Don’t Let the Bugs Out

A guide to Issue Tracking, and it’s role in Software Development

Beth Schechner
Elementool

Don’t Let the Bugs OUT

A Guide to Issue Tracking, and it’s Role in Software Development

What is an Issue and Why Must You Track it?

In software development, an *Issue* or *Bug* is a glitch in your software system that causes the program to behave in a way other than designed. Issues can range from the show stopper, a fatal error that causes irreparable damage to your software system, to the very minor glitch. Obviously, each issue in the system is not going to have the same importance, or urgency, attached to it. Critical issues, the show stoppers mentioned above, need to be solved first before you can do anything else. Lower urgency issues are the minor ones, which you or your team can put off until time permits. There are tons of other details, that are relevant to the fate of your software, to keep track of like who experienced the issue (was it a customer or someone within the company), when was the bug found, what exactly was it that the user experienced (in order to be solved, this needs to be extremely detailed!), what solutions were attempted, and much more.

An issue tracking system or tool will manage and maintain your lists of issues. Organizations commonly use Issue tracking systems in their customer support call centers to create, assign, update, track and resolve reported issues. These issues could come from customers directly, or even from the organization's other (non IT) employees. An issue tracking system will often also contain a knowledge base, which stores information on each customer, fixes for common problems, and other important and useful data.

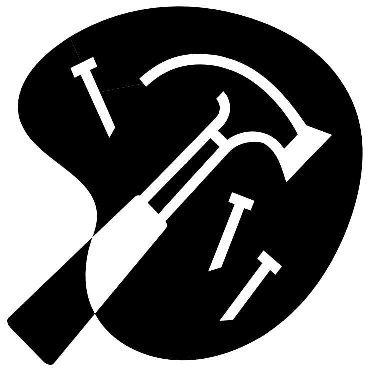
Why Issue Tracking is Important

Think of the software system or application that you or your company is creating and maintaining. If you're in the business of developing multi-player games, for example, it may not seem immediately obvious why you need to track and catch every possible issue or bug before release. However, imagine that your system is the software that will drive a jet plane, or hold a hospital's administrative records. Suddenly, an unintended action or issue could cause a major accident, or prevent a doctor from knowing about a patient's allergies. Of course, in most cases a glitch in your software won't have such a dramatic outcome, but it does drive home the importance of releasing software that behaves as designed and marketed. In any situation, releasing a product that doesn't work as advertised will cause harm to a company's reputation, and ultimately, its bottom line.

Using Tools to Improve Your Process

Ok, so you're convinced that you need to implement some sort of system to track issues. What now? Well, first you're going to need to take a close look at your development team and how they work. With so many different tools on the market (and there are tons!), it is best have an idea what you're going to need before you start shopping around. Is your team spread out in different offices or locations, or are they all in one room? Do you have a small team working on many projects, a large development team that works on only a few big projects at a time, or something in between? Do you want a hosted solution, or does your company prefer that you download and maintain software internally? While it may seem that there are many factors at work, asking (and answering) these questions will help you get a feel for what type of system will work best for your needs. Implementing an issue or defect tracking system is a great way to find and track bugs when developing software systems. However, it's also useful for tracking updates in website development and managing projects with many moving parts. With so many options available, it's easy to get overwhelmed. Once you've taken a look at your company's needs and asked some key questions about your operations, you're ready to start shopping around.

While there are many options for issue tracking software, Software-as-a-Service (SaaS) tools are becoming increasingly popular. One of the many arguments I've heard for using a SaaS or hosted issue tracking tool, compared to an in-house developed tool or a purchased tool installed in-house, is that your developers are freed up to develop the company projects and further business goals, rather than developing and maintaining an in-house solution. Another, albeit a bit more obvious, advantage of SaaS is that users can access reports and records from any location. Users need not have access to your network where the data is saved; they only need an internet connection and a password. A dedicated issue tracking tool will also let several users access and alter accounts at the same time; obviously a clear advantage over using email or an excel spreadsheet to track issues!



An ideal issue tracking system will allow you to maintain control of the system, determining which users are able to, at any time, enter, change or delete information. You should also take a close look at the record database, where issues are recorded and all important details regarding the issue are stored. A good record database will let your users alter records individually or multiple records at once. They should also be able to track changes, include attachments and search through reports.

Reporting issues should be easy. If they aren't transparently reported then finding the issue and fixing it is going to be difficult, if not impossible. Your system should have room to add details and should let the person reporting the issues include all the facts and information that you decided above was important to your process.

The tool should also be fully customizable to allow the company to customize it based on their specific needs. It should allow them to edit the fields, add or remove fields, etc.

A good issue tracking system is almost required in today's business world. Bugs can be damaging to a company and can cost a company in both money and time (again, the hospital scenario should spring to mind). By understanding the features of a good issue tracking system you can make sure that you choose the features that will work best for your company and way tracking issues.

