Effective Test Status Reporting

Most software test teams exist to assess the software's readiness prior to release. To achieve this goal, two primary tactics are used:

1. Execute test cases or scenarios that are likely to find errors, resemble actual usage, or both.
2. Report the test results, the defects found, and defects fixed, which, collectively, make up the test status and reflect the quality of the software.

For the test manager, the first task category primarily involves managing inward: assembling the test team and test resources, designing a solid test system, implementing the test system, and running the tests in an intelligent, logical fashion. The second area of responsibility involves upward and outward management. Upward management includes how you communicate with your managers, other senior managers, and executive staff. Outward management includes communication with management peers in development, configuration/release management, sales, marketing, finance, operations, technical support, and so on. As a test manager, your effectiveness in reporting test status has a lot to do with both your real and perceived effectiveness in your position. To put it another way, your management superiors and peers measure your managerial competence as much by how well you report your results as by how you obtain your results.

Upward and outward communication of test status presents the test manager with different challenges. Even seasoned software managers may not understand your role. The field of software testing now includes a body of knowledge not even experienced test professionals can know completely. That makes the test manager's communication tasks even more difficult. Perhaps most importantly, when testing is done properly, delivering test status means relating "bad" news to your peers and superiors. Competent test managers usually don't tell the project management team, "All tests were run overnight; no significant bugs were found; ship it."

Management problems like these, unlike technical problems, don't often have hard-and-fast solutions. There are, however, techniques that can mitigate them. The following sections should help you, the test manager, become an effective presenter of test status.

The Facts

Reporting test status begins with the state of tests executed and bugs found. I organize my own test projects around structured, organized, measurable test systems with carefully devised test cases and formal bug tracking databases; from these I extract meaningful metrics and aggregate data. Some test managers use test systems with more documentation than I do, some are more comfortable with less, and you should use whatever test system works most effectively for you and your test organization. Regardless of the test techniques you select, I have found that the ability to measure your testing process against a baseline plan and to gauge system quality in terms of defect numbers promotes a deeper understanding than a purely qualitative approach can. Quantification of key indicators lays the foundation for understanding test status. As Lord Kelvin wrote, "If…you cannot express [what your are speaking about] in numbers, your knowledge is of a meager and unsatisfactory kind." A typical project can involve hundreds, even thousands of test cases and bug reports, so you have your work cut out for you.

When it comes time to prepare a status report or present at a status meeting, be ready to answer at least the following questions:

- How many test cases exist and what are their states (pass, fail, blocked, etc.)?
- How many bug reports have been filed and what are their states (open, assigned, ready for testing, closed, deferred, etc.)?
- What trends and patterns do you see in test case and bug report states, especially opened and closed bug reports and passed and failed test cases?
- For those test cases blocked or skipped, why are they in this state?

- Considering all test cases not yet run--and perhaps not even yet created --what key risks and areas of functionality remain untested?
- For failed test cases, what are the associated bug reports?
- For bug reports ready for confirmation testing, when can your team perform the test?

Since the answers to these questions will change hourly during test execution, you will always be slightly behind. A rush to report, however, will result in errors and incongruity. Balance timeliness, accuracy, and consistency carefully. I try to err on the side of the accuracy and consistency , but your management team may accept wooly results to get the latest data.

No matter how carefully I prepare myself, I find I can't know everything. For example, I can't speak from memory in detail about each and every one of three or four hundred active bug reports. Likewise, I can't answer specific questions about a single condition buried in one of two or three hundred test cases. When confronted with such questions, I smile and say something like, "Well, you win the game of 'Stump the Dummy!' I don't know the answer to that, but I'll research the issue right away if you'd like."

As long as you are usually well-prepared, forthright admissions of incognizance coupled with an unstinting offer to get the information immediately win out every time over winging it or dodging the question. However, do be ready to discuss the serious problems in detail. Astute managers will ask you fine-grained questions about any test case failure or bug report involving data loss, complete functional incapacitation, or impairment of a "crown jewel" feature.

Test Status and Project Progress

Once you have your facts together, relate those facts to your company's objectives. For most projects the goal is to deliver software that does enough, well enough, and quickly enough. How do your test case (quality risk) and bug (failure mode) findings affect that goal? Can the project team respond to your report without impacting budget or schedule? The test manager must consider these questions when reporting test status. For example, contrast the following two statements that could describe the same test findings:

> "We've completed our test suites, and what a mess. Test has found endless bugs, and most of these no one has even looked at yet, for gosh sake! We're finding more new bugs every week than development can fix. There's no way we'll exit System Test in three weeks!"

