Software Testing

Áron Erdélyi

2021.09.22.

Contents

1 Test management

1.1 Test independence and organization

Definition: Independent testing means that the testing is done by someone who is not associated with the producer nor the customer.

Benifits of independent testing:

- Avoid author bias
- No positive assumptions regarding the quality
- More effective

Drawbacks of independent testing:

- More communication and management is required
- Political gameplay with negative attitude
- Reports made by non–IT specialists take more efforts to investigate

Independence can be achieved by involving IT and non–IT specialists from other teams/departments.

1.1.1 Levels of independence

Independence is not a boolean value, rather a range:

1. The developer
2. Independent testers attached to the development team
3. Independent permanent test team, center of excellence within the organization
4. Independent testers or teams provided by the operational business unit
5. Specialist testers such as usability testers, security testers, or performance testers
6. Outsourced test team or testers, e.g. contractors or other organizations

1.1.2 Testing by developers

Advantages:

- They know the code well
- They will find the issues what a tester will not
- They can fix issues quickly and cheap

Disadvantages:

- ”Parental feeling” towards their code
- Focus on the ”positive paths”
- Limited end–to–end and real–user perspective
- Limited time to perform testing

Developers can contribute to testing by performing unit tests.

1.1.3 Testing by independent testers

Advantages:

- Testers want to break things
- Testers approach the system from the user's perspective
- Testing expertise can "smell" bugs
- Testing is more objective and consistent

Disadvantages:

- Extra cost
- Over-reliance on testers, insufficient testing by developers
- Testers can be seen as a bottleneck

1.1.4 Developers and testers collaboration

Collaboration among testers and developers is extremely important for success. Miscommunication can further affect the release date of the application.

Example: Collaboration in Scrum:

- Reviewing the acceptance criteria together
- User story estimation
- Developers can review the test scenarios
- Initial testing on the developer's machine with test data provided by testers
- The developer and the tester demo the new feature to the customer

1.1.5 Tester tasks

An IT specialist who creates an executes test cases to ensure quality, design integrity and proper functionality. Tasks of the tester include:

- Analyses and reviews user requirements and specifications
- Reviews and contributes to the test plan
- Creates and reviews test scenarios
- Sets up the test environment
- Collects test data and executes scenarios
- Uses test case management and bug tracking tools
- Automates tests

The test leader/manager plans, controls, administers and regulates the testing. Their tasks include:

- Creates high level testing documents and coordinates with management
- Plans an manages the testing process
- Estimates the resource needs
- Monitors the progress, adjusts the scope of testing
- Creates test report for the business and the management
- Depending on team size performs activities as the regular testers

1.2 Test planning

1.2.1 Test planning

Benefits of the test plan:

- Motivates the testers to learn and understand the system being tested
- It is a communication channel between the developer and tester teams
- Helps to identify the future resource needs
- Clearly defines the responsibilities of the testing team

Planning has to be a continuous activity. The plan should be a live document, and the planning should be performed in all life cycle processes and activities, since as the project goes on more information becomes available (requirement changes, new risks, etc...).

PDCA – management technique The concept of PDCA is based on the scientific method "hypothesis".

- **Plan:** Establish the objectives and processes necessary to deliver results
- **Do:** Implement the plan, execute the process, make the product. Collect data for charting and analysis
- **Check:** Study the actual results vs the plan and look for deviations
- **Act:** If the CHECK shows that the PLAN that was implemented in DO is an improvement then that will be the new plan.

1.2.2 Test planning activities

Common planning activities:

- Defining what and how will be tested
- Determining scope and risks
- Define levels of testing, entry and exit criteria
- Scheduling milestones (test analysis, design, implementation, execution and evaluation)
- Assigning resources and activities
- Defining deliverables by testers (amount, level, structure, template)
- Describe test environment hardware and software needs
- Selecting metrics for monitoring and controlling test preparation and execution
- Specify defect lifecycle
- Communication to the management

Common pitfalls:

- Test plans are often ignored once they are written
- Test case descriptions are often used as test plans. No test cases in a test plan!
- The schedule of a testing is often inadequate for the amount of testing that should be performed
- Testing is often postponed until too late in the development process
- Test planning is ignored in Agile

1.2.3 Risks

Definition: Risk is a factor that could result in future negative consequences; usually expressed as impact and likelihood.

The level of risk is calculated by multiplying the probability of the risk occurring by the impact is it did happen.

Risks that affects the project's capability to deliver it's objectives:

- Testing schedule is too tight
- Unexpected project scope expansion
- Not enough resources
- Illness or other loss of personnel
- Non–availability of independent test environment and accessibility
- Missing or inadequate test data
- Dependency on 3rd party teams and components

Product risk is the possibility that the system or software might fail to satisfy or fulfill some reasonable expectation of the customer, user, or stakeholder.

