Available online www.jsaer.com

***Journal of Scientific and Engineering Research*, 2019, 6(11):300-304**

Review Article

ISSN: 2394-2630
CODEN(USA): JSERBR

Overview on Software Test Estimation Techniques

Bhupinder Paul Singh Sahni

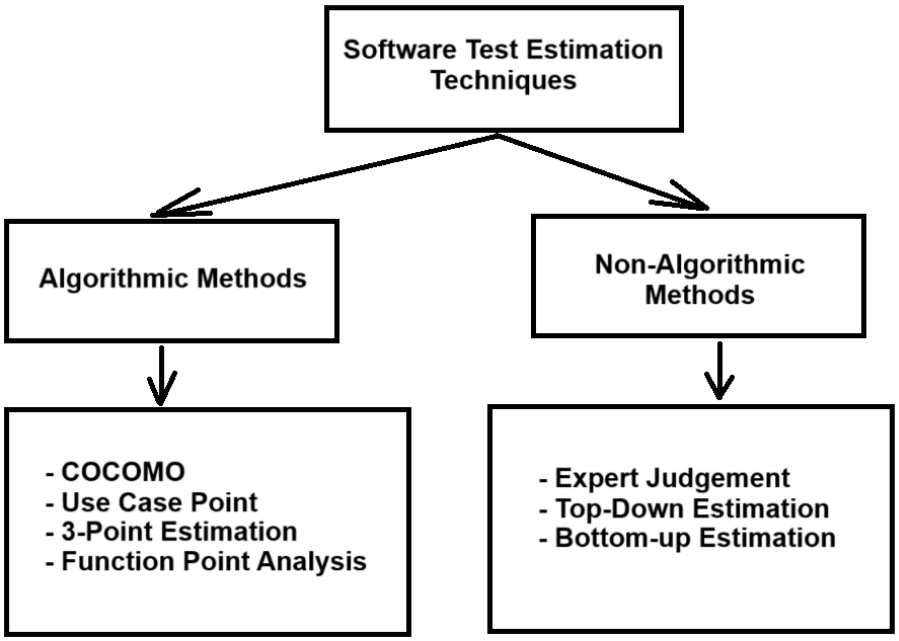
Email id: bhupinder.sahni@gmail.com

Abstract Software Testing is an important process of software development that is performed to support and enhance reliability and quality of the software. Using the correct Test Estimation technique and doing the Test estimation correctly is the key challenge Software Organizations face. There are various different ways through which we can estimate the testing efforts. This Paper presents an overview of different types of Algorithmic and non-algorithmic methods of Test estimation techniques as well as what kind of Projects each of these estimation techniques can be used. Every technique has its advantages and disadvantages. No one technique fits every Project. Depending on various factors like the project timelines, cost required, skills, tools, etc, a particular test estimation technique can be considered.

Keywords Software Testing; Testing Estimation; Software Estimation; Use Case Point; COCOMO; Function Point Analysis; 3-point Estimation.

Introduction

As more and more complex software is being developed, Software Testing is also becoming an important phase in the overall software development process. Multiple types of Testing like component testing, integration testing, system testing, functional testing, performance testing, and security testing are required to get a Bug free, reliable and robust Product into the market [1]. So, it becomes important that relevant estimation techniques are being used, depending on the type of testing required. Also, depending on the information available, cost and clarity of the Project, different Test estimation Techniques can be used to give a fair estimate of the testing tasks. Test estimation techniques have often been derived from generic software development estimation techniques, in which testing figures are as one of the phases of the software development life cycle.



Accurate estimates are not only beneficial to software engineers but to other stake holders as well. A good software estimation method leads to optimal utilization of resources and improves the quality of the end-product which results in higher levels of customer satisfaction levels. The incorrect estimation of testing effort often leads to an inadequate amount of testing, which, in turn, can lead to failures of software systems once they are deployed in organizations. Estimation is the most critical activity in software testing.

In this paper, we will be discussing different Algorithmic and non-Algorithmic methods used for doing Test Estimations. Every Test technique has advantages as well as disadvantages, but any can be used based on the overall requirement, clarity and cost of the Project.

