Test Effort Estimation Models Based on Test Specifications

Eduardo Aranha[1,2] and Paulo Borba[1]
[1]*Informatics Center – Federal University of Pernambuco*
PO Box 7851, Recife, PE, Brazil
{ehsa,phmb}@cin.ufpe.br

[2]*Mobile Devices R&D – Motorola Industrial Ltda*
Rod SP 340 - Km 128,7 A - 13820 000 Jaguariuna/SP - Brazil

Abstract

In competitive markets, test teams should be able to estimate the required effort to perform their test activities on the schedule. In this paper, we present a test execution and a test automation effort estimation model and their use for test selection.

All these models are based on the test specifications written in a controlled natural language. This characteristic may allow us to have better estimation accuracy, even for tests never executed before. The evaluation of these models and some additional benefits to industry are also discussed in the paper.

1. Introduction

In competitive markets (e.g., the mobile phone market), companies that release products with poor quality may quickly lose their clients. In order to avoid this, companies should ensure that product quality has conformed to its client expectation. A usual activity performed to ensure quality is software testing.

Software testing is been considered so important that organizations can allocate teams exclusively for testing activities. In such situations, test teams should be able to estimate the required effort to perform their test activities on the schedule and to request more resources or negotiate deadlines when necessary.

Several software development estimation models have been proposed over the years. Function Point Analysis (FPA) and COCOMO are examples of models used for estimating the effort to develop software products based on its characteristics. Regarding tests, Test Point Analysis is a model similar to FPA used for estimating the effort to define, develop and execute functional tests.

However, these models do not estimate the effort to execute or automate a given set of test cases, since their estimations are based on development complexity instead of execution complexity. For this reason, estimates are usually made only based on expert judgment and historical data, which generally leads to a lack of precision.

In order to estimate the execution or automation effort of a given test case or test suite, we propose test estimation models that regard test size and complexity measured from its specification written in a controlled natural language [6][12].

This paper is organized as follow. In Section 2, we present related works and explain why they are not appropriate for test effort estimation. Hence, Section 3 states our main research hypotheses. After that, Section 4 introduces a test specification language considered by this work. Then, Section 5 describes our proposed test effort estimation models and Section 6 shows how to evaluate them. Finally, we present some conclusions in Section 7.

2. Related work

During the last few decades, several models and techniques were created for estimating size and effort on software development. The surveys presented in [8] summarize the software estimation evolution so far.

Some of the related and renowned software estimation models are discussed here. The first model discussed here is Function Points Analysis (FPA) [4]. FPA gives a measure of the size of a system by measuring the complexity of system functionalities offered to the user. The size of system is determined in function points (FP), a unit-of-work measure, and this count is used for estimating the effort to develop it.

The Use Case Point Analysis (UCP) [9] is an extension of FPA and estimates the size of a system based on use case specifications. Both UCP and FPA regard the development complexity of a system, while

our proposed model regards the size and execution complexity of test cases.

The Constructive Cost Model (COCOMO) [3] converts size metrics such as FP and SLOC (source lines of code) into effort estimation for developing systems. Its formula uses effort multipliers and scale factors, and their values are defined according to the characteristics of the development environment, teams and processes used in the project.

Similar to UCP, Test Point Analysis [10] is a method for estimating the effort required to perform all functional test activities based on use case points. This model estimates the effort required for all test activities together, such as defining, implementing and executing all the tests. For example, it is not possible to estimate only the effort to execute test cases that were automatically generated.

In big software development companies, test teams are usually required to test different parts of one or more applications at the same time. In this context, we need a model to estimate the effort to execute any given subset of tests of any application in development.

3. Research hypothesis

We formulate our research hypotheses as presented next:

- The effort to execute tests can be accurately estimated based on the test specifications and risk factors.
- We can find the best test suite to execute by analyzing test coverage and execution effort.
- The effort to automate tests can be accurately estimated based on the test specifications and risk factors.
- The best test selections for automation are the ones that have higher manual execution effort and smaller automation effort when considering their expected frequency of execution.
- Test effort estimation models regarding the test specifications and risk factors are more accurate than models only based on historical data.
- It is possible to have test effort estimates automatically calculated with minimal human interaction when using estimation models based on test specifications and risk factors.

4. Test specification language

Test specifications are commonly written in natural language [5], leading to problems such as ambiguity, redundancy and lack of writing standard, which difficult not only the test execution, but also its complexity estimation.

