BEST PRACTICE
The Seven Habits of Highly Effective Testing Organizations

The Seven Habits of Highly Effective Testing Organizations
A Catalog of Today's Best Practices

By Lee Copeland

As a consultant, I am often asked to evaluate the testing practices of client organizations. I have found it useful to examine organizations through three different "lenses": the Capability Maturity Model (CMM) developed by the Software Engineering Institute; the Test Process Improvement model (TPI) developed by Martin Pol and his associates; and a catalog of industry "best practices" I've developed. I call them, a la Stephen R. Covey, the Seven Habits of Highly Effective Testing Organizations.

Since both CMM and TPI are well described in other publications, I'll use this space to focus exclusively on the Seven Habits. Each of these traits adds increased understanding, consistency, control, and improvement to your testing process.

1. A Separate Software Quality Assurance Organization

The Software Quality Assurance (SQA) function defines effective and efficient processes and standards for the entire software development process, including, but not limited to, testing. It defines the quality goals, the methods for reaching those goals, and the various roles and responsibilities of an organization-wide quality system. Among those processes are static testing (reviews, inspections, and walkthroughs) and dynamic testing (unit, integration, system, acceptance, regression, etc.). In addition, SQA audits the development and testing products and processes during and after the project and offers suggestions for improvement.

2. A Distinct Software Testing Organization

Software testing is now a recognized technical specialty within the software development process. By designating individuals as testers, we can focus their activities on the quality aspects of software. In a recent survey of attendees at the STAREast testing conference, 90 percent of all organizations questioned have a formal testing group. The placement of this group varies within the organization. The majority of testing groups are included within the development organization. Some testing groups are part of Quality Assurance. In other organizations, the testing group reports elsewhere. The placement of the organization is not as important as the fact that its roles and responsibilities are distinct from other groups and its contribution is valued by the organization.

3. Risk-Based Testing

The major facets of risk-based testing are as follows:

- Risk analysis—Identify the possible risks that a project faces; estimate the loss to the organization if that risk occurs; choose risk mitigation strategies and calculate their cost; compare the cost of the loss to the cost of risk mitigation and choose appropriately.
- Test planning and estimation—Create testing strategies based on the risks that have been identified and choose sets of test cases based on those strategies.
- Test tracking—Track the number of tests that have been defined, implemented, and executed; track the number of tests that have passed or failed; examine the test cases to determine the level of coverage they provide; evaluate the efficiency and effectiveness of the testing process.
- Analysis of testing's contribution—Determine the return on investment that the test group generates by discovering defects before they impact the organization's customers.

4. A Defined and Integrated Testing Process

A formally defined and integrated testing process includes these basic elements:

- Static testing of the work products of development, including requirements, design, code, and test strategies, plans, procedures, and data
- Dynamic testing at the unit, integration, system, acceptance, and regression test levels
- A managed testing environment
- The preservation of test artifacts including test strategies, test cases, test procedures, and test data specifically for reuse on subsequent versions of a single application or sharing between applications

5. Visible Project Planning and Tracking

Effective project planning requires a defined software lifecycle with stages of manageable size. The activities within each project are based on that lifecycle. Each stage should be planned, estimated,

and scheduled before work is done. This formal plan should be visible to all stakeholders in the project. As the development progresses, the actual activities should be compared to the plan. This comparison should also be visible to all concerned. Variations between planned and actual activities should be investigated. Differences may signal a misunderstanding about the size or scope of the project, a lack of skills or resources, or that the plan was faulty in some way.

6. Visible Defect Tracking and Reporting

Effective defect tracking requires the creation and maintenance of a database describing key aspects of each defect. Each recorded defect should include a unique tracking number, a description, the cause, who detected it, how it was detected, the cost of finding it, the estimated cost to the organization if it had been delivered to the customer, and the person assigned to repair it. Allow all stakeholders to examine this data. Look for defect patterns or trends that indicate that the defects are not merely random but are the result of a systemic problem within the development process.

7. Change Management

Change management is an umbrella concept that focuses on three basic areas:

- Requirements management—Requirements are reviewed to determine whether they are feasible and appropriate, clearly stated, consistent with each other and with the system as a whole, and testable.
- Release management—Rather than develop each change to a system individually, a number of changes are collected together into a release. When the release is defined, all the changes are designed, coded, and tested together. When complete, the release is moved into production. Release management establishes a common expectation regarding the length of time required to implement a change. It provides a mechanism to evaluate and integrate changes proposed by multiple sources within the organization, and it spreads the testing overhead over a larger base yielding a more efficient process.
- Configuration management—Defines baselines, controls changes, and reports the status of application software configurations and the hardware they run on. A library of current and past versions is maintained.

Summing Up

Take stock of your organization. Does it have these seven habits? If not, perhaps this will help focus your efforts. In my experience, these seven habits are the hallmark of successful testing.

About the Author

Lee Copeland has more than thirty years' experience in the field of software development and testing. He has worked as a programmer, development director, process improvement leader, and consultant. He has developed and taught a number of training courses focusing on software testing and development issues based on his experience. Contact Lee at lcopeland@sqe.com.

StickyMinds.com Weekly Column From 10/15/01