ISSN 2237-8960
Brazilian Journal of Operations & Production Management

SOFTWARE TESTING ESTIMATION: BIBLIOGRAPHIC SURVEY IN BRAZILIAN AND INTERNATIONAL ENVIRONMENTS

Fernando Oliveira de Araujo[a]; Luciana Vago Matieli[b]

[a] **Fluminense Federal University** (UFF) - Niterói, RJ, Brazil

[b] Accenture - Rio de Janeiro, RJ, Brazil

ABSTRACT

The methodologies used in the market regarding pricing practices related to software testing activity are strongly based on empiricism, without the parameters and criteria that are widely accepted, what does not confer the due credibility in terms of time and cost for businesses. Despite the highlighted theoretical and empiric importance assigned to the field of software testing, in Brazil, investigations on the theme software testing estimation are still scarce. This article provides a literature review of the main academic publications in national and international circulation which focus on the study of how to estimate software testing, aiming at fomenting and guiding further national researches specifically dedicated to this thematic. The review has provided both the characterization of how to estimate software testing and survey of the main existing techniques.

Keywords: Software testing estimation; methodology for software testing estimation; software quality

ABEPRO
DOI: 10.14488/BJOPM.2017.v14.n1.a2

Brazilian Journal of Operations & Production Management
Volume 14, Número 1, 2017, pp. 10-18
DOI: 10.14488/BJOPM.2017.v14.n1.a2

1. INTRODUCTION

Brazilian Telecommunication Companies usually hire outsourcing companies to carry out development and testing their systems. Every creation and/or change to computational systems, besides the software development, demands functional and/or performance tests to be performed in order to assure that the change has been made according to the functional and technical requirements specified.

Testing is a fundamental activity to assure the software quality. Some of the main challenges in tests are to match test cases and requirements correctly, in order to provide accurate information, estimations and follow up the test progress accordingly (LIOU, 2010).

According to Nguyen *et al.* (2013), software testing is an important activity in software development and maintenance projects, assuring quality, applicability and usefulness of software products. Actually, no software can be released without a reasonable quality of tests involved. In order to reach the acceptable quality, software project teams assign a substantial share of the total development effort to test performance. According to reports from the industry, software testing takes about 10-25% of the project total effort, and in some projects this number may reach 50%.

To Zhou *et al.* (2013), software testing, defined as the software systematic execution with the purpose of revealing failures, is an important stage to validate the software correctness. Software testing activities are composed of the definition of test cases and validation of the execution behavior. In general, the execution of test cases is limited, since validating all of the execution paths tends to be inevitable in terms of time and cost. Thus the quality of the test cases affects directly the software quality, and one of the features of quality test cases is the ability to detect failures that are still to be revealed.

To Lazic *et* Mastorakis (2008), software testing is an activity that can provide the product with visibility and the process with quality. Test metrics are among the facts that managers can use in the project in order to understand its current position and prioritize its activities, so that they can reduce the risk (or impact) of running out of time before the software is ready to release.

In the same line of thought, Aranha *et* Borba (2007) mentioned that software testing is an activity that has been aggregating more and more quality to the development process as well as the final product. It has been enriched by own methodologies, automation tools, and independent and trained teams. However, as well as the software development projects, testing projects has the same challenges and issues related to cost and time.

A common activity performed in order to assure the software quality is the execution of tests (ARANHA *et* BORBA, 2007).

A system testing estimation is subjective, as there are many variables that interfere with its final result, such as the knowledge level of the professional who is making the estimation, the features of each system to be tested, whether the system allows to perform parallel tests, so that it is possible to budget more professionals with little experience, whether there is, or not, a support documentation , among other variables. One of the biggest difficulties in the realm of software testing activities is to evidence clearly the activities that will be performed with its costs and times.

According to Patel *et al.* (2001), making an effective estimation of a software project is one of the most important and challenging ones. Only with a precise and reliable estimation it is possible to complete a project before the deadline. Estimations play a vital role in all the stages of the software development life cycle.

As Tronto *et al.* (2008) point out, a critical issue in software project management is how to make precise estimation of size, effort, resources, cost and time spent in the process development. Underestimations generated by time pressures may compromise the functional development and the software testing. Just as an overestimation may not match the project and generate non-competitive budgets.

