2011

# What if

## A question every tester must ask…

A collection of tips from a software tester.
Dive in, enjoy and lets discuss over Skype/Email.

Ajay Balamurugadas
http://EnjoyTesting.blogspot.com
10/16/2011

# What if...

Hello Reader,

Special thanks to you for downloading this book. This book is a small collection of my experiences in software testing. In 2006, I started my career as an Associate QA Engineer. Straight after my college, I dived into this job with a lot of energy. The entire corporate world was new to me. I was not aware of the term 'Software Testing'. After three weeks of training sessions, my first task was to execute the test cases. As you read these words, I have completed five official years of testing software. When I logged my first bug, I thought – 'What if' this bug was found after release? Years passed, many products were released, and I gained a lot of varied experiences.

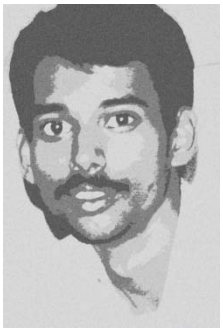
I made a few embarrassing mistakes too. There were few instances where *I wished that someone had warned me beforehand.* So, I started preparing a book of tips targeted at software testers. Special care has been taken to keep each of the 22 chapters short and to the point. Emphasis is on ready-to-use tips which would give you instant results. I do believe that there is no single best practice which would suit every context. Being a student of context driven testing community, I agree that there are good practices in context, but there are no best practices.

This book is heavily influenced by my experiences with industry experts, reading books, testing software, talking to customers, end-users, support team, testers, programmers and their managers. Do let me know if you have a topic in mind which I have not covered. Hope you enjoy the snippets while I wait for your comments...

Please feel free to contact me via email, Skype, Twitter or GTalk.

Email: ajay184f@gmail.com Blog: www.EnjoyTesting.blogspot.com

Skype/Twitter/GTalk: **ajay184f**



# Contents

## I am a fresher and I want to learn software testing

Hello Friend, welcome to the world of software testing. It is really nice to know that as a fresher, you are interested to learn software testing. Please note that the journey to learn about software testing will be hard and long although it will be very interesting too. There are very good resources shared by experts which will help you learn faster. At the same time, there are a lot of junk articles, posts which would constantly pull you back. At one point of this journey, you will face what is known as information overload. There is so much to learn and so little time. Let me highlight a few tips to help you stay on the right track to be a good software tester:

- Read this wonderful post by Shrini Kulkarni: http://shrinik.blogspot.com/2005/10/advice-for-budding-software-test.html This single post has a lot of powerful advice to everyone in the software testing industry. Remember that it is the skills that matter. Answers might get you a job but it's the skill set that defines you.
- Read a lot of books. Start with the two books – 'Lessons Learned in Software Testing' [Bret Pettichord, Cem Kaner and James Bach] and 'Testing Computer Software' [Cem Kaner, Hung Quoc Nguyen, Jack Falk].
- Avoid the 'Google is god' trap. When you search for a topic, you might get few good search results and many bad results. It is up to you to pick the good ones. Do not believe everything you read. Keep questioning.
- Create your account on Twitter & Skype. They are very good platforms to discuss software testing. You will be surprised by the wealth of information you can learn in one day if you follow the right people. Two sample lists are: http://www.softwarequalityconnection.com/2011/09/29-testers-to-follow-on-twitter/ and http://blog.utest.com/top-20-software-testing-tweeps/2010/05/
- Start a blog. Write about your experiences, what you learned, what you plan to learn. You can learn a lot from reading blogs too: http://www.testingminded.com/2010/04/top-100-software-testing-blogs.html
- Join the two forums: www.softwaretestingclub.com and www.testrepublic.com
- Participate in online competitions conducted by www.99tests.com and www.utest.com
- Sharpen your skills by practicing at www.weekendtesting.com. Whenever you get time, go through the old archives and practice them.
- Watch videos by Dr. Cem Kaner on Black Box Software Testing [BBST].
- Attend tester meets, courses and conferences. Meet testers to know about different contexts other than your context. Build your network of testers.
- Talk to different stakeholders - programmers, technical support, sales, product management and other team members. Learn from each one of them. You might get interesting test ideas.

## Practice testing

There are very few testers who consciously practice testing. As a tester, it is very important to practice different skills of testing - bug hunting, test reporting, generating test ideas, modeling, questioning and so on. When a tester has not practiced a skill enough, it takes a lot of time to perform in real time under pressure. As a tester, it is preferable to practice a skill in a fail-safe environment than to risk failure in front of stakeholders.

When do you practice? How do you practice? Who needs to help you practice testing?

Those who wait for others to help them practice testing hardly progress in their career.

### SourceForge (www.sf.net)

There are multiple projects on SourceForge which need testers. Even if the projects do not need any testers, what prevents you from testing? One can always download any application and test it. Share your test reports with the programmers or other friends and get feedback.

### Weekend / WeekNight testing

If you are a solo tester or a tester working on a single project for a long time, I suggest you to participate in Weekend / WeekNight Testing sessions. Each session is unique and brings in a diverse set of participants every time. It is a cost-effective way to practice testing in just two hours.

### Practice with friends

You could form a group with like-minded friends and test any software. You would be surprised at the variety of test ideas generated. It is a very safe environment where each one of you is helping each other.

### Beta testing

There are many products on the market for beta testing. Participate, practice and learn from the different feedback provided by the beta testers.

### Competitions

Competitions also offer a safe way to practice testing under time pressure. There is always something to gain by participating in the competitions. If you are good enough, you win the prize and it's an excellent opportunity to learn from other testers.

