FRI-2G.303-1-CST-03

ANALYSIS OF SOFTWARE TESTING TECHINQUES AND RESULTS MEASUREMENT METRICS

Tsvetelina Mladenova, MEng
Department of Computer Systems and Technologies,
"Angel Kanchev" University of Ruse
Tel.: +359 884 292 155
E-mail: tsmladenova@uni-ruse.bg

Abstract: *Software Quality Testing has always been a crucial factor when developing and delivering a product. The process of software testing refers to the evaluation of the software with the intention of finding an error in it [1].*

This paper reviews existing methods, techniques and web-based applications that create evaluate and measure software tests. In the process of the literature review, several testing methods are considered and their application is looked for in several existing systems. Some tests are being conducted and the results of every system is analysed and compared. Software metrics have a direct link with the measurement of the software quality [2] and having the correct measurement means is of an utter importance.

The goal of this study is to find the most suitable metrics and testing methods which will be used in the process of developing a software testing module in an ERP system.

Keywords: *Software Quality Testing, Software Metrics, Software Test Metrics, Software Development, Software Development Life Cycle (SDLC), Testing Tools, Software Testing Tools*

INTRODUCTION

The main goal of every software company is to deliver reliable, easy to use and high quality product to its clients. In order to achieve that the company has to ensure that the product it sells is error free and tested. The process of software testing refers to the evaluation of the software with the intention of finding an error in it (Sawant, et al., 2012).

The purpose of this paper is to study the most used testing models, the types of testing techniques and the metrics used when analyzing the results from the tests. The case study has the intention of being a starting point for a web-based testing system which will create, evaluate and analyze tests for software quality assurance.

EXPOSITION

Software Testing Main Principals

A common mistake among new and small software companies is the belief that their product is good enough and it does not need a quality testing check. This is an error that won't be seen in companies which has time-tested values and believes.

Software testing tests the application for things such as reliability, usability, integrity, security, capability, efficiency, etc.

The goals of every software evaluation are (Khajuria, 2018):

- Error detection - As this was already mentioned above, the main goal of the testing procedure is to find errors. The number of found bugs in later stages of the development will result in higher costs.
- Verification - The first stages of every software project are always gathering functional requirements. One of the main goals of software testing is to ensure that these requirements are met and done.
- Validation - The process of validation is similar to the verification but the main difference is that on this step the client's needs and requirements are per his wish and necessity.

- Customer satisfaction - After meeting the basic requirements a test cycle about the whole client experience should be done. This means that not only the functional requirements will be tested but also the visual ones.
- Quality - The whole purpose of software testing is to ensure that the final product will be reliable, stable, secure, scalable, and accurate and will meet the standard of the company that develops it.

Some authors are reducing the testing steps to just 4 – debugging, verification, validation and defect (Hailpern, 2002). According to Hailpern and Santhanam, the process of debugging involves analyzing the given program in order to find out if it meets the specifications. That process is also referred as the process of "diagnosing the precise nature of a known error and then correcting it" (Mayers, 1976). Here the processes of verification and validation are overlapping with the definition of Khajura, Shabaz and Taneja (Khajuria, 2018). The last step, defect, is more of a definition than a step. It states that every occurrence of a divergence from the specification can be viewed as a bug and therefore should be corrected (IEEE Standard, 1989).

Software Testing Models

A quick overview of the most used development models shows that there is a big overlapping between the terms for software development models and the ones for software testing. A more exhaustive research shows that this is because every of this development models has a testing phase implemented in it. The difference as expected is in the sequence and complexity of the testing itself. This paper considers three of the most used models: Waterfall Model, V-Model and Agile Model (Balaji & Sundararajan Murugaiyan, 2012).

1. Waterfall Model – particular for the model is that there is only one testing phase, at the end of the Software Development Life Cycle (SDLC) (Fig. 1). Defects found at this stage of the development costs more resources of the company and is inefficient. The requirements should be clear at the beginning of the project and the output of every stage is input of the next. That makes the management of the project harder and the testing process unreliable.

