Empirical Studies of Test Execution Effort Estimation Based on Test Characteristics and Risk Factors

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 this paper, we present the main goals, questions and hypotheses of our research related to test execution effort estimation. Here, we introduce a model that we developed for estimating test execution effort based on the test characteristics and risk factors.

In addition, we presented the planning of empirical studies for this model. These studies have the objective of configuring, calibrating and validating the model using expert judgment and statistical methods. Finally, we defined metrics to be collected and the methods to analyze them in order to answer our research questions.

1. Introduction

In competitive markets (e.g., the mobile phone market), companies that release products with poor quality may quickly lose its 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 [8]. 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 for executing a given set of test cases, since their estimations are based on SW development complexity instead of test execution complexity. For this reason, estimates are usually made only based on expert judgment or historical execution times, which generally leads to a lack of precision.

In this paper, we introduce a proposed test execution effort estimation model that regards test size and execution complexity measured from test specification written in a controlled natural language [14][9]. We also present the empirical studies planned to configure and validate this model for the mobile application domain.

2. Relevant prior work

During the last few decades, several models and techniques were created for estimating software development size and effort. Function Points Analysis (FPA) [6], for instance, 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) [12] is an extension of FPA and estimates the size of a system based on use case specifications. Although UCP and FPA regard the development complexity of a system, they do not specifically regard the complexity for executing or automating a given suite of test cases. For

this reason, they cannot be used to estimate the effort to perform these two activities.

Test Point Analysis [13] is an extension of UCP that estimates the effort required to define and implement functional tests based on use case points. This model also takes in consideration the characteristics of the product being tested (not the characteristics of the tests). For this reason, it estimates the effort of all test activities for the whole product, such as defining, implementing and executing tests. We are interested in a solution for individually estimate the test execution and the automation effort of any test suite.

The Constructive Cost Model (COCOMO) [4] converts size metrics such as FP and SLOC (source lines of code) into effort estimation for developing systems. This is done through the use of effort multipliers and scale factors, which represent the environment, the teams and the processes characteristics. Similarly to FPA and UCP, COCOMO is not used to estimate the effort to perform test execution or automation activities.

These models do not estimate the effort for executing any given test suite, since their estimations are based on software development complexity instead of its execution complexity. In the best case, these models can only estimate the effort for defining and executing all the tests for the whole system. We need a model to estimate the effort only to execute any subset of test cases.

3. Research goals, questions and hypotheses

The main goal of this research is the development and evaluation of an estimation model for test execution effort. We aim to create an estimation model with the following characteristics:

- Accurate estimates;
- Based on the test characteristics 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;
- Ease of automation.

The assessment of our research goal will be performed by answering the following main research questions:

RQ1: Can we achieve high estimation accuracy for test execution effort by regarding the test characteristics and the risk factors?
RQ2: Are the test effort estimation models based on test specifications and risk factors more accurate than the ones only using historical data?
RQ3: Can the proposed estimation models be automated?
RQ4: What is the cost to use the proposed test execution effort estimation model?
RQ5: Has the proposed estimation model a similar performance in different application domains?

Based on these main research questions, we formulate our research hypotheses as presented next:

RH1: The effort to execute tests can be accurately estimated based on the test characteristics and risk factors.
RH2: Test effort estimation models regarding the test characteristics and risk factors are more accurate than models only based on historical data.
RH3: It is possible to have out test effort estimates automatically calculated with minimal human interaction, leading to a low cost of use.
RH4: The proposed model can be applicable to different application domains.

In addition, we derived question RQ1 into more concrete ones (RQ1.1 to RQ1.8) in order to answer them in an easier manner during our empirical studies.

RQ1.1: What are the relevant test characteristics to estimate test execution effort?
RQ1.2: How to group the values of each test characteristic into influence levels (low, average and high)?
RQ1.3: What are the best weights to assign to each influence level of each test characteristic?
RQ1.4: Is the number of execution points a soundness measure for representing test size and execution complexity?
RQ1.5: What are the relevant risk factors for estimating test execution effort?
RQ1.6: How to group the values of each risk factor into influence levels (low, average and high)?
RQ1.7: What is the impact in test execution effort (weight) of each influence level of each risk factor?
RQ1.8: How accurate are the estimates using the proposed test execution effort estimation model?

4. 4. Test execution effort estimation model

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. Here, we consider that the test case specifications are written in a standardized way.

