An Approach for Calculating the Effort Needed on Testing Projects

Priya Chaudhary and C.S. Yadav

Abstract— **Software testing effort estimation has always been an on-going challenge to software engineers, as testing is one of the critical activities of SDLC. Accurate effort estimation is the state of art of software engineering, as it is the preliminary phase between the client and the business enterprise. The credibility of the client to the business enterprise increases with the accurate estimation. The earlier test estimation is done, the more benefits will be achieved in the testing life cycle.**

This paper proposes an approach for estimating the size and efforts required in the testing projects using test case point. The proposed model outlines all major factors that affect testing projects. Covering all major factors helps to do a fair estimation using the proposed approach.

This proposal is technology independent and supports the need for estimating, project efforts management and forecasting quality deliveries. This approach would be used by the individuals who would want to have good testing effort estimation for any given project under test.

The computation of proposed test effort estimation involves least overhead as compared to others.

Index Terms—**Test Case Point, Test Cycles, Test Effort Estimation, Testing Efforts Calculation.**

I. Introduction

Software testing is an indispensable aspect, directly influencing the quality of the software. In order to carry out a systematic testing, it is absolutely necessary to predict the effort required to test the software. Hence this paper proposes an approach for calculating the efforts needed on testing projects using test point analysis. A number of the methodologies that have been proposed in the past are code based, but when we have the code, it is too late. Therefore, the effort requirement for the software can be minimized, if the computation of test effort can be done in early phases of software development lifecycle (SDLC).

In this direction, we have captured the test case points from test case preparation, execution of the SRS Document of the project being worked upon. [16].

As SRS acts as a verifiable document, hence, estimation of test effort based on SRS will be early warning, systematic, less complex, faster and comprehensive one.

Use case based methods [9, 10, 12, 13] are also derived from software requirements and considers actor ranking, use case points, normal and exceptional scenarios and various technical and environmental factors in order to compute test effort based on a constant conversion factor. The limitation with these methods is that, it depends on the prudence of analyst.

However on the other hand, code based test effort estimation methods [4, 7, 8, 11] first considers the code, generates the test cases, execution points, productivity factors and later computes the test effort. These methodologies are computation intensive and amount of rework also gets increased.

Cognitive complexity based test effort estimation measures [6, 17] computes the cognitive complexity of the software and co-relates the cognitive complexity with test effort estimation. The measure uses line of the code and their subsequent basic control structure. The measure uses a tedious calculation to finally arrive at test effort.

However, the proposed measure considers requirements written in standard IEEE 830:1998 [1] format in order to obtain effective and precise result. Hence the strength of the measure lies in computation of requirement based test cases from requirement engineering document prior to estimation of software test effort. This will lead to early defect detection which has been shown to be much less expensive than finding defects during integration testing or later.

II. Related Work

During the last few decades, various models, methods and techniques have been developed for the estimation of software test effort. This section presents a survey of some leading papers describing the work carried out so far, for the estimation of software testing effort. The work presented in this paper is based on SRS. The Software Engineering Standard Committee of IEEE computer society [1] presents the guidelines for the documentation of software requirements using IEEE 830: 1998 format. Antonio Bertilino [2] discusses software testing roadmap for the achievements, challenges and dreams for software testing. Symon [3] discusses the computation of function point for the estimation of size and cost of the software. The paper also considers technical and environmental factors. Boehm [5] discusses constructive cost model (COCOMO) and its various categories for the estimation of software cost using cost driver attributes. Eduardo

Priya Chaudhary, *Department of Computer Science and Engineering, U. P. Technical University / N.I.E.T., ,Gr. Noida , India*
C.S. Yadav, *Department of Computer Science and Engineering, U. P. Technical University / N.I.E.T., Gr. Noida , India*

