Using Bug Taxonomy to Design Better Software Tests

By **Michael Stahl** - October 16, 2017
https://www.stickyminds.com/article/using-bug-taxonomy-design-better-software-tests

Summary:
In software testing, bug taxonomy involves defining feature categories and collecting lists of possible bugs in each category. These lists can be used to give inexperienced testers some starting points, to help experienced testers brainstorm new ideas, and to evaluate the completeness of a test case. Using an existing bug taxonomy can be useful, but creating your own is even better.

As part of the research and development department, my team develops new products. R&D has a halo of innovation and discovery, but as I am sure any of you who work in R&D knows, the relative part of the work where we actually do new things is rather seldom. Even when we develop something new, once the actual work starts, most of it means redoing things we have already done.

Some examples: You develop a new embedded wonder that solves world hunger. The final product must be field-upgradeable to be able to solve the ozone layer depletion on version 2.0, so you need a "FW update" feature. Haven't we done this before? Sure we did! Any embedded system has it. So once again we are facing the same challenges and (alas) the same bugs. It's the same situation for the install program, a database module, etc. If it's not our own team that developed something like it before, some other team did something similar and developed test cases. Yet here we are, inventing the wheel once again.

It's redundant and inefficient. There should be a way to convey accumulated knowledge between projects, between veteran and new engineers, between teams in your organizations, and even between different companies.

Getting Started with Bug Taxonomy

One way to share knowledge about testing is through bug taxonomy. As the classification term suggests, the general idea is to define feature categories and collect lists of possible bugs in each category.

These lists can be used to help experienced testers by providing more ideas about what needs testing, to give inexperienced testers a list of things that need to be tested and present product aspects they may have not realized need testing (like specific aspects in performance and usability), and to evaluate the completeness of a test case list by selecting items from the taxonomy list and checking whether they are already covered.

As an example, here is an excerpt of a list from the book *Testing Computer Software*, by Cem Kaner, Hung Q. Nguyen, and Jack Falk:

Performance

- Slow program
- Slow echoing
- Poor responsiveness
- No type-ahead
- No warning that an operation will take a long time
- No progress reports
- Problems with time-outs

Some of these items are fairly obvious functionalities that need to be tested, and others will trigger ideas for tests we may have missed.

For example, the bug “No warning that an operation will take a long time” triggers a number of validation ideas: Are there any functions in our product that may take a long time? Under what conditions? Do we provide progress tracking? Is the tracking accurate? It can also trigger thoughts about redundant features: Do we provide progress tracking for things that will take a very short time, with the result of a window popping up for split second and going away, leaving the user wondering what just happened?

Another interesting thing to note is that some of the ideas in the list have an expiration date. For example, the type-ahead functionality was a requirement for applications in the first generations of computers, when a fast-typing person would easily overrun the screen update speed. With the latest CPUs, this is usually not an issue.

Creating Your Own Bug Taxonomy

Although I am familiar with a number of taxonomy lists, I never really used them. I tried, but I admit I did not try too hard. So on one hand, I believe taxonomy lists are a good idea, but on the other hand, it can be inconvenient to use them.

The mindset and thought process of whoever wrote the list is not necessarily (more likely: necessarily not) identical to your way of thinking. The result is that items you’d expect to find under a certain category appear in a category that does not make sense to you. Take an example from the list above. The list developers decided to put “No progress reports” under Performance. I’d expect to see it under the heading User Experience. The language used to describe items, which by the nature of these lists must be terse and concise, also may sometimes be difficult for you to understand.

The result is that to reap the benefit from a taxonomy list for a specific product aspect, you must read large parts of the list, filter out many entries that are irrelevant to your product, and accept that you won’t understand some of the entries—a dubious experience.

So we have a good idea, but it was implemented in a way that is hard to use. Should we just give up on it?

I don’t think so. One option that I know from experience works well is to develop your own lists for things you encounter repeatedly. Because you wrote the list, the organization will make sense and the language will be clear to you. This increases the likelihood you will use the list on your projects.

I have one such list, prepared a long time ago, outlining aspects of API testing. Every now and then, when I learn or discover a new API coverage item, I add it to the list. No, I will not share this list, for the reason above: that it is written in a format I understand but will not be immediately clear to others. When I do share it, I personally walk through it with the recipients to make sure they understand it. For me, this list is very valuable, and I've used it numerous times. I also know of other lists created by others in my company for their own needs.

This anecdotal evidence does not qualify as a rigorous statistical survey, but I do think it points to a possible direction: Bug taxonomy works, but it can work better if you developed the taxonomy list yourself.

You develop such a list by collecting details over time. You can do this by yourself or as a group, but if you choose the latter, I recommend you invite not only testers, but also developers and product managers, to contribute ideas. You can even take a deep breath, go over published lists, and extract ideas that are relevant to your situation. Even if you won't find anything directly reusable, it is very likely to trigger some ideas.

As a starting point, you can find a useful list of **login test ideas here**. It is also a good example of using a mind-mapping tool in the creation of taxonomy lists. For **a comprehensive article on bug taxonomy**, check out this paper by Giri Vijayaraghavan and Cem Kaner.

So good luck, and if you develop something you feel is useful and clear enough to share, please consider publishing it somewhere public!