Universidade Federal de Pernambuco
Centro de Informática

Pós-graduação em Ciência da Computação

Estimating Test Execution Effort Based on Test Specifications

Eduardo Henrique da Silva Aranha

Tese de Doutorado

Recife
Janeiro de 2009

Universidade Federal de Pernambuco
Centro de Informática

Eduardo Henrique da Silva Aranha

Estimating Test Execution Effort Based on Test Specifications

Trabalho apresentado ao Programa de Pós-graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco como requisito parcial para obtenção do grau de Doutor em Ciência da Computação.

Orientador: *Prof. Paulo Henrique Monteiro Borba*

Recife
Janeiro de 2009

To my family.

Acknowledgements

✓ First of all, thanks to my parents, wife and daughter, for their support and patience.

✓ I would like to thank Paulo Borba, my advisor, for his great instructions and support during this research.

✓ Thanks to the professors Cristiano Ferraz, Renata Souza and Carla Monteiro, for their teaching and support given during this research.

✓ Many thanks to all professors, students and employees of the BTC Research Project and the Software Productivity Group for their valuable comments about my work and for their support during my empirical studies.

✓ Thanks to Luís Cláudio, Rogério Monteiro, Edson Fontes, André Lacerda, Alexandre Mendonça and the other employees and scholarship holders of the CIn/Motorola program that colaborated during this research.

✓ Thanks to the students Eduarda Freire, Raquel Cândida and Veristiana Carvalho, for their help during the execution of the empirical study ES4.

✓ Thanks to the students André Ribeiro, Fernando Souza and Ivan Machado, for their help during the execution of the empirical study ES5.

✓ Thanks to the students Filipe de Almeida, Thiago Diniz, Vitor Fontes and Guilherme Carvalho, for their hard work during the implementation of tools and scripts used in this research.

✓ Thanks to the student Roberto Ferreira, for their help during the evaluation of alternative experimental designs for the empirical study ES6.

✓ Thanks to the Motorola Brazil Test Center and the National Counsel of Technological and Scientific Development (CNPq) for their finantial support during my doctorate.

✓ The suggestions of well-known researchers in the area, such as Barry Boehm, Emilia Mendes, Ricardo Valerdi, Andrea de Lucia and Carmine Gravino, as well as comments from the anonymous reviewers of our papers allowed us to identify and explore several opportunities in our research. We are grateful to them for their valuable comments.

✓ Finally, I would like to thank God for giving me the strengh required to complete this stage of my life.

Resumo

Em mercados competitivos (por exemplo, o de celulares), empresas de software que liberam produtos com baixa qualidade podem rapidamente perder os seus clientes. A fim de evitar isso, essas empresas devem garantir que a qualidade dos produtos atendem a expectativa dos seus clientes. A atividade mais comum realizada para garantir a qualidade é teste de software.

Além disso, teste está sendo considerado tão importante que organizações podem alocar equipes exclusivamente para exercer atividades de teste. Em tais situações, as equipes de teste devem ser capazes de estimar o esforço exigido para exercer as suas atividades dentro do prazo e para solicitar mais recursos ou negociar prazos, quando necessário. Na prática, as consequências de ter estimativas ruins são onerosas para a organização: redução de escopo, atraso nas entregas e horas extras de trabalho.

Visando uma melhor forma de estimar esforço de execução de casos de teste, esta pesquisa tem o objetivo de propor uma medida de tamanho de teste e de complexidade de execução baseada nas próprias especificações dos testes, sendo estes escritos em uma linguagem natural controlada ou pelo menos padronizada. Também pretendemos definir e validar um método de medição para a métrica proposta.

Nesta pesquisa, pretendemos propor modelos de estimativa de esforço de execução de testes que consideram a medida de tamanho e complexidade de execução de teste proposta. Sendo assim, identificamos fatores de risco relacionados com as atividades de execução de testes. Além disso, é também nosso objetivo calibrar e avaliar os modelos de estimativa propostos aqui através de estudos empíricos dentro do domínio das aplicações móveis.

Palavras-chave: Esforço de execução de testes, tamanho de testes, modelos de estimativa, estudos empíricos.

Abstract

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.

In addition, software testing is being 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. In the practice, the consequences of poor estimates are costly to the organization: scope reduction, schedule overrun and overtime.