Aranha [4] uses a controlled natural language (CNL) tool to convert test specification into natural language and estimate test effort. It also computes test size and test execution complexity measure. Zhou, Xiaochun [9] presents an experience based approach for the test suit size estimation. Aranha and Borba [8] discusses about test execution and a test automation effort estimation model for test selection. The model is based on the test specifications written in a controlled natural language. It uses manual coverage and automated test generation technique. Nageshwaran [13] presents a use-case based approach for test effort estimation and considers weight and environmental factors with a constant conversion factor for the computation of test effort. Zhu Xiaochunet. al [10] presents an empirical study on early test execution effort estimation based on test case number prediction and test execution complexity. Erika Almeida et. al. [12] discusses a method for test effort, based on use cases. It uses parameters like: actor ranking, technical and environment factor related to testing like test tools, input, environment, distributed system, interfaces etc. for the calculation of test effort.

Kushwaha and Misra [17] discusses CICM and modeling of test effort based on component of the shelf (COTS) & component based software engineering (CBSE).

Further the CICM is compared with cyclometric number. The measure is able to demonstrate that the cyclometric number and the test effort increase with increase in software complexity.

Sharma &Kushwaha [15, 16] discusses the improved requirement based complexity (IRBC) based on SRS of the proposed software and also presented an object based semi-automated model for the tagging and categorization of software requirements. Contribution of the researchers towards this issue and a roadmap showing the various approaches used for the test effort estimation is summarised in table 1.

Table 1.Commonly Used Measures.

S. No.	Researchers	Technique
1	Antonio Bertilino	Software testing Roadmap
2	Symon	Function Point
3	Boehm	COCOMO
4	Eduardo Aranha	Controlled Natural Language(CNL)
5	Zhou, Xiaochun	Experienced based Approach
6	Aranha and Borba	Test execution & test automation effort estimation model
7	Nageshwaran	Use Case based Approach
8	Erika Almeida	Method for test effort based on Use Case
9	Kushwaha and Misra	Components of The Shelf(COTS) & Component Based Software Engineering(CBSE)

III. PROPOSED APPROACH

Lot many traditional techniques exist for estimating efforts on a project for either development or testing. All techniques take into account user experiences, past history and data to generate estimates, there by estimates differ because user experiences and past historydiffer from individual to individual across the globe. Another requirement lies where industries have really good estimation techniques for development but no concrete model exists for testing, though techniques exist.

In development there is a fairly good model which comprises of all the attributes that are sufficient to compute function point analysis (FPA) [3] for any software. The function point measure includes five parameters i.e. external input, external output, interfaces, file and enquiry. These parameters act as the bases to measure the size of the software for the estimation of function point. But, black box testing does not contain knowledge on software intrinsic qualities like functions, code, modules etc.

Proposed model suggest that:

1. Requirements can be used for estimation on which functions or modules are created in development.
2. External inputs in function points can be replaced by test case points (generated on the basis of requirements and the test cases intended to build around requirements.)
3. Development attributes like External inputs, external outputs, external queries etc. Can be replaced by testing attributes such as number of interfaces being touch by the particular test case covering any requirements, number of validations being made.

TCP analysis is an approach for doing an accurate estimation of functional testing projects. This approach emphasizes on key testing factors that determine the complexity of the entire testing cycle and gives us a way of translating test creation efforts to test execution efforts, which is very useful for regression testing estimation. TCP analysis generates test efforts for separate testing activities. This is essential because testing projects fall under four different models: Test Case Generation, Automated Script Generation, Manual Test Execution, and Automated Test Execution. Though the proposed approach focuses on Test Case Generation and Manual Test Execution.

TCP Analysis uses a 6-step process consisting of the following stages:

1. Identify Requirements
2. Calculate TCP for Test Case Generation
3. Calculate TCP for Test Case Execution
4. Calculate other related tasks (One time task & miscellaneous task)
5. Calculate Productivity Task
6. Calculate Total TCP

Given below is an overview of different phases

To determine the TCP for Test Case Creation, first determine the complexity of the Test Cases. Some test cases may be more complex due to the inherent functionality being tested. The complexity will be an indicator of the number of TCPs for the test case.

IV.