Typical options for risk control include:

- **Mitigate:** We take preventive measures to reduce the likelihood or the impact
- **Contingency:** We have a plan or perhaps multiple plans to reduce the impact
- **Transfer:** We get another party to handle or accept the consequences of a risk should it occur
- **Ignore:** Ignoring or accepting the risk and it's consequences should it occur.

Example: "Development is slower than expected and cannot deliver the code on time."

- **Mitigation (prevent):** Testers support devs by creating mock services, tell scenario ideas from previous tests, help in requirement analysis
- **Contingency (handle):** Reduce the number of tests to be executed (priority, defined in advance) and ask help from other testing team (more testers) or enable overtime work.
- **Transfer:** Ask the project manager to have an extended deadline. Ask DEV management to prioritise features developed to deliver early testable parts.
- **Ignore:** We note the risk and discuss with the stakeholders that in such case the product quality will be compromised.

1.2.4 Entry criteria

Define when to start a testing related activity, measure whether the system is ready for a particular test level. This is a gate between the stages of development to reduce the wasting of time. This criteria can include

- Code ready
- Documentation ready
- Previous level tests are passed
- Test environment and data available

1.2.5 Exit criteria

Defines when to stop testing, measure whether the test level has achieved its objectives. THis criteria may include:

- All tests run
- Schedule
- Defect density or reliability measures
- Coverage of code, functionality or risk is adequate
- Out of time/money

The criteria for successful completion are set at the beginning of the project before the pressures associated with delivery are felt. These criteria are based on a consensus of all the stakeholders. Decisions are visible and simple at the transitions from one test level to another and at delivery.

1.3 Test progress monitoring and control

1.3.1 Test progress monitoring

Measuring the progress of the process, providing feedback and visibility. Tells us where we are and how much to do:

- Gives a snapshot on the quality of the system under test
- Useful for testers, developers and the management team
- Gathers information for the future test plans

Common test metrics

- Percentage of work done
- Defect statistics
- Test coverage
- Dates of test milestones
- Subjective confidence of testers in the product

1.3.2 Test reporting

Summarizes the information about test activities:

- What happened during period of testing
- Give information and metrics to support recommendations and decisions about the future
- Final test report, regular status reports

Useful info to provide:

- General quality
- Level of confidence
- Remaining bugs and risks

1.3.3 Test control

Test control is used when the reality differs from the plans. This may include

- Reprioritizing tests
- Changing the test schedule
- Reviewing of product risks
- Adjusting the scope of the testing
- Setting entry criterion requiring fixes to be retested by a developer before accepting them into a build
- Reduce functionality
- Postpone the release
- Test after the release

1.4 Incident management

Definition: Incident is any unplanned event occurring that requires further investigation, anything where the actual result is different to the expected result. This can be raised at any time throughout the software development lifecycle.

An Incident is not necessary a defect. It is decided during the incident analysis (aka Triage). Root causes of non-bug incidents:

- User error
- Badly configured test environment
- Corrupted test data
- Wrong requirements
- Unrealistic scenarios
- Usability issues

1.4.1 How to report

A good incident report

- Provides developers, testers, business analysts and other parties with feedback about the problem to enable reproduction, isolation and correction
- Provide test leaders a means of tracking the quality of the system under test and the progress of testing
- Gives ideas for test process improvement

1.4.2 Priority vs Severity

Definition: Priority defines the order on which we should resolve a defect (Low < Medium < High).

Definition: Severity defines the impact that a given defect has on the system (Cosmetic < Minor < Moderate < Major < Critical).

1.5 Configuration management

The purpose of configuration management is to establish and maintain the integrity of the products during the project/software life cycle. CM helps to uniquely identify the

- Tested items
- Tested documents
- Test cases and scripts
- Test environments

2 Tool support for testing

Test tools can be used to support one or more testing activity. This includes tools that help directly in testing and test data preparation tools.

2.1 Types of test tools

- Improve the efficiency of test activities by automating repetitive tasks
- Improve the efficiency of test activities by supporting manual test activities throughout the test processes
- Automate activities that cannot be executed manually
- Increase reliability of testing

2.2 ISTQB classification

- Management of testing and tests
  - Test management tool
  - Requirement management tool
  - Incident management tool
  - Configuration management tool
- Static testing
  - Review process support tool
  - Static analysis tool
  - Modelling tool
- Test specification
  - Test design tool
  - Test data preparation tool
- Test execution and logging
  - Test execution tool
  - Test harness, unit test framework tool
  - Test comparator
  - Coverage measurement tool
  - Security testing tool
- Performance and monitoring
  - Dynamic analysis tool
  - Performance testing, load testing, stress testing tool
  - Monitoring tool

2.3 Benefits and risks

Benefits:

- Reduction of repetitive work
- Greater consistency and repeatability
- Objective assessment
- Ease of access to information about tests or testing

Risks:

- Unrealistic expectations for the tool
- Over–reliance on the tool