In order to better estimate the execution effort of a given test case or test suite, this research has the objective of proposing a measure of test size and execution complexity based on the test specifications written in a controlled or standardized natural language [119]. We also want to define and validate a measurement method for our proposed measure.

In this research, we aim to propose test execution effort estimation models that regard the test size and execution complexity measured from the test specification. For that, we identified risk factors related to test projects. Also, we aim to calibrate and evaluate the proposed estimation models through empirical studies on the mobile application domain.

Keywords: Test execution effort, sizing, estimation models, empirical studies.

Contents

List of Figures

List of Tables

CHAPTER 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 their clients' expectations. A usual activity performed to improve quality is software testing, which is the act of exercising the software with the objective to detect failures during its execution [66].

There are several approaches for testing an application. One important approach is the functional testing, which was defined based on the idea that programs can be considered to be a function mapping inputs to outputs [66]. This approach is considered to be "black box", that is, the only information used for designing the test cases is the software specification.

Software testing is an important activity that usually requires a significant effort. There is a well-known rule of thumb saying that approximately 50 percent of the elapsed time and more than 50 percent of the total cost of typical software development project is spent in testing [100]. Although the development of new testing technologies is reducing this impact over the time, software testing still demanding significant effort.

For this reason, several techniques still being developed not only to improve the effectivity of the tests, but also to improve the test process efficiency. For instance, Model-Based Testing (MBT) appears as a promising approach to control software quality and to reduce the inherent costs of test processes [44]. MBT can significantly improve test coverage and reduce the test design effort [111] [105], since it automatically generates test cases from the software specifications. However, the effort for executing these generated tests is still significant, mainly when tests have to be manually executed due to restrictions such as:

- Short time to makert, where the time required to automate tests is only acceptable for software maintainance, but not for the first software release.
- Tests of multimedia (sounds, video, etc.) and hardware features that have limited support of existing test automation frameworks.

Also, since the use of MBT tools for automatically generating tests usually achieve better test coverage, increasing the number of tests and consequently the effort to execute them.

In addition, software testing is being considered so important that organizations can allocate teams exclusively for testing activities in order to achieve unbiased test results [34]. These test teams are usually requested to test features of different software applications. Mainly 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.

A review of surveys [98] on software effort estimation showed the difficulty to have accurate estimates. Some of the practical consequences of having poor estimates in testing projects are:

- Scope reduction: in order to keep the planned schedules and budget, only part of the tests are executed. This means that only parts of the software will be tested, increasing the chance of escaped defects be found only by the client.

- Schedule overrun: in this case, the testing phase takes more time than planned, increasing the costs and delaying the delivery of software releases into the marketing. In some high competitive markets, this option is not viable.

- Overtime: the testing phase is finished on the schedule, but exceeding the regular working hours. Here, the cost may be high, as well the negative impact on team motivation.

In industry, all these consequences are costly and should be avoided.

1.1 Problem Statement

The effort required for executing tests should be properly estimated by test managers. For that, it is necessary to have good measures related to test execution effort and accurate estimation models. When regarding software development effort, several estimation approaches and models have been proposed over the years for estimating software development effort [26], such as Function Point Analysis (FPA) [47] and COCOMO [27].

Regarding testing effort, Test Point Analysis [101] is an approach similar to FPA used for estimating the effort of the whole testing phase of a given information system. However, these approaches do not estimate the effort required to execute a given set of tests, since their estimates are based on software size and development complexity instead of test size and execution complexity [11]. For this reason, estimates made by test managers are usually based on expert judgment [18], which generally is subjective and not repeatable.

Also, there are several cost drivers for software development reported in several papers and used by several software development estimation models, but few of them are also related to test execution. In fact, some cost drivers for test execution can also be particular to specific industrial settings. In summary, we have a lack of measures and cost drivers related to test execution, limiting the accuracy that estimation techniques can achieve for test execution effort.

1.2 Context

This work is part of a multi-disciplinary research initiative [126] in partnership with a important organization on the mobile application domain for developing new software testing technologies, such as the automatic generation of tests based on software specifications. In particular, the focus of this research is to increase the accuracy of test execution effort estimates and to automate the estimation process. In this way, **we should automatically generate test specifications and the estimated effort to execute them**.

