucial and considerable aspect of software bug reporting is to judge and identify how many bugs reports should be made with different issues of similar nature and on the contrary, if many issues seem to be originated from a single source. Confusion arises quite often when someone comes across this kind of a situation that presents perplexity or difficulty to figure out what exactly may be the root cause. One answer is that one should avoid raising two or more issues in a single bug report if issues are not related. There is a possibility that the principal issue raised in the bug report gets resolved and the rest associated with it is overlooked or ignored. A bug report should focus on one specific issue at a time. No different issues should be clubbed together that may create confusion and which may also lead to waste of time raising the left out issue later once again. Similarly, if there are multiple problems in an application but the solution to the whole is just changing one particular structure of code, there can be one blanket bug filed instead of logging multiple bugs. For, e.g. if you don't find the shortcut keyboard command for five different items in a menu of your application, you don't need to report five different bugs, one bug is sufficient to address these.

Comparative Analysis

A good comparison always makes your case persuasive, effective and cogent. At times, it becomes essential for a tester to give an analogy about certain behaviors to advocate or support his claim. It becomes easier when you have any previous version of a product, which you are

testing currently, or a similar kind of product offered by competitors, which has a reasonable market share or which can be considered some sort of a standard. This is very effective and easy to convince the developer if you can provide such an estimation of similarities and differences you observe between your test applications with any other.

Finding Important Bugs

Finding as many bugs as possible is always needed and appreciated but the most critical task is to find important bugs. However, is there a way to classify important bugs with non-important bugs? Important bugs in this context are those that can be found while verifying basic test cases and if missed, there is a high probability that the customer or the user would be disappointed and may report it or call up customer support. Along with finding general and remote bugs, tester should also concentrate more on finding those bugs, which may never occur with basic workflows offered in an application.

A Careful Thought

Duplicate Bugs

Creating many duplicate bugs leads to inefficiency and incompetence although in any practical businesslike scenario, this cannot be avoided completely. Attention should be paid in identifying duplicate bugs, as it is a repetition of time and effort made. It is not always easy for a tester to know the exact nature of fix needed while distinguishing different bugs looking at the behavior shown at the front hand by the application under test. Because the same underlying fix may resolve many problems in a single go. One way to minimize duplicate bugs is to simply search in the database with required keywords and to find out if there is already a bug logged on that issue. Sometimes, due to dauntingly complex and vast bug database structure, it may not be easy to search for similar bugs quickly, then it is advisable to log a bug although it may afterwards be found as identical to some other bug. One should not waste enough time looking at too long or too diverse bug list while searching, rather in such situations, it’s better to have a duplicate bug than no bug at all. An essential point to be noted here is that if you find a bug, which has been reported earlier, but if you have some valuable or additional information to contribute, write a comment to the existing bug report.

Bugs Not Going to Get Fixed

Quite often it seems that testers get de-motivated or discouraged when they come across situations where bugs filed by them are being deferred by management either justifying the reason as not worth effort or don’t have enough time and may be fixed in some future version. It is a typical tester’s psychology that they want to take decisions about what the quality of the product should be. Testers mindset may not always tally with what the business needs and users or customers may actually prefer. So, at times, depending on the customer specific demands or requirements raised by marketing, high severity bugs of some areas of the application may get deferred and small minor annoying bugs get fixed. Remember that fixing a bug is a business decision and it involves the management people. Testers should try to understand the user workflow and give an insight from the customer’s as well as the organizational point of view.

Not a Bug or As Designed

Think of a situation where you have filed a bug but the developer returned it with a comment 'Not A Bug' or sent it back with a comment that it is behaving correctly the way it should. If the specification is silent about this or if there is no other document to support this, you

now wonder what should be the expected behavior. Should it really behave in this way or it's a bug? Care should be taken whether logical reasoning backs the findings. There may be hundreds of situations where it will never be possible to get all information in a document about how the software will behave with innumerable combinations of features of a product. In such a situation, if there is any confusion, all these issues should be raised to concerned authorities or the management to take a decision on it.

Bugs Get Fixed Automatically

Sometimes, we also encounter problems where we find that bugs that used to exist, without any change-list get fixed automatically by surprise. You must be thinking that without a straightway fix, how can these bugs go away? You know that the developer has not marked such bugs to verify but you have found that these get fixed. Sometimes, it becomes inevitable for a developer to rewrite his codes. Similarly, situations also bound him to change code routes, change dependencies, etc. which, as a result, creates such magical situations. As a result, existing bugs go away and new bugs started appearing. A tester needs to perceive and comprehend the reasons for such changes and be prepared for these kinds of bug disappearance.

Bug Report "To and Fro"

Due to lack of coordination or may be not having sufficient information, sometimes a bug transfer process is repeated many times between the tester and the developer. Tester files a report, developer returns it seeking some more info which is again sent back by tester with some new info and so on. It may often create chaos in the bug reporting process. Lot of time and resources are being wasted if a bug is moved many times between the tester and the developer. To avoid such situations, a tester should always be careful while reporting a bug to mention all relevant information, which is useful to reproduce the bug. One needs to have a wider vision about the problem and provide complete information accordingly whenever it is asked for.

Personal and Emotional Statements

Frequently changing specification or unclear expected behavior may often create differences of opinions. It is a well-known fact that software testers carry bad news about the software and therefore they sometimes face unpleasant reactions. In any situations, one should avoid arousing passion or strong emotion, especially anger or belligerence etc. while making comment in the bug report. In addition, a tester should never have a feeling of underestimating a code and criticize accordingly. Bug report, being a technical document, should not carry any humor or sarcastic statement and tact and diplomacy should prevail while advocating bug findings. Sometimes, due to difference of implementation of code vs. expectations, there may be some kind of misunderstanding of concepts between the developer and the tester. It's always a good practice to sort out differences verbally rather than making it reflected in the report.

