



QUARDEV

Integrated Software Solutions

Quardev Monthly

Volume 3, Issue 1

January 2009

Welcome!

Welcome to the Quardev Monthly - the new version of our newsletter, now published monthly instead of quarterly. We look forward to sharing insights and helpful information in the areas where we know a thing or two - testing, quality assurance, technical writing and documentation, project management, and consulting.

This month we are highlighting Test-

ing - with an article by Ross Mortimer about Testing Smart, which has tons of tool and strategy suggestions.

Enjoy the newsletter with our compliments and please contact us with questions, comments, or article ideas.

-The Quardev Crew!!

Testing Smart

Ross Mortimer, Lead SDET, Quardev, Inc.

How do you successfully test software when the odds are against you? Where do you start? What do you need? Who can you count on? All these questions take on a heightened importance when you are dropped into the middle of some crazy project and told that your job is to make sure that the crazy project doesn't blow up. Usually these crazy projects are over schedule, over budget, or both. Projects can morph completely out of scope and the test team is often the last in the line to deal with the software.

There is only one way to get through this - test smart. But what does that mean?

Step 1: Talk, write, phone, visit, but COMMUNICATE!

Open communication up as wide as possible. Learn who the players are and what stake they hold in the pro-

ject. Let them know who you are. Make sure that everybody that has an interest in this software knows what the score is in real time. Don't sit on information and don't assume that just filing a status report to your immediate lead is going anywhere and you're off the hook. Talk to project managers, engineers, tech support staff and even marketing and sales or even the CEO if needed and not only relay information from your testing, but share any potential impacts you see on the others. One difficult thing to remember when trying to establish good communications is that you may be ignored. If this happens try to avoid drama. No one wants a testing diva in some meltdown. Once the information is passed on, you have done your job. You don't have to win all the battles. Be honest, be accurate, and be concise.

(Continued on page 2)



QUARDEV

Testing Smart, Continued

(Continued from page 1)

Step 2: What do you have and what do you need?

Part of Step 1 is to establish what resources you really have and hopefully negotiate what you really need to be successful in your tasks. You should try to model part of your testing to an environment that is close to your user. Know how they work and what they expect. Testing has an element of checking the design, but if you don't understand your user-base, how can you possibly have their interests at heart? You wouldn't expect a short-order cook to be an expert in medical equipment? There is a human factors aspect to consider:

1. Target audience is an expert on the intended use of what you are testing and a power user of the System Under Test
2. Target audience is an expert on the intended use of what you are testing and a novice user of the System Under Test (SUT)
3. Target audience is not an expert on the intended use of what you are testing and a novice user. of the System Under Test (SUT)

Compile a list of available resources and a list of missing resources. Once complete, write a rationale for getting what you need. See how much you can get through channels and then where resources are missing, make a note in the test plan and the impact it will have.

Step 3: Have a realistic test plan.

A test plan should be your contract with the rest of the team. It states what you can and cannot do. Avoid creating a test plan that is rigid and hard to maintain or obsolete and irrelevant.

What makes sense?

I recommend my common-sense template. Not original and you may recognize it, but you can't go wrong.

1. What is being tested (or not)??
2. Who is doing the testing (or not)?
3. When is it being tested (or not)??
4. Where is it being tested (or not)??
5. Why is it being tested (or not)?
6. How it is being tested (or not)?

Write as little or as much for each step to suit your needs. Remember you have two target audiences, the person who is approving the plan and the people who are actually doing the work. Clearly specify any and all assumptions and risks at each stage and remember this is your contract with the team, so if doubt arises, this is your first line of defense.

This can be as simple as checklist of "To Do" items with names to a masterwork of detail and pretty graphs, but if it doesn't tell your story, chuck it. This is about what you REALLY are going to do on the project. A well-written test plan pitches itself. The reader should nod at the end and think "Yep, that's the way to do it..."

Step 4: Gather your tools.

If you are working in a company that is been around for a bit, a lot of the tools you need for testing may already be around for you to use. If possible reuse as much as you can (so long as it will not have a negative impact on you).