However, the problems can be reduced using controlled natural languages. A controlled natural language (CNL) [12] is a subset of natural language with restricted grammar and lexicon in order to have sentences written in a more concise and standard way.

This restriction reduces the number of possible ways to describe an event, action or object. The test specifications considered by this work are written using a CNL described here. In a simplified way, each sentence (test step) in the specification conforms to the following structure: a main verb and zero or more arguments.

The verb identifies the action of the test step to be performed during the test. The arguments provide additional information about the action represented by the verb. For instance, the phrase "*start the message center*" has the verb start (action of starting an application) and the required argument the message center (application to be started).

The CNL can have its lexicon and grammar extended for specific application domains. For example, the list of possible verbs and arguments may be different between the mobile and the Web application domains.

The context of this work is related to testing mobile applications for Motorola Brazil Test Center site at the Informatics Center/UFPE. Hence, the considered controlled natural language in this paper reflects this domain [12] [6]. Although not essential, the use of CNL in test specifications is recommended when using the test estimation models presented next.

5. Test estimation models

The main goal of this research is the development and evaluation of estimation models for:

1. Manual test execution effort;
2. Test coverage vs. execution effort analysis;
3. Test automation effort;
4. Cost-benefit analysis for prioritizing manual tests to be automated.

The development of the first two models is already in progress. We aim to create the estimation models regarding the following characteristics:

- Accurate estimates;
- Based on the test specifications in order to be applicable to any given suite of functional test cases;

- Based on risk factors (team experience, test environment, tested product, etc.);
- Good performance for existing tests and for new ones;
- Easy of automation.

The details about these test estimation models are presented next.

5.1. Manual test execution effort

In this section, we present a test effort estimation model developed during our research [1][2]. As illustrated by Figure 1, the input of our estimation model is a test suite and the output is the estimated effort in man-hours required to execute all tests in the suite.

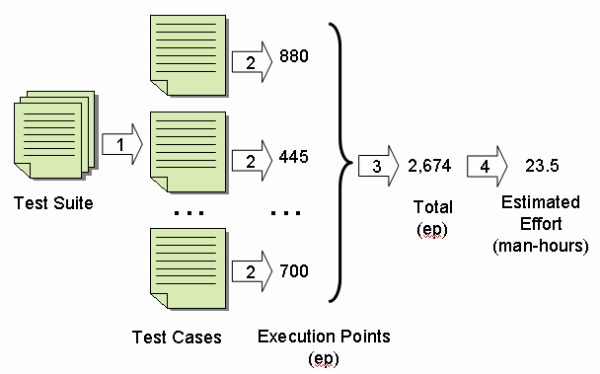

Figure 1. Overview of the test execution effort estimation model.

Our test execution effort estimation model works as follows. First of all, (1) we individually analyze the test cases existing in the suite. During these analyses, (2) we assign for each test case a number of execution points, a unit of measure defined in this work for describing the size and execution complexity of test cases. This number is useful for comparing test cases. For example, considering the same environment conditions, a test case assigned with 1000 execution points should require twice the effort needed to execute a test case assigned with 500 execution points.

After that, (3) we sum all the execution points measured from the analyzed test cases. This total describes the size and execution complexity for the whole test suite. Finally, (4) we estimate the required effort in man-hours to execute all tests in the test suite based on the total number of execution points. In this last step, we use test productivity or risk factors (similarly to COCOMO) to do the effort estimation.

Step 2 is also detailed in Figure 2. First, (a) we individually analyze each test step of the test specification. We analyze each test step according to a list of characteristics (C1 to Cn). Each characteristic considered by the model has impact in the size and execution complexity of the test and (b) this impact is rated using an ordinal scale (Low, Average and High).

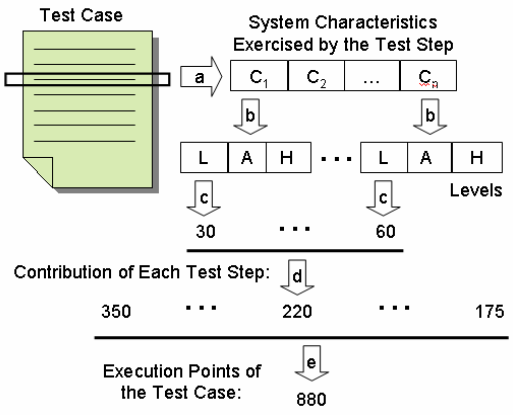

Figure 2. Step 2 in details.

