

A Little Paranoia Can Go A Long Way: Part II

Testing and bug-tracking is not an exact science, though most of us would like it to be. Feeling bad vibes can support a good risk analysis.



Roman Civin

ROMAN CIVIN explores what makes us shudder about localisation testing.

How many bugs are enough? A strange question, indeed. Everyone knows that measuring productivity and quality by bug numbers is, to say the least, a disputable activity, and that productivity in testing is a far trickier variable. Let us open up some of the suitable approaches you can take as a test lead who wants to be clear on bug definitions, avoid unnecessary bugs and help the team learn more.

The assumption is that most customers want only relevant bugs logged. Once you log a bug in the testing phase, everything becomes more expensive, because each bug is managed and meets several people during the cycle. To wit:

- **Find ways to eliminate the bugs prior to product testing:** This can be done through automation pre-testing, proper software QA prior to testing phase, well planned globalisation and internationalisation, and well-timed milestones to name a few.

- **Know which behaviour can be ignored,** so that 99% of what to enter as bugs actually gets fixed. Your "not-a-bug" list could (not) contain these items:

Localised dialogue boxes do not have to match source layout, if they appear OK
Context menu hotkeys

Non-localised components:
Product names
Very trivial cosmetic bugs, e.g. 1 pixel misalignments
Functional bugs caused by a missing key component

- **Provide training and feedback to your teams** to avoid duplicate and non-reproducible entries, not only before the project starts but also as it progresses.

- **Be sure to have the testing types well defined in your testplan,** in order for your engineers to perform the right testing. Otherwise you might end up with dozens of minor language errors that your Turkish tester finds on the product's functional testpass, which you know cannot be fixed. Or one and the same bug might get logged 4 times only with a different platform specified. Solve like issues before you start!

- **Focus on "postponed" and "won't-fix" bugs in previous versions,** especially ones with low severity and priority: do they need to be logged?

- **Focus on "postponed" and "won't-fix" bugs in previous versions,** especially ones with low severity and priority: do they need to be logged?

Some of this might help you avoid logging irrelevant bugs. Be aware that you may have a project or a language where your customer sees relatively few bugs logged, and therefore becomes concerned and suspicious. They will ask you, for example, why there are so few bugs, whether your team is on track and if not why not. There really could be something wrong, so do dig down and be ready to explain.

Your test engineers are undoubtedly smart, but it always seems right to teach them to rely on their common sense when in doubt. In case of behaviour that you saw the first time around it is better to log the problem once and consult with the customer on how to deal with further occurrences.

Prep tasks	preparation before project; OS setups and config; build setups; additional installations such as tools; server labs setup and config
Testing tasks	carrying out tests; running scripts
Admin. tasks	logging results; analyse automation logs; bug verification; bug closure; investigation on potential or ambiguous bugs; build/component problem solving; queries

Tracking the nose on your face

What is the optimal level of tracking on a test project? Let us take the viewpoint of a vendor test manager who wants to keep the project budget under control, and who is ready to answer the following customer question anytime: "What were these hours spent on here and why should I approve it?"

There is a Czech saying that goes, "To get a lot of music for little money". Most

Fig. 1: Three test tracking categories for advanced activity-based projects

customers today require excellent quality under very tight budgets. The majority of test projects today are activity-based, as opposed to resource-based, and majority of test management is thus with their business partner. The difference is that you get volumes in testing hours for an activity-based project, while for resource based your customer would say, "I need 20 people for three months to test my new release," and will pay for those.

There are several risks to consider when the test PM decides how to set up the project tracking:

- **Despite good intentions for transparency, tracking standards may get complicated or garbled** and do more harm than good.

- **Activity types are easy to follow but they don't give you the information you want - tracking does not allow for good reports and budget control.**

- **There is no tracking** and you never know where have all the hours gone.

- **You do not report what you spend** despite a tracking process in place.

So what should you track and how?

First of all, be sure your tracking will work with the structure of your budget and

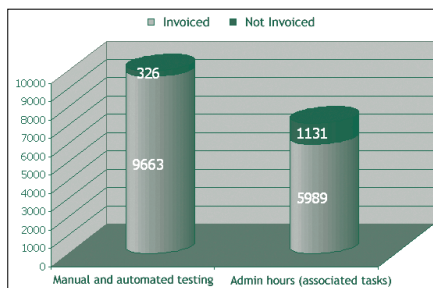


Fig. 2: High level test project results: Tracking systems should allow you to go deeper into the task level and unveil inefficiencies.

- **In specific test areas consider testing only target files.** When you perform the same checks repeatedly on many builds and cannot automate this, you might focus on testing the target, rather than ill-conceived comparisons with the source. For example, you can apply this approach in the testing of a web learning course.

volumes from the customer. If the customer gives you volumes for preparation, manual testing, automated testing and final testpasses, consider tracking these task areas. Be aware that the tracking process will be each test engineer's daily routine. Fig. 1 shows three test tracking categories for advanced activity-based projects. It is very useful in the beginning, when you need to benchmark many tasks and verify estimates. Later in the project you can, to a large degree, discard this tracking and use higher level means. Be reasonable and keep this a process with a purpose that is understood by all, otherwise it is a waste of time.

It is worthwhile to introduce something like an invoicing checkbox in addition to the categories, specifying which tasks can be invoiced to the customer and which can not. Tracking activity groups allows you to provide precise reports and see where you are with estimates and actuals (See Fig. 2). Moreover, post-mortem analysis and evaluation of this data is priceless — you will see clearly what went wrong and where to make changes next time.