I recommend you to read this post
**http://www.huibschoots.nl/wordpress/?p=274**

## Testing competitions

As I write this chapter, I have participated in a variety of testing competitions and won prizes at quite a few of them. I would classify a testing competition into one of the following categories with each one requiring a different strategy to win.

- **Online competitions**

These are the ones conducted online and offer good prize money for your effort. It could be a bug hunting competition, blogging competition spread over few weeks. The prize money could be a few hundred dollars or a free conference ticket. Some other competitions last for several days. Examples of sites conducting such contests include www.99tests.com and www.utest.com. In such competitions, the early bird has a good chance of winning.

- Know the rules and deadlines. They play a very important role in your final position. Understand your opponent's strengths, weaknesses & make use of this knowledge.
- Build a model of the product under test, emphasize on the actual deliverables. Define your strategy to effectively deliver what is required. Suppose you spend a lot of time on testing notes, you may not win if it is a bug competition and awards those who log the bugs first. In this situation, it is better to log bugs as soon as possible than spending time on creating testing notes.

- **Testers meet**

These competitions are conducted during testers meet. Testers form teams or participate solo and such contests last for a few hours.

- Quick tests and cheat sheets should be of great help. Do you have them NOW?
- Make use of popular heuristics to find inconsistencies in the product. Ex: HICCUPPS or the FCC CUTS VIDS touring heuristic can help you.
- Make use of the diverse skills among your team members.

- **Official testing competitions & competitions for fun**

These competitions are conducted in your organization and the rules vary from one organization to another. Some of them include the challenges you and your colleagues or friends have over a cup of tea. These help you gain credibility among your colleagues. Work on them along with your daily work. Take help from your colleagues.

And for the competitions for fun, I have only ONE tip – *Have Fun* ☺

Good Luck

## Solo tester

Being the solo tester for any project is an interesting experience. You will be the one who has to analyze all the specifications, attend the meetings, log bugs, talk to programmers and so on. In short, you would be the one responsible for the testing activity of the project. If you look at the positives, you have greater responsibility in addition to the freedom associated with the solo tester. The entire test setup is yours and you usually learn a lot from this experience.

On the other hand, there is only one set of eyes testing the application. The tests you conduct are largely influenced by your thought process alone. So, what if you are the solo tester of a project? In my five years' experience, I have been a solo tester for majority of the projects. Here are few tips that might help you enjoy the role of solo tester:

- Many projects have been successfully completed with just one tester testing it. Quality of tests matter more than the number of testers.
- Talk to your programmers and other stakeholders often to know more about your testing. It is good to know the issues early than to repent later.
- Provide demos to your colleagues and make a note of their questions, test ideas and concerns/issues.
- Be prepared with a cheat sheet, a checklist or a mind map describing the entire project. This activity would save lots of time when additional testers join the project. It also helps when you have to move out of the project and some other tester replaces you in this project.
- When you prepare a consolidated cheat sheet / mind map / checklist, it highlights your understanding of the project and helps fix the loopholes quicker.
- Challenge your colleagues to find bugs in your product. Sometimes, a different set of eyes is what is needed to expose a bug.
- Try different techniques, approaches and heuristics to add more value to the project. Have a look at the list of different mnemonics **http://www.qualityperspectives.ca/resources_mnemonics.html**
- In some projects, a solo tester might not be the best option. If you find it difficult to test the software alone, inform the stakeholders about this issue.
- Use your network. Have all your "products/documents" reviewed by peers or people in your network.

## I just got my test machine

Congratulations. As a tester, if there is one machine you need to take good care of, it's the test machine. As you test on different machines, your test results might vary. How do you setup your test machine? Which software is a must for the test machine?

Few key factors to be considered:

**Anti-virus:**

Install the antivirus software used in your organization. Install the updates to the operating system. Update the browsers if you need to test on the latest version.

**Test tools:**

Install the tools you would use regularly in your testing. I have the following tools installed on my test machine: Dropbox, Fast Stone Capture, Tree Size Free, Unlocker, Webex Recorder, Rapid Reporter, FreeMind and Skype. Some organizations have a strict policy about the software installed. Follow the rules and regulations.

**Enable remote desktop connection:**

There might be cases where you might need to access your machine from meeting rooms or from other machines. Enable/Disable remote connections to this machine based on your need. If you are not supposed to change system password, restrict remote access to a select set of users. You would not like anyone to disturb your test environment.

**Configure VNC [Virtual Network Computing]:**

I enable VNC so that I can share my desktop with programmers and they can troubleshoot or investigate any issue on my test machine.

**Change system password:**

One of the safest ways to preserve your test environment is to change the default password and not share it with anyone. This helps keep elves away.

**Take backup:**

You never know when something goes wrong. It is recommended to have a restore point to which you can always rely on. Even to save a particular environment, take backup.

