INFORMATION SYSTEMS IN MANAGEMENT

Information Systems in Management (2018) Vol. 7 (1) 47−58
DOI: 10.22630/ISIM.2018.7.1.5

SOFTWARE TEST MANAGEMENT APPROACH FOR AGILE ENVIRONMENTS

MICHAŁ PAWLAK, ANETA PONISZEWSKA-MARAŃDA

Institute of Information Technology, Lodz University of Technology

Software testing is a very broad term that includes a wide variety of topics. They range from technical like testing techniques and measurements, to more organizational like planning and management of testing. Ability to plan, design and create efficient tests is the most critical ability for any good tester. The paper presents *Kungfu Testing*, which is a testing approach based on advice and best practices advocated by experts in the field of testing. The method is intended to provide a step-by-step instruction of managing testing activities in a project environment. The presented approach was designed to work with and complement the agile development methodologies due to their widespread use and popularity.

Keywords: management of IT projects, software testing, management of testing process

1. Introduction

It is widely believed that a testing process includes only performing tests, which involve executing software and evaluating results. In reality, the process of testing is more complex and consists of different activities. According to International Software Quality Board (ISTQB) [1, 3, 4], the process includes: planning, control, defining test conditions, designing, choosing test cases, executing test cases, evaluating results, evaluating exit criteria, reporting the process and the system under test, closing the testing phase.

In addition, reviewing documents (including source code) and static analysis is included in the testing process. ISTQB [1, 2] also defines the following objectives of testing:

- finding defects,
- gaining confidence about the level of quality,
- providing information for decision-making,
- preventing defects.

The first point is the most basic and obvious objective of testing. Finding defects and fixing them is the easiest way of increasing software quality. The second point refers to an evaluation of whether the software works as expected and meets the requirements. The third point considers data that can be gathered during the testing process. A number of found defects, a percentage of requirements covered by tests or current quality level can all be taken into account during a decision-making process related to the software. The last very important point advocates that proper planning, designing of tests, ensuring testability of the produced code, a thorough review of requirements and inclusion of quality attributes into them from the start, can prevent the defects and increase the overall quality of the produced software.

The presented paper analyses the issue of *Kungfu Testing*, which is a testing approach based on advice and best practices advocated by experts in the field of testing. It provides a step-by-step approach to managing testing activities in a project environment.

The paper is composed as follows: section 2 gives the outline of testing process in software development project. Section 3 deals with main aspects of software testing management while section 4 presents the proposed approach for testing management process in IT project, named *Kungfu Testing*.

2. Process of testing in software development projects

Software testing is a very broad term that includes a wide variety of topics. They range from technical like testing techniques and measurements, to more organizational like planning and management of testing [2, 5, 6]. Software testing can be divided into six topics, each describing a different area of knowledge related to testing. The first topic, *Software Testing Fundamentals*, covers concepts, terminology and rules of testing. It also describes relationships between testing and other processes in software design. *Test Levels* refers to testing activities conducted on different levels of abstraction and their different objects and purposes. The topic *Test Techniques* describes a classification of basic test design techniques used for planning and generating tests. *Test-Related Measures* cover methods and measurements for monitoring and controlling the progress of testing and the product quality. *Test Process* refers to the testing process understood as a combination of

concepts, techniques and measurements in a single controlled process. The topic covers test planning, evaluation and organization. Finally, *Software Testing Tools* describes tools that can be used for automation of some of the testing activities. Detailed descriptions of the testing areas that can be supported with tools are also included in this topic.

Software testing can be classified by different levels, mostly related to development phases [1, 2, 3]. The phases are differentiated by a purpose and object of the testing. There can be many levels of testing. Unit tests verify the smallest independent and testable parts of the source code, which are referred to as units. Integration testing is performed to verify the interactions between software components and systems. System testing validates the integrated system against the initial project goals. This type of testing is based on functional and non-functional specifications. Acceptance testing is designed to verify the final product. Acceptance testing is typically done by a customer and its goal is to assess whether the software satisfies acceptance criteria. There are many other classifications of tests and types of tests, but it is not the purpose of this paper to present them.

3. Management of software testing

Software testing is a very complex and difficult process, but it is also important to remember that is not independent. It is limited and conditioned by a development process and its methodologies. A good example of it would be testing in agile methodologies. Developers perform a large part of testing so there is a risk of testers repeating tests if communication between both groups is lacking. A frequently changing specification might prevent automation of testing. These are just a few examples of how the development process may influence testing. However, the relationship between these two processes is two-sided. Testing can force changes in a project when some critical defect is identified, or can delay a release if quality standards are not met [4, 6, 8, 10].