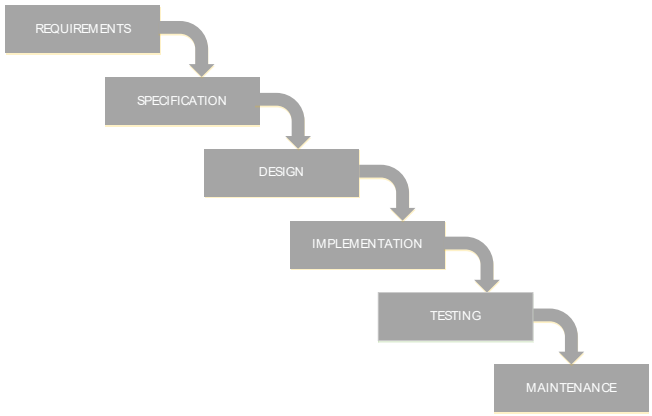

Fig. 1 Waterfall Development and Testing Model

2. V-Model – the V-Model is a modification of the Waterfall Model originated by the limitation of the Waterfall model in terms of testing phases. Fig. 2 shows that every phase of the development is followed by a testing phase and very often the developers and the testers are working parallel. Disadvantage of this model is the need for complete update of the documentation and retesting if a change in the requirements is made. That makes this model more suitable for the needs of today's development methods.

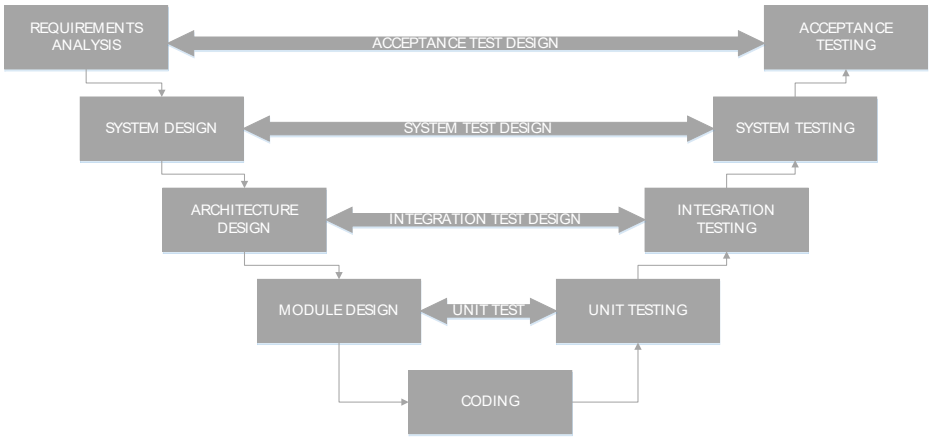

Fig. 2 V-Model for Software Development and Testing

3. Agile Model – the agile model is the most flexible model among the three models considered in this paper. A particular trait of this model is the fast development and implementation of a ready product. The project can be split in stages (sprints) and the client will receive updates, finished modules etc., in short periods of time. The division of the project into seemingly smaller projects will guarantee that every sprint consist of the five stages of development (Fig. 3). concerning the testing this is a complete cycle of development and testing in time.

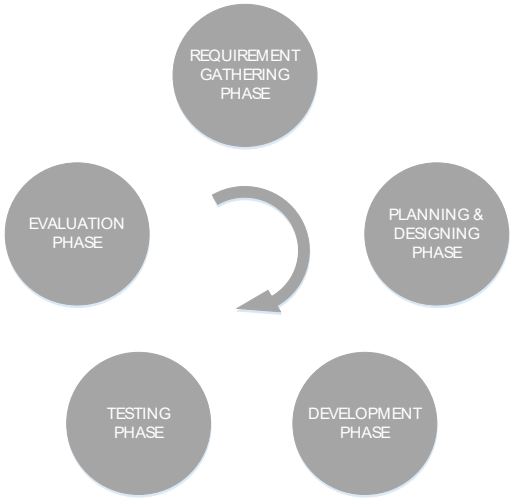

Fig. 3 Agile Development and Testing Model

Software Testing Life Cycle (STLC) and Software Testing Types

Regardless of the chosen model of development and testing the process itself consist of several steps that need to be done in order for the test to be evaluated. As it was said before the aim of every test is to find the errors in the application and to point them to developer which in turn to fix them. Therefore some guides when creating a test should be followed. Fig. 4 shows a basic STLC and the obligatory steps that should be taken when planning, creating and evaluating a test. Following these steps every test creator should be able to decide what type of test will be performed.

Fig. 4 Software Testing Life Cycle

The types can be divided in three main categories – functional, non-functional and maintenance. Fig. 5 shows the subcategorization of these three branches (Maheshwari, et al., 2019).

The functional types of testing are performed when the requirements should be validated and their implementation checked. This category consist of checks for unit functionalities, ways of integration, error handling etc.