A. Calculate TCP for test case generations

Table 2. Calculate test case generation complexity based on the factors given below.

S.No	Test Case Generation Complexity Factors	Weights
1	The number of steps in the Test case Assuming that test cases are atomic and that they test only one condition, the number of steps will be an indicator of the complexity.	
2	Interface with other Test Cases This usually involves calling other test cases from this use case. Ex. Address Book lookup from Compose Mail	

Weight = (((Validation x 1+test steps x 2+ Interface x 3)/5) x 3)

Table 3.Calculating complexity weights.

S.No	Test Case Category	Complexity Factors			Complexity Weight
		Validation	Test steps	Interface	
1	Simple	2	3	0	5
2	Medium	3	5	0	8
3	Complex	4	8	2	16

Calculate Rough Test Case Points for Test Case Generation by using the below formulae:

Rough TCP-G = (Simple Test Cases X its complexity weight) + (Medium Test Cases X its complexity weight) + (Complex Test Cases X its complexity weight)

Table 4.Calculating the rough TCP for test generation.

S.No.	Feature/ Screen	Number of test cases			Test Case Point
		Simple	Medium	Complex	
		5	8	16	
1	Screen1	2	0	0	10
2	Screen2	0	6	0	48
3	Screen3	2	0	0	10
4	Screen4	3	0	0	15
5	Screen5	1	0	0	5

The TCP estimates above might be affected by certain application level parameters. This might increase the effort required. To factor the impact of such parameters, we will increase the TCP by an Adjustment Factor. Some of the Application level parameters that might have an influence on the TCP are listed in the table below:

Table 5. The Application level parameters that might have an influence on the TCP

Sl. No.	Factors Adjusted	Adjustment Weight
1	Not present or No affect	0
2	Incidental influence	0.05
3	Moderate influence	0.1
4	Average influence	0.25
5	Significant influence	0.5

The weights in the table are only indicative and will be subject to the application for which estimation is being carried out.

Table 6. Calculating adjustment b weights based on the test preparation factors.

Test Preparation Factors	Adjusted Weight
Domain Complexity	0
Integration with other devices	
Multi-lingual Support	
Re-usable scripts generated for execution	
Documentation quality	
Detailed report with screenshots	
Total Adjusted Weight	0

Adjustment Factor = 1 + Total adjustment weight

Each of these factors is scored based on their influence on the system being counted. The resulting score will increase the Unadjusted Test Case Point count. This calculation provides us with the Test Case Point Generation count.

TCP-G =Rough TCP -generation X Adjustment Factor

B. Calculate TCP for Test Case Execution

Table 7. Calculate test case execution complexity based on the factors given below.

S.No	Manual Execution Complexity Factors	Weights
1	Pre-conditions This usually involves setting up the test data. It also includes the steps needed before starting the execution. E.g. to test whether the printing facility is working fine, the pre-conditions would include opening the application, inserting a record into the database, generating a report and then checking the printing facility.	
2	Steps in the Test Case + If the steps themselves are complex, the manual execution effort will increase. Please refer to the legend below to determine the complexity factor.	

Weight = (((Test steps x 1+ Validation x 2+ Interface x 3)/5) x 3)

Table 8.Calculating complexity weights.

S.No.	Test case name	Test Case Category	Complexity Factors			Complexity Weight
			Test Steps	Validations	Precondition	
1	TC 1	Simple	3	2	0	4
2	TC 2	Medium	5	3	1	8
3	TC 3	Complex	8	4	2	13

Table 9.Calculating the rough TCP for test execution.

S.No.	Module name	Feature /Screen	Number of test cases with test type and weight				Test case point
			Simple	Medium	Complex	Highly Complex	
			4	8	13	0	
1	Module 1	Screen 1	2	0	0	0	8
2		Screen 2	0	6	0	0	48
3		Screen 3	2	0	0	0	8

Calculate Rough Test Case Points for Test Case Execution by using the below formulae:

Rough TCP-E = (Simple Test Cases X its complexity weight) + (Medium Test Cases X its complexity weight) + (Complex Test Cases X its complexity weight)

Test Cases need to be manually executed in all the scenarios. To arrive at the additional Test Case Points, the TCP -E (single scenario) will be added for every scenario for which manual execution is required.