After that, (c) we assign weights (execution points) for each characteristic according to its selected value. To calculate the total number of execution points of a test step, (d) we sum the points assigned for each characteristic. Then, (e) we measure the size and execution complexity of a test case by summing the execution points of each one of its test step.

Our proposed test execution effort model should be configured according to the target application domain in order to maximize the estimates accuracy. For instance, the list of characteristics (C1 to Cn) and their number of execution points assigned to use depend on the target domain. This model configuration is done by expert judgment or by running empirical studies.

5.2. Test coverage vs. test execution effort analysis

When executing tests, it is important to achieve the highest possible level of test coverage [13]. Nevertheless, in practice, the resources are limited and it may not be possible to execute all test cases as desired. Then, the tester has to select a subset of tests to execute.

Some times it is possible to know the contribution of each test case for the test coverage before executing them, such as when using some Model-based testing (MBT) approaches [11]. Using our proposed test effort

estimation model, it is also possible to know the effort to execute each test case individually.

In this context, we are investigating the ways to select the best set of tests to execute. For instance, the test selection can be supported by a tool that generates graphs such as the one shown in Figure 3. In this sample graph, each dot represents a test case. In general, tests with high coverage and low effort to execute are good candidates for selection.

5.3. Test automation effort

Another important and time-consuming activity is test automation. In general, automated tests are cheaper to execute, since they usually require few human interaction. For instance, a tester can control the execution of several automated tests at the same time. In addition, automated tests can be schedule to be automatically executed at night or on the weekends.

Nevertheless, test managers must consider the cost to automate the tests. For this reason, we intend to create an estimation model for test automation effort. This other proposed estimation model is also based on the test specifications written in CNL.

In this way, we need to identify the list of characteristics and risk factors that are relevant to estimate test automation effort.

5.4. Test automation vs. manual execution analysis

Although test automation has a lot of benefits, it may not be possible to automate all test cases. One of the main reasons is the cost of automation. The automation effort must be paid by its resulting benefits. In this research, we intend to create tools that combine the outputs of both test execution and automation effort estimation models and other relevant information in order to support test selection for automation.

For instance, the tester can select tests to automate by analyzing the effort to execute and to automate each test using graphs, as the sample shown in Figure 4. In addition, the model will regard other variables such as the expected test execution frequency and the available time to automate the tests.

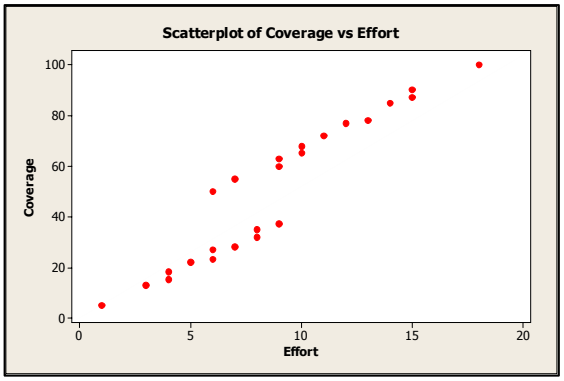

Figure 3. Test coverage vs. execution effort analysis.

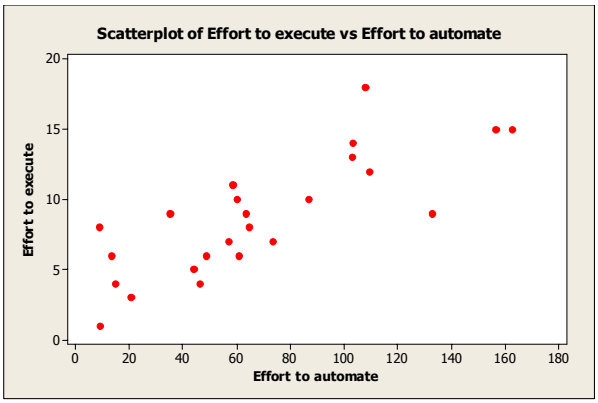

Figure 4. Test automation vs. execution efforts analysis.

6. Empirical studies

In order to evaluate the proposed estimation models, we plan to run some empirical studies. For the test execution estimation model, we have the following studies:

Expert judgment – the use of expert judgment through a Delphi panel to create a first version of the test execution effort estimation model. We already did this panel with six testers and we achieved a consensus in two to four rounds per question about the list of characteristics and risk factors to consider in the estimation model, as well their influence levels and weights according to the mobile application domain.