Using an Issue Tracking Tool

Once you’ve picked your tool, based on the criteria you defined while reading the last section, there will be new processes to learn and rules to figure out. Implementing an Issue Tracking system for the first time can mean reforming an entire company’s processes, and essentially starting over from scratch.

Create a system for compiling and writing reports, and pay attention to format as well as content. Make a list of what you think should be present in any report and take a look at previous issue reports to see what details have been included; if you have any questions why certain items are in reports, then ask! Ask your team members, ask management, raise your hand, or send an email. Just don’t stay quiet! Communication with the development team, QA team, management and clients is of utmost importance. Issue reports are a major component in successful issue tracking (and application lifecycle management in general) and it’s important that they be both accurate and uniform.

Issue tracking software isn't too difficult to use, but as a beginner, it can be awfully easy to make things harder than they really are. Yes, you will probably make mistakes as you learn your new software and get used to new processes, but knowing what some of the common mistakes are can help you to avoid them.

An up-to-date and user friendly issue tracking system won't take the place of well trained developers, nor is it intended to, but it will help your team to do their jobs better. However, a tool that you don’t know how to use isn’t very effective at all. To make the most out of both your tools and your team, take advantage of any training or support offered by your software vendor. Having a tool that no one on your team knows how to use can be a bigger waste of time than not having a tool at all!

Managing and Issue Tracking

Ok, so now you have an established process for issue tracking, and a tool to help you manage it. You’re done, right? Wrong! Having an effective IT and QA team is incredibly important to Application Lifecycle Management as a whole and to Issue Tracking and Testing in particular. Having strong IT and QA teams involves both team work and a strong manager. It takes a lot of different people working together to identify, track and address all of the possible bugs in a software system.

Since just one uncaught bug can crash an entire software system (think of the plane, or hospital scenario), effective management in Issue Tracking and QA can make the difference between a successful and an unsuccessful software product.

First off, having a dedicated team for testing will help to ensure that all bugs are properly dealt with. Communication within teams, among members and with clients once the application has been released can help to prevent problems before they arise. No one person is going to find every bug, but two people may very well find the same bug separately – and sometimes it may take two people working together to uncover and resolve an issue. Providing your team with an outlet for questions, comments and other chatter can help to prevent duplicate issue reports. Encouraging ongoing communication among team members can result in faster development lifecycles; developers can easily ask questions of each other about similar bugs and fixes, so no one needs to reinvent the wheel. Sharing information on past and present bugs, as well as current tasks and projects, will make it easier for the team to find and take care of every possible issue.

Having a testing team that communications and shares tips is great, but you're not done yet! If your team is unorganized, then they might as well not be tracking issues at all. No one wants to micromanage (or be micromanaged, for that matter), but some oversight is absolutely necessary. Having a system in place for periodic checks and for generating reports makes it easy for both you and your team to know for certain which bugs have been dealt with and which bugs still need to be addressed. Also, to tie back into greater

Application Lifecycle for a second here, regular reports will show whether or not a project is on time, and whether your application is meeting the original business requirements.

If your issue tracking tool is organized, then chances are that you'll have a properly working application ready for your target release date. Still, it can't hurt to throw out a reminder every so often. Since most software projects are on fairly strict timelines, establishing milestones and setting reminders for certain tasks will reduce unforeseen delays. If a major component isn't going to be ready and working on time, it's best to know as soon as possible, so that the necessary resources can be redirected.

Issue tracking is a major part of any software development project, combined with a well organized, informed, and trained team is one of the quickest ways to success.

The Lifecycle of an Issue

Issues are part of every product development process, like it or not. How your team tracks the bugs that you find during development, and after release, ultimately determines the success of your product. Bugs that are found but not properly logged and tracked, as mentioned above (repeatedly!), might slip through the cracks and be discovered after release. This, obviously, is no one's ideal scenario but it DOES happen. Every issue logged is in a constant state of motion. If you're using a static spreadsheet to track your issues, there's no way to accurately convey where your issue is, how it's been dealt with and what's going on with it RIGHT NOW. For client facing projects, spreadsheets also don't provide any efficient way to keep them updated in real-time – an update based on data culled from a spreadsheet only presents a snapshot of the project. In fact, by the time your client views your report, the status may have already changed! Using an issue tracking tool helps you keep a handle on your evolving roster of issues and bugs during the Application development lifecycle. However, to effectively use your issue tracking tool and manage your issue tracking team, it helps to have an idea of what happens during the lifecycle of an issue.