Orchestrating and outsourcing multiple testing teams

Your testing department has probably experienced something like this: You win a proposal, a great opportunity that you and the team have spent weeks on. The actual project takes 2 months to plan and prepare, but, suddenly, your close and trusting customer suddenly requires you to start in two weeks! It looks like you should join capacities to build a team of fifty test engineers for a 6 month project in no time and make sure it is one team that works. In your mind, the risk analysis looks beastly. Splitting the teams and locations could do more harm than good, especially in software testing and engineering. After all, you have been highlighting the value-add through team centralisation during the entire sales cycle. The list of inefficiencies you could suffer starts rolling out before your eyes. Chain handoffs, missing and lost information, triplicated preparation effort, duplicate, disparate quality standards, and low productivity, not to speak of problems with daily throughput, which is key for this project. These could be the potential results of testing on several sites simultaneously. But you do have capable staff at the central and branch offices, reliable testing partners, and a backup resource solution in place with 10 trained test engineers which you can have set-up and running in one week, don't you? Is it not a scary experiment to split the testing team in three on one project? What do you lose and what do you gain? What should you do to create a reliable and dynamic model with three parties involved? And how would the project run in this setup? The answer might be outsourced testing.

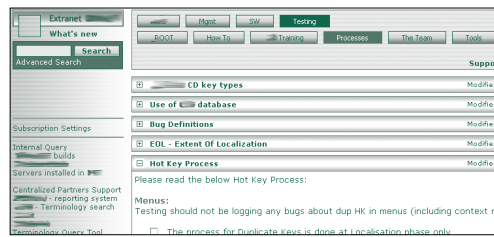


Fig. 3: Example of a project extranet

Let us assume there are 15 languages to test and a less reliable build plan which makes it hard to predict the weekly workload. Your solution is to create three main test locations; one at your Asian branch, one at your business partner in your city and one in your headquarters.

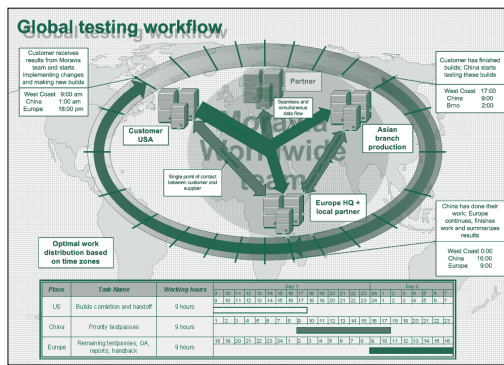


Fig. 4: Making the most of global workflow

Prerequisites for a successful virtual team:

■ **Technology, infrastructure and process must be seamless.** If your branch offices and partners use the same hardware, have appropriate file transfer and connectivity, are familiar with your shared database tools and implement your test processes, then half of the problems are solved.

■ **An excerpt from a checklist unified for all locations could look like this:**

- Test machine: DELL GX260
- Upload/download internet bandwidth: 4Mbit
- File transfer: customer replication server X
- Backup transfer: Secure FTP
- Servers: Exchange 2000 SP1
- Project sharepoint: <http://extranet.companysite.com/project>
- Project knowledgebase

Obviously it is best to play an easy tune first and verify these processes on a small scope project. Then you can be confident to play the symphony.

Efficient team model:

- **Maintain a single point-of-contact with the customer,** and have experienced local test leads on each production site.
- **Be sure there is only one project manager;** the production sites on large projects usually have a local PM.
- **Communication and information sharing:** Be sure the testkit is available to all and is shared from one location together with the testplan. If not provided by the customer,

create your own and share it on the project extranet portal. IP phone, Netmeeting, instant messenger and telephone are your daily means. Extranet online reporting, tracking and assignment tools are more than useful (See Fig. 3).

Making multiple locations an advantage:

The Earth is round and $3 \times 8 = 24$: While making sure your working hours overlap, you can do more in one day by setting up the appropriate workflow (See Fig. 4). The customer hands off to a single vendor contact and the Asian team start right after handoff. Then the European teams take over. At the end of the European day you can report the overall results to the customer. This solution is great when you need fast tested content turnaround. It relies on precise completion tracking and online reports within vendor production sites.

■ **Centralising, sharing, and work distribution experience:** The centralisation does not get sacrificed. It is there at the level of test leads that are responsible globally. The processes are set as if the team was sitting just in 3 rooms of one building. The right team needs to do the right stuff — in this case it would be the double-byte languages that get tested in Asia. One production site would usually specialise in demanding server testing. However, Asian teams should have the same capabilities of testing Latin character languages. This model allows for a more dynamic resource planning within the whole testing group.

■ **Return on Investment (ROI):** An added benefit of this model is that it allows for better strategic positioning due to performing testing in price-attractive regions.

We test engineers and (test) managers should pretend potential risks are a little more serious than they appear. Not waving them off can provide you with a probing analysis and will give you more control over your project results. In this short series we have tried to show a few real-life examples, tips and methods that I believe help better identify and clarify the stakes of project preparation, test planning, tracking and team models. Be a little paranoid at sensing issues; then reasonable, QA-conscious and aware of customer value while solving them. ■

Roman Civin is a Senior Manager with Moravia IT who oversees the Testing Division in Moravia's Headquarters in the Czech Republic. He is responsible for European and Asian functional and linguistic testing services, including localisation QA, automation and internationalisation. Roman, a native Czech, has a degree in English Language and Literature. He can be reached at RomanC@moravia-it.com