Algorithmic Methods of Test Estimation Techniques

There are various Algorithmic methods of Test Techniques used to estimate efforts required for Testing a Project. Algorithmic test techniques involve the use of algorithms or systematic procedures to design, execute, and analyze tests. Depending on the nature of the system under test and the testing objectives, different techniques or combinations of techniques can be used. We will study some of the commonly used Algorithmic methods of test techniques.

A. COCOMO: COCOMO (Constructive Cost Model) is one of the algorithmic approaches to estimate software cost. It uses mathematical formulas and calculations to estimate the cost of a project. COCOMO provides a structured approach for estimating the effort required for various activities in software development, including testing.

COCOMO is made up of a hierarchy of three progressively in depth and exact forms [2]:

Fundamental Model
Transitional Model
In-depth Model

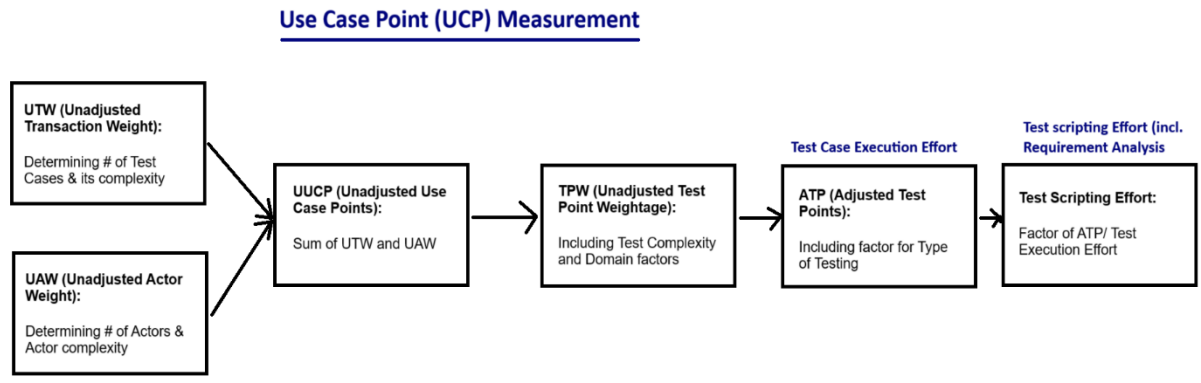
Fundamental COCOMO estimates the effort for small to medium sized software projects. It uses a simple formula based on the size of the software product measured in lines of code (KLOC) and a set of cost drivers. The COCOMO cost drivers are grouped into four categories: product, computer, personnel, and project attributes. The number, nature, and numeric ranges of significant cost drivers may vary from organization to organization [3].

Transitional Model is an extension of Fundamental COCOMO model and is suitable for estimating medium to large-sized software projects. It incorporates additional factors such as product complexity, development environment, and team experience.

In-depth COCOMO model is the most comprehensive variant and is suitable for estimating the effort required for large and complex software projects. It considers a wide range of factors like resource skills, project attributes, and hardware attributes [4].

COCOMO model is a common test estimation model as this model is flexible to changes [5]. It has a structured approach to software cost estimation. But COCOMO is a complex model, especially the In-depth model which considers numerous project factors and attributes. This complexity may make it challenging for inexperienced users to apply effectively.

B. Use Case Point (UCP): Use Case point (UCP) method is one of the commonly used size estimation methods in software development. The use case points method is a software sizing and estimation method based on use case counts called use case points [6]. It considers factors such as complexity, test case types, actors and environment dependencies. It starts with measuring the functionality of the system based on the use case model in a count called Unadjusted Use Case Point (UUCP) [7]. To calculate UUCP, parameters like Unadjusted transaction weight (UTW), Unadjusted actor weight (UAW) and Actor Complexity are calculated. After the calculation on UUCP, Test Points are calculated, which helps in determining Test Execution effort, Test Scripting effort and ultimately total testing effort required for a Project.



Use Case Points Estimation Technique is commonly used where project estimation is done based on the use cases. The advantage of using Use Case Point Estimation is that it can be measured early in the project life cycle after making suitable assumptions. Therefore, the onus lies on how well an estimator has understood the system which will ensure correctness, completeness and accuracy of use cases [6]. UCP is easy to use and does not call for additional analysis.