The non-functional testing is considered when the application will be tested for performance, speed, load time and volume, usability, etc.

The maintenance tests are the ones that are mostly done at the end of the development cycle. These types of verifications gives a guarantees that the program has been installed or uploaded correctly and works in the production system of the client. There are several other cases when maintenance verifications are used – when there have been made changes to the original code of the program. Doing a maintenance check is mandatory in order to be sure that the new code is not interfering with the old one (Maheshwari, et al., 2019).

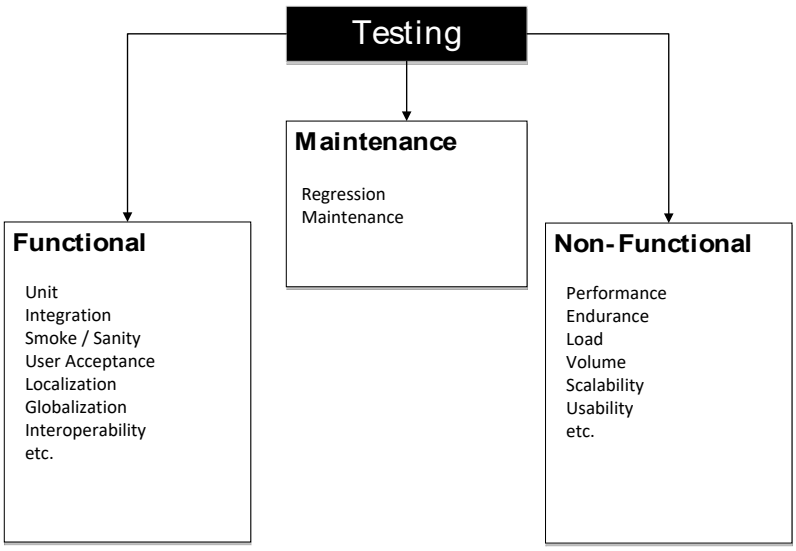

Fig. 5 Software Testing Types

Result Metrics

It's said that "you can't manage, what you can't measure" (Tom, 1986). Coming from that statement the role of the result metrics is critical.

The process of testing itself can't give enough information about the quality of the product, and if the standards of the development company have been followed. The testing will give the number of defects in the program and will be a guide to the developers when fixing the said defects. But if

planned right the testing can gather enough data which can be further analyzed and transformed so that the simple process of testing a program can become an existential part of the project management.

Every phase of evaluation should end with deep analyzation of the collected data and conclusions about the quality of the product and the team. In order to do that a set of result metrics should be used. They can give an overview of the development process and be an indicator for the future work of the team. The set of metrics should be carefully selected so that they are adequate for the specific project.

Product Metrics

The so-called "Product Metrics" are used when we want to measure the size and complexity of the program (Belachew, et al., 2018). The size of the program is a factor when considering the quality of the software originating from the assumption that the longer the program is the better it's been developed. Of course that can't be enough affirmation but it's considered among other metrics.

A disadvantage of sorts is that these types of metrics are only available once the product has been developed. Therefore they can't have a predictive character.

These metrics can measure external and internal attributes of the products – software usability and reusability, correctness, complexity, testability (Bhatti, n.d.).

Cyclomatic Complexity

The cyclomatic complexity metrics was first proposed by Thomas McCabe in 1976 for measuring the software especially for the Design phase (A. H, 1996). The process of measurement views the program as a graph. The cyclomatic number is the number of paths to reach the entire graph. That technique is good enough for small programs but when the software is long it becomes harder to count the number of paths. McCabe then proposes counting the number of the basic paths instead of all paths. The Cyclomatic Complexity can be expressed as the following equation:

$$\boldsymbol{V\ (G) = E - N + 2P} \quad (1)$$

Where:
V (G) – Cyclomatic Complexity
E – Number of edges
N – Number of nodes
P – Number of connected components or parts

Process Metrics

Process metrics are used when the developers and the team leaders want to get an overview of the number of defects found, the time it took for the developers to fix them, testing time and other quality related factors and indicators.

The measuring of these types of metrics can begin as soon as the development begins and that makes them appropriate for time predictions and quality control.

In many cases when teams wants to make performance optimizations they analyze the results from these metrics from previous projects.

Table 1 shows some of the testing process metrics summarized by (Lee & Chang, 2013) and proposed by (Premal & Kale, 2011), (Kaur, et al., 2007) and (Farooq & Quadri, 2011).

