Model Simulation for Test Execution Capacity Estimation

Eduardo Aranha[1,2]
ehsa@cin.ufpe.br

Paulo Borba[1]
phmb@cin.ufpe.br

José Lima[2]
Jose.Lima@motorola.com

[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

It is important for a company to know if it is able to attend the clients' requests on time and according to their expectations, mainly in competitive markets. A usual activity performed to ensure quality is the test activity. Software testing is been considered so important that organizations can allocate teams exclusively for test activities.

In this context, it is important that test managers know the capacity of their test team. This paper defines a capacity model for test execution teams and presents the use of simulation technique for estimating teams capacities for executing tests in mobile applications. We also report our experiences using this model in an organizational setting.

1. Introduction

The quality of a company is not defined just observing the quality of its products. There are several other characteristics that influence on how the clients perceive the company quality. An important one is the company's ability to attend client requests on time. Therefore, it is important for a company to know if it is able or unable to attend the clients' requests according to their expectations.

Functional test execution, for example, is an important activity to ensure product quality. Since it is generally done at the end of the development cycles, it should not be a bottleneck in the process. Actually, organizations can allocate teams exclusively for test activities. In particular, test execution teams should be properly planned in order to be able to execute the necessary tests on the requested time and without having idle resources.

Several software development estimation models have been proposed over the years, such as the Constructive Cost Model (COCOMO) [1] and the Function Point Analysis (FP) [2]. However, none of these models estimate specifically the required effort for test execution activities, but generally for the whole development effort.

These models also estimate the required effort based on the product characteristics. Nevertheless, we usually do not know previously these characteristics, so as we cannot use these models alones to estimate a team capacity. A solution for these cases can be simulating possible situations and verifying the team performance [4].

In this paper we define a capacity model, present the simulation technique and use them to estimate the capacity of test execution teams. In Section 2, we define a capacity model for test execution teams. Section 3 presents the capacity model simulation. Then, Section 4 presents the use of our capacity model in an organizational setting. Finally, we present related works in Section 5 and our conclusions in Section 6.

2. Test execution capacity model

During the work presented here, we wanted to investigate the current capacity of test teams for executing test cases. We measure the capacity of a test execution team by the number of test cycles that can be executed per day. A test cycle is a group of test cases selected according to some criteria. For example, a test cycle can be created to group all tests required to cover a feature that has been modified.

The capacity of an execution test team considered here is determined by the number of test cycles executed per day. In a test cycle, the number of tests and the time required to execute them are usually different and unknown. This fact makes difficult the capacity estimation.

In order to calculate the capacity of a test execution team for executing test cycles, we divide its available manpower (man-hours per day) by the effort required to execute a test cycle (hours per test cycle).

$$Capacity = \frac{ManpowerPerDay}{EffortPerCycle}$$

The result of this expression is the number of test cycles executed per day. The available manpower per day for executing tests is obtained multiplying the number of testers by its working hours per day:

$$ManpowerPerDay = NumberOfTesters \bullet WorkingHoursPerDay$$

The number of testers is usually a constant, as well the working hours by each tester. For obtaining the effort required to execute a test cycle, we sum the time spent (in hours) in environment setup and test execution activities:

$$EffortPerCycle = EnvSetupEffort + TestExecutionEffort$$

The effort required to execute a test cycle is obtained multiplying the team productivity (average time in hours required to execute a test case) by the number of test cases in the test cycle:

$$TestExecutionEffort = TeamProductivity \bullet NumberOfTestCasesPerCycle$$

In this model, we do not detail how the environment setup effort is calculated. When creating a model, the level of details to be used should be established carefully. More details do not necessarily mean precision improvement. When detailing an activity, you can loose precision when forgetting to include some internal steps that directly affect the total effort or when the process is not clearly defined.

The cost to measure the effort of internal steps should also be considered before deciding to include them in the model. SIPOC diagrams, top-down charting and other process mapping techniques [8] can support a better understand of the process and its inputs in order to build new models.

