TEST EFFORT ESTIMATION AND ITS TECHNIQUES

Poonam Kumari[1], Nikita Bakshi[2], Yamini Pathania[3]
Dept. of Computer Science Engineering
Chandigarh University

ABSTRACT: Testing is the most important quality assurance measure of software. Test execution becomes an activity in the critical path of software development. In this case, a good test execution effort estimation approach can benefit both tester managers and software projects. Test Effort Estimation is an important activity in software development. The test effort can be calculated on the basis of cost effort, quality and time required for testing. Software test effort estimation is a complex activity that requires knowledge of key attributes that affect the outcomes of software projects. The critical problem is needed lot of data, which is often impossible to get in needed quantities. Hence, Software test effort estimation has become a challenge for IT industries. In this paper software metrics are described for cost estimation technique and some test effort estimation techniques are also described.
Keywords: Test effort estimation, FP (Function Point), UCP (Use Case Point) and TPA (Test Point Analysis), LOC (Line of Code).

I. INTRODUCTION

Testing is the process of evaluating a system or its components with the intent to find whether it satisfies the specified requirements or not. In other words, testing is executing a system in order to identify any errors, bug or missing requirements in the actual requirements [4].Software testing is a process of executing a program or application with the intention of finding the software bugs. It can also be stated as the process of validating and verifying that a software program or application or product. The testing techniques are useful for cost effective way of testing. Test estimation is a management activity which approximates how long a task would take to complete. The test effort estimation is one of the major and important tasks in test management. Estimation is done because it is expected to help in predicting how much will the project cost and when will the project get completed. Test effort is important for triumphing planning for a testing process. The software testing consumes more effort, which is nearly 50-60% of the total effort t[1]. In software development, the test effort estimation is an important activity. The test effort refers to the expenses for tests. The test effort is related to the quality, cost and time required for testing. It defines the relation between test cost and failure cost [5]. The cost of software development is due to the human effort and most cost estimation methods focus on this aspect and give estimates in terms of person-months. Accurate cost estimates are critical to both developers and customers. They can be used request for proposal, contract negotiations, scheduling, monitoring and control. There are number of competing software cost estimation methods available for software developers to predict effort and test effort required for software development. Testing is an important activity to ascertain software quality. Big organizations can have many development teams. Their products being tested by overloaded test teams. In such condition, test team managers must be able to properly plan their schedules and resources and estimates for the required test execution effort can be an additional criterion for test selection. A good test execution effort estimation approach can benefit both tester managers and software projects.

Software testing used following methods for test effort estimation. These are:

- Use case point estimation
- Function points / Test point Analysis
- Ad-hoc method
- Experience Based - Analogies and experts
- WBS
- Delphi technique
- Three-point estimation
- Percentage of development effort method
- Percentage distribution
- FIA (finger in the air) or best guess

II. SOFTWARE METRICS

In the test effort estimation are used many software metrics. This section provides some background information on software conventional metrics used in software test effort estimation like size oriented metrics and function oriented.

A. Size oriented metrics: Size oriented metrics are widely used but their validity and applicability is widely debated. Derived by normalizing quality and or productivity measures by considering the "size" of the software .These are two types:

- Source Lines of Code (SLOC): is software metric used to measure the size of software program by counting the number of lines in the text of the program's source code. This metric does not count blank lines, comment lines, and library. SLOC measures are programming language dependent [8]. SLOC also can be used to measure others, such as errors/KLOC, defects/KLOC, pages of documentation/KLOC, cost/KLOC [4].
- Deliverable Source Instruction (DSI): It is similar to SLOC. The difference between DSI and SLOC is that "if-then-else" statement, it would be counted as one in SLOC but in DSIcounted several.

B. Function oriented metrics: Function-oriented software metrics use a measure of the functionality delivered in normalization value. Since 'functionality' can be measured indirectly using other direct measures. Function points are derived using an empirical relationship based on countable measures of software's information domain and assessments of software complexity.

- Function Point (FP): It is a unit of measurement to express the functionality of software. FP is programming language independent, making ideal for applications using conventional and nonprocedural languages. It is based on data that are more likely to be known for evolution of project. Function types are :
  - Number of user inputs
  - Number of user outputs
  - Number of user inquiries
  - Number of files
  - Number of external interfaces

III. TEST EFFORT ESTIMATION TECHNIQUES

The multiple techniques are used in test effort estimation. In this section we have explained some techniques like UCP (Use Case Point Analysis), FP (Function Point) /TPA(Test Point Analysis)etc.

