Managing a Software Test Team

by James Bach

Whether you're an individual tester assigned to find bugs for a team of developers, or the manager of a testing department with 75 testers, you have an uphill job. When testing, you can't be sure you will catch all the problems, or even all of the important ones. You can criticize the product, but you can't directly improve it. Your work results in little that's tangible, so people assume there isn't much to it.

If you're new to test leadership, let me assure you that these problems are normal and manageable. There is hope. In this article, I offer a set of principles and guidelines for being a successful test lead. These principles come from my own experience as a test manager and consultant, and from many mentors and colleagues who helped me learn the craft.

The environment of testing

No tester is an island. Beyond the technological issues inherent in testing, and the details of your test strategies, the problems you face in testing depend largely on how your project environment is structured.

In my experience, most test teams operate in projects that provide little support for the needs of a good testing process. There's nothing sinister about that. It's just that few people know much about testers, testing, or the dynamics of software quality. What we don't understand, we generally don't support. Your job as a test team lead is partly to make the test process visible, along with its benefits and consequences, so that the environment will support it better.

There isn't room, here, to detail all of the aspects of the typical project environment for testing, but here are a few points to keep in mind.

- No matter what management says about product quality, they usually don't behave as if it has to be especially excellent. The fact is, very few software products really have to be especially excellent. Software quality is rarely a selling point, it's only that the noticeable lack of quality in your product can be a selling point for your competitors.
- No matter what you do, you'll never have enough staff to do what you'd like to do.
- The requirements and specifications for the product, most of which will never be written down, will be vague, or outdated whenever you do see them.

The mission and tasks of testing

Everybody shares the overall mission of the project-- something like "ship a great product" or "please the customer." But each functional team also needs a more specific mission that contributes to the whole. In well-run projects, the mission of the test team is not merely to perform testing, *but to help minimize the risk of product failure*. Testers look for manifest problems in the product, potential problems, and the absence of problems. They explore, assess, track, and report product quality, so that others in the project can make informed decisions about product development. It's important to recognize that testers are not out to "break the code." We are not out to embarrass or complain, just to inform. We are human meters of product quality.

The heart of testing is examining a product system, evaluating it against a value system, and discovering if the product satisfies those values. To do this completely is impossible. Trust me on this. No testing process, practical or theoretical, can guarantee that it will reveal all the problems in a product.

However, it may be quite feasible to do a *good enough* job of testing, if the goal of testing is not to find all the problems -- only the critical ones; the ones that will otherwise be found in the field and lead to painful consequences for the company. The thing is, it can be difficult to tell the difference between a good job of testing and a poor one. Since many problems that are likely to be found in the field are just plain easy to find, even a sloppy test process can be accidentally sufficient for some projects. And if the product doesn't have many critical problems to begin with, even crippled test process may seem successful. Systematic good enough testing, on the other hand, is more than just a process of finding problems. It also provides a *reasonable confidence* on which to base decisions about the product.

> ***A note about quality assurance.*** *The role of quality assurance is a superset of testing. Its mission is to help minimize the risk of* project *failure. QA people try to understand the dynamics of project failure (which includes product failure as an aspect) and help the team prevent, detect, and correct the problems. Often test teams are called QA teams, sometimes in the enlightened belief that testers should evolve into the broader world of QA, and sometimes just because it sounds cooler.*

If you take the mission of reducing risk seriously, then you need a test process that provides a balanced perspective on the quality of the product, and a reasonable confidence of revealing critical problems that may be lurking there. This requires effective collaboration with other members of the team, as well as users and other stakeholders who may not be on the project team. It must be coordinated in lock step with all of the other activities of the product development lifecycle. Testers themselves must have the right skills and knowledge to design and perform the test process. Finally, you as a test lead have to support all of these tasks.

That makes six major tasks of testing. There isn't room here to expand on all of them, but here's a little more detail on each one.