> "We have run all 517 planned test cases as of last Saturday. Currently, the bug tracking system shows a backlog of 247 bug reports either open or assigned to a developer but not yet resolved. The good news is that we only found ten new bugs this week, down from about 40 per week over the last three weeks of System Test. However, because the fix rate since the start of System Test is approximately 30 bugs per week, I believe our target System Test exit date three weeks from today is at significant risk. Perhaps we should consider a detailed bug review meeting to defer issues we feel are not critical for this release? After that discussion, we might have to address the current schedule."

The first statement paints the schedule as impossible and quality as hopelessly bad. The second statement explains test status in numbers, summarizes the risk to the project's success, and offers a suggestion for ameliorating that risk. Which statement will your management peers and superiors find most aligned with the company's concerns?

The Bearer of Bad News

While critical, just communicating in the context of project success won't suffice. Make sure people don't shut their ears or, worse yet, kill the messenger when you deliver bad news. Effective test managers are skilled at telling reasonable people particulars they don't want to hear. This is really hard to do, and the keys to success will vary for each manager and each management team. However, the following techniques can start you on the right path.

- Be calm and patient. A rational presentation communicates facts better than an emotional one, and you may have to explain some issues more than one way before everyone understands.
- Treat your fellow managers with respect and consideration. I never deliberately humiliate my management peers with embarrassing status reports.
- Don't assume or accept responsibility for quality unless it's in your job description. You are not Don Quixote, Lone Champion of Quality, you are a risk management expert, competent to report on specific tests and findings. Let your facts speak for themselves, let management pick what gets fixed, and let development do the fixing.

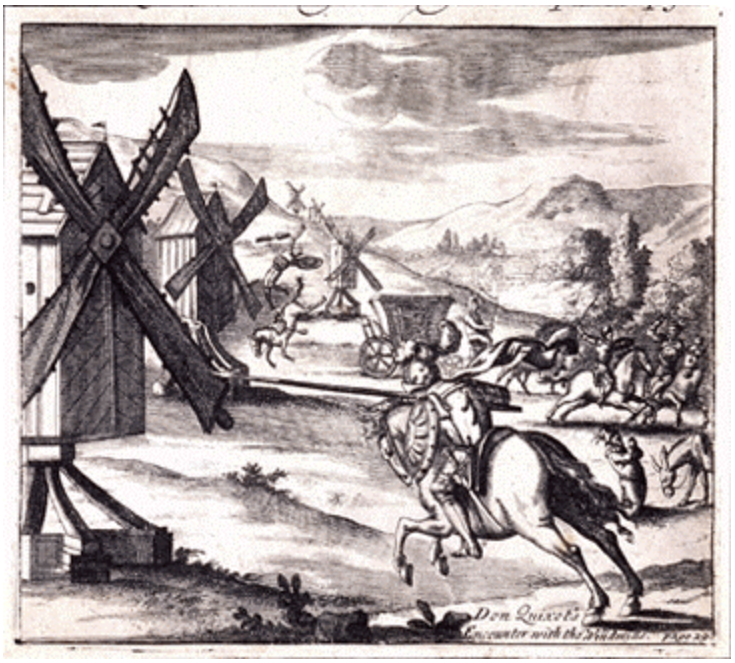


Figure 1: Don Quixote or Inflexible Test Manager?

- Try to report test status, even specific failures, in terms of schedule, budget, features, and quality, not core dumps, dereferenced uninitialized pointers, unsupported widgets, and the like. Those who share my technical background will have to learn, as I did, that "geek-speak" does not impress non-technical managers as often as it confuses them.
- Don't gloat about bugs! As a talented test manager, Reynolds MacNary, once told me, "Be optimistic on the outside, pessimistic on the inside." Your test team should find bugs, and you should be satisfied with your organization's professionalism when they do. Nevertheless, bugs *are* bad news for the project. Resist the temptation to present anything other than an appropriately somber demeanor when presenting failure data, especially when you'd just love to tell someone, "I told you so."

For formal meetings, as opposed to e-mail, posted, or other non-interactive test status reports, consider the following points as well.

- Arrive ready for status meetings with slides and supporting documentation. The more senior the audience, the fewer slides you should bring, but the more carefully you should prepare.
- Walk your audience through your slides, charts, and reports, pointing out the good news and the bad. What's self-evident to you, the test professional, may have no meaning to your management peers and superiors.
- Take suggestions and questions with a smile but avoid reacting. People are problem-solving animals, so you (and the development manager) will probably get lots of suggestions during test status discussions. Listen, consider, and be willing to discuss, but don't be in a hurry to change your carefully crafted test plan or processes in response to a proposal from a project status meeting.