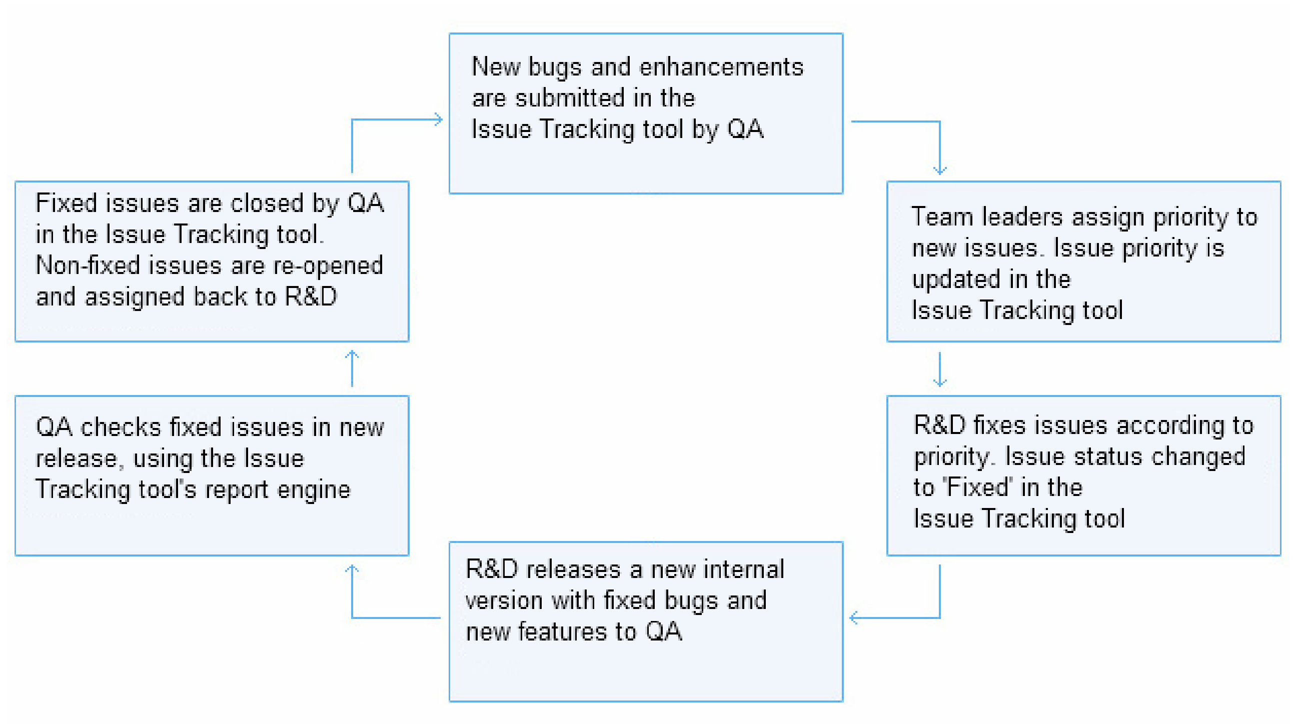
An issue lifecycle, in most cases, is pretty straightforward. Your QA team, in testing, submits any issues or bugs that they find into your issue tracking system. If you're developing for a client, you can set up your issue tracking system so that clients can log right in and submit bugs as well, and in some cases, issues will be submitted by users via a web form or helpdesk once your product has been released.

Once an issue is in the system, you'll need to determine its importance and assign it to a team member to resolve. Is this issue a "show stopper," or is it only a "minor glitch"? Setting priority will let your team handle the biggest issues first, before tackling the less important ones. Every user who signs into your issue tracking tool from this point on will see any issues that have been assigned to them, as well as their relative importance.

So you're a developer, you logged into your account this morning and saw that 3 new issues had been assigned to you, and that two of them had high priority. Once you get started working on them though, you and your project manager need to make a decision. Not every bug is worthy of being fixed and not all bugs can be fixed right away. You may just decide to pass on some of the lower priority issues until you have more time, or you may decide to scrap a feature, and not address the bug at all. When, finally, you've addressed all the logged bugs you're ready to release your tool. Congratulations!

But wait, you're not done yet! Management has decided to release another version of your application and has added new features. So you start back over with QA, but this time, they only have to check the new features because you've found all the previous issues which are logged and their history is easily accessible in your convenient issue tracking tool!

To recap: 1. Issues are submitted by QA, Clients and End Users. 2. Management assigns priority to each new issue, and delegates it to a team member. 3. Developers log into their accounts and can view all issues they need to address, along with their relative priority. 4. Logged issues are handled according to priority and either fixed or thrown out. 5. Your software is released! 6. A new application is developed which solves the previous issues and incorporates new features. 7. The new features are checked by QA and any new issues are submitted. Repeat.