With the presented model, we can estimate the capacity of a test execution team. However, in order to use this model, we must know the team productivity, the number of tests per cycle and the time required to setup the test environment. These values are not constants for every test cycle. The team productivity and test environment setup, for instance, can change according to environment characteristics that also change over time, such as network, time pressure, etc.

If you have historical data, you can analyze it and extract some representative measurements of the data. The most common and easy to calculate is the mean. Unfortunately, some times the means do not represent a data set appropriately as we wish, especially when the data are not symmetric.

In addition, a worst situation occurs when you do not have historical data. In this case, you have to guess all these values. In the next section, we overview model simulation, a technique used to minimize the problems listed here, supporting the choice of the more appropriate values to be used as input for our model.

3. Model simulation

A model is an abstract representation of a system that shows the relations of system inputs and outputs. Model simulation is a useful technique to simulate real situations (model inputs) in order to verify the system behavior (model output) in those situations. Model simulation is commonly used to predict the performance of a system or to compare system alternatives [4]. In particular, the Monte Carlo method [3] is used for simulating a model using probabilistic distributions.

When defining a model, some of its inputs may have fixed values, such as the number of days per year. For all others inputs, we must indicate what are its possible values and their probability to occur using probabilistic distributions. A probabilistic distribution is a function that assigns probabilities to events occur [3]. In our case, the probability that each input values occurs.

The normal distribution is the most common probabilistic distribution, where we have the mean and a standard deviation delimiting most of the values variation. Most of time, defining the probabilistic distribution of a given input is not an easy task. It can be done by intuition or using tools to fit the most appropriated distribution. The skewness and the kurtosis, for example, are measures that respectively indicate the lack of the data symmetry and whether the data are peaked or flat relative to a normal distribution. For instance, a normal distribution should have skewness near zero and a kurtosis higher than 3.

When using intuition instead of data analysis, there are several probabilistic distributions defined with their applicability. For instance, the Poisson distribution [9] can be used to express the probability of a number of events (software defects, for example) occurring in a given time interval. On the other hand, when using the statistical tools, a set of historical data is required.

The simulator works as follow. For each input, a random value will be generated according to its probabilistic distribution, representing a possible situation. After that, the model will generate output values, called forecasts, as a response for the simulated input values. This process is repeated by a configured number of times or until achieving a statistical

confidence level for its results. Finally, the simulator is ready to present charts and other information about the generated values.

The next sections present the simulation of our test execution capacity model and a test based on historical data.

3.1. Simulating the test execution capacity model

In our study, we estimated and validated the capacity of one test execution team. During this work, we selected a simulator tool and configured it with our model. For inputs with fixed values, such as the working hours per day and the current team size, we simply set these fixed values. For the other inputs (number of test cases per cycle, team productivity, and environment setup effort), we had to analyze them and define their probabilistic distribution.

For the number of test cases per cycle, team productivity, and environment setup effort, we defined their probabilistic distribution based on historical data. We had access to 5 months of test execution data. We separated this historical data into two parts. The first 3 months of data was used to define the probabilistic distributions of each input, and the last 2 months of data we used to calculate the real achieved capacity in order to test the estimated capacities generated by the model simulation. A more elaborated cross-validation technique [6] could be used to avoid bias, but we did not use it due to its cost and since we verified a stable process during the analyzed period.

During the analysis of the historical data, we verified its precision and consistence, keeping attention on the outliers [8], observations with values very different from the rest of data. When an outlier is an event that is not part of the normal process, it must be removed from the analysis. The process should also be improved in order to remove or minimize the chance of this special cause occurring again.

After we verified the data consistence and removed outliers, we used the simulation tool to fit and test the more appropriated probabilistic distribution. The team productivity and the environment setup effort fitted the normal distribution. The number of test cases per cycle, in its turn, fitted the lognormal distribution [9]. As these distributions were found by analyzing real measures, we just validated them graphically using expert intuition.

We also have to define the forecasts, representing the values we want to estimate. In our model, only the capacity variable was defined as a forecast, since it is the information we are interested in estimating. Finally, we set the simulation to run at most 5000 trials and to use a confidence level of 95%. When achieving this confidence level, running more trials will probably not change significantly the results. We then started the simulation and analyzed the results.