The TCP estimates above might be affected by certain application level parameters. This might increase the effort required. To factor the impact of such parameters, we will increase the TCP by an Adjustment Factor. Some of the Application level parameters that might have an influence on the TCP are listed in the tables above.

The weights in the table are only indicative and will be subject to the application for which estimation is being carried out.

Adjustment Factor = 1 + Total adjustment weight

This calculation provides us with the Test Case Point Generation count.

TCP-E =Rough TCP – Execution X Adjustment Factor

C. *Calculate Other Tasks*

C.1. One time tasks (person hrs.)

- Consider Test Strategy preparation time
- Consider Environment setup time
- Consider time, frequency and duration of team meetings
- No. of days depending on prior knowledge, times no. of staff.
- Consider setup of testing support tool like bugzilla, TD etc.
- Regression Testing Efforts
- Browser Compatibility
- Additional Configuration

C.2. Miscellaneous task (person hrs.)

- Consider documents like weekly, progress reports preparation time
- Consider review time of different arti. facts (Typically 10% of Total Preparation effort)
- Consider rework time for document preparation (Typically 5% of Total Preparation effort)
- Consider retesting time for bug fixes (Typically 10% of Total Execution effort)
- Consider regression testing time
- Consider test data preparation time
- Consider database creation time

D. *Calculate Productivity Task*

Productivity sheet, lists out various domain/application type. Industry standard or Historical data of Productivity in hrs. /TCP is collected. These are indicative figures based on the platform of choice. This figure will become more consistent and reliable as we collect historical data from projects that get completed and feedback productivity figures into the system.

Table 10.Table representing PCB productivity standards for test generation.

Domain/ Type of application	Test Generation Productivity in TCP/person- hrs.
Oracle Application	
Web Application	2
Client Server Application	
Database	
Mainframes Application	
Installer Applications	3.00

Table 11. Table representing PCB productivity standards for test execution.

Domain/ Type of application	Test Execution Productivity in TCP/person-hrs.
Oracle Application	
Web Application	5
Client Server Application	
Database	
Mainframes Application	
Installer Applications	6.00

E. *Calculate Total TCP*

The Total TCP is computed by summing up the individual TCPs for Test Case Generation and Test Execution.

1. Test Preparation Productivity (TCP/person hrs.): Organisation Level Productivity in TCP/person hrs.[Refer the Industry Standard Productivity figures]
2. Test Execution Productivity (TCP/person hrs.): Organisation Level Productivity in TCP/person hrs.[Refer the Industry Standard Productivity]
3. One time task (person hrs.)
4. Miscellaneous task (person hrs.)
5. Test case Generation effort = (test case preparation x preparation adjustment factor)/ test preparation productivity.
6. Test case execution effort = (test case execution x execution adjustment factor)/ test execution productivity.
7. Calculate the system test cycles conducted.

F. *Effort Calculation*

To translate the Test Case Points into the total person months involved based on your prior experience estimate the number of test case points per person month.

1. Total execution efforts(In Person Hours)= total test case execution efforts+ total system test cycles conducted X total efforts
2. Total Project efforts(In Person Hours)= total test case preparation efforts+ total execution efforts+ onetime task+ miscellaneous task

The total TCP is indicative of the size of the testing project.

V. Conclusion

The proposed work computes the requirement based test effort estimation based on IEEE-830: 1998 standard of requirement engineering document, soon after freezing the requirements. The proposed measure is reliable because it is derived from SRS of the software to be developed. This makes the estimation precise and perfect. Since test case point estimates provides an approach for early estimation, if used properly, may help organization in better product/projects management and tracking. Early estimates can help organization to gain early knowledge about the schedule, project length and thereby early planning to reduce their cost and gain profits. It facilitates early detection and process improvements. It provides a better view to decision makers. Better planning and estimation can be of great help in effective utilization of resources.

