Surveying the Terrain

Originally published on StickyMinds.com May 2, 2008

As testers we learn to be very good at communicating detail. Each bug log describes a single bug, showing specific symptoms, with specific steps to reproduce. If we've done a reasonable job of categorizing the bugs, we or our stakeholders can extrapolate some useful generalizations from our bug database about the state of the system at a given point in time.

We're also pretty good at providing whole-project dashboard-type information: assessments of quality and progress at a very high level.

But there are situations where neither of these is good enough. On troubled projects—or in fact in any circumstance when a project could benefit from a directional view of the state of testing and fixing, we need to give thought to presenting something more like a survey of the terrain. A Survey Report can be both a useful communication tool showing different progress in different parts of a system, and a helpful test management tool, showing you where to focus testing for best impact. In a previous column, *Pack Up Your Troubles,* I alluded to this kind of report as a way to present a test team's findings about system quality factually and unemotionally.

Here's a sample template for the kind of simple, structured and colorful report I find useful.

	Summary Assessment			Execution Detail				Verification Detail		
Functional Area - POS transactions	**simple (happy path)**	**medium**	**complex**	**GUI**	**complete txn to post**	**applicable discounts**	**other (specific to txn)**	**virtual receipt**	**printed receipt**	**inclusion in reports**
Sales										
Returns										
Rentals										
Manual Discounts										
Delivery										
Voids										
Post Voids										
Recall										
Suspend										

This Survey Report was for a retail Point of Sale (POS) system. As you can see, each row represents one functional object of the system, from a business point of view. (The original had many more rows, while this extract covers only the POS transactions.) Each column represents something meaningful to the business that either works or doesn't for each transaction. The table, plus a legend, gives management a survey of testing progress and the system state.

In this example, the first three columns give a summary for three levels of complexity. A green for "simple" sales means that all the columns to the right of the summary are green for that complexity level. In the legend I define a simple transaction as one with a

single-product sales basket, the most vanilla customer type, the most straightforward sales tax category, etc.

The columns to the right of the summary map out what works and what doesn't for any summary that's yellow or red. Maybe the GUI is fine for a complex sale and the transaction posts correctly, but the virtual receipt (what the cashier sees on the screen) doesn't match the printed receipt. Anything we haven't touched stays uncolored. I add a final column for "Comments" or "Notes".

If a cell is yellow or red, I put the bug IDs either in that cell or in one beside, so we know exactly which problems are getting in the way at each point. Yellow is for major severity bugs; red is for blockers or critical bugs. (Obviously these will vary somewhat depending on project standards.)

When a project team is under pressure, the testers may need to show why they can't test faster. This report shows clearly what you can and cannot do, and you can use it to reduce the temperature when there is contention about the actual state of system quality. And it's actionable information. Like a literal survey, the project management team can use it to decide where to go next. If no transactions are posting correctly, then fixing the post functionality will enable more testing. That will allow the test team to find the next area that's blocked. Testing information grouped into a survey like this helps people steer the project better than they can from individual bug reports or a set of dashboard numbers.

The key to making a Survey Report successful is getting the right functional objects in the rows, and column headers that progressively map out success for each group of objects—whether those are pages, transactions, or functions. Keep it as simple as possible, but don't try to constrain everything into one structure. It often makes more sense to use different column headers for different groups of functional objects.

After *Pack Up Your Troubles* went live, some readers asked me for an example of the kind of simple graphic report I mentioned. I sent them the example above, asking for feedback in return. Last week, reader Dave Russo related how he successfully used the report format in a challenging client situation, and sent me this variant he designed to report on a web application.

	Summary Assessment			**Execution Detail**			**Verification Detail**			**Bugs**	**Comments**
Functional Area	**simple (happy path)**	**medium**	**complex**	**Web tier & GUI**	**Business Logic tier (esp. calcs)**	**Database tier**	**Doc or Tracking Numbers**	**Printed Forms**	**Inclusion in reports**		

His column headers are simple and meaningful, and can be used to show incremental progress.

Here's a somewhat different example, showing an in-progress version of a Survey Report I used on a User Acceptance Test of a banking check processing system.

	SUMM	ASSESSMENT DETAILS									
			Functional		Reports		Procedures		Infrastruct		
Functional Area	Start next cycle	Ok Test?	R Y G	Bugs	R Y G	Bugs	R Y G	Bugs	R Y G	Bugs	Notes
Capture	G	G	G		G		G		Y	R115,	
Reject repair	G	G	G		Y	314	G		G		
Block Balancing	G	G	G		G		G		G		
I3 backend CL jobs	G	G	G								
Archive ingestion	G	G	G	315	G		Y	298, 299, 300	Y		UAT archive performance issue
Verify capture	Y	G	Y	315	G		Y	298, 299,	G		
Exception update (1)	G	G	G								
Exception update (2) - remote capture	R	R									
Query archive	G	Y	G						Y	316	

I used this Survey Report during the shakedown of a new build following an infrastructure upgrade. The column structure reflects a broader categorization of what is and is not working than the two previous examples. I added the “Ok Test?” column to indicate our assessment of how thoroughly we had tested each area. This structure worked well for a shakedown.

Always design your report in business & risk terms for the system you are working with, so all the stakeholders can understand it. The terrain for each system is different, so the surveys will be organized differently. If you design your report from the business point of view, management and sponsors will see increasing stability and progress towards completion in successive reports, in terms meaningful to them.

Because each project is unique, setup and maintenance of a Survey Report is entirely manual. Especially if you haven’t used the survey categories to model your test from the outset, it can be labor-intensive to populate initially. But I find that the process of designing the categories and analyzing our test results to populate the table is indispensable for the test team. And once we have done the initial work, I can keep it up-to-date via a quick daily review with my testers.

I have used project-specific Survey Reports many times. Stakeholders like them and find them easy to understand. Even hardcore metrics fans appreciate the concreteness and the picture they present of system terrain.