Experiment – almost 200 test cases of 3 different features were executed by 6 testers in a controlled environment. We evaluated the accuracy of the estimation model through a three-fold cross-validation.

Case study – the model will be used for 6 months by 4 test teams in context of the Informatics Center/UFPE and Motorola partnership. The achieved accuracy will be verified and compared to the one based on historical test productivity.

A similar sequence of empirical studies should be run in order to evaluate the test automation effort estimation model. The test coverage vs. test execution effort analysis and the test automation vs. manual

execution analysis will be evaluated during the planned case studies.

7. Conclusions

In this paper we presented our main research hypotheses related to test effort estimation models. These hypotheses are related to the research of estimation models for test execution and automation effort and prioritization.

We presented the estimation models for test execution and test automation effort estimations and their use for test selection.

All these models are based on the test specifications written in a controlled natural language. This characteristic support a better effort estimation to execute or automate any given set of test cases, even for tests never executed before.

The test execution effort estimation model was presented in more detail. We have started a sequence of empirical studies to evaluate it. The other models are at initial stages of research.

The expected results of this work have several additional benefits to industry. For instance, the use of our models also promotes the use of a standardized test specification language, reducing ambiguity and other types of problems.

Also, measures of test cases (such as the number of execution points) support a better capacity specification and testers productivity comparison. For instance, it is difficult to evaluate team capacity or tester productivity only based on the number of tests executed per day, since the tests may vary a lot in size and complexity. However, we can better state our capacity and compare test productivity using execution points, since it is a generic unit of work.

8. Acknowledgments

We would like to thank all anonymous reviewers who have helped us to improve this paper. The first author is partially supported by Motorola, grant BCT-0021-1.03/05, through the Motorola Brazil Test Center Research Project. The second author is partially supported by CNPq, grant 306196/2004-2.

9. References

[1] E. Aranha and P. Borba. An Estimation Model for Test Execution Effort. *International Symposium on Empirical Software Engineering and Measurement (ESEM 2007)*. September 20th and 21st, Madrid, Spain, 2007. To appear.

[2] E. Aranha and P. Borba. Measuring Test Execution Complexity. *2nd Intl. Workshop on Predictor Models in SE (PROMISE 2006)*. Philadelphia, Pennsylvania USA, 2006.

[3] B. Boehm, et. all. *Software Cost Estimation with COCOMO II*. Prentice Hall, 2000.

[4] D. Garmus, D. Herron. Function Point Analysis, Measurement Practices for Successful Software Projects. Addison Wesley, 2001.

[5] P. Jorgensen. *Software Testing, A Craftsman's Approach*. CRC Press, second edition, 2002.

[6] D. Leitão, D. Torres and F. Barros. Nlforspec: Translating natural language descriptions into formal test case specifications. M.Sc. Thesis, 2006.

[7] R. Mickey, O. Dunn, V. Clark. *Applied Statistics: Analysis of Variance and Regression*, third Edition, 2004.

[8] K. Molokken and M. Jorgensen. A review of surveys on software effort estimation. In *ISESE '03: Proceedings of the 2003 International Symposium on Empirical Software Engineering*, page 223. IEEE Computer Society, 2003.

[9] P. Mohagheghi, B. Anda, R. Conradi. Effort estimation of use cases for incremental large-scale software development. In *Proceedings of the 27th international conference on Software engineering* (ICSE'05), pages 303–311. ACM Press, 2005.

[10] S. Nageswaren. Test Effort Estimation Using Use Case Points. In *14th International Internet & Software Quality Week 2001*. June, 2001.

[11] A. Pretschner. Model-based testing. In *ICSE '05: Proceedings of the 27th Int. Conference on Software Engineering*, pages 722–723, 2005.

[12] R. Schwitter. English as a formal specification language. In *Proceedings of the 13th International Workshop on Database and Expert Systems Applications (DEXA'02)*, pages 228–232. IEEE Press, 2002.

[13] M. Whalen, A. Rajan, M. Heimdahl, and S. Miller. Coverage metrics for requirements-based testing. In *ISSTA '06: Proceedings of the 2006 international symposium on Software testing and analysis*, pages 25–36. ACM Press, 2006.