In this work, we consider **dedicated and independent test teams executing functional tests manually**, since these are the teams more impacted by the investigated problem. Also,

most tests on the mobile application domain is executed by only one tester, which means that **the effort required to execute a test is actually the time spent to execute it**.

Finally, we consider that software development teams request the services provided by test execution teams when they deploy internal or commercial software releases. Hence, **test execution effort estimates should be performed when these requests arrive**. We also consider that the tests for the product are already created (by model-based testing tools or by manual test design) and **only a subset of the tests will be executed**, according to the functionalities created and modified in the current software release.

1.3 Summary of Goals

In order to better estimate the execution effort of a given test case or test suite, this research has the objective of proposing a measure of test size and execution complexity based on the test specifications written in natural language. We also want to define and validate a measurement method for the proposed measure.

To improve the accuracy of test execution effort estimation, we want to propose estimation models that regard the test size and execution complexity measured from test specifications and that regard the effect of cost drivers related to test execution. For that, we also have to identify these cost drivers and to investigate their effects on test execution effort. Finally, we want to calibrate and evaluate the proposed estimation models through empirical studies on the mobile application domain.

1.4 Methodology

Size is usually the most important and most used variable in effort estimation models. Hence, it is important to know how to measure software size. For this reason, we first review the existing measurement methods of software size. Since we want to propose a better way to estimate test execution effort, we also review the existing software effort estimation models described in the literature. Then, we define a size measure and a measurement method for test size and execution complexity. During our empirical studies, this measure is validated in terms of validity of construction, practical use and benefits for the accuracy of test execution effort estimation.

We define some test execution effort estimation models based on our proposed measure of test size and execution complexity and the existing estimation model techniques. After that, we run a sequence of empirical studies to identify the relevant variables to include in the proposed test execution effort estimation models, to evaluate them and the achieved accuracy. Some supporting tools are also developed to support the execution of these empirical studies.

1.5 Summary of contributions

During this research, we produced the following contributions for the academia and industry:

- The proposal and validation of a measure for test size and execution complexity that is based on test specifications written in natural language. This characteristic better supports the estimation of effort to execute a given set or subset of functional tests, even for tests never executed before.
- The definition and implementation of an automated measurement method for sizing test specifications.
- The proposal and evaluation of estimation models based on the size of test specifications.
- Definition of systematic methods for configuring the proposed measurement method for specific application domain, which includes the identification and weighting of characteristics in test specifications that better explain test execution effort.
- Definition of systematic methods for identifying and weighting cost drivers related to test execution that can be particular to specific industrial settings.
- The configuration of the proposed measurement method and the investigation of cost drivers related to test execution for the mobile application domain.
- The development of a tool for measuring test size and for estimating test execution effort based on the test specifications.

1.6 Thesis organization

The remaining of this document is organized as follows:

- Chapter 2 presents a review of software size measurement methods. It shows the main software size measures and measurement methods that were proposed in the literature and used in well-known software estimation models.
- Chapter 3 presents a review of existing effort estimation models. This review is mainly based on books and papers that describe and compare estimation techniques, reporting their advantages and limitations, etc.
- In Chapter 4, we present the overall planning of our research, as well the development of a measure for test size and execution complexity and estimation models for test execution effort based on the proposed measure.
- Chapter 5 presents the planning, execution and results of our empirical studies used to: create, calibrate and evaluate estimation models for test execution effort; validate our proposed size measure for test size and execution complexity; identify cost drivers for test execution and investigate their effects on test execution effort.
- Chapter 6 relates this research to other works found in literature.
- Chapter 7 presents the final conclusions of this work, presenting opportunities for future work.

CHAPTER 2

Software Size Measurement Methods

Size is usually the most important and most used variable in effort estimation models [28]. In this chapter, we show the main software size measures and measurement methods that were proposed and used in software estimation models.

2.1 Analysis Criteria

To analyze the different software size measurement methods reported in the literature, we defined the criteria shown in Table 2.1. These criteria represent characteristics that should be considered when selecting a software size measure for estimating effort based on the project size.

Table 2.1 Set of criteria used to evaluate software size measurement methods.