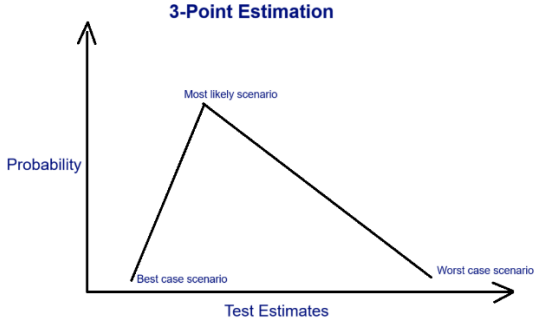
However, UCP can be used only when requirements are written in the form of use cases. Also, If the use cases are not well or uniformly structured, the resulting UCP may not be accurate. UCP is useful for initial estimate of overall project size, but they are much less useful in driving the iteration-to-iteration work of a team.

C. point Estimation: 3-Point Estimation technique is also known as PERT (Program Evaluation and Review Technique) considers 3 possible scenarios:

Best Case Estimate
Most Likely Estimate
Worst Case Estimate

Tasks are divided into smaller sub-tasks, and then each sub-task is estimated using 3-Point estimation. The estimates are found out by applying these values to the formula given below [8]:

Estimate = [(Best Case Estimate) + {4 * (Most Likely Estimate)} + (Worst Case Estimate)] / 6



The formula weighs heavily on the most likely estimate, putting emphasis that the most likely estimate would be closer to the actual estimate.

3-Point Estimation technique is used especially when there is uncertainty involved, and thus considers range of estimates, and not limits to a single point estimate. This helps in managing Project risks effectively. Since it gives a range of estimates, 3-Point estimation technique is flexible with the change in Project requirements.

However, for large and complex Projects, 3-Point estimation technique might not be useful because of the need to create tasks and subtasks to calculate 3-Point estimation might be very time consuming.

D. Function Point Analysis: Function Point Analysis (FPA) is a technique used for estimating the size and complexity of software based on the functionality provided to the end-user. The estimation is done in terms of function points. It is used for measuring complexity and size of software [9]. The idea of function points is assessing the size of the system in functions form they perform [10]. FPA refers to the standardized methods of determining software sizes by using functional constraints which determine the key features to be designed [11].

The first stage of function point analysis concerns identifying the forms of transactions to be made in the software applications. Secondly, the Testers evaluate and appraise the components of the software system. Lastly, the process involves evaluations of the general system characteristics.

FPA methods are universal and independent of programming language [12]. Size of the project is determined without the need for an analysis that considers the technology used [13]. FPA also has limitations such as taking into consideration the individual functionalities of aggregate elements do not allow for a complete reflection of the whole project complexity.

Non-Algorithmic Methods of Test Estimation Techniques

Having gone through some of the Algorithmic methods of Test Estimation techniques, now let's consider some of the non-Algorithmic Estimation techniques.

A. Expert Judgement: Expert judgment is a test estimation technique that relies on the knowledge, experience, and insights of domain experts or experienced professionals to estimate testing efforts, resources, and timelines. It involves getting input from individuals who have relevant expertise in the specific domain or technology being tested.

Expert Judgement requires consultation and collaboration with experts in their fields, having brainstorming sessions and then coming to a consensus to estimate for the Testing Project being

considered. These experts can be Software Testers, Developers, Project Managers or even Domain experts.

Since this Estimation Technique relies on external experts for estimation, there can be individual perspectives and experiences which may lead to different estimates from different experts on the same Project. Expert Judgement can be useful for small scale Projects, but not suitable for medium and large scaled Projects.

B. Top-Down Estimation: Top-Down estimation approach focuses on estimating the cost of a project from the overall properties of the project without any detailed project information. In this Estimation approach, existing knowledge of similar projects is used to calculate the overall timeline of the Project.

In top-down estimating, initial time estimates come from the opinions of experienced Testers in the field, who have done similar Projects in the past. These initial estimates are then modified to account for the characteristics or aspects of the current project.

This Test estimation technique is more beneficial while the project is still in its early stages and the planning horizon is quite long. There is no need for detailed information about the project, so it is used for high-level estimates. The major disadvantage is that the ballpark estimate is very inaccurate [14].

C. Bottom-Up Estimation: This is the exact opposite of the top-down estimation methodology. In this technique, the cost of every component of the software is derived and then the result is obtained by combining these elements to get the total estimated cost of the project [14]. The aim of this estimation technique is to obtain a proper estimate that will be an accumulation of the estimates of the smaller components of the software.