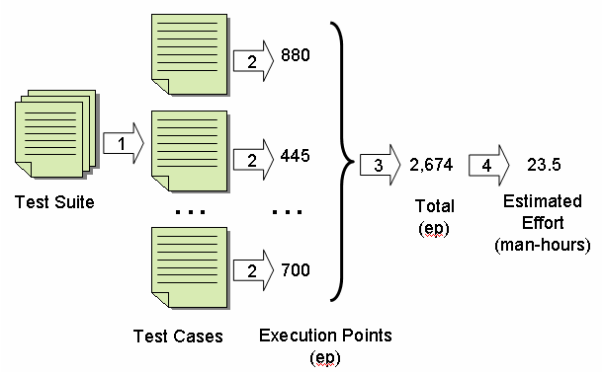

Figure 1. Test execution effort estimation model overview.

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.

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. Next, we give more details about Steps 2 and 4.

The Step 2 of our model is detailed by 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 (C_1 to C_n). 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).

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 estimation model should be configured according to the target application domain in order to maximize the estimates accuracy. For instance, the list of characteristics (C_1 to C_n), their influence levels and weights should be calibrated through empirical studies in the target application domain.

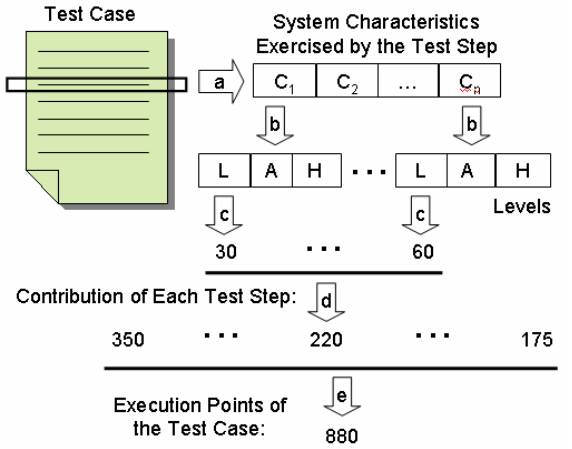

Figure 2. Assigning execution points to a test case.

For the Step 4, we have two approaches we can use. The first approach is the simplest one, which is applicable when we have a static test environment and test productivity. Here, the estimated effort is calculated based on the execution points of a test suite and a conversion factor (CF).

As illustrated by Figure 3, (f) we first calculate the conversion factor by dividing execution effort by the number of execution points. For that, testers can measure the test size and execution complexity of several test cases. Then, the execution time of the tests should be collected from a historical database (if available) or by executing them. Then, to estimate the effort to execute new test suites, we just need to (g) multiply its number of execution points by the conversion factor.

The second approach to use in Step 4 regards the risk factors existing in a dynamic environment, such as team experience, environment conditions, test precedence and product stability. All these factors impact somehow the test execution effort

Hence, to calculate the effort to execute a test suite, we use a mathematical equation that is based on the execution points and the risk factors. The list of relevant risk factors and the coefficients (impact) to be

used for each of them in the equation may depend on the application domain.

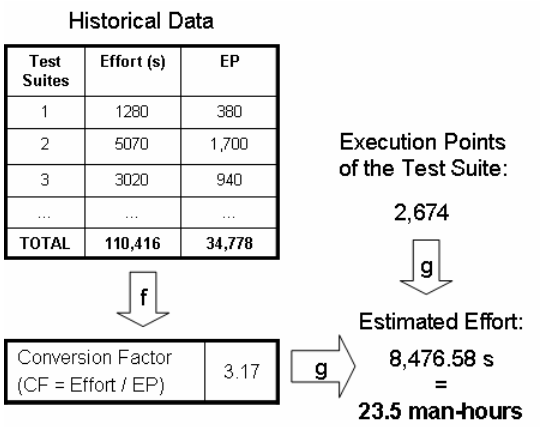

Figure 3. Test effort estimation.

As discussed in Section 5.1, we will perform some correlation analyses to define a regression model and use regression analysis to create the equation. For instance, COCOMO has its equation for SW development effort defined through a similar process, identifying linear and exponential relationships. If we find similar relationships between the test execution effort and each related risk factor, we will have the following regression model:

$$\text{Effort} = A * CD * (EP)^{ED}$$

Where:

A is an effort coefficient.
CD is an effort adjustment factor derived from cost drivers.
ED is an exponent derived from scale drivers.
EP is the test size and execution complexity in execution points.