Criterion	**Description**
Artifact	Type of artifact that is measured.
Attribute	What characteristic is being measured.
Restriction	Any restriction that is applicable to the measured artifact.
Availability	Phase or time where the artifacts are available and the metric can be collected.
Standardized	Describe if there are well-defined measurement procedures and a concensus (standard) in the community.
Deterministic	Describe if different people can obtain different results when measuring the same artifacts.
Measurement cost	The cost to obtain the measure.
Calibration cost	The cost to calibrate the measurement method, if necessary.
Required backgound	Describe if there is any dependency of expertise for ensure the quality of the measure/measurement method.
Automation	Capability of the measurement method to be automated.
Validity	Describe if there is any critique about the validity of the measure.

The main sources used to find the presented software size measures and measurement methods were books, papers and surveys related to software metrics and effort estimation models.

2.2 Source Lines of Code (SLOC) Counting

There are two types of SLOC, the Physical SLOC and the Logical SLOC [109]. Physical SLOC depends on the style and format of the programming language. It may include comments and blank lines and it is easer to count. Logical SLOC attempts to measure only the number of statements and it is harder to count. Some advantages of SLOC are:

- The counting can be automated, although the tools are usually developed for specific languages due to their different syntaxes and structures. Examples of tools are the CodeCounter [38] and SLOCCount [122].
- It is an intuitive metric, since bigger programs have more SLOCs than smaller ones.

However, there are also some problems with SLOC:

- The number of SLOCs is dependent upon the skill level of the programmer. For instance, skilled programmer may write fewer lines of code for the same functionality when compared to a beginner programmer.
- For the same application, the support of the programming language (structures, instructions, etc.) affect its number of SLOCs.
- Lack of counting standards. A recent initiative is trying to define what to consider when counting SLOCs [104].
- It is difficult to have early estimates of the total number of SLOCs of the application to be developed [81][99].
- The use of code generators can automatically generate a great amount of source code, reducing the correlation between SLOC and development effort.

2.3 Function Points Analysis (FPA)

Function point [8] was developed by Allan Albrecht of IBM as a standardized metric for measuring the functions of a software application from the user's point of view. It provides a technology independent estimate of the size of the final program, since it is related to specifications rather than code. His first publication about FP was in 1979 [6] and an extended version was published in in 1983 [8]. Then, in 1984, the International Function Point Users Group (IFPUG) [55] was set up to clarify the rules, set standards, and promote their use and evolution.

In industry, Function Points Analysis (FPA) is one of the most known measurement process used for measuring software size based on the software specification [47] [80]. Since its initial description in 1979, this method was evolved as shown in Table 2.2. More details about the improvements found in each version can be seen in [4]. In addition, IFPUG Functional Points Analysis is an ISO certified functional sizing method.

FPA measures function points in 7 steps [47] that are described next.

Table 2.2 Official versions of the function points analysis.

Year	Version
79	Albrecht 79 [6]
83	Albrecht 83 [8]
84	GUIDE 84 [7]
86	IFPUG Release 1.0 [56]
88	IFPUG Release 2.0 [57]
90	IFPUG Release 3.0 [58]
94	lFPUG Release 4.0 [59]

Step 1: Determine the type of counting.

Three different types of function point counts were defined to represent the three major types of software projects:

- *Development:* this type of count is associated with the development of new software applications.
- *Enhancement:* this type of count tries to size the enhancements to be done in applications already in production.
- *Application:* this type of count are done on applications already developed, usually as a “baseline count".

Step 2: Identify the counting scope and application boundary.

The counting scope is determined by the purpose of the count, identifying the systems, applications or parts of applications that will be sized. The application boundary is the border between what is being measured and external applications and users.

Step 3: Identify all data functions and their complexity.

The data functions are classified as Internal Logical Files (ILF) and External Interface Files (EIF). ILF is a group of logically related data or control information that is perceived by the user and maintained inside the application boundary. EIF is a group of logically related data or control information that is perceived by the user but maintained inside the boundary of another application. For instance, an EIF counted for an application may be an ILF for another application.

FPA defines guidelines to identify ILFs and EIFs. Then, their complexity are calculated by counting the number of Data Element Types (DET) and Record Element Types (RET) DET are unique user-recognizable non-repeatable fields or attributes. RET are user-recognizable

subgroups of data elements contained whitin ILFs or EIFs. There are also guidelines defined to identify DETs and RETs in the FPA documentation.