From the point of view of management, software testing can be divided into two main phases, each split into stages (Fig. 1).

During the *planning phase,* test manager must obtain the information what to test, how long the testing will take, what strategy of testing should be utilized and finally who will conduct the testing and how. The required information is acquired during risk analysis, test estimation, test planning and test organization stages respectively. It is important to note that in this case, the risk analysis means any method of analysis that helps to determine the important areas to test. It may consist of requirements-based testing, testing based on operational profile or any method presented further in this section.

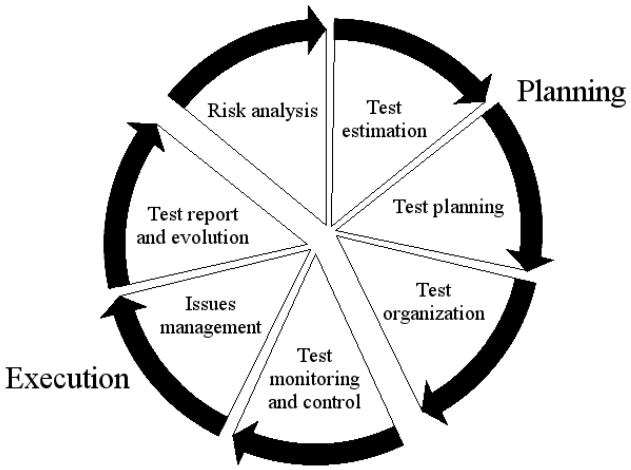

Figure 1. Software test management phases [9, 11]

The *execution phase,* on the other hand, is concerned with obtaining the information about the performance of the testing process itself, whether the testing proceeds according to the schedule, dealing with expected and unexpected issues and evaluation and improvement of the testing process. These activities are conducted in test monitoring and control, issue management and test report and evaluation stages respectively.

Each stage has its own set of methodologies, which help to facilitate the activities required to complete its goals. Managing software testing is not an easy task. Managers must constantly deal with many different types of challenges. Some may be caused by people and some by the project itself. In general, the challenges the managers face can be connected with: project limitations and conditions, stakeholders, software production cycle, testing and human factor.

4. *Kungfu Testing* approach to software testing management

Kungfu (Chinese: 功夫, Pinyin: gōngfu) in its original meaning refers to any skill achieved through a hard work and practice. Therefore, *Kungfu Testing* can be described as a testing skill obtained from training and experience. *Kungfu Testin*g is an approach of testing based on advice and best practices advocated by experts in the field of testing. The method is intended to provide a step-by-step instruction of managing testing activities in a project environment. Given the popularity of agile development methodologies, the presented approach is mainly intended to be used in the agile environments. The following subsections present the description of *Kungfu Testin*g process.

4.1. Example application – *ErasmusBooking*

ErasmusBooking is an exemplary online service intended to provide all mobility students of Lodz University of Technology (LUT) with information about partner institutions and ability to book a place in any of them.

The functionalities implemented in *ErasmusBooking* are made available via dedicated webpage similar to the hotel-booking website *Booking.com*. *ErasmusBooking* offers easy to find and use search interface. The list of search results displays the most important information about each found institution, namely: localization of each of the institutions, ranking of the institutions, availability and a short description of the offer. However, the key functionality available from this sub-page is booking of a place in the institution. The booking is made with the student account, so the information about bookings is available to LUT's staff.

4.2. *Kungfu Testing* planning phase

Every project is different in terms of goals, available resources and targeted market. Software development is no different and it is possible to divide them into two broad categories: *contract-driven* and *market-seeking*.

In the first case, the company's main goal is to fulfil the contract. Typically, in such projects, there is some form of a requirement list at the beginning of the project. It is bound to change sooner or later, which must be kept in mind by test manager. In the second type, there is no single client and no single specification. The company is searching for clients by trying to provide them with the best service. In this case, testing may be performed against expectations of many possible customers.

In the both cases, the test manager should take an active part in the requirement specification. The main goal is to ensure that testability features will be included in the project's schedule and budget from the very beginning. This will increase the testing team efficiency and effectiveness [12, 13, 14]. Next important activity of the test manager at this stage should be inquiring about possible non-functional requirements, like stability, security etc. They are often overlooked, both by clients and project managers, which results in lack of time and resources for their later inclusion [11, 12, 15].

In case of the *ErasmusBooking* exemplary system, a list of functional requirements is provided along with a description of the system purpose. No detailed non-functional specification is provided. However, *ErasmusBooking* is emulating Booking.com, so it can serve as a reference point for assessing the non-functional requirements. Standards and regulations can be found on the website of World Wide Web Consortium (W3C). Additionally, because students are the target users of *ErasmusBooking*, they can be interviewed to find what they expect from such a system.

