

Test Execution Effort and Capacity Estimation

Eduardo Aranha^{1,2}, Paulo Borba¹
{ehsa, phmb}@cin.ufpe.br

¹*Informatics Center
Federal University of Pernambuco
PO Box 7851, Recife, PE, Brazil*

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

Abstract

A usual activity performed to ensure software reliability is functional testing, where organizations can allocate teams exclusively for testing. With tight schedules and limited resources, test managers shall be able to estimate the effort required to execute the test suites requested by their clients (usually development teams), and also to estimate its capacity for attending the requests.

Several models and techniques were created for estimating software development size and effort. Some renowned examples are Constructive Cost Model and Function Point Analysis. However, these models cannot be used for estimating the effort to execute a given test suite, since they are based on system characteristics instead of test characteristics.

In this context, we developed a model for estimating test execution effort based on the execution complexity of test cases written in a controlled natural language (CNL). A CNL is a subset of natural language with restricted grammar and lexicon, where the sentences are written in a more concise and standard way.

In a simplified way, each test step of a test case conforms to the following structure: a verb identifying the action of the test step and arguments providing additional information. Examples of test steps in CNL for the mobile phone domain are “*Launch the message application*” and “*Select the inbox menu item*”.

This natural controlled language can have its lexicon and grammar extended for specific application domains. In this way, the list of verbs (actions) and arguments may be different between the mobile and the Web application domains, for instance.

In order to estimate the effort required to execute test cases, we first evaluate their execution complexity. For example, creating a message is more complex than reading it. Regarding test cases written in CNL, we can evaluate their execution complexity based on their specification. We evaluate the execution complexity of a test case by performing the following activities for each test step defined in the specification:

- Evaluate the complexity level for executing the action described by the test step based on the functional and non-functional system characteristics exercised by it.
- Rate the execution complexity level in execution points, a quantitative measure of execution complexity defined by this work.

Examples of functional characteristics are the number of navigations between screens, average number of pressed keys and the number of items to verify during the test. Regarding the non-functional characteristics, we have characteristics as system performance and the use of network.

For each characteristic, we evaluate it as low, average or high complexity. Each complexity value has assigned a specific number of execution points. Guidelines were defined through expert judgment (Delphi panel) for evaluating and rating the complexity levels. Finally, the execution complexity of a test case is measured by the sum of the execution points of each of its test steps.

After evaluating the execution complexity of each test case, we estimate the effort in man-hours required to execute them. In a stable environment, we can do these estimations multiplying the total number of execution points by the historical test productivity. Here, we consider test productivity as the average proportion between execution time and number of execution points of each test case (time spent per each execution point).

Nevertheless, team experience, tools and other environment conditions usually change over time. In these situations, these changes are represented by linear and scalar factors that adjust the estimations, similarly as done in COCOMO. These factors are defined according to its influence in the test execution time using multiple regression analysis and expert judgment. Examples of these factors are test experience, use of tools, software stability, etc.

We also use this effort estimation model for estimating the capacity of test execution teams. The execution capacity considered here is determined by the number of test suites executed a day. Since the characteristics of these tests change (and consequently its execution times), we use the simulation technique for generating test suites with different characteristics and then estimating the effort to execute them. After that, we use the generated estimations with team manpower information to verify the team capacity, the probability to achieve a desired capacity and others additional information.

The use of a controlled natural language reduces the ambiguity helping the complexity measurement. Actually, the number of possible ways to describe the same test step in a controlled language is minimal, and we also observed that a small but concise controlled language can support a high number of different test cases.

Based on these considerations, the method for measuring test execution complexity can be optimized as follows. The complexity evaluation of test steps are recorded and reused whenever possible in the complexity evaluation of other test cases. Over the time, the number of necessary evaluations tends to decrease. With these optimizations and the possibility to automate all steps of the method, except the test step complexity evaluation, the costs for applying this method are significantly reduced.