In this case, the risk factors are classified in cost (linear relationship) and scale drivers (exponential relationship).

5. Empirical studies

We plan to run some empirical studies with the objective to create, configure, calibrate and test our estimation models for the mobile application domain. The objective of this section is to describe these studies.

Table 1 summarizes the design of our empirical studies of test execution effort estimation and it is discussed next. We discuss the planned empirical studies and present the definition of the metrics to be collected. Also, we show the methods that we plan to use for analyzing the data.

5.1. Description of the empirical studies

In this section, we overview the studies planned by this work.

ES1 – Expert judgment: this empirical study uses 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. This study helps to answer questions RQ1.1 to RQ1.3 and RQ1.5 to 1.7.

ES2 – Experiment: in this study, almost 200 test cases of 3 different features were executed by 6 testers in a controlled environment. After collecting the data, we performed two different analyses. In the first analysis, we evaluated the model created during the ES1:

- We evaluated the accuracy of the estimation model created during the ES1 through a three-fold cross-validation. We used the training sets to calculate the value of the effort coefficient A of our effort equation. Since it is a controlled environment, we use the value 1 for the CD and ED variables of our effort equation in this experiment. Then, we used the test sets to observe the mean estimates accuracy and to calculate its confidence interval. (RQ1.8)
- We also used all the collected data to empirically demonstrate the soundness of the test size and execution complexity metric. For instance, we verified that tests intuitively considered similar (with respect to test size and execution complexity) have similar number of execution points assigned to them. (RQ1.4)
- We also identified the steps of our estimation process that can be automated. (RQ3)

We will perform a second analysis, which aims to use statistical methods (ANOVA, statistical tests, etc.) to identify:

Table 1. Planned empirical studies of test execution effort estimation.

Question	Metric	Empirical Studies					
		Mobile Application Domain				Desktop Application Domain	
		ES1	ES2	ES3	ES4	ES5	ES6
		Expert judgment	Experiment	Survey	Case Study	Expert judgment	Case Study
RQ1.1	TestCharacteristics	Delphi panel	Statistical test	Questionnaire		Delphi panel	Statistical test
RQ1.2	$CharacteristicValues_{c,l}$	Delphi panel		Questionnaire	Clustering algorithms	Delphi panel	
RQ1.3	$CharacteristicWeight_{c,l}$	Delphi panel	ANOVA				ANOVA
RQ1.4	InconsistentMeasures1, InconsistentMeasures2		Empirical demonstration				
RQ1.5	RiskFactors	Delphi panel		Questionnaire	Statistical hypothesis test	Delphi panel	Statistical hypothesis test
RQ1.6	$RiskFactorsValues_{c,l}$	Delphi panel		Questionnaire	Clustering algorithms	Delphi panel	
RQ1.7	$RiskFactorsWeight_{c,l}$	Delphi panel			Regression Analysis		Regression Analysis
RQ1.8	MMRE, Pred(0.25)		Cross-validation / confidence interval/ Statistical test		Cross-validation/ confidence interval/ Statistical test		Cross-validation/ confidence interval/ Statistical test
RQ2 and RQ5	MMRE, Pred(0.25)				Cross-validation/ Statistical test		Cross-validation/ Statistical test
RQ3	StepsNotAutomated		Empirical demonstration				
RQ4	$\#EvaluatedTestSteps_n$				Cross-validation/ Statistical test		Cross-validation/ Statistical test

- If the test characteristics identified in ES1 are statistically significant to the accuracy of the effort estimates. (RQ1.1)
- The weights of the influence levels of each test characteristic according to the mobile application domain. (RQ1.3)

We developed a tool for collecting the execution time spent in each test step of the test cases. Table 2 presents an example of the information used in the ANOVA. The main results of the ANOVA are: the statistical significance of each characteristic (C_1 to C_n) with respect to the execution effort; the impact (weight) in the effort when we change the values from Average to Low or to High in each characteristic.

ES3 – Survey: the purpose of this survey is to identify characteristics and risk factors not highlighted during the Delphi panel (ES1). We will apply a questionnaire to approximately 60 testers of different test execution teams. (RQ1.1, RQ1.2, RQ1.5 and 1.6).

ES4 – Case study: we planned to use in practice the estimation model calibrated by the ES2. The model will be used for 6 months by 4 test teams in context of the Informatics Center and Motorola partnership.