A .Use case point analysis (UCP): Test effort estimation using UCP is based upon use cases (UC). UC is a systems behavior under multiple conditions. It is based on requests from a stakeholder. Use case capture contractual agreements between these stakeholders about the systems behavior. Thus, the primary task of UCP is to map use cases (UC) to test cases (TC). Hereby, each scenario together with the corresponding exclusion flow for each UC serves as input for a specific test case [1]. Basically, the amount of test cases identified through this mapping results in the corresponding effort estimation. So the multiple factors in use case give a direct ratio to the testing effort. Use case is a document which specifies different users, systems or stakeholders interacting with the concerned application. They are named as 'Actors'. The interactions accomplish some defined goals protecting the interest of all stakeholders through different behaviour or flow termed as scenarios. UCP is comprised of six basic steps that determine a projects required test effort:

- Obtain unadjusted actor weight (UAW)
- Determine unadjusted use case weight (UUCW)
- Calculate unadjusted use case points (UUCP) UUCP = UAW + UUCW
- Determine the technical/environmental factor (TEF)
- Compute the adjusted use case point (AUCP) AUCP = UUCP * [0.65 + (0.01 * TEF)]
- Arrive at final effort using a conversion factor Total Actual Effort = AUCP * CF

B. Test Point Analysis (TPA): Test point used for test point analysis (TPA), to estimate test effort for system and acceptance tests. TPA (Test Point Analysis) only covers black-box testing it doesn't covers white box testing [1]. Hence, it is always used with FPA (Functional Point analysis), that does not cover system and acceptance test cases. FPA and TPA merged together provide means for estimating both, white and black box testing efforts. Test Point analysis is used for estimation of system testing or acceptance testing. It helps to calculate the test hours required and the risk involved in software project [9]. Risks can be identified by comparing the test point analysis (TPA) estimate and the pre-determined hours.

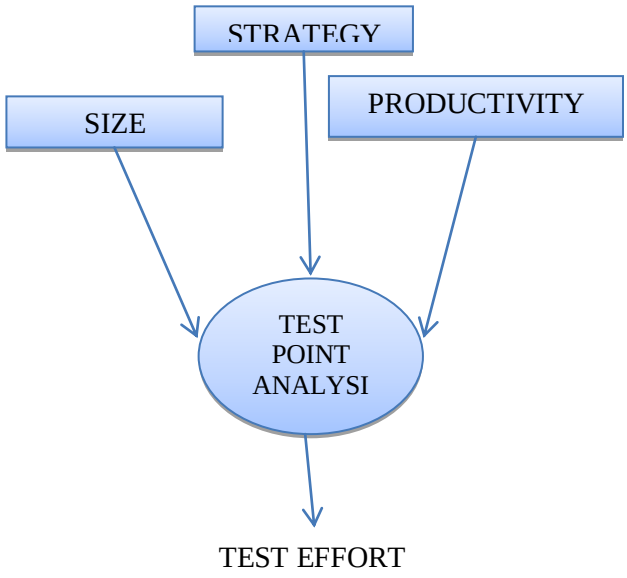

Fig1: Strategy Based Effort Estimation

i. Size: The first element to be considered is the size of the information system. For test point analysis purposes, the size of an information system is determined mainly by the number of function points assigned to it. However, a number of additions or correction need to be made, because certain factors which have little or no influence on the number of function points are pertinent to testing. The factors are followed:

- Complexity: - Complexity relates to the number of conditions in a function. More conditions almost always mean more test cases and therefore a greater volume of testing work.
- Interfacing: - The degree of interfacing of a function is determined by the number of data sets maintained by a function and the number of other functions, which make use of those data sets. Interfacing is relevant because these "other" functions will require testing if the maintenance function is modified.
- Uniformity: - It is important to consider the extent to which the structure of a function allows it to be tested using existing or slightly modified specifications.