4.2.1. *Kungfu Testing* analysis phase

Assuming some specifications of the product have been collected, it is now important to start making test plans. The purpose of this stage is to determine objectives and scope of testing. In order to do that, the test manager has to analyse the product and identify areas that can be tested in the given project limitations. *Kungfu Testing* advocates using risk analysis as a way of detecting these areas. There are many different methods of risk analysis, but in this approach, *Product Risk Management (PRisMA)* is a method of choice [13].

It is argued that it is not testers' responsibility to make a decision whether the product is ready or not. This role belongs to project managers and other stakeholders. Test teams are just information providers for the decision-makers.

The analysis stage can be facilitated using *Product Risk Management* method (Fig. 2). The main idea behind this method is a creation of a product-risk matrix. The matrix contains numeric values of risks of each identified item. Depending on these values, a different testing approach should be employed.

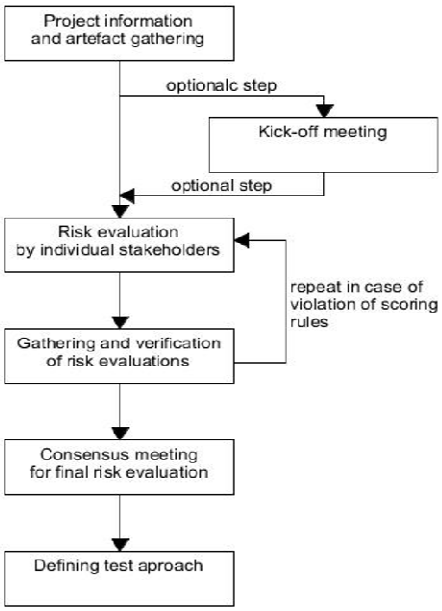

Figure 2. *PRisMA* process diagram of risk evaluation [13]

Project information and artefact gathering is the first phase of this method and consists of the following steps:

- gathering input documents – determining and collecting data relevant to product risk analysis, performing static analysis of the obtained documents,

- identifying risk items – listed with own identifier and description; no more than 30-35 risk items in order to keep the process functioning correctly,
- determining impact and analysis factors – described in a high-level document for the whole organization,
- (optional) defining weights for each factor – the scale from 1 to 5, also called *Likert scale*, is recommended,
- selecting stakeholders – including different roles from business and production, e.g. project manager, developers, system architect and end user,
- defining scoring rules.

Risk evaluation by individual stakeholders phase consists of the stakeholders scoring the risk items according to the presented rules. These scores are based on personal stakeholder assumptions which should be also documented. *Gathering and verification of risk evaluations* phase consists of collecting the results of evaluation by stakeholders and checking its correctness. Test manager proceeds to average scores of every risk item so that individual scores can be placed in the product-risk matrix.

Consensus meeting for final risk evaluation phase can be started after a completion of the product-risk matrix. During it, a common understanding of perceived product risks should be achieved and may result in modifications of the requirement list, use cases or user stories.

Define a differentiated test approach – basing on the risk position in the matrix, the test objects can be identified and prioritized. This defines test scope of the projects, from which necessary test design techniques can be derived.

Taking into consideration the *ErasmusBooking* example, there are a few sources of information that can be used for examination and evaluation of it. Having collected the necessary sources, the risk analysis stage can be started. Firstly, five main risk items are identified for *ErasmusBooking* service: lack of communication between components, incorrectly displayed data, illegible interface, business logic defects, slow response times.

Next step is to select factors that affect each risk. With risk items and factors identified, a grading scale must be selected. There is no high-level document describing the evaluation process so it is assumed that any scale can be chosen. Following the guidelines from *Kungfu Testing* the *Likert scale,* from 1 to 5, is selected with no additional constraints for scoring.

4.2.2. *Kungfu Testing* test estimation phase

The next activity, after the analysis stage, is to estimate time, resources, cost and skill required for completing the test activities (Fig. 3).

Estimation can be done by combining different approaches to the task estimation – the test should be divided into smallest possible tasks, each should be de-

scribed in detail. Next step is assigning the tasks to specific team members, who then proceed to estimate the best, worst and most likely estimates for their tasks, in order to conduct the three-point estimation.

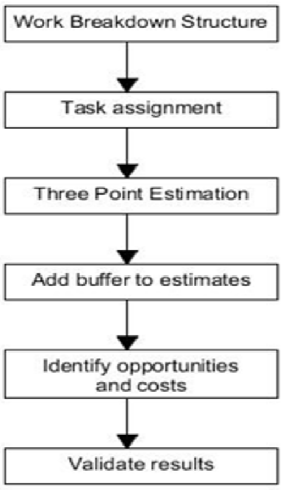

Figure 3. *Kungfu Testing* test estimation process

When the group estimation is completed, the test manager should analyse the estimates and consider additional factors that may be involved.