Table 1. Process Test Metrics

Test Metric	Definition	Formula
Test Case Productivity (TCP)	Gives an overview of the productivity	Total Test Row Steps / Efforts (Hours)
Test Case Effectiveness (TCE)	Gives the relation between the number of defects detected during testing and the total number of found defect in the program	Dt / (Df + Du) * 100
Test Efficiency (TE)	Determines the efficiency of the testing team and gives an indicator of the defects missed out during the testing phase	Dt / (Dt + Du) * 100
Bad Fix Defect (BFD)	Defects who when fixed resulted in a new defect are considered bad fix defects	Number of Bad Fix Defects / Total Number of valid Defects * 100%
Defects Rejected Percentage (DRP)	Number of falsely marked defects	(Number of rejected defects / Number of Total Defects) * 100
Critical Defects (CD)	Number of critical defects found	(Number of Critical Defects / Number of Total Defects) * 100
Test Improvement (TI)	Shows the relation between the number of defects found by the testers and the number of line codes	Number of Defects / Source lines of code in thousands
Test Cost as a ration of Development Cost (TCD)	Shows the relation between testing cost and development cost	Total cost of testing / Total cost of development of the product

CONCLUSION

This paper consist of the main three development and testing models, categorization of testing types and an overview of the metrics used when analysing the results from the tests evaluation.

The metrics presented in this paper are enough to give a team manager a detailed view of the quality of the development and the developed product.

The goal was to make an observation of the most used models, types and analyzation methods in order to plan and develop a testing tool that considers these factors. The tool should be able to track the development process as well as the tests execution.

ACKNOWLEDGEMENT

The study was supported by contract of University of Ruse “Angel Kanchev”, № BG05M2OP001-2.009-0011-C01, " Support for the development of human resources for research and innovation at the University of Ruse “Angel Kanchev”. The project is funded with support from the Operational Program " Science and Education for Smart Growth 2014 - 2020" financed by the European Social Fund of the European Union.

REFERENCES

A. H, T. J., 1996. Structured Testing: A Testing Methodology using the Cyclomatic Complexity Metric, s.l.: National Institute of Standards and Technology Gaithensburg.

Balaji, S. & Sundararajan Murugaiyan, D. M., 2012. Waterfall Vs V-Model Vs Agile: A Comparative Study on SDLC. International Journal of Information Technology and Business Management, 2(1).

Belachew, E., Gobena, F. & Nigatu, S., 2018. Analysis of Software Quality Using Software Metrics. International Journal of Computational Science & Application, Volume 8.

Bhatti, H. R., n.d. Automatic Measurement of Source Code Complexity, s.l.: Master's Thesis Computer Science and Engineering Lulea University of Technology.

Farooq, S. U. & Quadri, A. M. K., 2011. Software measurements and metrics: Role in effective software testing. International Journal of Engineering Science and Technology, 3(1), pp. 671-680.

Hailpern, B. S. P., 2002. Software debugging, testing and verification. IBM System Journal, 41(1), p. 4.

IEEE Standard, 1989. IEEE Guide to the Use of IEEE Standard Dictionary of Measures to Produce Reliable Software. New York: IEEE Standard 982.2-1988.

Kaur, A., Suri, B. & Sharma, A., 2007. Software Testing product metrics - a survey. Proceedings of National Conference on Challenges & Opportunities in Information Technology.

Khajuria, V. S. M. T. N., 2018. Software Testing: A Error Finding Technique. 7(2277-2723), p. 318.

Lee, M.-C. & Chang, T., 2013. Software Measurement and Software Metrics in Software Quality. International Journal of Software Engineerins and Its Applications, 7(4).

Maheshwari, H., Rana, I. & Goswami, P., 2019. A REVIEW OF TOOLS AND TECHNIQUES USED IN SOFTWARE TESTING. JETIR, 6(4), pp. 262-266.

Mayers, G., 1976. Software Reliability: Principles and Practises, New York: John Wiley and Sons, Inc..

Premal, B. N. & Kale, K. V., 2011. A brief overview of software testing metrics. International Journal of Computer Science and Engineering, 1(3/1), pp. 204-211.

Sawant, A. A., Bari, P. H. & Chawan, P. M., 2012. Software Testing Techniques and Strategies. s.l., s.n., pp. 980 - 986.

Tom, D. A., 1986. Controling Software Projects. s.l.:New York: Yourdon Press.