In Figure 1, we can see the values generated by the simulator tool for the productivity variable, according to its defined probabilistic distribution. The figure shows the distribution curve and the frequency (probability) of each generated value. The capacity values are irrelevant here and were hidden due to information safety. The simulator ran 4995 trials and achieved the configured confidence level.

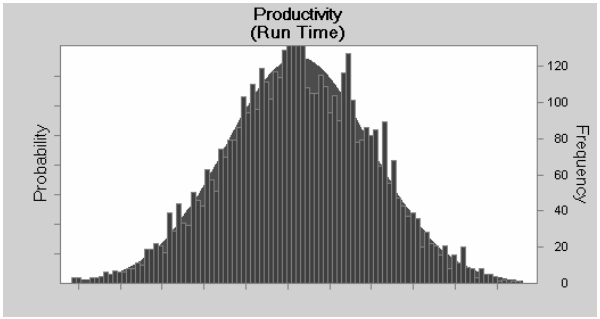

Figure 1. Generated values for productivity according to its distribution curve.

For each trial, the tool calculates a capacity value. All generated capacity values are then summarized in a chart (see Figure 2).

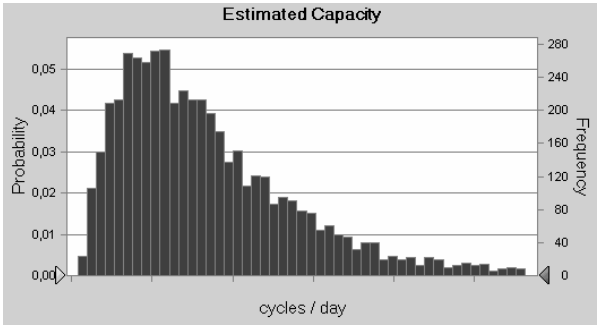

Figure 2. Possible values for capacity calculated during the simulation.

Now, we can consider the mean or the median of the capacities generated in each trial as our estimated team capacity. We can also consider other possibilities, as such the most probably capacity value or the capacity that achieves a specific certainty level. Regarding to the certainty level, the median achieves approximately 50% of certainty, as approximately 50% of the generated capacities are smaller and the others are larger than it. For the analyzed data, the mean gave us 37% of certainty, indicating that in approximately 37% of the trials, the achieved capacity was at least the mean value.

Other measures presented by the tool, such as standard deviation, variance and range, they are useful to make the decision of which value to choose for the team capacity. In addition, the simulator also presents the contribution of each input to the variation of the generated values. This last information can be useful to identify points to investigate and improve the process in order to reduce variability and increase precision.

3.2. Testing the capacity model simulation

The precision of a simulation model is measured by the closeness of its outputs to that produced by real systems [3]. In order to test our capacity model and the use of simulation, we compare the estimated capacity with the real number of test cycles executed per day obtained from the last two months of the analyzed historical data.

While analyzing the historical data, we verified that overtime occurred in specific dates. As we want to know what is the capacity to execute test cycles without needing overtimes, we consider a reduced number of test cycles executed per day. The reduction was done proportionally according to the verified overtime. We also disregarded the data about two days that have suffered effect of special causes. At the end, we had the capacity achieved on each day and the mean for the whole period.

We then compare the real observed capacities with the estimations generated by three different approaches using our capacity model: use of historical means as model inputs; use of simulation and considering the mean of the generated capacities; use of simulation and considering the median of the generated capacities.

Table 1 summarizes the comparison results, showing the percentage error between the real capacity (mean of the whole period) and the estimated capacity according to each estimation approach. It also shows the percentage number of days that achieved a smaller capacity than the estimated.

As we can see, all approaches resulted in smaller capacities than the real one verified for the whole period. Although considering the estimated capacity as the mean of the capacities generated by simulation resulted in the smallest error for the whole period, there are several days in the historical data that do not achieved that estimated capacity.