**Create folders:**

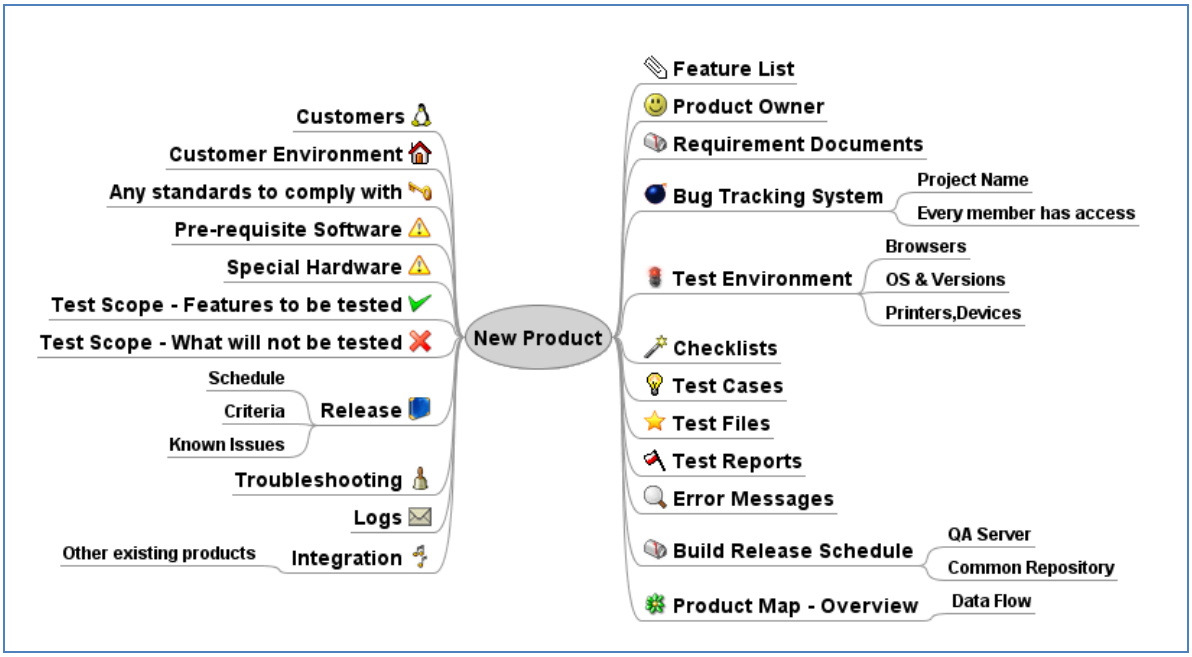
Create different folders based on their purpose - Screenshots, Test Files, Builds, and Test Reports. Even at a later date, this helps you to know about your testing quickly.

## New project – new product

How happy you will be when a new project or a new product comes your way and your test manager nominates you as part of the test team? I am always happy when I get a chance to test a totally new product. In fact, I look forward to the challenge of learning the product from scratch, reading and understanding the multiple documents, product workflow, customer requirements, logging the first bug of the project and so on. Sometime, you might be involved in testing a new product.

If you have always worked on updates to products which were already in the market and have never experienced testing the first build of a new product, here are some tips:

- Get to know the programmers, product managers, test team, technical support team as soon as possible. You never know when any or all of the interactions might give you the information to help you test better.
- Try using the following mind map, fill the information and circulate to the stakeholders. Ex: What you assume as the customer environment might not be the same as the actual customer environment.




- Lookout for teams who test similar products and check if there is any learning you can apply to this new product.

## So many test cases to execute

When I had a lot of disposable time, I used to go to other teams who were working on different projects. I would request a few minutes of their time and present them a test idea related to their application. Sometimes the test idea would highlight a bug. In some cases, the bug would be a known issue and in other cases, I would get a strange (for me) reply: "There is no test case for it." Don't be surprised if I tell you that in those cases of missing a straight forward scenario, the entire testing activity was based on test cases alone.

The procedure was simple:

"Read requirement documents, write test cases, log bugs, write a test case for every bug (if not present), test and release".

Following the above procedure meant that there existed a lot of test cases to be executed. Suppose you are asked to join such a project team, what do you do?

What if there are a lot of test cases?

Do not panic, I have been in these situations and it is definitely manageable.

Here are few tips that worked for me:

- Spend a few hours out of your daily schedule and perform time boxed testing sessions. This activity helped me find bugs faster.
- Highlight to your management that writing a test case for every bug found takes valuable time out of actual testing. Customers would be happier if more and more bugs are discovered and fixed rather than all bugs being documented as a test case.
- Do not test the software. Use the software. How do you feel using the software? What kind of issues are you facing? Write down your experience, any garden paths and report them as issues.
- Reduce the total number of test cases to a manageable number. Do not fall into the trap of equating quantity with quality.
- Have checklists and guideword heuristics for a particular module/feature of your application. They are easy to maintain – analyze, report and update.
- If management still insists on using test cases, spend some time with team members and have a concise list of robust test cases. Once again, remember that quality of tests matter more than quantity of tests.

## Very few test cases

Testing is an act of investigation and discovering quality related information about the software. To an extent, test cases help a tester find bugs. Teams who do not practice exploratory testing find that testing with help of test cases is easy. In such cases, test cases act as a guideline to the testers. It gives them confidence about test coverage. Teams are not afraid of leaving a feature untested. Test cases document acts as a map for testers who need direction and guidance.

Suppose you are one such tester. You join a team where most of the testing activity is based on test cases and there are very few test cases. What do you do? How many more do you need? Is there a formula to find out how many test cases should be executed for a project? According to me, there is no such formula to find the total number of test cases. If there exists such a formula, beware – that might be a trap.