In the same line of thought, Liou (2010) mentions that the purpose of software estimation is to generate realistic estimations, both for the project team and for the client (the project sponsor). An inaccurate estimation may cause problems that no cost may recover. Besides that, producing an estimation intended to please the client, but not in accordance with the project team, may lead the project to failure.

In this context, the importance of software testing activities is reinforced, in addition to a correct estimation. Nevertheless, in the international academic environment, the number of publications concerning specifically software testing estimation is small. When searching the bases Scopus and Web of Science with combined words “Software Testing” and “Estimation”, and “Software Test” and “Estimation”, 188 publications were found; however from the titles and abstracts, only 7 publications that had some relation to the theme concerned could be selected.

A specific lack of technical and scientific documents in Portuguese also stands out, which represents a fragility to the organizations and professionals who work in software testing activities – still strongly based on empiricism.

Given the theoretical challenge above, this study aims to provide a systemization of the technical and scientific literature on methodologies and software testing activities,

Brazilian Journal of Operations & Production Management
Volume 14, Número 1, 2017, pp. 10-18
DOI: 10.14488/BJOPM.2017.v14.n1.a2

also incorporating an investigation on the border of the state of marketing practices related to software testing activities.

As expected contributions, this work aims to reduce a gap seen in the literature, concerning systemization of studies related to software testing, aiming to consolidate the main evidences and propositions presented.

In terms of structure, this work is organized in 4 sections, namely: the methodology used in the study is described in section 2; section 3 is dedicated to describing the results and discussions; and section 4 presents the conclusions and suggestions for further studies.

2. METHODOLOGY

The bibliographic survey was made from indexed periodicals in the multidisciplinary databases Scopus and Web of Science, accessed in the period from November 11th, 2013, to May 20th, 2014, through CAPES Periodicals Portal.

The first researches were carried out in the databases SCOPUS and Web of Science using the following keywords separately: "Software Testing"; "Software"; "Estimation". This first search initial goal was to identify and investigate the general amount of publications in the respective databases, with no restriction. As a result of this first research, it was possible to evidence many themes related to the three keywords searched separately. Within the field of software testing were found themes on failure prediction in large software projects, software automation, regression test, estimation of the number of faults found along the tests, test data generation (inputs), estimations for test cases generation, among others.

Because of the high dispersion and variety of records found, it was necessary to refine the research, so that it was necessary to combine some words, "Software Testing" AND "Estimation" and "Software Test" AND "Estimation", intended to restrict the research universe, besides considering only the articles in the field of computer science.

After carrying out these searches, the 25 duplicate records found in the bases SCOPUS and Web of Science were properly discarded, remaining a total 188 articles to be analyzed, from the titles and abstracts.

From the analysis of the titles and abstracts of the 188 articles, 7 were selected for reading and full analysis, being considered relevant and lined up with the study concerned. Then the result achieved is shown, focusing the presentation on the main references for each topic taken into consideration. The data of the articles are illustrated in Table 1.

In an analog way to the researches in the bases Scopus and Web Science, a survey was carried out in the database of SciELO.Org in the period from May 25th, 2014, to May 28th, 2014. The results of this research evidenced many themes related to three keywords searched separately: "Software Testing"; "Software Test"; "Estimation", being necessary to combine some words, "Software Testing" AND "Estimation" and "Software Test" AND "Estimation", intended to restrict the research universe.

Table 1. Articles found

Author (Publication Year)	Article Abstract
Christodoulakis D. *et* Panziou G. (1990)	Optimal estimation techniques are applied to predict the future behavior of software systems.
Jiang, Li-Xin; Han, Wan-Jiang; Yan, Chen-Chen *et* Shi, Bo-Ying (2012)	In this paper, a function testing size estimation model based on testing steps is proposed. The model applies to black box testing. A global analysis on COCOMO model is carried out. Its basic steps include: design test cases, sum total test steps, define the parameters in the model based on the pattern of model, define size factor, and calculate test size.
Liou, Jing-Chiou (2010)	In this paper, we present a parametric model for software test estimate along with a test graph for matching test cases with requirements and test cases analysis to aid in producing more accurate estimates.
Nguyen, Vu; Pham, Vu *et* Lam, Vu (2013)	This paper describes a simple process, which we name qEstimation, to estimate the size and effort of the software testing activities. The process incorporates a proposed approach to measuring the size of the test case based on its checkpoints, preconditions and test data, as well as the type of testing.
Srinivasan, Krishnamoorthy *et* Fisher, Douglas (1995)	This article describes two methods of machine learning, which we use to build estimators of software development effort from historical data.
Tronto, Iris Fabiana de Barcelos; Silva, José Demísio Simões da; *et* Sant'anna, Nilson (2008)	This paper aims to offer alternative methods for those who do not believe in estimation models, based on artificial neural networks and regression.
Zhu Xiaochun; Zhou Bo; Hou LI; Chen Junbo *et* Chen Lu (2008)	This paper proposes an experience-based approach for test execution effort estimation. The approach is characterized by a test set as a 3-dimension vector which combines test case number, test execution complexity and the tester's experience.