1. Monitor product quality.

1.1 Quality Planning: explore and evolve a clear idea of the requirements for your product.
1.2 Quality Assessment: compare your observations of the product with the quality standard.
1.3 Quality Tracking: be alert to changes in quality over the course of the project.
1.4 Quality Reporting: keep the team informed about your findings.
Quality monitoring, overall, is the core service of the test team, and supports their mission of helping to minimize the risk of product failure. The remaining testing tasks support this service.

2. Perform an appropriate test process.

2.1 Test Planning: decide what test techniques to use, and how they will be applied.
2.2 Product Analysis: understand what the product is and how it works.
2.3 Product Coverage: decide which parts of the product to test, and assure that they are covered.
2.4 Test Design: design tests that fulfill the test plan.
2.5 Test Execution: execute tests that fulfill the test designs and test plan.
A test process is appropriate if it provides an acceptable level of confidence in the quality assessment. The less sure you need to be about your quality assessment, the less extensive your test process should be. Don't insist on extensive testing for low risk components. But, if you need to ship a high quality product, than you need to work hard at the tasks above.

3. Coordinate testing schedule with all other activities in the project.

Make sure you know what's going on in the project. Many other processes beyond the test team can impact the test process. Make sure you are included in planning and status meetings.
When testing is on the critical path, management is strongly tempted to cut it short. So, don't stand directly in front of the project freight train, if you can help it. Start testing the moment that a barely testable product is available, build up a backlog of bug reports, then try to keep development working up to the last minute, fixing bugs, while insisting on careful change control to help assure that no reckless modifications are made to the code base. If you can safely perform incremental, spot regression testing on changes, rather than multi-week cycles of general testing, then the developers stay on the critical path throughout most of the project.
By the way, if your project has a sloppy release process, all your work in testing could go out the window at the last minute, due to one ill-advised bug fix that introduces a slew of new problems.

4. Collaborate with the project team and project stakeholders.

Since testers are merchants of information, their success lies in large part with the quality of their information pipelines to and from the other functions of product development, as well as to stakeholders who aren't even on the project team.
If you can establish a connection to actual or potential users of the product, they can help you improve your notion of acceptable quality. Developers give more credence to tests that involve realistic situations, so collecting data from users for use in test design is a good idea. Technical support people often hear from users who have concerns about quality, so they are useful to pal around with, too. They can also help you understand what kind of problems are likely to generate costly support calls.
Virtually all products have some kind of online help or user manual. Whoever writes them has to examine the product. When they find problems, they should bring them to you, or at least copy you on them. You can help them understand the product better, and they can help you spot patterns of problems. Also, a user manual is an invaluable tool to use as a test plan. Whenever the technical writers ask you to review a manual or help file, I'd suggest dropping everything else for a week and do nothing but test against the writing. You are guaranteed to learn more about the product and find important bugs to boot.
The marketing team produces fact sheets and ads, and those should be tested. Besides, any claim that goes into an advertisement gives you more leverage to argue for a high quality standard. Marketing people can help you understand which parts of the product they believe are most critical for impressing customers in the field, which helps justify the resources to test them.
Your relationship to management dictates how well you will be allowed to do your job. Do what you can to make management understand and value your test processes, and especially your mission. The key thing with management is to track historical data about customers experiencing critical problems, then show them how a better process might prevent future mishaps. People learn best from their own failures.
Your most important relationship is with the developers. This is the one that makes your job or breaks it.

What Testers need from Developers

Information

- *What the product is and how it works.*
- *How the product is intended to be used.*
- *What parts of the product are at greater risk of failure.*
- *The schedule for remaining development work.*
- *Status and details of any changes or additions to the product.*
- *Status of reported problems.*

Services

- *Provide timely information to the testers.*
- *Respond quickly to reported problems.*
- *Stay synchronized with the project schedule.*
- *Collaborate to set an appropriate standard of quality.*
- *Involve testers in all decisions that may impact them.*
- *Seek to understand the test process.*

What Developers need from Testers

Information

- *Problems found.*
- *What will be tested.*
- *Schedule for testing.*
- *Status of the current test cycle.*
- *What information testers need to work effectively.*

Services

- *Provide timely information to Development.*
- *Seek to understand the general process of creating software.*
- *Seek to understand the product and it's underlying technologies.*
- *Respond quickly to new builds.*
- *Stay synchronized with the schedule, and don't delay the project.*
- *Collaborate to set an appropriate standard of quality.*

5. Study and learn.

5.1 Technology Study: learn about component technologies used in your product.
5.2 Market Study: learn how your users think, and who your competition is.
5.3 Testing Study: strive to know your job well enough to teach it to others.
5.4 Experiential Learning: preserve notes, collect metrics, and review problems found in the field.
The world of technology changes so fast that it takes constant learning to get good at testing it. Good testing is so hard that we need to be constantly looking out for ways to do it better.

6. Lead the test team.

6.1 Resource Estimation: figure out what you need to do the job.
6.2 Resource Acquisition & Allocation: get the resources you need and distribute them wisely.
6.7 Tester Assessment & Improvement: assure that the testers are performing up to par.
6.3 Pressure Management: be a buffer between testers, and the pressure to ship.
6.4 Relationship Management: establish and monitor relationships to other roles on the project.
6.5 Information Management: assure that information is gathered and distributed to the right people.
6.6 Process Management: assure that the test process is sound, synchronized, and visible.
6.7 Quality Management: work with development to maintain a reasonable standard of quality.
Testers who report directly to the development team, without an intervening test manager, are at a disadvantage, because these important leadership activities are shortchanged. A good test manager makes good testing possible by monitoring all the rest of the activities, solving problems with them, and speaking with one voice about them to management.
It's hard to pull a test team up out of accidentally sufficient testing and toward a test process that's more professional. Few people are prepared for this challenge, when it's first thrust upon them. I certainly wasn't prepared when I found

myself in the job. The good news is that you can readily become better at it than most, just by striving to be good enough.

What are developers?

For the purposes of this article, I'm calling anyone who actually creates any aspect of the product a developer; whoever it is who creates quality and is the object of attention by those of us who assess quality.

What is their mission?

Naturally, developers want to ship a great product, or at least a good enough one. But rather than thinking in terms of minimizing the risk of failure, as testers do, developers are committed to ***creating the possibility of success****. Their mission positions them at the vanguard of the project, while testing defends their flank. There's a natural tension between these roles, because achieving one mission can potentially negate the other. Any possibility of success creates some risk of failure. The key is for each role to do its job while being mindful of the other's mission.*

What is their focus?

Developers, by and large, are focused on the details of technology, rather than the more abstract concerns of process or quality dynamics. The reason for this is simple: you can't create software without skill in manipulating technology and tools, and such skill requires a lot of time and effort to maintain. To produce software of consistently high quality, a lot of other kinds of skill is required, but consistently high quality is more often a desire than a need. Most developers believe that their knowledge of technology is the best key to survival and success in their field, and they are right.
Developers are usually not focused on testers and what they do, until the latter stages of the project when development work is more or less defined by the outcome of testing. Testers, on the other hand, are always focused on the work of developers. That mismatch is another source of tension.

What problems crop up between testers and developers?

The differing focus of either role can create a situation where each role looks amateurish to the other. Developers, who know that high quality becomes possible only by mastering the details of technology, worry that testers have too little technology skill. Testers, who know that high quality becomes achievable only by taking a broad point of view, worry that developers are too involved in bits and bytes to care about the users. Each role creates work for the other, and people performing either role often have little understanding of what the other guy needs.

How do you achieve a good relationship with them?

I've found that the one key to relating to developers is to refer to your common mission of shipping a great product. Express the attitude that testing is here to allow development to focus on possibilities without falling prey to risks. You're going to provide information to them about risks. If you can't agree on the risks, then perhaps you can agree to watch closely what happens when the product is released, and learn from that experience.
Another key is to be clear about what each of you needs from the other. Don't assume that your needs should be obvious to them.
Finally, master as much of the technology as you can.