So in this case, you want to add more test cases. Few tips from my experience:

- Understand the mission. Write test cases that help you meet the mission. Identify the quality criteria covered by the current test cases.
- Prepare a list of user scenarios. Who might use your product and what might be their use cases? Talk to your sales, marketing and tech support teams to identify the potential customer base. Write test cases to cover the most common scenarios.
- If the product is already in the market and there have been issues reported by the customers, study them. Do you think any particular feature is totally untested for lack of test cases? Write test cases for those features.
- Study the requirement documents, find out the requirements which you feel are not covered by existing test cases. Write test cases to test the uncovered requirements.
- Requirements and the understanding about the product and the project evolve over a period of time. The tester can now think of more test cases as and when (s) he tests the product. Write down those points, write test cases based on the current understanding.
- Provide the test reports of your initial testing sessions to the stakeholders. Ask their feedback on tests, missing tests, unimportant tests and repetitive tests. Based on the feedback, write test cases to have a consolidated test suite.
- Ask the testers who wrote the test cases about the process involved in writing them. Did they have enough time? Did they miss any test cases due to lack of time? Concentrate on the pointers. They know the best as they are the authors of the current test suite.

## Handling multiple projects

There are quite a few instances where multiple projects run in parallel. Sometimes due to availability of testers, a single tester might have to test multiple projects. There might be situations where only you know about those projects and they all have the same priority. What if a customer issue crops up in project A when you are busy testing project B? The fun begins when more variables come into picture like different domains, test environments, bug trackers, programmers, product managers, testers, tools and processes. Each of the product managers wants their product to be tested first but you are the solo tester.

What if you have to test multiple projects in parallel?

- Have a separate checklist highlighting each of the products variables. A subset of the checklist might look similar to this table:

| Parameter | Project A | Project B |
|---|---|---|
| Version | 1.5.0004 | 4.5 PP3 |
| Machine Name | http://pqm28.com/abc | testmachine - 2 |
| OS | Windows XP, SP3 | Windows 7, 64-bit |
| Browser | IE 8, FF 4, Chrome 10 | NA |
| Bug Tracker | Seibel | TestTrack |
| Programmers | Ashish, Ankit | Eveline, Harish |
| Sharepoint | http://myshare.com/ProjectA | \\projects\Project B |
| Released Version | 1.5.0000 | 4.5 PP1 |
| Username/Password | Central/C@pitol | Administrator/ajay |

- Have separate test environments. A lot of time can be saved if a separate machine is dedicated to each of the projects.
- Time-box your testing sessions and dedicate a few sessions to every project based on priority. It is better if you devote uninterrupted blocks of time for testing project A and then repeating the same with project B. Juggling between projects at the same time might have an adverse effect on the quality of testing.
- Focus on the task at hand. It is very easy to get distracted when multiple tasks pile up. Always remember that everyone is more interested in the quality of the project than quick completion of multiple projects.

## Prioritizing tasks

Are you a tester who is constantly bombarded with multiple tasks that you are unsure of which task to complete first? Do all the tasks occupy the top slot in your To-Do list? Do you complete a task and then realize that other tasks were to be completed first? Do you have trouble prioritizing tasks?

Here are my suggestions:

- Have a clear line of communication with those who assign you the tasks. When both of you have different deadlines in mind, problems arise. It is always better to understand the priority of the task before you start the task.
- Avoid distractions – do you get distracted by phone calls, meeting requests, emails or instant messages? You would be surprised by the amount of time you would save by avoiding such distractions.
- Make use of time zone differences. If there are two people who assign you tasks, and only one of them works in the same time zone as yours, you know whose task takes the higher priority. If you need information from someone who is online only for a few more hours, make sure you get answers to your questions during that time.
- Have a brief outline of your schedule – meeting time, breaks, setup time etc. Once you know your schedule, you are well prepared to agree on a deadline for any task.
- Understand if a particular task is blocking multiple high-priority tasks. Complete the task early so that other tasks can be started when required.
- Finish the task whose result is needed right now. Ex: If you have to reply to a customer who is facing problems with the application and you have to update the antivirus on your test machine, which task will you finish first? It depends. If your test machine is virus-infested and spreading virus to other machines on the network, it makes sense to update the antivirus immediately. At the same time, if the customer is facing a showstopper, it is worth giving him your full attention.
- Ask the 'What if' question. What if you don't complete any of the tasks? What are the consequences? Who will suffer the most? The answer should help you decide which task should be completed earlier?
- Which tasks can be completed only by you? Suppose there are two tasks – task A & task B. Task A can be completed by any member of your team while task B can be completed only by you. Which task would you pay attention to? What if you start off with task A?
- Finally, note that prioritizing tasks is not an easy task. Remember the lessons learned when you did not prioritize correctly. Making mistakes is okay as long as you don't repeat them.

## Save testing time

One of the traps many testers fall into is wasting a lot of time. Apart from avoiding the tasks which do not contribute to the final goal, there are many other ways testers can save a lot of testing time. As a tester, I believe that we need to spend more time testing the software. As time is constant, we might have to change the way we work to save time. Here are some tips that might be useful to you:

- **Unlocking your screen**

Every time your machine gets locked after fifteen minutes of inactivity, you need to unlock using CTRL + ALT + DELETE combination. Then you enter the password. By tweaking the registry, you need to enter just the password.

- **Connecting to remote machines**

When you need to access multiple machines on the network, one way is to Run > mstsc and then entering the machine name, password and the domain. Instead save the credentials as a shortcut. By double clicking on the shortcut, you will be connected to the machine.

- **Remember shortcuts**

As testers, we might have to install and un-install applications many times. Do you know that Run > appwiz.cpl launches the 'Add/Remove Programs'? Suppose you take a print screen and paste on MSPaint application, the canvas size is more than the image size. Do you manually resize the canvas size? Before pasting, I press CTRL + E, adjust the canvas size to 1 x 1 and then paste the image. This ensures that the image size is always greater than the canvas size and I need not spend time cropping.