Source: The authors own

The search in the ScieELO.Org database did not return any publication pertinent to developing bibliographic review; although finding 19 articles, none matches the research central theme. Figure 1 illustrates the abstracts of the researches in the databases Scopus, Web Science and SciELO.Org.

3. RESULTS AND DISCUSSION

According to Lopes *et* Nelson (2008), most of the estimation models applied to development also aim at estimating the test effort given to the importance of this activity. In this way, it was possible to identify eight different themes for discussion about software testing estimation: (i) Estimate produced from a software development project effort distribution; (ii) Estimate produced by a percentage

Brazilian Journal of Operations & Production Management
Volume 14, Número 1, 2017, pp. 10-18
DOI: 10.14488/BJOPM.2017.v14.n1.a2

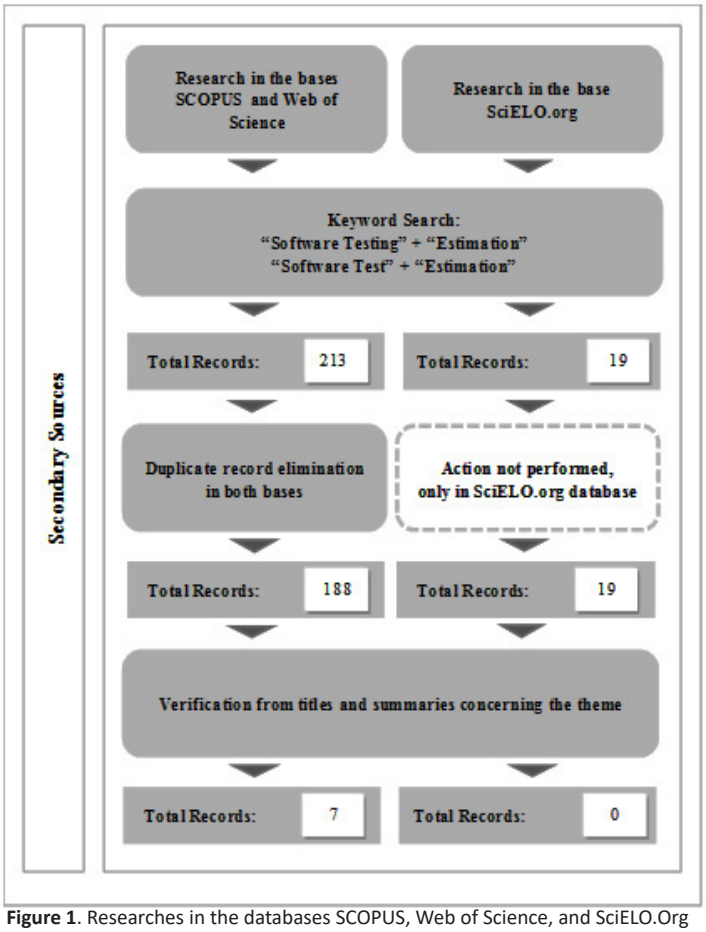

Figure 1. Researches in the databases SCOPUS, Web of Science, and SciELO.Org
Source: The authors own

of the time for development; (iii) Estimate produced on function points (FP); (iv) Estimate produced on use case points (UCP); (v) Estimate produced on test cases points (TCP); (vi) Estimate produced by test point analysis (TPA); (vii) Estimate produced from testing projects historic bases; (viii) Estimate produced from Simple Estimate Test (SET).

3.1. Estimate produced from a software development project effort distribution

To Patel *et al.* (2001), traditionally an estimate of effort for tests was a percentage of the remaining stages of the development life cycle. This approach to estimation is more prone to errors and brings more risk of delaying the deadlines for releasing the product.