While these ideas should get you started, keep your eyes and ears, heart and mind open as you deliver unpleasant news. Emphasize and adopt styles that work-and drop those that don't-with your colleagues.

Reports and Audiences

In addition to using your best gentle bedside manner with your fellow managers, you need to communicate at the right level. Present trend charts and summary reports to senior and non-technical managers, who need to understand the overall trends. Save the detailed reports about bugs and test cases for development, operations, and technical support managers who can best understand them. Let's look at some examples from which you can extrapolate to your own particular circumstances.

To show trends, two charts, bug reports opened/closed and test progress, provide concise, appropriate information. Figure 2 is an example of the open/closed chart. It shows the daily and the cumulative find and fix counts for bugs reported, which gives management an idea of the defect trends. A flattening cumulative open curve indicates stability, or at least the inability of the test team to find many more bugs with the current test system. A cumulative closed curve that converges with the open curve indicates quality, a resolution of the problems found by testing. Overall, this chart gives a snapshot of product quality as seen by testing, as well as telling management about the bug find and fix processes.

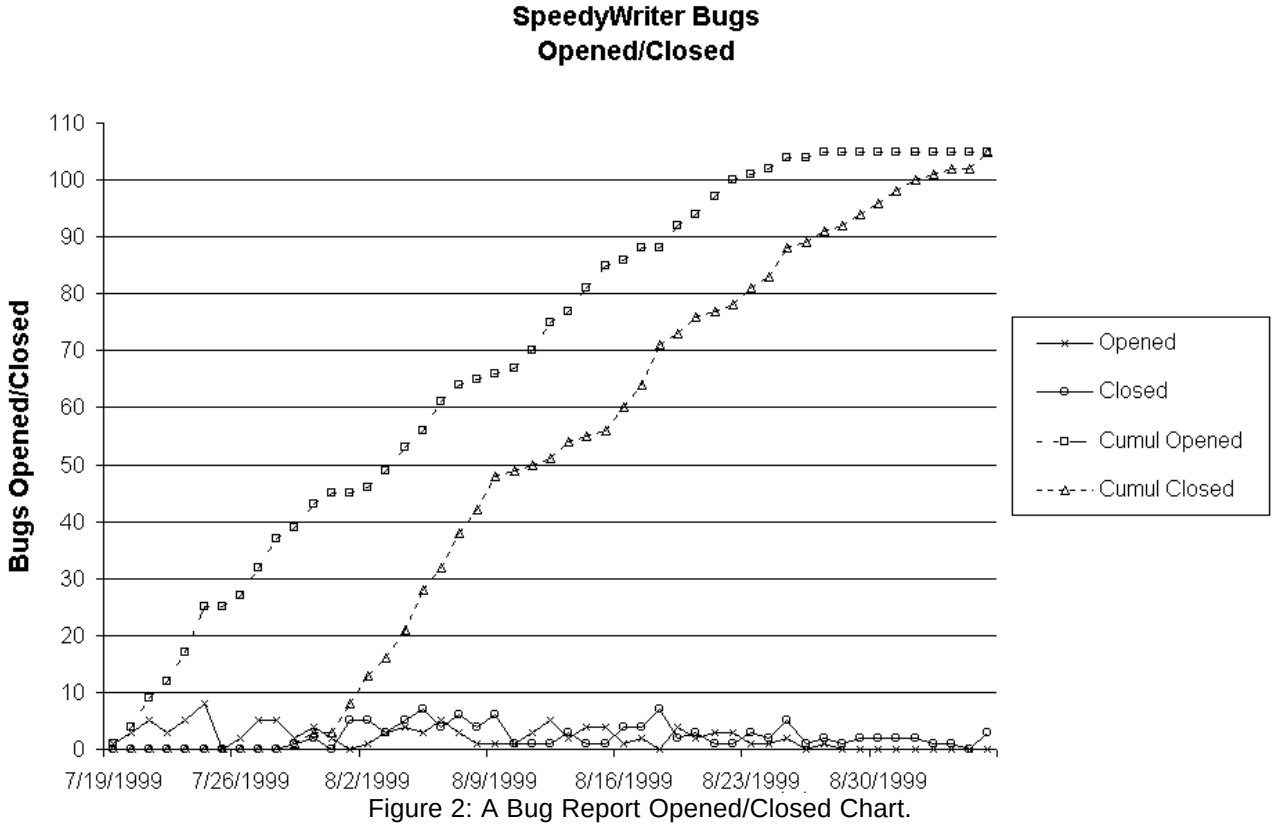


Figure 2: A Bug Report Opened/Closed Chart.