Let's just assume though, you are in a startup or a group that has little or no resources and you need to be extra clever about what you do. You probably don't have a lot of money for purchases so let's look at what you can get for free and how it can help you.

I have listed what I consider to be a good collection of tools that will allow you to do a lot. Unfortunately, I can't go as in depth as to why I think you need

Testing Smart, Continued

(Continued from page 2)

these apps I as I would like because this is beyond the scope of the article, but hopefully I can tackle some of this going forward. Don't forget to add learning the tools into your planning.

To find these apps you can Google search them (or your preferred search engine):

1. If you can buy Microsoft Office, you should. It just makes life that much easier. It is probably the easiest way of making quick tools for testing. If you can really be indulgent, I cannot recommend enough Microsoft OneNote. It is one of the most versatile and expandable tools I have found for collecting my notes, tracking my testing and keeping a repository of specs, tech notes and even my daily status reports. You can use OneNote as a collaboration tool amongst your team to make a brain trust to retain your group knowledge over time. I love OneNote.
2. If Microsoft Office is not an option, I suggest Sun's Open Office for free. It has a lot of the basics of Office and with a little practice will get you up and running pretty easy. It will allow you the freedom to design your own processes.
3. FreeMind – a mind mapping piece of software that lets you brainstorm ideas for test cases and other non-linear thinking. FreeMind, is well, free.
4. yED – a free flowcharting type app that allows you to create workflow type charts and easily re-arranges them when you need to add or delete data.
5. TreePad – a free utility that allows you to capture scraps of data and organize them for later use.
6. StickyNotes (or some variant) – Post-it notes for the computer. It is always great to be able to jot down quick notes and this is a matter of personal preference, but I find it helpful in keeping reminders of what I am seeing when I am working through a tough bug.
7. ScreenHunter Pro – A free screen Capture app that will help in creating visual supplements to your documentation and screen captures of bugs you may find.
8. Microsoft Virtual Desktop Manager (from their PowerToys suite) – This allows you to have four desktops available to you to flip back and forth and keep your screen for being too cluttered.
9. Notepad++ - A seriously robust upgrade from Notepad. This is a free app that has all the benefits of Notepad but so much extra. If you've used something like Textpad or UltraEdit, you will be quite at home.
10. XMLPad – Since it seems that XML is so prevalent these days, I like this free editor.
11. GanttProject – A free MS Project knock off. Pretty good for lightweight tracking of tasks. A good way to sanity-check scheduling and managing testing. One drawback is it works in whole day increments, so really short duration tasks are hard to manage.
12. EssentialPIM – a free, handy little personal information manager that looks like MS Outlook for ease of use.
13. Serif Plus PhotoPlus – Serif Plus releases their older versions as free software as an incentive to get their new stuff. If you want a free Photoshop knockoff that is not GIMP (weird UI), I suggest this as an alternative. You have to Google "Free Serif" to find it. They also have several other apps, a free drawing program, page layout, and Web site designer. They are teaser apps, so some functionality is not turned on, but the basics are there and all quite usable.

Testing Tools (Windows)

1. Bug database – you can get a canned bug database of all shapes and sizes. I don't have a specific recommendation since your organization's

Testing Smart, Continued

(Continued from page 3)

requirements are unique. Some things that I would strongly suggest:

- A bug entry form that is clear and easy to use
- A good query mechanism. It is amazing how many bug bases are really pretty and clever and yet searching them for duplicate bugs or similar bugs works poorly.
- A good reporting mechanism for bug metrics and reports.

The plus of a canned database is you don't have to build it. The downside is that you can't build it around how you really work. I like homegrown tools because of the flexibility and scalability, but I admit if time is not on your side, it is better to go with something proven than something that half works.