According to Pressman (2005), the recommended effort distribution throughout the software process is often referred to as the rule 40-20-40. This rule recommends that 40% of the effort is reserved for analysis and design, 20% of the effort is reserved for coding, and 40% of the effort is reserved for testing. This effort distribution should be used only as a guideline and it varies according to the features of each project. The rule is very simple to be applied, as illustrated in Figure 2.

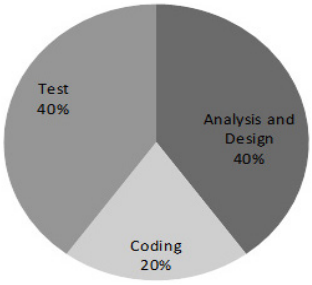

Figure 2. Effort Distribution (40-20-40)
Source: Pressman (2005)

To Matieli (2014), applying a percentage to estimate testing effort, the important points such as the coverage and the complexity of the tests are not considered and may have systems that require much more complexity in the tests. In these cases, the testing effort estimated by this rule tends to be much less than what should be performed.

Currently the rule 40-20-40 is under attack. Some believe that over 40% of the global effort should be spent during the analysis and design. On the other hand, some supporters of agile development argue that less time should be spent "in the front" and that a team should move rapidly to the building process.

Brazilian Journal of Operations & Production Management
Volume 14, Número 1, 2017, pp. 10-18
DOI: 10.14488/BJOPM.2017.v14.n1.a2

3.2. Estimate produced based on the percentage of time for development

According to Lopes *et* Nelson (2008), in this technique the test effort estimated is based on the development estimated value of time/effort (number of lines of code or points per function), i.e., the number of lines of code or points per function is counted and a percentage is applied to this result to estimate the testing time. According to Matieli (2014), a positive aspect of this technique would be that after counting the number of lines of code or function points, applying a percentage, the testing estimation is produced more agilely. However in this technique the test nature, environment, complexity of test cases, among other variables, are not taken into consideration

To Nageswaran (2001), this method is not based on any scientific or technical principle. Delays in schedules may vary 50-75% from the estimated time. This method is far from being the most used.

3.3. Estimate produced on function points (FP)

To Lopes *et* Nelson (2008), the estimated effort and test case is determined by function point estimation, according to Capers Jones. The formula used is: number of test cases = (function point) to the power 1.2. It can produce accurate estimates as long as the analysis of function points is also accurate. Capers Jones' estimation feature different efforts depending on the consideration made due to having a polynomial increase, where the number of test cases equals (function point) to the power 1.2. The number of test cases grows as the system size grows. Therefore, when we consider the estimation for each use case and then we add the result achieved from the use cases, the result of effort will be less than when we consider the project total size. The bigger the size in function points, the bigger will be the effort spent. Another point is that this technique does not provide a test estimate in hours and it does not consider any characteristic of a software project.

According to Mastorakis *et al.* (2009), the disadvantage of using this technique is that the requisites are required in advance.

3.4. Estimates produced on use case points (UCP)

To Aranha *et* Borba (2007), the estimate produced by use case points (UCP) is an extension of the estimate produced by the function point (FP) and it estimates a system size based on each use case specification. Both UCP and FP consider a system development complexity.

According to Lopes *et* Nelson (2008), the effort estimated for the test is found by multiplying the adjusted UCP with a conversion factor. This conversion factor denotes the number of men/hour that represents the test effort required for a combination of language and technology. This conversion factor is determined by the organization for the given combinations. Use case points (UCP) was developed based on function points, and in the philosophy both methods are based on is the same, i.e., the functionalities seen by users are the basis to estimate the software size. The lack of universal patterns to build use cases makes it difficult to compare projects from different organizations. Thus, if the criteria used to build the use cases are very diverse, there is no way to assure that the use cases will be measuring the same thing.

3.5. Estimate produced on test case points (TCP)

To Patel *et al.* (2001), the estimate based on test case points is considered one of the most accurate estimations for functional tests due to emphasizing factors that determine the complexity of the cycle of tests as a whole. This technique combines four stages of the testing process: test cases generation, scripts implementation for automated tests, manual tests execution, and automated tests execution. It can also be used in processes where one or more stages of the process are applied.

Still according to Patel *et al.* (2001), as said above, the TCP analysis generates test efforts for the separate test activities. This is essential, because the test projects tend to fall under four different models. Although, in practice, most of the test projects are a combination of the four models of execution, as illustrated in Figure 3.