Figure 3 is an example of the test progress chart. The top curve is the plan for test case execution. The lower curves represent test cases completed—including a of passed and failed cases-and those blocked. This chart gives managers some insight into how many test cases remain to be run, what proportion of test cases can't be run, and the relative pass and fail situation. Essentially, this chart addresses the expediency of the test process against the plan over time.

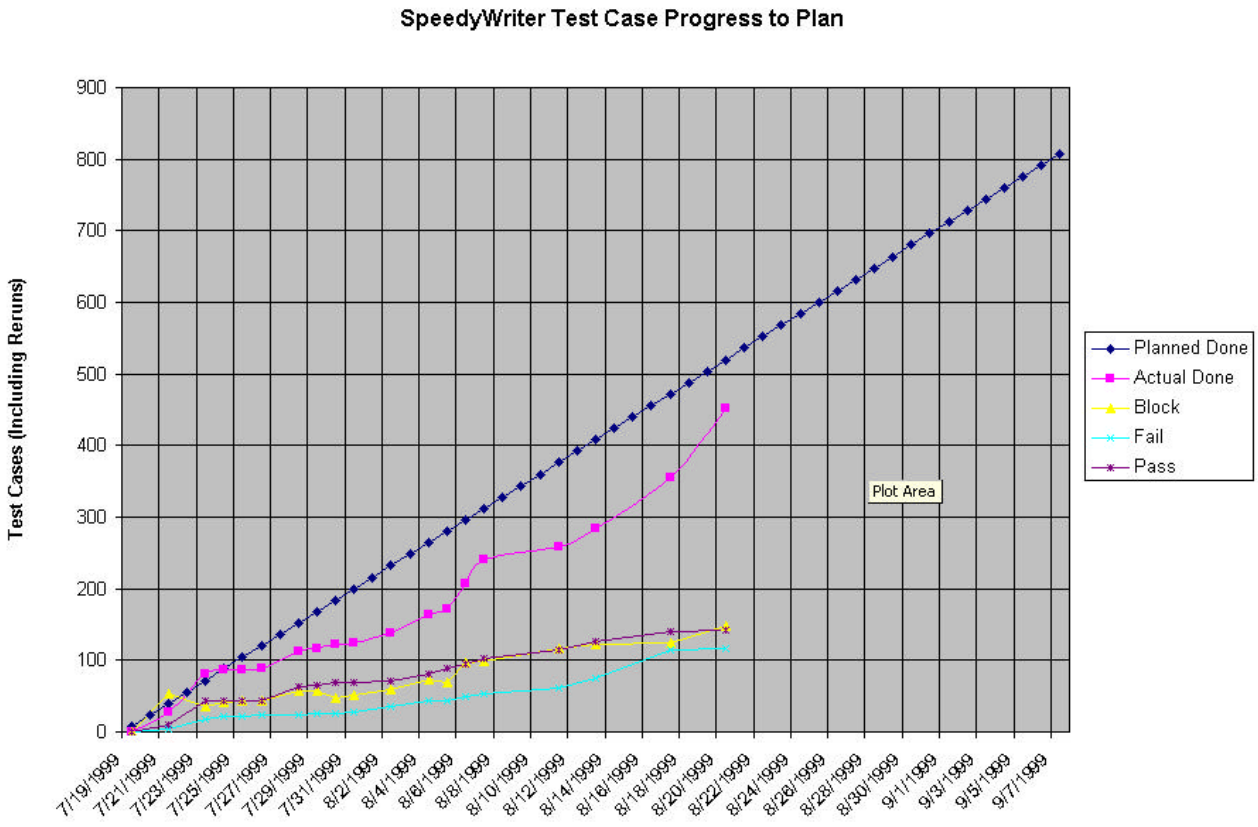
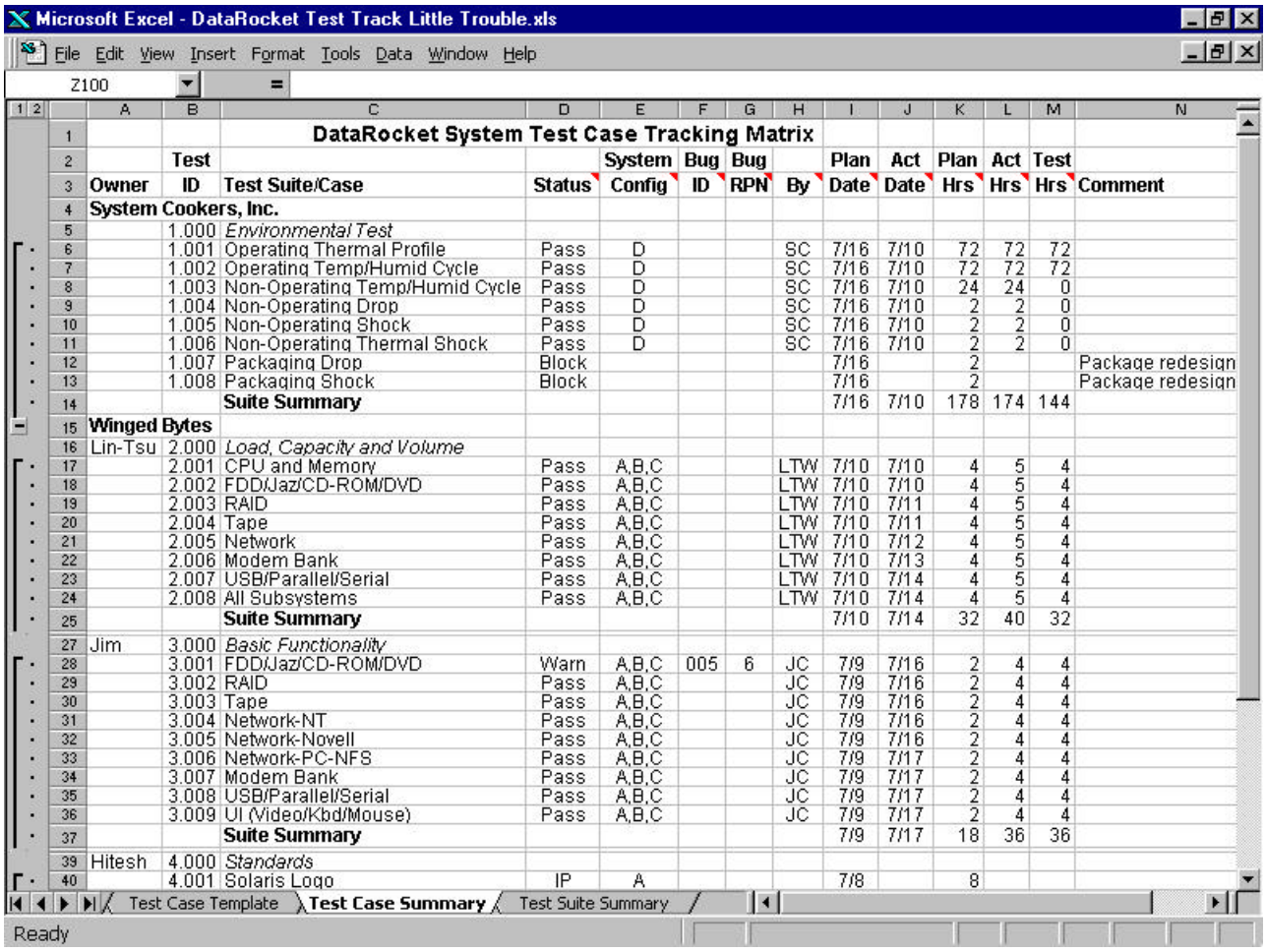