- **Use tools to help testing**

Install tools like Greenshot or Jing which help edit images quickly and save them to a pre-defined folder every time you press Print Screen. Have you tried Rapid Reporter? It is a brilliant time saving tool for a tester.

- Record your testing activity as much as you can. It is better to have lots of recorded videos than re-test the scenarios again.
- Save time by using Hexawise if you need to test multiple combinations and you do not know which combinations to test.
- Make a list of special characters with their ASCII codes, the location of test files, test reports – Print them & pin it on your cubicle. It saves a lot of time than spending time searching for them.
- Google is a powerful tool. Do not ignore it.

## I want to create a mind map

Have you ever used mind maps? Do you want to learn how to create a mind map or where one can use mind maps? A mind map might help you organize your thoughts quicker. There are multiple short videos by Tony Buzan on mind maps, their usage and how they are better than lists. Darren McMillan has a dedicated blog post on how he uses mind maps for different purposes in testing
http://www.bettertesting.co.uk/content/?p=956

You can start mind mapping using a paper and pencil. Draw the main topic of your mind map at the center of the page. Branch out sub-topics or ideas from the main topic. Keep adding points, use different colors and draw images where appropriate.

Slowly, shift to a mind mapping software - Freemind or MindMeister. You can install FreeMind onto your machine while you can access MindMeister online. Do not judge your initial maps. The flow of ideas is more important than the map. Concentrate on generating ideas than the design of the map.

You can try creating mind maps for the following topics:

- Parameters involved in a successful tweet.
- Components of a bug.
- Test ideas to test a mobile phone.

Practice creating a few maps using the mind mapping software. Once you gain confidence, try the next level - collaborate and create a mind map. Work with another tester and edit a mind map simultaneously. Can you think of ideas without getting biased by another tester? Mind Meister allows multiple people to work on a single mind map.

Generate test ideas for a product on a mind map. Test with the help of the map. How does it feel? Try updating your test results on the same map. Color or highlight the ideas that have been tested. Report your testing results for a complete feature.

Create mind maps to explain a feature to someone. A mind map conveys a lot of information quickly compared to a document. It is easy to branch out, update, and edit a document using a mind map. Try presenting about a topic using a colorful mind map. At CAST 2011, James Bach presented his keynote with help of a mind map. There have been many webinars which were based on a few maps.

Some of the tools for mind mapping are FreeMind, BluMind, MindMeister, XMind, and NovaMind. A list is available on Wikipedia.

Remember that the first few maps might have a lot of information missing. As you practice mind mapping, you will benefit by the maps you create. Keep practicing.

## A particular type of bug is found

In a test cycle, there might be instances where each tester specializes in a particular type of bug. Some might find usability bugs while others identify functionality bugs. Here is a pattern – tester A finds type I bugs while tester B finds type II and so on. Is there a problem here? I would say that it depends on what troubles you. If you are worried about missing a category of bug every test cycle, it might be a problem. If you are happy about the diverse skills and the bugs uncovered by the team, I do not see any problem.

Even if you feel that there is no problem, how about learning to find bugs other than the type of bugs you find every test cycle. There might be scenarios where one of the testers is unavailable. How will you make sure that the bugs are not missed?

Here are few tips to learn how to test for different types of bugs:

- Be the note taker when the tester is testing. Try to understand the thought process of the tester. Once the tester has finished testing, ask questions.
- Test and prepare the test report and get feedback from the expert tester within your organization or friends circle.
- Read the bugs logged by the tester and understand the test idea. Why did you not think of the test idea? What similar test ideas can you think of?
- Ask the tester to present his learning to the team. It could be in the form of a presentation, diagram, mind map or a video.
- Participate in forums discussing a type of testing. There are groups within these forums discussing only about a particular quality criteria. Discuss and learn from them. Two forums stand out: www.softwaretestingclub.com and www.testrepublic.com
- There are many competitions where testers log different types of bugs. Study those bugs. Examples of sites include – www.99tests.com ; www.utest.com
- Remember that practice is the key. The more you practice sharpening a skill, greater are the chances of improving the skill. Make sure that you have the right guidance. Compare your bugs against other testers' bugs.
- There are enough videos on Google by good testers highlighting the different testing techniques. There are many webinars by skilled testers. Register, learn and practice.
- There is free online Skype coaching given by Anne-Marie Charrett. Make use of this wonderful opportunity. Ping 'charretts' on Skype.
- Write an article highlighting your knowledge. Post it on the internet and ask for feedback. Experts are ready to help you if you are passionate and show an interest to improve your skills. It is you who has to take the first step.

## Analyze a bug

As testers, there might be many instances when you might have to analyze a bug. It could be a bug faced by a customer after the product release. It could be the bug found well before the product release but the programming team finds it hard to fix. Even before logging a bug, it would be a good idea to analyze the bug – find its root cause, find the simplest steps to reproduce the bug and try to maximize the impact of the bug.

How would you feel if a programmer highlighted a more severe effect of your bug? What if an additional step to your bug crashed the system? What if the scenario you mentioned was more generic than highlighted in the bug? From my experience of finding and reporting bugs, I have few tips which have helped me analyze a bug:

Three things differentiate a bug – steps, root cause of the bug and customer impact.

What are the steps to recreate the bug? Are the steps mentioned in the bug report the only steps? Are all the steps mentioned in the bug report required? What happens if you miss a step? Can you recreate the same bug in less number of steps?