An advantage seen in this technique is the emphasis on test automation, and the flexibility of estimating software tests that are run: only manually, only automated, or both manual and automated. The technique does not establish any criteria to transform the points achieved by test case in test effort.

3.6. Estimate produced on test point analysis (TPA)

According to Souza *et al.* (2010), among the effort estimation techniques for software test, test point analysis (TPA) is considered in literature as the most consistent. However, although the technique authors claim that it was used by some organizations and they had good results, no detailed reports or measuments to prove such results were found.

To Veenendaal *et* Dekkers (1999), the estimated effort is to define, develop and run functional tests, based on the software development complexity (derived from the function point analysis technique), also considering the tests and productivity strategy.

Still according to Veenendaal *et* Dekkers (1999), the greatest benefit from TPA is in managing to gather factors

Brazilian Journal of Operations & Production Management
Volume 14, Número 1, 2017, pp. 10-18
DOI: 10.14488/BJOPM.2017.v14.n1.a2

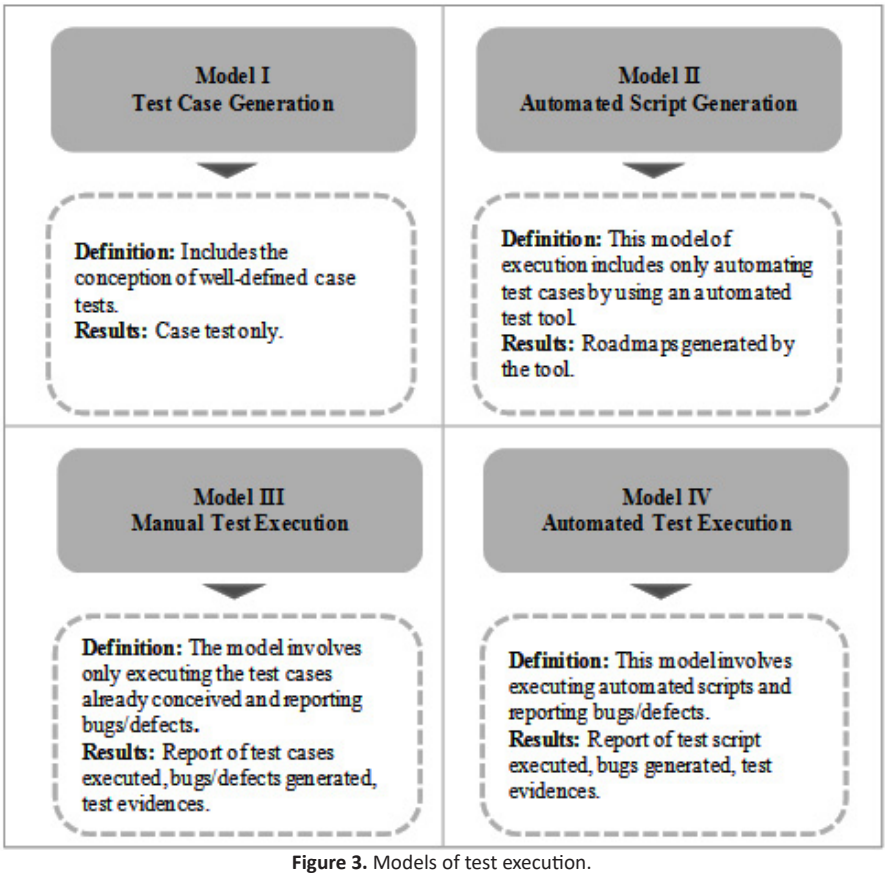

Figure 3. Models of test execution.
Source: Adapted from Patel *et al.* (2001)

that influence the specific effort of one of the stages of the development process in a systematic way, producing more accurate results. Furthermore, it allows evaluating the test effort per activity and takes into consideration the stages of planning and control that are essential to any project success and should be considered in effort estimation techniques.

This TPA technique can be considered complex and difficult for use and interpretation. There are several variables such as system size in function points, considering also the quality characteristics to be tested dynamically, dynamic or static testing team productivity, so that their values are defined through analysis of the many factors that composed them (Lopes *et* Nelson, 2008).