Figure 3: A Test Case Progress to Plan Chart.

To provide more detail on test status, give your management colleagues test case and bug summaries, or even detail reports. A test case summary report will usually list each test case, line by line, with its current status, associated bug reports, test configurations, and so forth. Figure 4 shows an example of this report. A bug summary report will show bugs, one per line, with the bug ID, summary or abstract, date opened, severity, and other such information. In some circumstances, detailed test case and bug report reviews are appropriate, but the typical one-hour status meeting won't allow for that level of granularity.



Microsoft Excel - DataRocket Test Track Little Trouble.xls

File Edit View Insert Format Tools Data Window Help

Z100 =

DataRocket System Test Case Tracking Matrix

	Owner	Test ID	Test Suite/Case	Status	System Config	Bug ID	Bug RPN	By	Plan Date	Act Date	Plan Hrs	Act Hrs	Test Hrs	Comment
4	**System Cookers, Inc.**													
5		1.000	*Environmental Test*											
6		1.001	Operating Thermal Profile	Pass	D			SC	7/16	7/10	72	72	72	
7		1.002	Operating Temp/Humid Cycle	Pass	D			SC	7/16	7/10	72	72	72	
8		1.003	Non-Operating Temp/Humid Cycle	Pass	D			SC	7/16	7/10	24	24	0	
9		1.004	Non-Operating Drop	Pass	D			SC	7/16	7/10	2	2	0	
10		1.005	Non-Operating Shock	Pass	D			SC	7/16	7/10	2	2	0	
11		1.006	Non-Operating Thermal Shock	Pass	D			SC	7/16	7/10	2	2	0	
12		1.007	Packaging Drop	Block					7/16		2			Package redesign
13		1.008	Packaging Shock	Block					7/16		2			Package redesign
14			**Suite Summary**						7/16	7/10	178	174	144	
15	**Winged Bytes**													
16	Lin-Tsu	2.000	*Load, Capacity and Volume*											
17		2.001	CPU and Memory	Pass	A,B,C			LTW	7/10	7/10	4	5	4	
18		2.002	FDD/Jaz/CD-ROM/DVD	Pass	A,B,C			LTW	7/10	7/10	4	5	4	
19		2.003	RAID	Pass	A,B,C			LTW	7/10	7/11	4	5	4	
20		2.004	Tape	Pass	A,B,C			LTW	7/10	7/11	4	5	4	
21		2.005	Network	Pass	A,B,C			LTW	7/10	7/12	4	5	4	
22		2.006	Modem Bank	Pass	A,B,C			LTW	7/10	7/13	4	5	4	
23		2.007	USB/Parallel/Serial	Pass	A,B,C			LTW	7/10	7/14	4	5	4	
24		2.008	All Subsystems	Pass	A,B,C			LTW	7/10	7/14	4	5	4	
25			**Suite Summary**						7/10	7/14	32	40	32	
27	Jim	3.000	*Basic Functionality*											
28		3.001	FDD/Jaz/CD-ROM/DVD	Warn	A,B,C	005	6	JC	7/9	7/16	2	4	4	
29		3.002	RAID	Pass	A,B,C			JC	7/9	7/16	2	4	4	
30		3.003	Tape	Pass	A,B,C			JC	7/9	7/16	2	4	4	
31		3.004	Network-NT	Pass	A,B,C			JC	7/9	7/16	2	4	4	
32		3.005	Network-Novell	Pass	A,B,C			JC	7/9	7/16	2	4	4	
33		3.006	Network-PC-NFS	Pass	A,B,C			JC	7/9	7/17	2	4	4	
34		3.007	Modem Bank	Pass	A,B,C			JC	7/9	7/17	2	4	4	
35		3.008	USB/Parallel/Serial	Pass	A,B,C			JC	7/9	7/17	2	4	4	
36		3.009	UI (Video/Kbd/Mouse)	Pass	A,B,C			JC	7/9	7/17	2	4	4	
37			**Suite Summary**						7/9	7/17	18	36	36	
39	Hitesh	4.000	*Standards*											
40		4.001	Solaris Logo	IP	A				7/8		8			

Test Case Template | **Test Case Summary** | Test Suite Summary

Ready

Figure 4: A Test Case Summary Report.

For a general listing of the kinds of reports appropriate to each audience, see table 1. Examples of the reports and charts listed in that table—and others—can be found in my book, *Managing the Testing Process.* The charts and reports found in my book are applicable to most test organizations, with the right customizations. Stephen Kan's *Metrics and Models in Software Quality Engineering* provides a more thorough and advanced treatment of the subject, but many of his metrics and charts are only available to those test managers working in an organization with a mature software development process in place.

Chart/Report	***Target Audience***
Hardware allocation plan or HW/SW logistics database reports	IT/MIS managers, facilities managers, project managers, executive management
Test coverage analyses	Development managers, project managers, sales and marketing managers, customer and technical support managers, executive management
Defect analyses	Development managers, project managers, sales and marketing managers, customer and technical support managers, executive management
Bug detail report	Developers, development managers, project managers
Bug summary report	Development managers, project managers, sales and marketing managers, customer and technical support managers, executive management
Test case details	Developers, development managers, project managers
Test case summary	Development managers, project managers, sales and marketing managers, customer and technical support

	managers, executive management
Test suite summary	Development managers, project managers, sales and marketing managers, customer and technical support managers, executive management
Test case progression	Development managers, project managers, sales and marketing managers, customer and technical support managers, executive management

Table 1: Test Status Reports and Appropriate Audiences for Them.