Are you sure that you have found the root cause of the bug? Is it the root cause or yet another symptom of the real bug? What tests confirm that it is the root cause of the bug?

Finally about the customer impact – does it really impact the customer? Is there any other scenario more severe than the one you found? What happens if you alter the data a bit to discover more critical effect of the bug?

As highlighted in BBST Bug Advocacy course, when you want to find the follow-up bug, try the following:

- Change the steps.
- Change the data being loaded in the program – it could be the test files.
- Change the program settings – you could change the preferences the program uses for the particular test.
- Change the platform settings – operating system, browser.

Once you have tried the above points, you will have more information about the bug. It is a good idea to analyze the importance of 'time' factor for the bug. What happens if you delay the steps or if you perform the steps quickly? Does the time delay between steps have an adverse effect?

Also, search in the bug database to determine if similar bugs are logged. Do they have a pattern? Are you too reporting one of the symptoms? Can you talk to the programmer? Maybe increase the debug level and find clues.

It should be interesting to analyze a bug. Good Luck!

## Invalid bugs

As testers, we find information about the product. Management makes decisions based on the information it has. One of the ways testers provide information is through "Bugs". In most of the cases, once a bug is logged, it is assigned to the development lead and to the programmer. Either it is fixed, deferred, or rejected.

What if the bugs are rejected?

What to do when the bugs are labeled "invalid", "not a bug", or "rejected"?

Overall it poses a big problem to the project if there are lots of invalid bugs:

- The tester does not understand the application.
- The programmer does not understand the bug reports.
- Testers and programmers might have different views on the application's behavior.

In my experience as a software tester, I have followed a few tips which might help you too.

- Read the comments provided by the programmer.
- Do not hesitate to talk to the programmer directly if it is possible.
- Close the bug if you are convinced with the justification provided by the programmer.

*If you disagree with the comments,*

- Re-open the bug immediately.
- Add your comments to the bug and assign it back to the programmer. If you and programmer have different opinions, try to resolve them. In case you cannot resolve, assign the bug to the product manager.

Remember that as testers, we are information providers and NOT decision makers.

For example, suppose your application uses a third-party HTML editor for composing emails. Though the HTML editor will be part of the final product shipped to the market, the programming team has no control over the code of the HTML editor.

In some cases, the entire module might be revamped in the next release. Considering the cost vs. value, it might make more sense to fix any bugs once the new code is written. So, before you open/close a bug marked as invalid/rejected/not a bug/deferred, read through the comments thoroughly. If you do not understand, collect information before taking any action on the bug.

## Non-reproducible bugs

There are times when bugs are hard to reproduce. Commonly, they are termed as 'non-reproducible' bugs. Programmers find it difficult to fix a bug if they are unable to reproduce it on their machine. If a bug is easily reproducible, the chance of it being fixed seems to increase rapidly. As testers, we must ensure that most of the must-fix bugs are fixed.

If you are facing a lot of non-reproducible bugs in your testing, some of these tips might help you:

- Take notes of your testing activities. It could be on a notepad, sheet of paper, Rapid Reporter, Session Tester, Jing or any other tool. I personally recommend Rapid Reporter.
- Record your testing activity using screen capture tools. I use WebEx Recorder during my testing sessions.
- Try to find the critical condition for the bugs.
- As stated in the BBST Bug Advocacy course, vary each of the following parameters and take notes
    - The steps
    - The options and settings of the program
    - The data loaded into the program
    - The software and hardware environment
- Talk to the programmer about the bug. Provide as much information as possible: screenshots, probable steps, logs, video recording, system configuration, build details, test results etc.
- Study the bug database to find similar bugs or bugs with similar symptoms. They might give you further insights about the non-reproducible bug. Also, remember that a particular bug might be reproducible only on the nth attempt.

Example: An application did not crash when the user imported jobs. If the user imported jobs ten times, it crashed. The application would again crash only after nine more attempts.

Recommended Reading:

How to Investigate Intermittent Problems – James Bach

Anticipating and Dealing with Objections: Irreproducible Bugs – Dr. Cem Kaner

## Unclear bug reports

I am assuming that either you are a tester or someone who manages testers. One of the most important task of a tester is to file bug reports. Every individual is unique and have their style of bug reporting. Some testers write excellent bug reports while some bug reports make no sense to the reader. For such testers, bug reporting might not be their forte. They might be very good at bug hunting and bug investigation – but what if their bug reports are unclear?

Writing good bug reports is as important as investigating a bug and its effects. It is a well-known fact that good bug reports add to testers' credibility. Good bug reports make it easy for everyone to understand the bug, replicate (if necessary) and take steps to resolve it. So, if you need tips on writing good bug reports – here are few of them:

- Understand the testing culture of the team and the organization. Do you file every bug report? Is there a bug template? Do you follow the bug template? Does the programmer always ask for more information even though you have filed the bug report as per the template?
- Include a brief summary of the bug – (Component) (Effect) (Step)

  ***Good Example*:** Call Log: Missing entries after five calls in twenty minutes.
  ***Bad Example*:** When a user calls many times, he does not see few entries in the call log.

  Observe how the summary is framed using just ten words. If the summary is not catchy, few people ignore your bug report. Make sure unnecessary words are omitted.
- Include the pre-requisite or the base state of the system such that the reader can also reproduce the bug easily. Do not assume that everyone is replicating the bug from the same state of the machine.
- Highlight the critical condition. Sometimes, the critical condition is lost between a lot of information and the bug report is returned with the remark – Not Reproducible.
- Screenshots, test files, video recording, log files, backup of the machine state are useful attachments to a bug report. Remember that the extra effort at the bug reporting stage saves a lot of time later.
- Some bugs are specific to a particular test environment. Make sure that you include the test environment details.
- If you feel that the current bug is related to a similar bug, cross-reference that bug. A list of related bugs helps to figure out the root cause.