Bottom-up estimation technique is used when the project team identifies all the test components and determines the lowest level component to derive estimates.

This Test Estimation Technique gives better estimates than Top-Down Estimation, but also has some disadvantages. The accuracy of the estimation is high, but it can also take a lot of time to create this level of estimate, which means that the estimation can be expensive to create. Also, all the components estimates are added to give the overall Test Estimate, but all the components could be having dependencies that one activity has to be finished before other can start. So, all these factors need to be considered in order to have a reliable estimate.

Conclusion

There are various Test Estimation Techniques that can be used Testing efforts of a Software Project. Each estimation technique has advantages as well as disadvantages. The technique that can be used for one Project may not necessarily be extended to another one. Test estimation technique is essential for effective project planning, resource allocation, and risk management in software development and in Testing. So, depending on the Size of the Project, cost involved, variables known, time available, etc. test estimation techniques should be used with thoughtful consideration.

References

[1]. ISO, "ISO/IEC DIS 29119 - 1, 2, 3, & 4: Software and Systems Engineering - Software Testing -Part 1, 2, 3, & 4," International Organization for Standardization (ISO), Geneva, 2012.

[2]. Musilek, Pedrycz & Sun, (2002), "On the Sensitivity of COCOMO II Software Cost Estimation Model". Proceedings of the Eighth IEEE Symposium on Software Metrics (METRICS.02) 0-7695-1339-5/02.

[3]. R. Fairley, "Recent Advances in Software Estimation Techniques", Proceedings of the 14th international conference on Software engineering, pp. 382-391, 1992. doi: 10.1145/143062.143155

[4]. K Chauhan, M. Gupta, "COCOMO Estimates Using Neural Networks", MECS, 2012.

[5]. S. Chirra, H. Reza, "A Survey on Software Cost Estimation Techniques", Journal of Software Engineering and Applications, vol. 12, no. 6, pp. 226-248, June 2019. doi: 10.4236/jsea.2019.126014

[6]. M. Kirmani, A. Wahid, "Use Case Point Method of Software Effort Estimation: A Review", International Journal of Computer Applications, vol. 116, no. 15, pp. 43–47, April 2015.

[7]. P. Jena, S. Mishra, "Survey Report on Software Cost Estimation using Use Case Point Method", International Journal of Computer Science & Engineering Technology, vol. 5, no. 4, pp. 280–287, April 2014.

[8]. M. Maniyar, M. Hakeem, M. Zafar, "Effort Estimation for Performance Testing of Software Applications", International Journal of Innovative Research in Computer and Communication Engineering, vol. 6, no. 12, pp. 9557-9563, Dec 2018.

[9]. A. Srivastava, S. Abbas, S. Singh, "Enhancement in Function Point Analysis", International Journal of Software Engineering & Applications (IJSEA), Vol.3, No.6, 2012.

[10]. M. Aljohani, R. Qureshi, "Comparative Study of Software Estimation Techniques", International Journal of Software Engineering & Applications (IJSEA), vol. 8, no. 6, pp. 39-53, Nov 2017.
[11]. G. Komal, P. Kaur, S. Kapoor, S. Narula, "Enhancement in COCOMO Model Using Function Point Analysis to Increase Effort Estimation.", International Journal of Computer Science and Mobile Computing, Vol. 3, 265-572, 2014.
[12]. T. Grzeszczyk, J. Pelszynski, "Possibility of Using Function Point Analysis in Project Evaluation", World Conference on Management Science and Human Social Development (MSHSD 2017), pp. 257-261, Dec 2017. doi: 10.2991/mshsd-17.2018.48
[13]. M. Basavaraj, C. Karthish, "Software estimation using function point analysis: Difficulties and research challenges.", Innovations and Advanced Techniques in Computer and Information Sciences and Engineering, pp. 111-116, 2007.
[14]. L Hareton, F. Zhang, "Software Cost Estimation. In: Handbook of Software Engineering and Knowledge Engineering", Emerging Technologies, World Scientific Publishing Co Pte Ltd., Singapore, vol. 2, pp. 307–324, 2002. https://doi.org/10.1142/9789812389701_0014