All the different reports, metrics, and charts promulgated by tool vendors, standards organizations, consultants, and others can be confusing. Each has its own strengths and weaknesses. They are communication tools, like languages. Some languages are more facile at expressing certain concepts than are others. To use a well-known example, the Inuit language has many different words for "snow," each expressing a nuance that requires adjectives like "sticky" or "powdery" in English. Your management peers and superiors will be most receptive to your status reports when they directly communicate the data points they find interesting, and they'll know these reports when they see them. They probably won't tell you, "Bring me a Raleigh chart if you want my attention"; by trial and error, question and answer, you will discover how best to tell the test status story to them.

In addition to matching your reports to your audience, you must consider frequency. You may want to update and distribute daily those charts that take just a minute or so for experienced managers to scan, such as the test progress and opened/closed charts. You should probably only distribute more detailed reports once or maybe twice a week, such as the bug and test case summaries, which demand thirty minutes or more to review fully. The source data-test case and bug report details-can only get management attention once or twice during a typical project. Imagine management attention as a gas tank: You have a limited amount of it to go the whole project's distance.

Tuning your Message

In the preceding paragraphs, I made some generalizations about reports by role and title, but don't miss the opportunity to tailor your reports to the individual. I have worked with marketing managers who were willing to spend time in the lab, observing test case failures. Some development managers may want to review your test cases to make sure their team's unit testing is complementary.

Regrettably, some managers who find the test situation particularly unpleasant may respond dysfunctionally by trying to bully, drown out, or discredit you. In an environment where one or two managers resort to this childish tactic, while most other managers are receptive to your message, maintain your position, staying reasonable and on-message, calm and consistent. In meetings where these tactics are used on you, focus on trying to communicate clearly to those who are listening. Build your relationships with your receptive management colleagues and don't let your nemeses distract you from the facts.

In *The Art of War*, Sun Tzu cautions the reader that the wise man does not confront the tiger in its den, so likewise the wise general does not engage in battles from a position of weakness. I offer this metaphor not to recommend war-like behavior towards your fellow managers, but to encourage you to get the test organization's natural allies aware of and involved with test case status and bug resolution. Why go it alone? For example, in a shrink-wrap or mass-market setting, groups like marketing, sales, and customer support will be very interested in your test results, while in the IT or IS test manager will find NOC, operations, and finance to be the interested parties. Prepare special status e-mails, have informal face-to-face discussions, and even attend other manager's status meetings with selected reports.

When talking to developers and development managers, it helps if you can relate to their perspective. Some developers may not understand why you run particular tests, or they may want to have a discussion about why certain bugs are happening. While this kind of dialog is usually very helpful, beware of some developers' tendency to trivialize a bug simply because they understand it better. Developers I've worked with sometimes argue that, since they understand why the bug happens, that makes it less of a real problem. "Oh, that's just a natural consequence of the way XYZ functions work, you know." I often have to bring these folks back to earth by explaining how the failure affects the customer, and why the customer won't care about the underlying mechanism of the failure mode. Be ready to relate test cases back to critical

quality risks, either in the realm of usage scenarios or unacceptable business exposure. A reasonable presentation of your position, together with an understanding of the development situation, can keep the relationship positive.

Conclusions

I have outlined some techniques for effectively fulfilling a key test management role—reporting on test status. From a solid test system, you can gather key metrics and data points that allow you to report on objective facts, not opinions or impressions. Your status report, verbal or written, should then present these facts in the context of organizational priorities, usually delivering an acceptably good product with the required features in a timely fashion. Also, when delivering a status report or discussing status with management peers and superiors, be sensitive to the way in which you present bad news. Not only is your manner important, but also is picking the right reports for the right set of people. These reports need not be inflexible; you can customize your communication as appropriate to each one of your colleagues. A practical test manager should master these styles of communicating test status as critical elements of managing upward and outward.

Sidebar: Grounding Test Status in Well-Written Bug Reports

The ideas presented in this article will work only if the underlying test execution you're reporting is professional quality work. One element of the foundation of solid test status information is a thorough suite of tests, intelligently executed. Various methodologies for building such suite have been covered extensively in the test literature. Select and adapt test techniques that accord with your quality risk management needs, supervise the performance of the planned tests with alacrity, and you'll have solid test execution.