According to Bastos *et al.* (2007), TPA considers as the basis for its analysis the size of the system to be tested from the information gathered along with the development team. In the definition and calculation of the number of testing hours, the system size in function points is the initial calculation base in these estimations. The tests are also affected by the following factors:

a. The degree of the test process complexity. By this, we mean to say that very complex systems tend to consume more testing hours than the simpler ones, as expected.

b. The quality level intended to be reached with the tests. Evidently, the requirement of very high quality levels will demand more test effort. There are cases of financial systems, for example, that should be failure-free, or have a near-zero failure rate prediction when entering production. In these cases the test effort will tend to be bigger.

c. The degree of user involvement with the tests. The more users get involved in the test activity, or validation, the better results are achieved and, very often, the less will be the testing effort.

d. The interfaces that the functions under test have as files. The bigger the number of files involved in a test case, for example, the more difficult will be to test the system or software. If the test case, for example, maintains only one file, testing it and assessing the results achieved will be much easier.

e. The quality of the system under test (the defect recurrence cycle). When the system has development or design defects, certainly the work of testing will be more onerous. In this case, it is true the precept that the cost of defects grow in geometric progression the later they are detected. Production defects are much more expensive than project defects.

Brazilian Journal of Operations & Production Management
Volume 14, Número 1, 2017, pp. 10-18
DOI: 10.14488/BJOPM.2017.v14.n1.a2

f. The coverage level expected with the tests. The system requirements will usually establish the coverage level required by the tests. For example, to some systems, the load or performance tests are indispensable, whereas to others, where the access level is too low, such tests can even be dismissed.

g. The test team's experience and productivity (measured through historical indicators). It is evident that the test team's quality and management are linked to the effort spent to run the tests. There are organizations that use historical bases to estimate and assess their testing process. This information also serves as a base for measuring the team's productivity.

h. Automation degree of the tests. The automation tools allow for higher levels of productivity, as they facilitate repeating tests already run, as many times as necessary, in addition to facilitating documentation of test cases.

i. The test environment quality, including its ability to simulate the production environment. The test environment should be really close or equal to the production environment, because in this way many compatibility problems between the environments are prevented. This will be especially important to stress, load and performance tests.

j. The system documentation quality and, especially, requirements. As the requirements are the basis for the system development, we will have a process in chain, as the test only shows defects occurrences, however it does not solve the problem resulting from poorly designed project. In this case, the correction costs tend to be very high, as well as the test effort will be much more. Using test point analysis will allow the test process to have a measurement of its own to determine its size, what is justified when the test is treated as an independent activity, keeping a connection with the system size in function points.

3.7. Estimate produced from historical bases of test projects

According to Lopes *et* Nelson (2008), the effort estimated through a historical base is based on the collection of information stored in the projects databases, where the business requirements are the basic information for the estimations. In order for the process of estimating the effort to be actually consistent, it is necessary that all the historical data are extremely organized and systematic, so that the figures produced have the most possible accuracy. To Matieli (2014), this technique is positive, because it produces a history of projects inside the organization, so that it is possible to make inquiries of analog projects in order to make the most consistent estimations, produce statistical data. However, in order for the process of estimating effort to be really consistent, it is necessary that the historical data records are extremely organized and systematic so that the figures produced have the most possible accuracy.

The creation of historical bases for the projects estimations is a source for elaborating indicators (SOUZA *et al.,* 2010).

3.8. Estimate produced from Simple Estimate Test (SET)

According to Matieli (2014), the Simple Estimate Test (SET) was proposed according to the combination of technical and scientific literature, with the distribution of answers from the empirical investigation applied along with experts in the field of software testing.

Still to Matieli (2014), the technique can be used to estimate functional tests taking into consideration variables such as test cases definition, test cases complexities and its respective classifications, hours of test cases planning, hours of test cases execution, hours of retest, and hours of test management, besides the schedule variable. These variables for composing the model, its respective weights and percentages were defined after a semantic analysis of the answers to a questionnaire applied to experts in large multinational company of the telecommunication sector. As illustrated in Figure 4.

The option for the value range rather than fixed values was present, considering that projects of different complexities demand different times for the activity concerned.

In order to estimate using the technique, it is necessary that the professional knows how the system works. It is seen in the SET that the incorporation of the professional reasonableness concerns the project and the organizational environment specificities, besides technical knowledge in defining and classifying the test cases, are requirements that determine a good estimation of the complexities shown in Table 2.