On the other hand, the approach without simulation resulted in the largest error for the whole period, but all days in the historical data achieved at least this other estimated capacity. Considering the estimated capacity as the median of capacities generated by simulation, in its turn, resulted in an intermediate error when compared with the other approaches.

Table 1. Estimated and real capacities comparison summary for the experiment.

Estimation approach	Estimation compared to the real capacity	Number of days with smaller capacity
Without simulation, using means as input.	43.27 % smaller	0.00 %
Estimation considering the median of capacities generated by simulation.	34.47 % smaller	8.11 %
Estimation considering the mean of capacities generated by simulation.	11.73 % smaller	35.14 %

4. The capacity model in an organizational setting

The customers of test execution teams are, in general, development teams or other internal teams from its organization and partners. Since the applications should pass the tests before going to the market, the test execution teams take a lot of pressure to execute the tests quickly. It is a big challenge to negotiate more resources or more time to perform the requested tasks. This challenge is even bigger when you are based only on opinions, instead of analysis and observations.

In this work, we used the presented model and the simulation technique for supporting the planning of a test execution team for mobile applications. This test execution team works for Motorola Brazil Test Center site at the Informatics Center/UFPE and its historical data were used for simulating and testing our capacity model, as shown by Sections 3.1 and 3.2.

The main use of the estimated capacity using our approach was to verify if the team capacity were sufficient for reaching the customer requirements. Although the occurred overtimes had indicated that the team was overloaded, it was difficult to quantify the overload. The estimations provided by our capacity model confirmed the overload and provided information about how much overloaded the team was. This quantification is done observing the difference between the requested and the estimated capacities.

The capacity estimations highlighted necessities and limitations of the test execution team. These necessities and limitations were used together with other information during schedule negotiations. We observed that the team requests for new resources and schedule reviews were more easily justified and

accepted when using information based on the statistical analysis.

For almost two months, the test execution team performance was monitored. At the end of this period, a significant number of its testers were moved to other teams and new testers were incorporated. These changes had modified the team productivity and the simulation model did not capture it automatically. In order to do that, the input probabilistic distributions should be reviewed and adapted to the new situation.

While monitoring the team performance, we investigated the causes of the lognormal distribution for the number of test cases per test cycle. We analyzed the data to verify if there were distinct groups of test cycles with different sizes. We easily demonstrated that using the histogram shown by Figure 3.

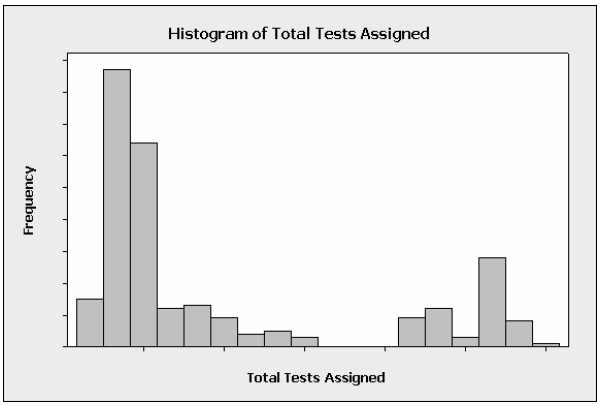

Figure 3. Histogram of the number of tests per test cycle.

In the histogram, we can quickly identify at least two distinct groups of test cycles apparently having different probabilistic distributions for its sizes. Time series plots and other charts suggested similar conclusion.

Supported by the expert intuition of the test team, we filtered and analyzed the data using the mobile technology, phone model, software stability and other attributes that could be correlated with the number of tests per test cycle. In this way, we identified the attributes that influenced the cycle size.

This information was useful to support process improvements. We also believe that this information can be used to refine our model.

5. Related work

A capacity model can be defined based on an effort model. There are several effort models available for software development. Some examples are the Constructive Cost Model (COCOMO) [1], the Function Point Analysis (FP) [2] and the Use Case Point Analysis (UCP) [5]. The COCOMO, in addition, has some extensions as COQUALMO, for quality prediction, and COCOTS, for off-the-shelf software integration.

