A Little Paranoia Can Go a Long Way

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 explores what makes us shudder about localisation testing.

Roman Civin

Testing and bug-tracking is not an exact science, though most of us would like it to be. The more experienced we get as testers or test managers, the more aware we are of things lurking around corners, and the what-could-go-wrongs that will go wrong. Sounds a little paranoid, right? In this first instalment about what makes us shudder in localisation testing, we describe some elementary and more complex issues that testers encounter (or not) while providing some hints to help solve them. After reading, I trust you will be a little more paranoid too!

A healthy approach to testing should be tinged with a bit of paranoia. Add in a few parts of mistrust and you will be in good shape. Feeling bad vibes can support a good risk analysis, detect inappropriate test coverage while suggesting improvements, avoid redundant preparation, ameliorate an inadequate test plan, and for good measure, find leaks in the test process for a given product. Bearing this in mind, we must remain QA-conscious and flexible; otherwise we can become paralysed, instead of being an inconsolable worrywart. Being "appropriately mistrustful" while adhering to a "customer is always right" attitude is a contradiction that must live together in the area of localisation testing. The following are several real-life examples. If you are a localisation testing expert, you may have demons such as these discussed here.

Test preparation control: create, then choke?

Even some advanced projects have just a week of test preparation. You receive the key information and competencies only a few days before the start date. We must *always* be ready for this scenario and I will discuss this in one of the following issues of *Localisation Focus*. For the moment, let us focus on projects for customers who have a major product release once every year or two. Before the behemoth gets rolling, a lot of effort and trust is put into innovation of the cooperation model and the process or tools with the client's established partners. A dream, right?

What more could you wish for? You are invited to contribute, suggest, implement and you have time to adapt and optimise your own environment for the new big project. It adds great value and helps build a lasting relationship. Together, you decide to upgrade the test database by reporting features and allowing the partner team to test the new/fixed features. You cooperate on creating a tracking and reporting model; detailed and elaborate testing categories and a benchmarking model for testing volumes. Now imagine that upstream, you (the vendor) provided other complex services for the software publisher including translation, software engineering, etc. for this project. You are so ahead of yourself that the feedback and preperation stage can last a quarter.

The situation then becomes something like this: the testing stage typically would start much later than the localisation stage, yet we are encouraged to work on areas that should happen a few months later. As the testing start comes closer and the localisation kick-off meeting is well behind you, the then-promising testing initiatives receive less attention and are put aside unfinished as "extras", because things like product-readiness and tool key functionality problems emerge as a priority. You waste your effort on something that looks useful early on, but cannot eventually be applied. Needless to say, the change is permanent and there is no point complaining about it. But it makes your team reflective; mournful for great ideas that are gone forever.

What would a more-than-slightly paranoid tester, who felt these feelings, suggest for a better test prep the next time he is invited?

Consider just a few simple things:

- Plan the test preparation as a sub-project with the customer and try to agree beforehand on how both parties might benefit. (You may not want to invoice the customer for this)
- Make sure there is a team appointed for this cooperation
- Make sure the start date is not too early, not too late.
- Make sure localisation kick-off and testing kick-off are two different milestones

What is clean?

As an everyday occurrence, the paranoid or not-so paranoid tester prepares and tunes up the operating system images for testing. It seems obvious what a clean image means, but an experienced tester knows there are a few potential pitfalls.

Voice inside head:

Can I keep the testing OS clean and secure?

Other voice inside head:

Sorry, I cannot reproduce your bug. Is there something wrong with my clean system?

There are ways to be sure your work is protected and your test machine is not littered with installations from the company logon script, which installs many 'useful' things. Make sure no antivirus gets installed on the machines, or other internal software. Verify with your admin team as to what is the best topology to keep the testing domains secure from outside attacks. You can use a test domain, which is protected from the outside world and has a trust with the production domains. Allow access only to key sharepoints in the production domain, which are indispensable for the team. The test domain should work with your test servers only and admin rights are usually granted for the testers who work with them.

Voice inside head … again:

Should I allow others to edit the image?

Other (pesky) voice … again:

If you use a disk image casting server in your company to store and manage system images, it pays to have the important project images set as "read only" and allow for documented copies. It reduces the risk of your perfect image to be undesirably modified or even corrupted.

Voice of doubt:

How do I know it is the right language?

Voice of reason:

Make sure you are testing on the right language. You have tested the product on 3 languages already and now you run on the fourth, say one of the 10 new EU entrants-Slovenian-using an image someone else created. If you test the language for the first time and you are not a proficient speaker, make sure it is Slovenian by learning (at a minimum) basic phrases of the language you are testing: Yes, No, OK, Cancel, Next, File, Edit.