## Missed a bug

There are some days when you or your test team finds a good bug. Sometimes, you miss a bug. When the customer reports a bug, it is always a source of learning. Either you learn about your testing or the way the customer uses the software.

'Why did you miss the bug?' 'How did you miss the bug?' Do you face such questions in your day-to-day tasks? Does anyone ask you these questions as soon as a bug is found in production? How do you respond to such questions? Do these questions freeze your thought process? Do you test with the constant fear of missing a bug? Are you or your team members judged based on the missed bugs?

According to me, there is no perfect testing and not all bugs can be found during test cycles. If we consider the cost vs. value scale, sometimes the missed bugs are costly to find. So, what to do when you or the test team misses a bug?

Few points to consider are as follows:

- Was this bug a one-off miss or a simple bug bound to be found by good testing?
- Was there a test idea to find the bug? Was this test executed? If not, why? If yes, what made the bug hide? Why was this test idea missed? What can be done to avoid such instances in the future?
- Is it an environment specific bug? Did you test with the same test environment? What was the critical condition responsible for the bug?
- Was this issue observed yet not recognized as a bug? What factors contributed to this observation – lack of domain knowledge, poor testing, unreasonable deadlines, difference in opinion among stakeholders?
- If the tests were not executed due to lack of time, what other tasks were considered? Which of those tasks can be ignored in the next test cycle?
- Are there any lessons which can be applied to other projects? Is there any indication that similar bugs are being missed in other projects too?
- Could this bug be found during unit testing itself? If yes, spend some time analyzing the cause of missing the bug.
- Think if you missed a trick by not using the appropriate testing tool. Could the tool have helped you find the bug? Maybe there is a need to change the testing process and this bug is just a symptom of a bigger problem.
- Consider which testing techniques would have helped find the bug. Which quality criteria were ignored?
- As you get to know more and more about how the customer uses your product, ask yourself the question: 'Which features do you focus on during testing? It is better if you run the important tests on the features that the customer will use than hundreds of tests on features that they will not use.

## Regress old bugs

One of the types of testing that some testers and management gain confidence by is Regression Testing. Suppose the test team is provided with a build which has lot of bug fixes. The management is more focused on the quality of the existing features and wants the test team to focus on regression testing. How do you start? What do you focus on? With so many bugs to verify and regress, what is your approach? Do you have time to test all the combinations? Do you verify just the fixed bugs or do you want to explore related features?

In the projects I was involved with, Regression testing especially – regression of old bugs was given a lot of importance and rightly so. Some critical bugs were re-opened and new test ideas were discovered. When projects are tested over duration of few months, testers tend to learn more and more about the product. After a lot of tests and interactions with multiple stakeholders, the tester's knowledge increases.

The new tests helped us find new bugs. So, what if you need to regress old bugs?

- Make sure you understand the exact steps of the bug logged. It could be a total time waste if you try steps which have nothing in common with the bug and you expect the bug to appear. It gives a false sense of confidence that the bug does not appear in the current build.
- If you have access to the code or the programmer, try to know the cause and the solution to fix the bug. Is the fix a temporary fix? Was the programmer able to identify the real cause of the bug?
- Where else can similar bugs appear? How about tests to check if there are no similar bugs in the product?
- Test to determine if the original bug is fixed. There have been a lot of cases when other test cases seem to work well but the original bug was re-opened. Have you tested the original bug?
- What about the related features, steps other than the one mentioned in the bug report? Can you be sure that the features which were bug-free are working fine even now?
- Use your product knowledge to think of scenarios which might uncover some bugs. After few weeks of testing, the increased knowledge will help you look beyond the original bug.
- A lot of focus is usually on the recent bug fixes and the bugs which were fixed a few test cycles ago. One of the important points to consider in regression testing is if the original purpose of the product is still fulfilled. Do the core and the critical components continue to work as expected? Make sure you test it.

## Test reporting

You might be a very good tester in finding bugs. Your team might look up to you when a hard-to-reproduce bug is to be discovered. One of the skills of a good tester is 'Test Reporting' skills. What is the use if a tester is unable to report his testing activity to the stakeholders? I attended the Test Reporting session by Ben Kelly at CAST 2011. He highlighted two types of reporting – Pull reporting & Push reporting.

Pull reporting is when someone asks you for a report. Push reporting is when you provide someone with a report. You are the initiator.

When you prepare a test report, some of the important questions to ask yourself are as follows:

**Purpose:**

Do the test reports serve its purpose? Who will read them? Is the test report clear and easy to understand? Are you using terms or abbreviations which need an explanation?

**Type of report:**

There are different types of test reports. Bug reports, quick testing report, comparison with similar software, comparison with previous versions, test ideas report, test coverage etc. Each report highlights different aspects of test results. Make sure you understand what kind of report is expected from your testing session.

**Format:**

What is the format of the test report? Is it a pdf, html, .wrf, .csv or a .mm? Is it the format expected by the stakeholder? Do they need special software to read your report? It will not help me if you provide a test report that I cannot read if I do not have the software used to create the report.

**Timing:**

What is the frequency of the report? Does your test lead or manager need them daily, weekly or after every testing session? Is someone else waiting for your test report? Do you send the individual reports or consolidate to a single report?