New bugs and enhancements are submitted in the Issue Tracking tool by QA
Team leaders assign priority to new issues. Issue priority is updated in the Issue Tracking tool
R&D fixes issues according to priority. Issue status changed to 'Fixed' in the Issue Tracking tool
R&D releases a new internal version with fixed bugs and new features to QA
QA checks fixed issues in new release, using the Issue Tracking tool's report engine
Fixed issues are closed by QA in the Issue Tracking tool. Non-fixed issues are re-opened and assigned back to R&D

Issue Life Cycle

Finding and Submitting Issues

So you've been convinced to implement an Issue Tracking system, you've picked out a tool and you have your team using it. You also took a quick look at how the lifecycle of an issue, to get a better idea of how your software will fit into the whole process. Now you're ready to find and submit issues to your system.

This part is pretty straight forward, but it can be easy to get hung up on details. One of the easiest mistakes to make is to spend too much time focusing on one single issue. Sure, you want to identify and track as many bugs as possible, but be careful not to spend all your time tracking down that one elusive bug. Sometimes you're better off moving on to the next item on your list, and maybe letting someone else on the team take a crack at it.

Another common misstep is overanalyzing an issue - making a simple problem into a more difficult and complex problem. This actually applies to many things in life, though in issue tracking it’s undeniably easy to do. Making too much of a simple issue will only create more work for you and your team, and likely end up being reported as a duplicate in the records of your new issue tracking software.

Since generating an issue (bug) is a major component of issue tracking software, and one of the bigger reasons that you probably chose to implement a tool in the first place, having the correct information in your submission is essential. The first entry in an issue lifecycle is the most critical one. It is this first entry that everything else in the issue's history is built upon. For this reason, everyone involved your development and QA teams should understand the three main components of submitting an issue.

First: how to get to, or where to find, the issue. This piece should explain to anyone reading the report how to see the bug, and it should be detailed. Anyone who might be unfamiliar with the issue should be able to experience the bug by reading and following these directions. How are you going to fix an issue if you or your team members can't find it every time?

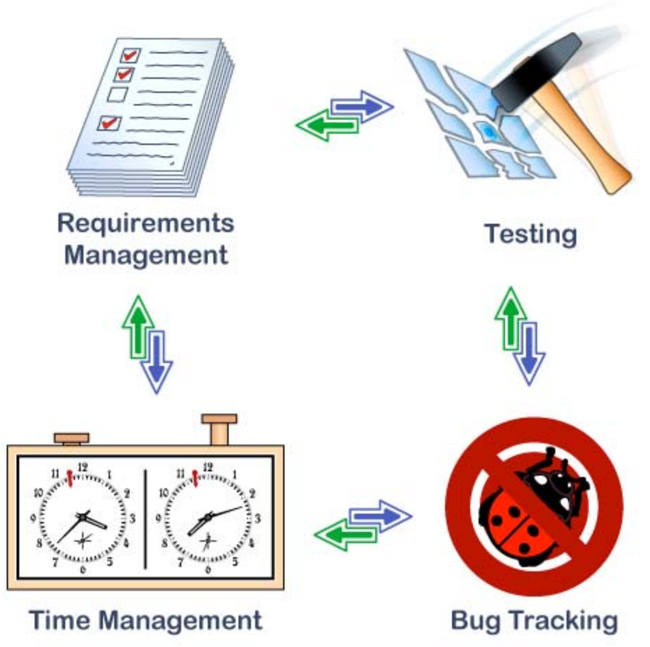
Second: what should have happened if the bug wasn't present? This should also be really detailed; it's obviously important to explain the desired action, so you and your team know when the issue has been successfully resolved.

Third: what occurred because this issue was present? Or what did the application do, that it wasn't supposed to? Here is where you should explain what effect the bug had and how it changed your program. This, again, should be detailed. Are you picking up that detail is important in issue tracking? If the issue tracking team knows exactly what the issue is doing, or what chain reactions it's causing, they can figure out the best way to fix it.

Writing a detailed and informative issue is essential to optimizing your issue tracking software. Make sure that all three essential elements are included in each issue submission, and you can ensure that you're playing an effective role in handling bugs in your company's software system.

How Issue Tracking Fits into ALM

Application lifecycle Management is the processes of conceiving, developing and releasing a new software application or ALM. ALM links business management (looking out for time, money, function, market and practicality) with software development (developing, testing, tracking and fixing). This is all made possible by a suite of tools and applications that both facilitate and integrate the two groups. Creating a process for issue tracking, and implementing a tool to help you do this, is going to help facilitate both testing and future development. Using an issue tracking system give you a clear overview of development needs including bugs, improvements and fixes, and the current status of each – and where you currently fall in your application lifecycle.



Application Lifecycle Management: Common Steps

An issue tracking tool is the chassis of the ALM, and you can’t run a truly efficient process without it. An issue tracking tool, when well implemented, gives managers a way to oversee the different tasks, stages and people involved in developing your application, and track progress of your entire project.