Have the specific keyboard layout in front of you, as well as the national language standards (see www.microsoft.com/globaldev/reference/keyboards.aspx) or use a utility that displays the graphic keyboard layout.

So, what are the things that a good clean system should NOT contain? Install nothing but the product on the clean system, unless there are other requirements. It is wise to have just the localised input language installed on the machine and not any additional reference keyboards. Fig. 1 shows US Windows XP environment with Asian language pack and inputs installed. Though the Simplified Chinese input is not turned on, the files are there on your machine and can have impact on the product you test, not to speak of accidental keyboard switchings. Tip: Pretend you only speak or write the language you are testing.

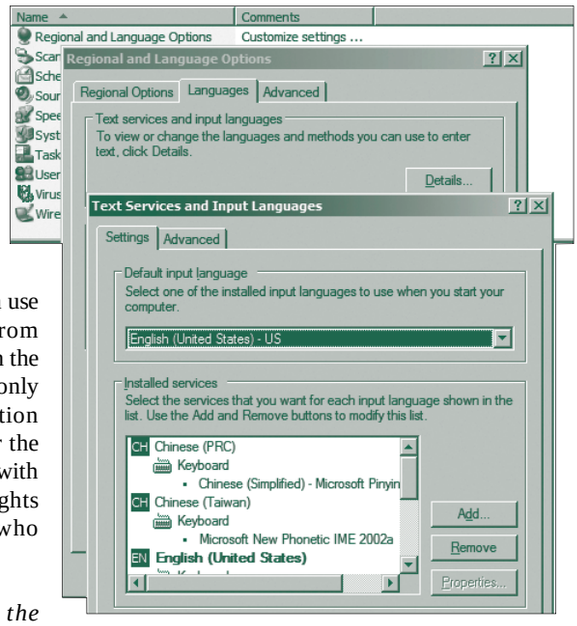

Fig. 1: US Windows XP environment with Asian language pack and inputs installed

Coverage and team on a global project

Say you have a web service product in 11 key European and Asian languages and relatively high-level test cases. The customer puts their trust in you as a global, experienced testing partner and would leave the language-specific approach up to you. Often, they do not make a distinction between French and Traditional Chinese testing and they expect the same level of quality and efficiency on each language. (Perhaps, though, double-byte support and Asian internationalisation has already been identified as a potential risk). As a senior tester or test project manager your job is to decide on the right approach in the test plan. Do Chinese Traditional, Chinese Simplified, Japanese and Korean (CCJK) require Asian language speakers to do the testing? Do you need to split the team and test in several parts of the world?

Questions to consider:

- Is the main task to verify localised product user interface and functionality with arbitrary localised input in the local environment?
- Does your analysis tell you the largest amount of problems would be User Interface (UI) problems?
- Could a large part of testing be effectively covered and supported by suitable tools — UI checkers lke Tool Proof? Automated scripts?

In this case there may be little purpose for a Korean speaker testing Korean. The value here is that a smaller team can do the whole job and leverage more languages, be it in Asia or Europe, without impairing the product quality.

Are text input methods or dialects important during the testing or are there features like "search" to be tested?

Is any linguistic testing in play, i.e. do you need substantial contextual verification or identification of specific localised text in combined or dynamic features of the product? Does the tester have space for variation and decisions?

The point here is that if the test cases do not read like an idiot box (just do what you are told), you should rely more on tester experience and a proficient speaker testing Chinese could be an advantage. Inasmuch as they can uncover serious problems not directly described by test cases, they can decide to put a low priority or ignore verifications that do not apply for their language (e.g. alphabet order or specific formatting). A good solution here could be two international teams; one in Asia, one in Europe, integrating the geographic area language experience. If your process and project management is well synchronised, distance is not a real problem: you can appropriately answer customer needs and maintain value by having the right people do the right level of testing.

Next time we will look at the following topics:

- How many bugs are enough?
- Should we select a scenario that is likely to produce the most bugs? What to consider?
- Tracking the nose on your face!
- What is the optimal level of tracking on a test project?
- Multiple testing teams and outsourcing
- Is it not scary to split the testing team in two on one project? What do we lose and what do we gain?
- You are too far away – Don't I need a partner who lives closer?
- Why do customers want a test partner who lives next door? ■

Roman Civin is a Senior Manager with Moravia IT (www.moravia-it.com), 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