Table 2. Criteria for the complexities classification

Complexity	Definition	Weight
Low	Little user interaction with the program, not much processing time to produce a piece of data.	1
Average	User reasonable interaction with the program, normally in the same interface, slightly long processing and more than one verification.	2
High	More than one different user interaction with the program, normally in different interfaces, processing and results validation are long.	3
Very High	Users need to interact a lot with the program in more than two interfaces and validate a big number (five or more) of results.	4

Source: Matieli (2014)

Brazilian Journal of Operations & Production Management
Volume 14, Número 1, 2017, pp. 10-18
DOI: 10.14488/BJOPM.2017.v14.n1.a2

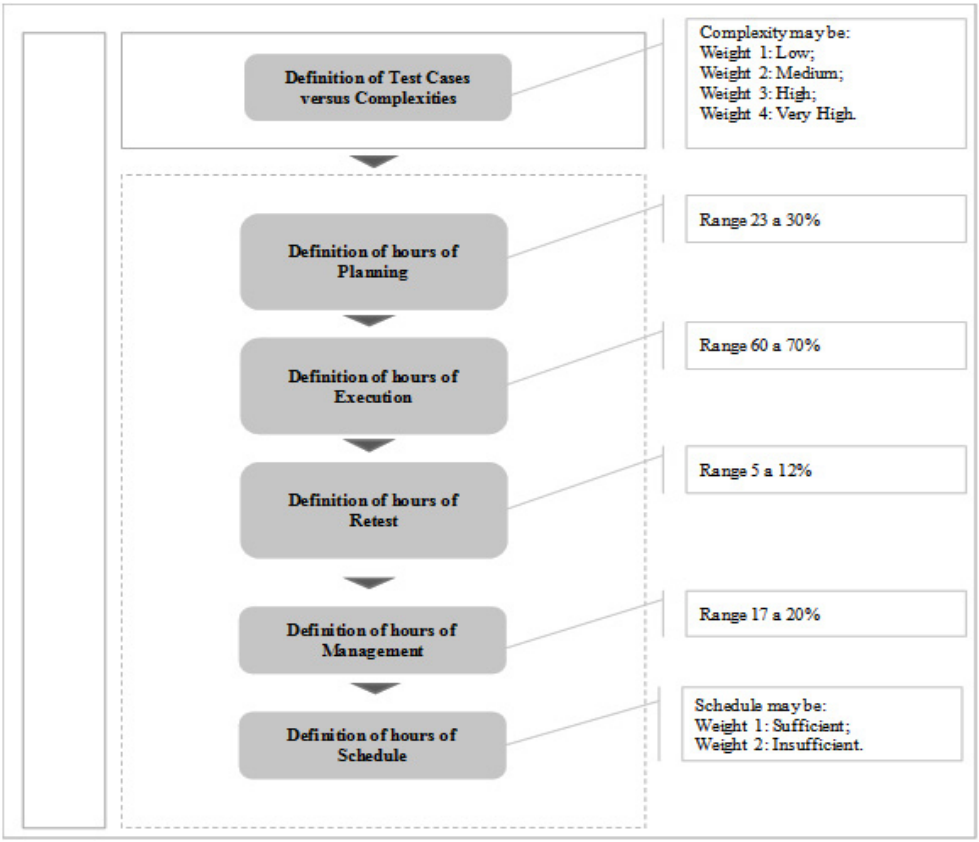

Figure 4. SET – Simple Estimate Test
Source: Matieli (2014)

From test cases definition and classification, the planning, execution, retest, management and schedule hours are calculated, achieving final estimation value in hours.

To some critics, this technique may be considered difficult to use and interpret, because it is necessary to understand the project complexity functionally in order to be able to identify within each range the most appropriate value and thus achieve a final value in hours. Another important point is that the proposed technique needs to be applied in actual business projects (MATIELI, 2014).

4. CONCLUSION AND FUTURE SUGGESTIONS

During the study, it was seen that some techniques exist in the literature; however, none is dominant, better or recommendable, because there is no dominant, better or more dominant technique either in the literature, or in the practices of big organizations. In practice, the companies invent their own methods, mostly with no technical or scientific foundation.

In this sense, this work provides a summarized overview of the main techniques developed for dealing with the issues inherent to the evident empiricism in marketing practices, representing a relevant input for both researchers and practitioners who work or wish to work in this field.

As suggestions for further studies, we recommend selecting some existing techniques in the literature and their application in actual corporate projects, in order to make a comparative assessment concerning the degree of accuracy of each test for specific situations.