Most of these models estimate the effort to execute a set of activities in the development process. The Test Point Analysis [7], in its turn, estimates the time needed to define and implement functional tests based on use case points. These models also take in consideration the characteristics of the products been developed or tested while doing the estimations.

The model defined here is specific to estimate the capacity to execute test cases in terms of test cycles, which is the motivation of this work. Our model also abstracts the characteristics of the test cases and the product being tested, as well some factors that may affect the team capacity (team experience, environment, training, etc.) and that the related models regards. This fact should be a problem only when these factors or the product characteristics change drastically.

We can define a new capacity model using these effort estimation models and simulate it. For example, we can consider the capacity to develop software as the division of the manpower per day by the effort to develop a product estimated using the FP analysis. Unfortunately, defining the probabilistic distributions for these detailed models can be very costly.

6. Conclusions

In this paper, we presented a test execution capacity estimation model and tested the use of simulation in order to improve the estimations precision. We tested this model using historical data and applied it for a mobile phone test execution team at the Motorola Brazil Test Center site at the Informatics Center/UFPE.

The use of means or medians of the generated capacities during the simulation allowed us to have a better precision considering the whole validation historical data. The simulator also gave additional information about the estimations precision through the certainty levels. Using higher certainty values, e.g. 80% or 95%, there will be more confident results about the minimal (or similarly the maximal) capacity that a test execution team can support.

In addition, the test execution capacity estimations provided by our model allowed us to present the test team necessities and limitations based on estimations supported by statistical fundamentals, important characteristic for justifying schedules reviews and new resources requests.

Although the results of this work suggest that we can achieve better precision using simulation, we cannot say in which situations this is true. In order to

do that, we need to apply this model in different test teams and to perform statistical tests to verify the results representativeness.

As the use of simulation requires we analyze the behavior of model inputs, it can help us to better understand and improve our processes. For instance, we identified attributes that influenced the number of tests per test cycle and we used this information for improving resource planning. Furthermore, the simulator showed the main source of variations, useful information for supporting process improvement.

A disadvantage of the simulation technique is that the precision of its results depends of the use of proper probabilistic distributions for variable inputs. Finding the correct probabilistic distributions may be a difficult task. It may require the availability of reliable historical data and validation through expert judgment.

Additionally, the probabilistic distributions are dependent of factors such as team experience, use of processes and tools, etc. When these factors change, model estimations may become invalid until we set proper probabilistic distributions for the new situation.

We also modified the presented capacity model in order to estimate the team size required to achieve a desired capacity. This information suggested the number of additional testers required for supporting the customer requests.

Some interesting investigations can extend the results discussed here. For example, we identified similar test cycles and organized them into groups according to some attributes. Including the probability distribution information about each distinct group in our capacity model may improve the estimations precision. We also believe that the use of simulation can be applied for models different from the presented here. For example, a software development capacity model can be defined and simulated in a similar way.

7. Acknowledgments

We would like to thank Rogério Monteiro, Carlos Albuquerque, André Lacerda and Alexandre Mendonça for their help with the organizational setting. We also thank the anonymous referees, whose appropriate comments helped improving the 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.

8. References

[1] Boehm, B., et al, Software Cost Estimation with COCOMO II, Prentice Hall, 2000.

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

[3] Jain, R., The Art of Computer Systems Performance Analysis Techniques For Experimental Design, Measurement, Simulation And Modeling, John Wiley, 1991.

[4] Kapoor R., Chen L., Lao L., Gerla M., Sanadidi M., "CapProbe: a simple and accurate capacity estimation technique". In *Proceedings of the 2004 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 67–78. ACM Press, 2004.

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

[6] Montgomery, D., Design and Analysis of Experiments, 5th Edition, John Wiley, 2000.

[7] Nageswaren, S.. "Test Effort Estimation Using Use Case Points.", available for download at http://www.cognizant.com/cogcommunity/presentations/Test_Effort_Estimation.pdf, June, 2001.

[8] Pyzdek, T., The Six Sigma Handbook, Mc Graw Hill, 2003.

[9] Ross, S., Introduction to Probability Models, Eighth Edition, Academic Press, 2002.