Identifying opportunities and costs refer to two good practices. It refers to a preparation of a cost model of testing in which a simple simulation of costs can be performed. This will allow the test manager to answer inevitable questions like "what would happen if some tests were to be skipped" or "what would be gained through performing this type of testing" [3]. Having model like that puts the test manager in a better position when validating, negotiating resources and schedule with management, which is also the last step of the process.

4.2.3. *Kungfu Testing* test planning phase

The objective of this stage is to create a test plan document. It serves as a frame of reference for testing activities, which will be monitored by the test manager. What is important during its creation is to have a clear purpose of it in mind. A test plan can be a useful tool for managing tests but it can also be a product itself. Often, detailed test plans are required for successful software delivery, to satisfy some standards or norms or may be objects of a business transaction on their own.

There is no single proper template for test plans. The best plans are those, which fulfil their role in the project. That is why the most important thing is to

understand each of the fields within it and to properly utilize them. Creating detailed documentation for its own sake is entirely pointless waste of resources and time. However, *Kungfu Testing* proposes the following fields to be included in plans: business background, requirements, risks, test objectives, test scope, test design approaches, task estimations, test team, test resources, suspension and resumption criterion, metrics and methods of measurement, test deliverables, norms, standards and constraints.

4.2.4. *Kungfu Testing* test organization phase

In general, a *test team* consists of: *test manager*, who directs and organizes work on the project; *testers*, who create and execute tests and *test administrator*, who manages and maintains a test environment. In addition, developers who create tests as part of TDD can also be considered to be part of the testing team. Their contribution is crucial in agile development, where unit and integration tests provide continuous feedback on the stability of the system under tests.

There are many techniques and methods of team management and teamwork. They depend on managers, organization and team members.

A good practice for testing is rotating the testers across features. This practice brings many advantages. Firstly, it keeps the staff from getting bored from constantly doing the same thing. Secondly, it keeps the testers form becoming overly specialized in a single area of the project, which decreases the value of such testers. In addition, it prevents knowledge gaps in case such specialists leave. Thirdly, different testers have different strengths, which should be taken into consideration during task assignment. Finally, a good practice is to work in pairs. Testers are encouraged to share, explain and react to ideas which bring them to better focus and trigger more ideas. It helps testers concentrate at the task at hand and makes the tasks of bug replication and analysis easier.

Testing of *ErasmusBooking* is organised in accordance with the *Kungfu Testing* recommendations and agile development principles. The testing is conducted in close cooperation with developers who are responsible for low-level tests, by working in the same place and in pairs. Testers provide a different point of view, combining business perspective and developer perspective on the project.

4.3. *Kungfu Testing* execution phase

The test execution phase is concerned with fulfilling the test plan and ensuring that the testing provides the best possible service to stakeholders. This subchapter is focusing on providing methods and guidelines for test execution, reporting, evaluation and improvement of testing and development processes.

Test execution phase in *ErasmusBooking* example is conducted in accordance with project development approach conducted in agile. The tests are conducted during two-week sprints. Testing is highly automated and performed in parallel to

development, so a continuous feedback can be provided. This enables quick fixes and constant improvement of *ErasmusBooking* service.

4.3.1. *Kungfu Testing* test monitoring and control phase

Monitoring and control of testing processes are the most important tasks of test managers at this stage. There are four key parameters that should be monitored during this phase: cost, schedules, resources and quality. They can be monitored by collecting various measurements and metrics. Cost, schedules and resources are relatively easy to follow – test managers must only regularly check the state of their budget and verify whether work is delivered on time and resources are in required quantity. Measuring quality of work and performance of staff is a different and difficult problem.

The test managers should consider reading the artefacts created during testing like: bug reports, code, and documentation. Having done so, test managers can evaluate individual strengths and weaknesses of their staff. It is possible to create a scale for various skills and assesses them for individual team members.

ErasmusBooking uses several measurements, which counterbalance others' side effects to report progress. They measure progress by answering the following questions:

- How much of *ErasmusBooking* has been tested?
- How much of the planned testing has been done?
- How many problems have been found, and how many are critical?
- How many defects were overlooked by the testing team?
- How much of the testing is being blocked by unfixed defects?

4.3.2. *Kungfu Testing* issue management

The test teams are not responsible for controlling and ensuring the quality of the development process. However, it does not mean reporting such issues should not be performed. After all, there may be problems with the testing process itself and it is not always in test managers' power to fix them.