REFERENCES

Aranha, E.; Borba, P. (2007). An estimation model for test execution effort. Available in: <http://www.academia.edu/8455490/An_Estimation_Model_for_Test_ Execution_ Effort> Accessed: January 22, 2015.

Bastos, A.; Rios, E.; Cristalli, R.; Moreira, T. (2007). Base de Conhecimento em Teste de Software. Martins, São Paulo.

Christodoulakis, D.; Panziou, G. (1990) Modelling software reliability prediction with optimal estimation techniques, University of Patras, Jan./Feb., vol. 32, No. 1.

Jiang, L.-X.; Han, W.-J.; Yan, C.-C., Shi, B.-Y. (2012). Research on size estimation model for software system test based on testing steps and its application. 2012 International Conference on Computer Science and Information Processing (CSIP), China, p. 1245 a 1248.

Lazic, L.; Mastorakis N. (2008). Cost Effective Software Test Metrics. Wseas Transactions on Computers, 7(6), 599-619.

Brazilian Journal of Operations & Production Management
Volume 14, Número 1, 2017, pp. 10-18
DOI: 10.14488/BJOPM.2017.v14.n1.a2

Liou, J. (2010). On Software Test Estimate and Requirement Tracking. 19th International Conference on Software Engineering and Data Engineering, USA, pp. 57-62.

Lopes, F. A.; Nelson, M. A. V. (2008). Análise das Técnicas de Estimativas de Esforço para o Processo de Teste de Software. Available in: <http://ebts2008.cesar.org.br/artigos/EBTS2008-Analise_das_Tecnicas_Estimativas_de_Esforco.pdf> Accessed: February 15, 2013

Matieli, L. V. (2014). Proposta metodológica para elaboração orçamentária de testes de software. Dissertação (Mestrado em Sistemas de Gestão). Niterói: Universidade Federal Fluminense. Orientador: Prof. Fernando Oliveira de Araujo.

Mastorakis, N.; Mladenov, V.; Kontargyri, V. T. (2009). Proceedings of the European Computing Conference. Springer, Grécia.

Nageswaran S. (2001). Test Effort Estimation Using Use Case Points (UCP). Available in: <http://www.bfpug.com.br/Artigos/UCP/Nageswaran-Test_Effort_Estimation_Using_UCP.pdf> Accessed: January 06, 2015.

Nguyen, V.; Pham, V.; Lam, V. (2013). qEstimation: A Process for Estimating Size and Effort of Software Testing. International Conference on Software and Systems Process, USA, pp. 20-28.

Patel, N.; Govindrajan, M.; Maharana, S.; Ramdas, S. (2001). Test Case Point Analysis - Cognizant Technology Solutions. Available in: <www.stickyminds.com/getfile.asp?ot=XML&id=2566&fn=XUS373692file1.pdf> Accessed: January 06, 2015.

Pressman, R. S. (2005). Software Enginnering: a practitioner's approach. McGraw-Hill, USA.

Srinivasan, K.; Fisher, D. (1995). Machine Learning Approaches to Estimating Software Development Effort. IEEE Transactions on Software Engineering, v. 21, n.2, p.126-137.

Souza, P. P. de; Barbosa, M. W.; Silva, A. R. da (2010). Estimativa de Esforço de Teste no Auxílio da Garantia da Qualidade de Software. Available in: <http://www.spinsp.org.br/artigos/Revista_PBQP_2_Edi%C3%A7%C3%A3o.pdf> Accessed> January 07, 2015.

Tronto, I. F. de B.; Silva, J. D. S. da; Sant'anna, N. (2008). An Investigation of Artificial Neural Networks Based Prediction Systems in Software Project Management. The Journal of Systems and Software, 81, 356–367.

Veenendaal, E.V.; Dekkers, T. (1999). Test point analysis: a method for test estimation. Project Control for Software Quality, Editors, Rob Kusters, Adrian Cowderoy, Fred Heemstra and Erik van Veenendaal. Shaker Publishing.

Zhou, B.; Okamura, H.; Dohi T. (2013). Brief Contributions: Enhancing Performance of Random Testing through Markov Chain Monte Carlo Methods, IEEE Transactions On Computers, 62(1), 186-192.

Zhu, X.; Zhou, B.; Hou, L.; Chen, J.; Chen, L. (2008). An Experience-Based Approach for Test Execution Effort Estimation. The 9th International Conference for Young Computer Scientists. China.