**Tools:**

What tools do you use to prepare the report? Does the tool have options to export to the required file format? Do you spend time converting the report to the desired format? Is your test team expected to use a particular tool? I like Rapid Reporter. It exports your report to a csv file or an html.

## You have INR 5000 to spend

One of the key points testers must focus on is to improve their skills. There is no one single way to improve your skills. Some improve their skills by reading books, attending courses, participating in tester meets or even practicing with their friends. Gaining skills is neither easy nor a short term task. It takes lots of dedication and right practice to achieve good skills.

As experts say, work on building your reputation. One day, the world will know you through your reputation. So, how much are you ready to invest to improve your skills? What if you have INR 5000 to spend on improving your skills? How will you spend the amount? Let me tell you how I would spend the amount of INR 5000.

Apart from the thought of investing in a bank or stocks, here are few quick tips that have helped me improve my skills:

- Try to find a group of friends who share the same interests. Imagine a group of 3 testers each buying two books. Instead of one tester buying six books, the group has six books. Everyone can exchange the books after they finish reading the book. This also encourages good discussion and debates.
- Suppose you do not meet any such people and you are all alone with INR 5000. Spend INR 600 and buy the two books – 'Lessons Learned in Software Testing' [Bret Pettichord, Cem Kaner and James Bach] and 'Testing Computer Software' [by Cem Kaner, Hung Quoc Nguyen, Jack Falk]. These books are a great addition to any tester's collection.
- Attend the testers meet conducted in various cities. You would be surprised by what you can learn in such a short time. Testers' meets are ideal to meet testers, share and learn about different contexts.
- Scout for training courses offered by experts on the internet. Sometimes, they do not cost much. Demonstrate your passion and skills to the tester and request for a discount. You never know unless you ask. I attended 'Rapid Software Testing' course and it helped me a lot.
- If you are someone who benefits more through classroom style of learning compared to self-learning, AST Courses match your expectations perfectly. Register yourself at AST. I completed BBST Foundations and BBST Bug Advocacy courses. You can take online coaching classes from Michael Bolton, James Bach or Anne-Marie Charrett.
- Register for testing competitions. Invest in any source which would help you earn more money. Example: There is a security testing competition – spend money to learn about security testing. Gain skills which will help you win the competition.
- I would like to quote Robin Sharma:

  'To double your income, triple your learning'.

## You are going on a vacation

You are the STAR tester of your team and after many busy months, you get a well deserved vacation. What do you do during your last few days before the vacation? What if you are going on a long vacation? Can your colleagues continue their testing without disturbing you on your vacation?

Here are few tips to work on before you start your uninterrupted vacation:

### Out-of-office message

Make sure you have configured the Out of Office message. It is good to let others know when you would respond to their emails. Also, it helps others to know whom to contact in your absence. Do mention the contact points for different projects.

### File permissions and folder sharing

You don't want to get back to your office and discover people complaining that they did not have access to a file or folder. The problem? You forgot to give them access.

### Licenses, invoices and devices

It makes sense to hand over the devices you used for testing so that others can use it if their devices are broken. If there are any important invoices or documents, share it in a common location so that you will not be missed. If you are the sole owner of a list of licenses or codes necessary for your application, make sure you share them with one of your colleagues. Let them take care of this responsibility while you are away.

### Keys

If you were in charge of the inventory or the lab and you have the keys, it is time to hand them over to someone else in your team. You don't want the locks to be broken when you return. Do you?

### Schedule sessions

If someone in your team is dependent on you for a particular task, schedule sessions where you can train them about the task. You could give them few sessions on how they can carry out the tasks in your absence or provide a document highlighting the steps.

### Assign tasks

If you are the senior tester who assigns the task to your colleagues, then make sure you assign tasks for your entire vacation. You do not want your colleagues to be wasting their time as you did not assign them sufficient tasks.

## Valuable Resources

Apart from the many links shared in this book, here are a few more which will help you:

Rapid Software Testing: (James Bach and Michael Bolton)

http://www.satisfice.com/rst.pdf

Resources on Exploratory Testing, Metrics, and Other Stuff: (Michael Bolton)
http://www.developsense.com/resources.html

Software Quality Characteristics: (The Test Eye)
http://thetesteye.com/posters/TheTestEye_SoftwareQualityCharacteristics.pdf

The Little Black Book on Test Design (Rikard Edgren)

http://www.thetesteye.com/papers/TheLittleBlackBookOnTestDesign.pdf

Essential Test Design (Torbjoern Ryber)

http://www.ryber.se/wp-content/EssentialTestDesign.pdf

Free Magazines: (Simon Schrijver)

http://simonsaysnomore.wordpress.com/2011/09/11/free-magazines/

This is the end of this book. Hope this book is useful to you and your team.

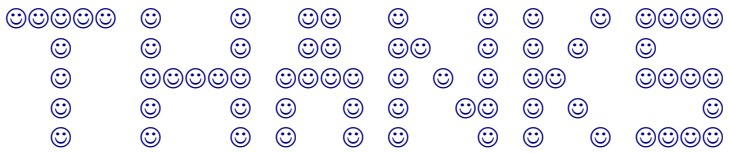
If you have any comments on any of the chapters or the book in general, feel free to contact me. I will be more than happy to discuss about the topics.

**My contact details:**

Email: ajay184f@gmail.com

Skype/Twitter/GTalk: ajay184f

LinkedIn Profile: http://in.linkedin.com/in/ajaybalamurugadas





*Ajay Balamurugadas*