There are three steps of issue management. The first step is to record every encountered issue – it should have assigned priority, tracking status and owner responsible for it. Next step is to report documented issues to a high-level manager. The final step is control and it is a responsibility of a project manager to monitor the reported issues and possibly find solutions for them.

4.3.3. *Kungfu Testing* report and evaluation stage

Reports are the primary product of most testers. They shape the perception of testers in eyes of their receivers. It is test managers' job to ensure that reports generated by their staffs are [10]: actually written (not just reported verbally), num-

bered (for identification purposes), simple (they describe only one problem at a time), understandable (makes the bug more likely to be fixed), reproducible and legible.

There is no certain and dependable formula for deciding when the testing is done. The best way to assess this, is to use experience, skill and judgement of test teams. However, because testing is an information gathering process, theoretically it can be stopped when enough information has been gathered. Enough, in this case, means that based on the collected information, it may be reasonably assumed that there are no important or critical defects or undiscovered problems in the final product.

ErasmusBooking advocates close cooperation between testers and developers. For that reason, many defects are fixed instantly after a conversation between the tester and the programmer. Defects that are more severe are reported using *Trello* platform [15]. A standard report contains a build version, defect severity, defect description, steps required for its reproduction and expected results.

5. Conclusion

Software defects and lack of quality may result in losses, not only in money but in human life as well. In order to reduce these risks, software testing must be conducted. Correctly performed, testing does not only mitigate the mentioned dangers but also builds confidence in both software and developers, which in turn brings generates profits. Different types of tests accomplish different goals: they ensure delivery of correct functionalities; increase stability, security and reliability of software, etc. Each of these goals represents an increase in quality of software, which is a result of software testing.

Kungfu Testing provides test managers with lightweight methods for conducting risk analysis and creating test plans. This forms a frame of reference for testers, who often lack such item especially in the agile methodologies, which focus on communication instead of documentation. This philosophy often results in no documentation at all. The problem is solved by providing methods of creation of such documents based on communication and close cooperation between stakeholders. Each element proposed by *Kungfu Testing* (e.g. risk analysis, test plan and test estimates) can evolve with the whole project, according to the current needs and requirements.

Kungfu Testing's advice and guidelines for various issues are based on already existing techniques and methods employed by various experts in the field. This assures some level of reliability of them and thus the whole approach. It provides a ready to use set of steps and guidelines for starting and completing testing projects.

REFERENCES

[1] International Software Testing Qualifications Board, *Certified Tester – Fundation Level Syllabus*, International Software Testing Qualifications Board, 2011.

[2] IEEE Computer Society, Guide to the Software Engineering Body of Knowledge Version 3.0, IEEE, 2014.

[3] A. Roman, *Testing and Software quality, Models, techniques, tools*, PWN 2015.

[4] L. Crispin and J. Gregory, *Agile Testing. A Practical Guide for Testers and Agile Teams*, Pearson Education, Inc., Boston, 2009.

[5] International Software Test Institute, *Software Testing Levels*, http://www.test-institute.org/Software_Testing_Levels.php, 2017.

[6] K. Zimitrowicz, *Quality of IT projects, Software development and testing*, Helion, Gliwice, 2015.

[7] International Software Testing Qualifications Board, *Standard Glossary of Terms Used in Software Testing*, version 3.1, 2015.

[8] P. Dissanayake, *How to draw a Control flow graph & Cyclometric complexity for a given procedure*, DZone/Performance Zone, 2014.

[9] IEEE Computer Society, *IEEE Standard for Software Reviews and Audits*, Software & Systems Engineering Standards Committee, New York, 2008.

[10] C. Kaner, J. Falk and H. Q. Nguyen, *Testing Computer Software*, John Wiley & Sons, Inc., 1999.

[11] IEEE Computer Society, *IEEE Standard for Software and System Test Documentation*, Software & Systems Engineering Standards Committee, New York, 2008.

[12] C. Kaner, J. Bach and B. Pettichord, *Lessons Learned in Software Testing. A Context-Driven Approach*., New York: John Wiley & Sons, Inc., 2002.

[13] E. van Veenedaal, *Practical Risk-Based Testing. Product Risk Management: The PRISMA Method*, UTN Publishers, Manchester, 2011.

[14] J. M. Bryson, *What To Do When Stakeholders Matter: A Guide to Stakeholder Identification and Analysis Techniques*, University of Minnesota, Mineapolis, 2004.

[15] R. Black, *Practical Tools and Techniques for Managing Software and Hardware Testing*, Indianapolis: Wiley Publishing, Inc., 2009.