Proactive Approach

Raising Immediate Alarms

Generally, in complex applications, due to large number of bugs in the database, it is not always easy for the management to identify all those bugs, which needs immediate attention, although the priority and severity of bugs are mentioned in the bug reports. A tester is aware of the testability of his areas and if something goes terribly wrong, apart from simply writing a bug, it is also equally important for the tester to raise his concern. It is done in a situation where a tester feels that his testing is blocked or hampered and therefore he needs an early fix to proceed

with testing. Without any delay, the bug should be communicated to the respective developer with a note why it needs a quick fix. A prompt action is needed from the tester in such situations.

Clear up all testable bugs

You must have faced situations when your manager comes to your desk asking you about the reasons for not regularly clearing your testable bugs. Once a developer returns your bug with his fix, tester's most primary job is to verify and update the status at the earliest. Your responsibility is not just over once a defect is reported. You need to keep a track of every bug fixed and prioritize the act to verify it along with continuation of regular testing activities. It gives a clear picture about the quality of fix and accordingly the shape of the application you are testing. Management also closely monitors at the find and close ratio of bugs that you report.

Bug Hunts

Organizing and participating in some sort of bug hunt across the team also helps finding out hidden bugs. A tester should act very positively towards this approach and welcome people to spend some time on the area that he is testing. If you are working in a team, you can invite other members of your team to look at your areas. Similarly, you should also act in the similar manner to participate in other projects if required. Different teams can spend few hours to make an effort to find bugs if you have multiple teams in your organization. It always gives result, as this is generally not the kind of formal testing which a tester normally does. Lot of good companies have this culture because they believe it is better to catch bugs with every possible manner before it reaches the hands of the customers. A formal recognition or a prize can also be awarded to the people who find the best bugs in the application.

Study Previous Bug Reports

It is considered a very good habit to look at and study bugs filed by experienced tester or old bugs if these are available in your database. Exposure gives you confidence and enhances your testing skills. If you are new in testing, treat this practice strictly as a learning process and do it in your spare time. It gives you new dimensions in your thoughts, which you might not have taken care of while doing your regular testing activities. You can learn new ideas, techniques, systematic procedures by which a complex or scientific task is accomplished in these defect reports.

Unwanted Workflows

Late Findings

What do you do if you find a major bug just before the product is going to ship? Everyone knows the fact that 'to err is human' and therefore a tester may miss a good bug at the beginning and may catch it at later stage of the project. There is always a high risk associated with late finding of major bugs as fixing it may need another round of testing which also leads to delay the process. A tester's responsibility also increases, as he needs to take care of the possible effects of the late fix. Although late findings are unwanted, tester should treat this as a learning experience. However, late findings should never be ignored and it's always advisable to report a bug and raise the matter to the management immediately.

Low Bug Counts

Do you think that your efficiency is measured by your bug count? Its Yes and No. How reliable is the code you are testing or how difficult is your area of testing? Many people advocate that bug counts are considered to be poor measurements of tester's performance as quality of bug

is much more important than the quantity. A tester needs to pay more attention in detecting quality errors rather than simply finding out remote or corner cases. You can prioritize your finding patterns with respect to various stages of your application. However, the number of bugs at the same time cannot be ignored totally also. There should always be some sort of balance between the nature of the area you are testing and defect findings

Not Finding Already Fixed Breaks

There is every possibility that in complex projects, fixing of one bug may break another already closed bugs. Effort, which is put to find a bug and the same to get fixed, will go waste if we miss to find if there is any break in already fixed bugs. In most of the software projects, a complete regression is being done for closed bugs just to make sure that already closed bugs don't reappear again. Although this practice cannot be done all the time due to time constraint, generally, before releasing a major milestone of a project, it is always taken care of. Monitor closely all your bugs and if you have any doubt on your earlier bugs, just check it once more.

Customers Catch, You Missed

What happens when a tester's work is done and finally the product is in the market? It is the most thrilling moment for the testing guy once customers started using the product. No tester expects that customer finds some serious flaws in the application, which is missed by him although it is a well-known fact that no software can be absolutely bug free. Also, it is an embarrassing situation, when your manager comes to you and asking why this bug is missed? One cannot deny the fact that there is every possibility that user may find unexpected bugs. But, that's the way of learning and understanding the job. Once you develop experience, different approach towards testing develops and helps you to handle issues before it actually reaches the customer.

To conclude, as you are in software testing, attention should be paid to manage your bugs efficiently in every possible manner. Difficulties may lead to evolving new thoughts and challenges associated with different testing environments, but at the end, the goal is to handle these correctly.

Reference:

The Bug Reporting Process - by Rex Black

About the Author:

Jitu Borah is a Software Quality Engineer working with Adobe Systems India Pvt. Ltd. He has over 9 years of experience in IT of which past 4 years are in software testing. He has worked on multiple products in print and publishing domain, which are widely used in the international market. Prior to Adobe, he was into software education and publishing. He has a deep understanding of entire Product Life Cycle and various processes involved in delivering a quality product of international repute. Exposed to various types of testing such as functional, installation, acceptance, performance etc. and aware of different test methodologies, techniques and tools. Extremely well versed with all the documentation processes, those are involved in the entire product life cycle.