Table 2. Execution effort and complexity evaluation of each test step.

Test Step	Effort (s)	C_1	C_1	…	C_n
Go to the message center	11	Low	Low	…	Low
Send a message	58	Average	High	…	Low
…	…	…	…	…	…

During this study, we expect to collect data about the execution of some thousands of tests. We will use the collected data to perform some analyses:

- Use clustering algorithms to group the possible values of each test characteristic and risk factor into influence levels (low, average and high). These groups were previous created through expert judgment in ES1. (RQ1.2 and RQ1.6)
- Use correlation analysis to identify the more appropriate regression model and then use regression analysis to:
  - Identify which of the risk factors identified in ES1 are statistically significant to the accuracy of effort estimates. (RQ1.5)
  - Automatically calculate the weights of the influence level of each risk factor. These weights are used to calculate the CD and ED variables of our effort equation (RQ1.7)
  - Evaluate the accuracy of the estimation model configured by ES2 by analyzing the estimation errors. (RQ1.8)
- Evaluate the accuracy of the estimation model configured by the improvements done by ES4. We used the training sets to calibrate the weight of the risk factors and the test sets to observe the estimates accuracy. (RQ1.8)
- Calculate the cost of using our estimation model by observing the number of times we need to evaluate the execution complexity of test steps. (RQ4)
- On the mobile application domain, compare the accuracy of our model with the others based on historical data. (RQ2 and RQ5)

ES5 – Expert judgment: the purpose of this Delphi panel is to identify characteristics and risk factors related to the execution of tests of desktop application, as well their influence levels. Five experts of the domain will attend the panel. This information will be used to configure the estimation model to the desktop application domain. (RQ1.1, RQ1.2, RQ1.5 and 1.6)

ES6 – Case study: we intend to calibrate and use the estimation model for 3 months, in a test team of desktop applications. The main objective here is to verify if our proposed estimation model can be generalized to others application domains. (RQ5)

5.2. Definition of metrics

We define here the set of metrics to be collected during the empirical studies to answer our research questions in a quantitative way [3].

MMRE: Mean magnitude of the relative error.

$$MMRE = 1/N * \Sigma MRE_i$$

Where:

$MRE_i = abs(RE_i)$
$RE_i = (est_i - actual_i) / actual_i$
N = number of estimates
est_i = estimated execution effort of the ith test
$actual_i$ = actual execution effort of the ith test

Pred(0.25): Percentage of estimates that are within 25% of the actual values.

$$Pred(0.25) = 1/N * \Sigma \ (1 \text{ if } MRE_i \leq 0.25, \ 0 \text{ otherwise})$$

TestCharacteristics: List of characteristics relevant to estimate test execution effort.

$CharacteristicValues_{c,l}$: List of values of the test characteristic c that have a given influence level l (low, average or high) on test execution effort.

$CharacteristicWeight_{c,l}$: Weight (quantitative impact) assigned to the influence level l of the test characteristic c with respect to test execution effort.

RiskFactors: List of risk factors relevant to estimate test execution effort.

$RiskFactorValues_{r,l}$: List of values of the risk factor r that have a given influence level l (low, average or high) on test execution effort.

$RiskFactorsWeight_{c,l}$: Weight (quantitative impact) assigned to the influence level l of the risk factor r with respect to test execution effort.

InconsistentMeasures1: List of pair of tests (t_1, t_2), where t_1 and t_2 are intuitively similar with respect to their size and execution complexity with significantly different numbers of execution points assigned to.

InconsistentMeasures2: List of pair of tests (t_1, t_2), where t_1 and t_2 are intuitively different with respect to their size and execution complexity with inconsistent numbers of execution points assigned to. For instance, a bigger test with a similar or a smaller number of execution points.

#EvaluatedTestSteps$_n$: Number of test steps of the nth test case that required human interaction to evaluate its execution complexity. In other words, the number of test steps never evaluated before, since they require manual evaluation only at the first occurrence.

StepsNotAutomated: List of steps of the estimation model that were identified as difficult to automate.

5.3. Summary of the main data analysis methods used in the studies

Delphi is a method that experts are invited to attend a panel for two or more rounds in order to achieve a consensus about a subject. The participants are coordinated by a facilitator who ensures their anonymity and structure the information flow between them. It promotes rapid consensus and avoid groupthink. We use this method to decide how to configure our estimation model based on expert judgment. For example, we use it to decide which risk factors and test characteristics to consider in the estimation model.