VI. Future Work

There may be different solutions for a single problem. Likewise, in many of the available approaches to test effort estimation, the Test Case Point approach is one. To make this research a more robust method of estimation over a period of time, it is required to make use of the available data from past projects. This will definitely contribute to the accuracy of test case point estimates and their enhancements. The estimation model is not claimed to be exact, but the approach offers significant practical advantages over other estimation techniques currently in use. Further research and experimentation will definitely provide more benefits in arriving at a method to validate the estimates.

This research paper is based on the analysis of the literature available on the estimations common across industry. To be more specific for the results obtained, one should have seen similar result on large empirical data. Further work can be done in this direction by validating each estimate on large volume of empirical data so that there is no chance of error in effort estimations.

REFERENCES

[1] Software engineering standard committee of IEEE Computer Society. IEEE Recommended Practice or Software Requirement Specifications", IEEE Inc. NY, USA, 1998.

[2] Antonio Bertilino, Software Testing Research: Achievements, challenges and Dreams, IEEE Future of software Engineering-FOSE 2007

[3] Charles R Symons. Function point: Difficulties and Improvements, IEEE Transactions on Software Engineering, Vol.14, No.1, Jan. 1988.

[4] Eduardo Aranha, Filipe de Almeida, ThiagoDiniz, VitorFontes, Paulo Borba. Automated Test Execution Effort Estimation Based On Functional Test Specification. Proceedings of Testing:

Academic and Industrial Conference Practice and Research Techniques, MUTATION 2007.

[5] Barry Boehm, Cost models for future software life cycle processes, Annals of Software Engineering, Special Volume on Software Process and Product Measurement, Neitherlands, 1985

[6] Thomas J McCabe. A Complexity Measure. IEEE Transactions on Software Engineering, Vol., SE-2, No. 4, Dec. 1976.

[7] AjithaRajan, Michael W Whalen, Mats P. E. Heimdahl.2007. Model Validation Using Automatically Generated Requirements-Based Tests. 10th IEEE- High Assurance Systems Engineering Symposium, 2007.

[8] Aranha E, Borba P. Test Effort Estimation Model Based On Test Specifications. Testing : Academic and Industrial Conference-Practice and Research Techniques,IEEE Computer Society, 2007

[9] ZHU Xiaochun, ZHOU Bo, WANG Fan, QU Yi CHEN Lu. Estimate Test Execution Effort at an Early Stage: An Empirical Study", International Conference on Cyber World , IEEE Computer Society,2008

[10] Qu Yi Zhou Bo, Zhu Xiaochun. Early Estimate the Size of Test Suites from Use Cases", 15th Asia-Pacific Software Engineering Conference, IEEE Computer Society, 2008

[11] DAnielGuerreiro e Silva, Bruno T. de Abreu, Mario Jino. A Simple Approach For Estimation of Execution of Function Test Case. IEEE-International Conference on Software Testing Verification and Validation, 2009

[12] Erika R. C De Almeida, Bruno T. de Abreu, Regina Moraes. An Alternative Approach to Test Effort Estimation Based on Use Case. IEEE-International Conference on Software Testing Verification and Validation, 2009

[13] Suresh Nageshwaran, Test Effort Estimation Using USE CASE Points. Quality Week 2001, San Francisco,California USA, 2001

[14] The Standish group research for staggering bugs and effort, http://standishgroup.com

[15] Sharma Ashish, Kushwaha DS, A Complexity measure based on requirement engineering document, JCSE UK, May 2010

[16] Sharma Ashish, Kushwaha DS, NLP based component extraction and its complexity analysis, ACM Sigsoft,, Issue 36, January 2011

[17] Kushwaha DS, Misra AK, Software Test Effort Estimation. ACM Sigsoft, Software Engineering Notes, Vol. 33 , No. 3, May 2008