ii. Test strategy: During the development or maintenance phase of software product the quality requirements will have been specified for the information system. The test activities

must allocate the extent to which these requirements have been satisfied. The system or subsystem quality characteristics to be tested are identified and lay down their relative importance [10]. The importance of each characteristic impact the thoroughness of the related test activities. The more essential a quality features, the more strait and thorough the tests have to be and the greater the volume of work. The importance of the various features should be determined in inference with the client when the test strategy is being formulated; the information can then be used as TPA input. In the TPA process, the volume of testing is calculated on the basis of the test strategy.

iii. Productivity: Productivity is not a new concept to anyone who has produced estimates on the basis of function points. In FPA, productivity is an expression of the relationship between the number of hours essential for a task and the measured number of function points. In TPA, productivity relates to the time necessary to realize one test point, as determinate by the size of the information system and the test strategy [9]. Productivity has two components: a productivity figure and an environmental factor. The productivity figure is based on the knowledge and skill of the test team, and is therefore specific to the individual organization. The environmental factor indicates the degree to which the environment influences the test activities to which the productivity is related. Influential environmental considerations comprise the availability of test tools, the amount of experience the team has with the test environment, the quality of the test basis and the availability of test ware.

IV. CONCLUSION

This paper focus on the existing software test effort estimation techniques. Also, presented background information on software metrics to be used for effort and cost estimation. Effort estimation is a complex activity that requires knowledge of a number of key attributes. Conventional estimation techniques focus only on the actual development effort furthermore; this paper also described test effort estimation. In fact, testing activities takes 40-60% total software development effort. Hence, test effort estimation is important part of estimation process. In future work, meta-heuristic techniques are apply on these test effort estimation techniques.

REFRENCES

[1] S. Aloka, Peenu Singh, Geetanjali Rakshit, and Praveen RanjanSrivastava, “Test Effort Estimation-Particle Swarm Optimization Based Approach”, Springer-Verlag Berlin Heidelberg, CCIS,2011.
[2] Garima, KailashBahl, “Software Project Estimation Techniques - Effort and Cost”, IJISET - International Journal of Innovative Science, Engineering & Technology,2014.
[3] Xiaochun Zhu, Bo Zhou, Li Hou, Junbo Chen, Lu Chen, “An Experience-Based Approach for Test Execution Effort Estimation”, 9th International Conference for Young Computer Scientists, IEEE,2008.
[4] KhaledHamdan, Hazem El Khatib, KhaledShuaib, “Practical Software Project Total Cost Estimation Methods”, MCIT 10, IEEE,2010.
[5] VahidKhatibi, Dayang N. A. Jawawi, “Software Cost Estimation Methods: A Review”, Journal of Emerging Trends in Computing and Information Sciences,2011.
[6] M. Jorgensen and M. Shepperd, “A systematic review of software development cost estimation studies”,IEEE Transactions on Software Engineering,2006.
[7] Eduardo Aranha, Paulo Borba, “An Estimation Model for Test Execution Effort”, 1st International Symposium on Empirical Software Engineering and Measurement,2007.
[8] F.S. Gharehchopogh, I. Maleki, S. Sadouni, “Artificial Neural Networks Based Analysis of Software Cost Estimation Models”, MAGNT Research Report,2014.
[9] E. Khatibi, R. Ibrahim, “Efficient Indicators to Evaluate the Status of Software Development Effort Estimation inside the Organizations”, International Journal of Managing Information Technology (IJMIT)2012.
[10] J.S. Chou, C.C. Wu,“Estimating Software Project Effort for Manufacturing Firms”, Computers in Industry, Vol. 64, pp. 732-740, Elsevier B.V.,2013.
[11] Srivastava, P. R., Varshney, A., Nama, P., & Yang, X. S., “Software test effort estimation: a model based on cuckoo search”. International Journal of Bio-Inspired Computation, 4(5), 278-285, 2012.
[12] PriyaChaudhary and C.S. Yadav, “An Approach for Calculating the Effort Needed on Testing Projects”. International Journal of Advanced Research in Computer Engineering & Technology,2012.
[13] Srivastava, P. R., Bidwai, A., Khan, A., Rathore, K., Sharma, R., & Yang, X. S. , “An empirical study of test effort estimation based on bat algorithm”,International Journal of Bio-Inspired Computation, 6(1), 57-70,2014.