Statistical hypothesis test is a statistical method used to test if a given hypothesis is true with a determined confidence level [11]. We use this kind of test to verify, for example, if a given test characteristic or risk factor is relevant for estimating test execution effort, or if the accuracy of proposed estimation model is better than the current one.

ANOVA is a statistical method to test heterogeneity of means by the analysis of group variances [11]. In our research, we use ANOVA to identify the impact on the test execution effort when varying the influence levels of the test characteristics. This information is used to calculate the weight of each influence level.

Empirical demonstration is the confirmation that a given theory is correct by observing it in the practice [5]. For instance, we empirically demonstrated the soundness of the execution points measured from the test cases.

Cross-validation is a method for generalizing the results of a model evaluation. The main idea is to partition a sample of data into folds (subsets) such that you test the model on a single fold, while the other folds are used to build the model. The use of this method reduces the probability of obtaining results by chance.

Confidence interval is an interval estimation built with a given confidence level that is generated from a random sample of an underlying population. Based on the results of the empirical studies, we use confidence intervals to have a better idea of our estimation model accuracy.

Clustering algorithms [7] are used to classify objects into different groups. We used these algorithms, such as the k-means clustering, to group the values of the test characteristics and risk factors into influence levels (low, average and high).

Regression analysis is a method used to create a mathematical model (called regression equation) that tries to explain the relationship between response and predictors variables [11]. We use this method to identify the weights of the influence levels of each relevant risk factor. We choose the regression model to be used (linear, quadratic, exponential, etc.) by doing linear and nonlinear correlation analyses between the effort and each risk factor.

5.4. Threats to validity

In this section, we list the main threats to the validity of our empirical studies and how we plan to control them.

Some relevant characteristics and risk factors may not be identified in ES1 and ES3: we can control this threat by analyzing the residuals of the regression analysis and the results of the ANOVA.

Tests may not be representative: if the tests used in our empirical studies were not representative, the results would be biased. During our studies, we will analyze the test specifications to verify how representative they are. If necessary, we can add more different tests to increase their representativeness.

Test environment not representative: we may not collect enough data regarding all risk factors if we have a not significant variety of situations (tools, team experience, etc.) in our studies. We can control this threat by analyzing the collected data and observing the discarded risk factors during the regression analysis.

Problems during data collection: the collected data may be incorrect or inconsistent, leading to invalid

results. We will guide the data collection and inspect the collected data to identify outliers, discarding the data always when necessary.

6. Conclusions

In this paper, we presented the main goals, questions and hypotheses of our research. We introduced a new estimation model that we developed for estimating test execution effort based on test characteristics and risk factors. We presented the planning of the empirical studies for configuring and evaluating this model. These studies are based on expert judgment, an experiment, a survey, case studies and the use of statistical techniques.

We also presented the metrics that will be collected during the empirical studies and how they will help to answer our research questions. In addition, we described the main methods used to analyze the collected data. Most of them are statistical methods that can have a significant contribution to the model accuracy. Then, we discussed about the main threats to validity of our empirical studies and how to control them.

Finally, we expect that the results of our empirical studies will not only validate our work, but also support the construction of new models, such as test coverage vs. execution effort analysis model, a test automation effort estimation model and a cost-benefit analysis model for prioritizing manual tests to be automated.

9. Acknowledgments

We would like to thank all anonymous reviewers who have helped us to improve this paper through their extremely relevant comments.

The first author is a PhD candidate 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.

10. 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)*, co-located with the IEEE Conference on Software Maintenance. September 24, Philadelphia, Pennsylvania USA, 2006.

[3] V. Basili, G. Caldiera, and H. Rombach. "Goal Question Metric Approach," *Encyclopedia of Software Engineering*, pp. 528-532, John Wiley & Sons, Inc., 1994.

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

[5] N. Fenton. Software measurement: A necessary scientific basis. *IEEE Transactions on Software Engineering*, 20(3):199–206, 1994.

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

[7] K. Jajuga, A. Sokolowski, H. Bock. *Classification, Clustering and Data Analysis*. Springer, 2002.

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

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

[10] H. Linstone and M. Turoff. *The Delphi Method: Techniques and Applications*. http://is.njit.edu/pubs/delphibook, 2002.

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

[12] 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.

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

[14] 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.