2. Test Case Manager – pretty much the same as the bug database requirements. However the nature of test cases can be as simple as paper checklists, so a high end solution is not always warranted.
3. Virtual machines – Virtual Machines (VMs) are a great way to test different configurations without having lots of hardware. You install software that allows you to have a computer running inside the computer. You can try different kinds of tests and not mess up your main computer. Testing with VMs does have its detractors. VMs are slower (with enough RAM not so bad) and can sometimes give you less than optimal results if you are testing something like networking or some types of hardware that are not meant to be shared with VMs. Microsoft Virtual PC does not support USB devices very well. So using a VM has its place.
- VMWare Server 1.x (not 2.x) – free application that enables you to run VMWare

applications and use the snapshot feature that allows you to rollback to a good state in your VM and is very helpful for testing and isolation.

- Microsoft Virtual PC 2007 – Like VMWare is free and allows you to run a virtual machine Microsoft style. There is no roll back feature, but you can get Windows VMs from the Internet that are from MS that are time-limited intended for browser testing (also free) which can save you having to buy full on extra copies of Windows if you are on a budget.

Automation

2. If your application is capable, you can automate with Microsoft's built in automation hooks like Reflection in .Net apps, or COM/COM++ in non-managed apps. You can use the traditional MS way with VBA or WMI. I once wrote a test harness in MS Excel that ran my automation and made pretty charts at the same time. Microsoft has a lot of different ways to do automated testing and you just need to pick a flavor that is right for you.
3. Autolt 3.0 – A free, robust Windows automation tool. It has a lot of good, easy to learn scripting assets and if programming is your thing, great utility.
4. AutomationBox – A free command line utility that you can use with Autolt 3.0 for some powerful test automation.
5. PC Magazine's InCtrl5 – a very useful utility (not free, but very cheap - \$7.97 for the download from PC Magazine) that is useful for capturing before and after software install states. It tracks everything installed or modified in your system. Excellent for isolating rogue files.
6. Microsoft Windows Powershell – A excellent shell for Windows that can give a Windows tester some of the benefits of Unix shell scripting. Also plays well with other MS technologies for automation like WMI and VBScript.

Testing Smart, Continued

(Continued from page 4)

7. Selenium IDE/RC – If you are testing a Web application, this might be an excellent free choice for you. An easy to use automation tool that enables you to develop your automation in several popular languages such as Java, Python, C# and even HTML.
8. PushToTest – a free web based framework. Has a bit of a learning curve.

Debuggers and Other Stuff

2. Windows Debugger (windebug.exe) – Get this and the Windows symbols for the different OSs. You'll win engineering friends with good debug stack traces!
3. Microsoft SysInternals Suite – Too many tools to go in depth on, but trust me, you need this suite. It is a great way to have tools that will get under the hood.
4. Application Verifier – A free stress test tool from Microsoft that will do all kinds of nasty things to your system to stress it out.
5. Microsoft Platform SDK – free and has a bunch of handy goodness in it.
6. Windows Resource Kits – these are tools that MS tech support uses and they have great tools for helping solve issues.
7. Microsoft Fiddler 2 – a free HTTP debugger if you are testing Web applications.
2. Try to maintain a consistent way of versioning your builds that allows you to reference a build that everyone will recognize. Some places I have seen do the .1, .2, .3, etc., naming, some by date. It only matters that it is consistent.
3. Keep your open bugs open and your closed bugs closed. Nursing along the bug database to be up to date is an amazing time saver in the end.
4. Same with your test cases. Not only keep them relevant, but mix them up to give variety. Running the same cases in and out will make testers lazy and think they can gloss over them. Lots of bugs get missed.
5. Meet with your key people frequently to stay in touch. Giving timely updates and getting updated will only help you react more effectively.
6. Don't burn out. Usually testing gets shorted in time and burning out only affects the end result. Remember it's the last thing you test before releasing into the wild that matters most. Ask for help when you need it.

In conclusion, this article only touches on the very tip of the iceberg but hopefully will give you some hints and clues for going in the right direction and save you pain that others have felt. Again testing is as much as an art as a formal process, the more creative and smart you are, the better your results will be. Don't be afraid of changes. Just be smart with when and how you do them.

Happy bug hunting!

Step 5: Start testing.

Since testing varies in so many ways wherever you are, I can only give you a few helpful hints to help you: