Critical Software Testing Processes

Rex Black: President and Principal Consultant
RBCS, Inc., Bulverde, TX

Key Words: Testing, critical test processes, test management, software development projects, software maintenance projects, test automation, manual testing.

Audience: Test managers, test leads, project managers, test engineers, development managers, configuration managers.

Introduction

As a test professional with almost two decades in the computer field, I'm pleased to see a new publication devoted to our field, the *Journal of Software Testing Professionals*, and honored to serve as a Contributing Editor. As my first task in this role, I have been asked to write and collaborate with other software testing professionals to produce a series of articles addressing what we consider critical processes that the test professional must master in order for her to succeed. In each of these articles, we'll examine one critical process closely, offering practical advice on what works and caveats on what can cause problems.

Series Scope

Let's start off by defining what this series is—and isn't—about by examining the notion of a "critical software testing process. " A process is a set of tasks and activities, some of which can happen in parallel, and some of which are sequential. The tasks and activities are related by the desired outcomes but do not necessarily have techniques or required skills in common. For example, the test automation process includes creating a business case for the effort, evaluating and selecting tools, and automating the test cases themselves. One forthcoming article will address this process. The individual activities involved in automation, though, we'll leave to other articles in this journal.

As I mentioned above, a critical process is a process you must master to succeed as a test professional. A process becomes critical when:

- You repeat the process frequently. Crisp handling of recurring processes leads to efficient and consistent execution of your day-to-day duties. For example, the bug reporting process occurs constantly during test execution.
- The process involves peers or superiors in the organization. Handling high-visibility processes properly builds a reputation for competence, reliability, and trustworthiness. In the test automation example, we must present a business case to management that convinces them to spend thousands of dollars on a tool, then to dedicate hundreds of person-hours to the automation effort.
- The consequences of process failure are unacceptable. The successful test professional is an expert at quality risk management. For example, to detect potentially embarrassing

product failures, one must analyze the quality risks that can affect the customer and focus the test effort appropriately.

Finally, to keep this series practical and inclusive, I'll focus on processes that most software test professionals will encounter as part of their jobs. Special-interest topics certainly have a place in this journal, but we'll keep these articles centered on common concerns.

I will also use this criterion of broad applicability to determine which critical processes belong in the domain of software testing, and therefore within the scope of this series. Rather than debate the various definitions of "software testing," let's agree that any critical process that confronts many of our peers in the software test profession must be a critical software testing process in some sense. However, let's exclude processes that are generic to all software professionals. Under this definition, I'll include articles about measuring test case coverage and managing releases of software into the test operation, but I'll exclude compositions about maintaining a good relationship with your manager or applying corporate personnel policies uniformly. The latter are good topics—probably critical processes for the test manager—but not critical software testing processes.

Types and Relationships of Critical Processes

I classify software testing processes as internal—involving only people on the test team—or collaborative—involving other one or more teams in the organization. I use the word "collaborative" rather than "external" because teams in effective companies accept each other's work through positive hand-offs; they do not toss incomplete or poor-quality deliverables over organizational walls. A collaborative process might involve accepting deliverables from another team, providing deliverables to another team, or both. A collaborative process might entail a sequence of deliverables spanning multiple teams.

For example, see figure one, which shows a simplified the number of processes (arrows) and teams (circles) involved in the overall software process. In this illustration, running tests is an internal process, as is tracking results. Delivering bug reports is a collaborative process that delivers data to the development team from the test team through the process of running tests. Likewise, sending release notes for a given build of the software is a collaborative process that delivers data to the test team from the development team into the process of running tests. Reporting test status is a collaborative process in which the test manager communicates to the management team the progress of testing and the assessed quality of the product using charts and metrics the management team understands. Finally, note the multi-stage collaborative process, releasing software to the test team, which I have broken down into collaborative subprocesses, shown in parentheses.

Since the test organization is where many deliverables converge and where independent assessments of quality are made, I suggest that collaborative processes are the most critical from the organization's perspective. Also, collaborative processes are harder to execute, due to the coordination, diplomacy, and negotiation that come into play when teams must adopt processes to support each other's needs.

In addition, many processes are interdependent, influencing each other. For example, the quality of the bug reporting process influences the quality of the test status reports delivered to management. The quality of these status reports, in the sense that they add value for the organization and build a perception of testing as a good investment, affects the test manager's ability to secure adequate resources. Such resources are a prerequisite for quality in the test execution processes, including defect reporting. In figure one, the outer sequence of process arrows involves all five teams and shows how test findings against a certain release influence the content of subsequent releases. A number of such feedback loops arise, and we'll take a look at how to manage these.

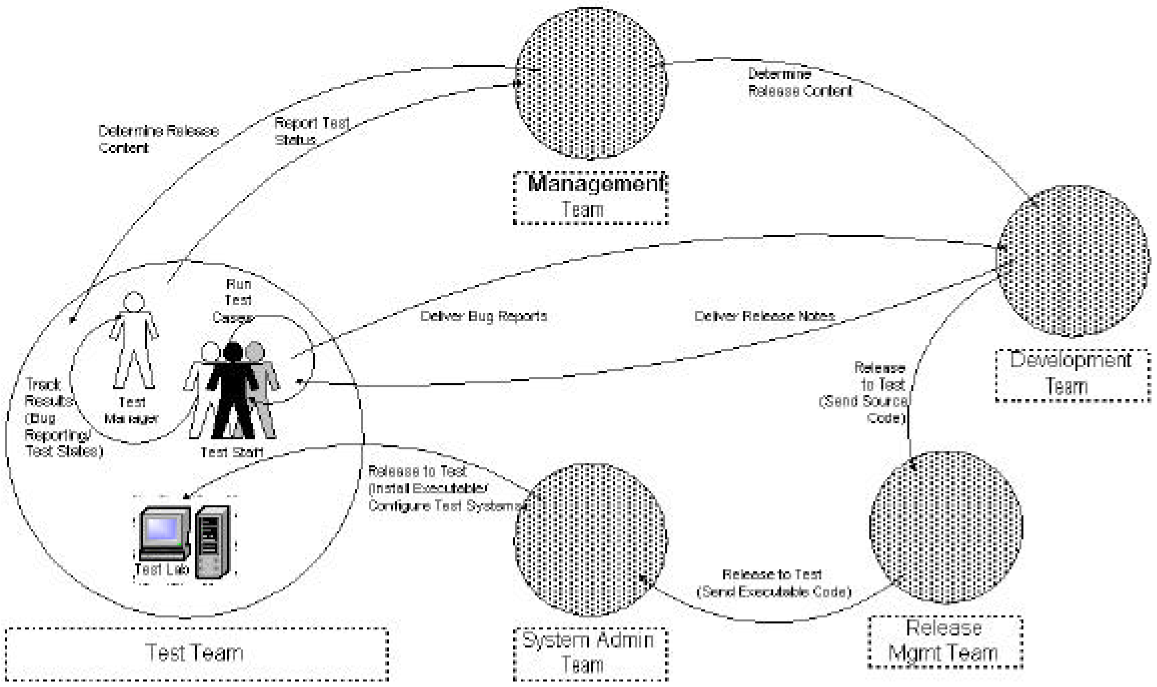
Management Team
Determine Release Content
Determine Release Content
Report Test Status
Run Test Cases
Deliver Bug Reports
Deliver Release Notes
Development Team
Release to Test (Send Source Code)
Track Results (Bug Reporting/ Test States)
Test Manager
Test Staff
Release to Test (Install Executable/ Configure Test Systems)
Test Lab
Release to Test (Send Executable Code)
Test Team
System Admin Team
Release Mgmt Team

Figure 1: Internal and collaborative processes

A Sampling of Critical Processes

As a preview of coming articles, allow me to propose a few critical processes that the next few quarterly editions of this journal will address.

- Planning the test effort. Planning the test effort is not the same as writing down test cases. Testing includes many logistical and organizational factors that the test manager must think through before the first test case is written. Planning also includes gathering facts from many players and obtaining peer and management support for your plan. We will look at various processes and standards that exist for testing planning to help you pick the right approach for your organization.
- Assessing and managing quality risks. Testing is primarily a quality risk management exercise, but sometimes we don't understand the critical quality risks before we start testing. A coming article will discuss ways to align our test systems with the customer's, user's, and other stakeholder's experiences of quality to make sure we spend our precious resources on the right tests.
- Testing software maintenance releases. While development testing has gotten most of the ink, the maintenance release testing process poses as many difficult and high-pressure challenges. An emergency patch for a critical bug requires a different test approach than a regularly scheduled enhancement release, so flexible processes are a

must. We will present a set of case studies, including discussions of the strengths and weaknesses of each method.

- Measuring and improving test coverage. How can we benchmark the extent to which our test system adequately covers the system under test? We typically measure coverage in terms of code, specifications, and requirements, but there are others. An upcoming article will survey the community of test professionals for their techniques.
- Distributing the test effort. With outsourcing, off-shore vendors, and cross-functional IT initiatives becoming a common part of software projects big and small, test organizations have opportunities to leverage the test capabilities of their partners. In addition, test labs, contractors, and consultants provide resources for filling gaps in your own test group. An article will discuss how to coordinate the different test teams involved, including deciding the tests to be run by each and communicating the results found.
- Dynamic test prioritization. Most testers have had to deal with situations where they can't run all their tests in the time allowed. Sometimes too little time is allowed for test development, leading to the need to cover many test conditions with a few test cases. Let's look at ways to develop effective and efficient test cases, and to prioritize them to allow for intelligent quality risk management regardless of the schedule.
- Defining test team roles, phases of involvement, and scope within the software organization. The testing process should not resemble "kiddie soccer": dozens of people following the ball around a large field trying to kick the hell out of it. By using test phases, each team can play the right position in testing the product. By making sure that the test team stays focused on the right areas of testing—and testing alone—the savvy test manager keeps her team effective. A coming article will discuss techniques for focusing the test effort, including those tricky cross-functional and managerial discussions where the test manager must make others commit to performing certain types of tests.
- Requirements and specifications gathering. Have you ever found yourself testing a product in substantial or total ignorance of how it should behave under certain conditions? Educated guessing, quizzing your peers, comparing your product to competitors, and considering the "reasonable customer's" perspective are all part of the process for dealing with these situations. We'll take a look at other ways, too.
- Growing and maintaining the test system. A static set of tests suffers from what Dr. Boris Beizer calls "the pesticide paradox": it will become less effective as developers learn to avoid making mistakes that trigger the tests. We'll address how to institute processes that continuously improve the test suites, adding new tests, eliminating obsolete tests, and updating existing tests as expected conditions change.
- Building a test repository. Test systems consist of test conditions, test cases, test tools, test suites, and other objects. In heterogeneous environments, these objects can span multiple operating systems, programming languages, and even test teams. Can you buy or build systems that will allow you to manage these disparate pieces? Let's look at techniques for building and using home-made tools and commercial software for test management.

- Managing software releases and configurations in the test environment. The software under test does not just magically appear in the test environment, nor does the test environment magically configure itself around the delivered code. A smooth flow of software from the development teams into the test lab, with careful control of the resulting installations, requires tight management and crisp communication, often under rapidly changing conditions. A future article will review the key considerations in these areas along with some case studies of release and configuration management done well and done poorly.

As I mentioned above, you may not have direct responsibility for all these processes, but they have a direct impact on the success of your organization. These cross-functional processes pose challenges to our managerial as well as our technical competence, and therefore I consider them especially interesting. In addition, I have deliberately selected some dowdy and unglamorous processes, because dull doesn't mean unimportant. You may not get applause from your manager for making sure that software releases move smoothly into the test lab, but failing to manage that process can bring attention of the wrong kind.

A Call for Participation

The authors in this series, myself included, will base these essays on case studies, experience on real-world projects, and surveys of the community of test professionals. This series of articles will harness the collective experience of a group of people who have worked on hundreds of projects. Each project teaches the test professional something about how to manage critical test processes. Likewise, others in the test field also have anecdotes and data to share about what works. Because of this hands-on approach, each of these articles will provide a set of action items you can implement to improve your test operation.

As a fellow test professional with your own unique experiences, you have your own set of concerns and ideas. As the customer for this journal, you are the ultimate arbiter of which processes are critical. Following the submission guidelines for the *Journal for Software Testing Professionals*, submit an abstract for an article in this series. If you're struggling with a critical process and looking to learn from your peers, let me know. I look forward to interacting with the readers extensively. My e-mail address is list below.

This series is about how we as test professionals can succeed in our most crucial roles. I invite each of you to join the discussion, especially if you would like to share your own thoughts, suggestions, and caveats on a critical test process.

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. He and his consulting associates help their clients with implementation, knowledge transfer, and staffing for testing and quality assurance projects. His book, *Managing the Testing Process*, was published in June, 1999, by Microsoft Press.