Another key element-high quality bug reporting-has received less attention. This is unfortunate, because good bug reports are as important to the actual and perceived quality of the test team's work as is the testing itself. Think about it: If you can't explain a problem to a developer, in terms he can understand, in a way he can use to debug a problem, what are the chances it will get fixed? If you can't put a "bumper-sticker" summary on a bug report that grabs management attention, how will you get them to care about the bugs your team found?

At the heart of a bug report is the failure description. Figures Y and Z show examples of a bad failure description and a good one, respectively. While writing great bug reports is an art form, applying the following eight tips will help your team improve:

1. Structure. A tester who uses a deliberate, careful approach to testing and takes careful notes, tends to have a good idea of what's going on with the system under test. When failures occur, the organized tester knows when the first signs of trouble manifested themselves.
2. Reproduce. Testers should check reproducibility of a failure before writing a bug report. If the failure doesn't recur, the bug report should still be written, but the sporadic nature of the failure should be noted. A good rule of thumb is three attempts to repeat the problem before writing the report.
3. Isolate. After trying to reproduce the failure, the tester should then proceed to isolate the bug. This refers to changing certain variables, such as system configuration, that may alter the symptom of the failure. This information gives developers a head start on debugging.
4. Generalize. After the tester has an isolated and reproducible failure, she should try to generalize the problem. Does the same failure occur in other modules or locations? Are there more severe occurrences of the same fault?
5. Compare. If a tester has previously verified the underlying test condition in the test case that found the bug, the tester should check these prior results to see if the condition passed in earlier runs. If so, then the bug is likely a case of regression, where a once-working feature now fails. Note that test conditions often occur in more than one test case, so this step can involve more than just checking past runs of the same test case.

6. Summarize. The first line of the bug report, the failure summary, is the most critical. The tester should spend some time thinking through how the failure observed will affect the customer. This not only allows the tester to write a bug report that hooks the reader and communicates clearly to management, but also helps with setting bug fix priority.
7. Condense. With a first draft of the bug report written, the tester should reread it, focusing on eliminating extraneous steps or words. Cryptic commentary is not the goal, but the report should not wear out its welcome by droning on endlessly about irrelevant details or steps which one need not perform to repeat the failure.
8. Disambiguate. Either in parallel with or immediately after condensing the verbiage, the tester should go through the report to make sure it is not subject to misinterpretation. The tester must avoid words or phrases that are vague, misleading, or subjective. Her goal should be clear, indisputable statements of fact.
9. Neutralize. As mentioned in the article, being the bearer of bad news presents one with the challenge of gentle delivery. Like overall test summaries, individual bug reports should be fair-minded in their wording. Attacking developers, criticizing the underlying error, attempting humor, or using sarcasm can create ill will with developers and divert attention from the bigger goal, increasing the quality of the product. The cautious tester confines her bug reports to statements of fact.
10. Review. Once the tester feels the bug report is the best one he can write, he should submit it to one or more test peers for a review. His colleagues should make suggestions, ask clarifying questions, and even challenge the assertion that the behavior is buggy if appropriate. The test team should only submit the best possible bug report, given the time constraints appropriate to the priority of the bug.

A bug report should be an accurate, concise, thoroughly-edited, well-conceived, high-quality technical document. These ten techniques can help you achieve that goal.

Summary
Arial, Wingdings, and Symbol fonts corrupt new files.
Steps to Reproduce
1. Started SpeedyWriter editor, then created new file.
2. Typed four lines of text, repeating “The quick fox jumps over the lazy brown dog” each time.
3. Highlighted all four lines of text, then pulled down the font menu, and selected Arial.
4. All text became corrupted into control characters, numbers, and random binary data.
5. Reproduced three out of three tries.

Isolation
Reproduced using Wingdings and Symbol fonts, but did not see the failure with Times-Roman, Courier New, or Webdings
On vague suspicion this was a formatting problem, saved file, closed SpeedyWriter and reopened file. Garbage remained.
Saving file before changing font prevents bug.
Bug does not occur with existing files.
Only happens under Windows 98, not Solaris, Mac, or other Windows flavors.
Regression
This function passed against build I1, so this failure is regression.

Figure 5: A Well-Written Bug Report

Trashed contents of new file that I created by formatting some text in Arial font.

Figure 6: A Poorly-Written Bug Report

Author Biography

Rex Black (Rex_Black@RexBlackConsulting.com) is the President and Principal Consultant of Rex Black Consulting Services, Inc. (http://www.RexBlackConsulting.com), an international software and hardware testing and quality assurance consultancy. His book, *Managing the Testing Process,* was published in June, 1999.