The complexity of the data functions are given in a Low, Average and High scale, according to the counts of DETs and RETs (see Table 2.3). Then, these ordinal values are transformed in numerical values as presented in Table 2.4. To illustrate this transformation, let us suppose the counts RETs=7 and DETs=3 for a given data function. According to Table 2.3, the complexity of this data function is rated as Average. For an ILF data function, this result corresponds to the weight 10 (see Table 2.4).

Table 2.3 Complexity matrix for ILFs and EIFs.

RETs	DETs		
	1-19	20-50	> 50
1	Low	Low	Average
2-5	Low	Average	High
> 5	Average	High	High

Table 2.4 Weights used for data functions.

Rating	Weights	
	ILF	EIF
Low	7	5
Average	10	7
High	15	10

Step 4: Identify all transaction functions and their complexity.

Transaction functions perform update, information retrieval, output and so forth. They are classified in External Inputs (EI), External Outputs (EO) or External Inquiries (EQ). EIs are elementary processes that process data or control information entered from the outside of the application boundary. EOs are elementary processes that generate data or control information to outside of the application boundary. EQs are elementary processes that retrieve data or control information from ILFs or EIFs to the outside of the application boundary.

DETs and File Types Referenced (FTR) are used to determine the transaction function complexity. FTR is the number of files referenced or updated in the transaction. There are other tables (see [47]) similar to Tables 2.3 and 2.4 with the complexity matrix and weigths for EIs, EOs and EQs.

Step 5: Determine the unadjusted function point count.

The unadjusted function point count is the sum of the points assigned to data functions and transactions functions.

Step 6: Determine the value adjustment factor.

The value adjustment factor (VA) summarizes 14 General System Characteristics (GSC) defined by FPA. When applied, the value adjustment factor can adjusts the functional size by a maximum adjustment of $\pm 35\%$ to produce the adjusted function points (AFP). This is considered one limitation of the FPA measurement method.

The list of GSCs includes complex processing, reusability, installation ease and the development in multiple sites. Each GSC is evaluated on a scale of 0 to 5. This value is called degree of influence. The sum of all GSC rates is called Total Degree of Influence (TDI). VAF is calculated based on the following formula:

$$VAF = (TDI * 0.01) + 0.65 \tag{2.1}$$

Step 7: Calculate the adjusted function point count.

Finally, the adjusted function point count is calculated by multiplying the number of unadjusted function points by the value adjustment factor (VAF).

According to [47], counting points with FPA has the following advantages:

- It is a language and implementation-independent metric.
- Available at an early stage of the software development cycle, since it is based on the software specification.
- It has an international and independent group promoting its use and evolution.

However, there are also some problems in this approach:

- The reasons for the assignments of specific values (weights) are not clear.
- Lack of a clear definition. As we can see in [3], different authors had different interpretations of FP: a measure of size, productivity, complexity, functionality or user deliverables.
- The expertise in the assessment of complexity, the interpretation of the specification, value judgments, perception of the object boundaries and other aspects may causes a variation in the counts. Experience in the application of function points is then an important factor in their successful application [81].

2.3.1 Critiques about FPA model construction

The validity of Function Points Analysis (FPA) with respect to the measurement theory was questioned by several researchers. In [3], the authors suggested that the function points interpretation should be reconsidered from a measurement perspective, addressing issues in the expert judgments and measurement scales and transformations throughout the measurement steps.

Barbara Kitchenham discussed in [70] about problems in the model construction. She pointed that absolute scale counts are reduced to ordinal scale measures and, for this reason, you can no longer apply other transformations to make them interval or ratio scale. Formally, we cannot add the label “simple" to the label “complex", even when using numbers (3, 5, etc.) as a synonym for them. Also, since we cannot multiply or divide a ordinal value, we cannot convert them to productivity measures.

Another reported problem with function points is the technology adjustment factor, that is based on a subjective assessment of 14 project factors on a six-point ordinal scale. In addition, some researchers found correlations between Albrecht function point elements. Actually, [71] and [64] found different correlations, suggesting that predictive models based on the sum of the elements will not be stable for different datasets [70]. Also, the authors of [71] identified that some function point elements were not related to effort.

For instance, they verified that an effort prediction model based on only two function point elements (input function points and output function points) had a similar performance of an effort model based on total function points. Also, the authors achieved almost the same result using stepwise regression and raw counts of the number of files instead of the total function points.

Another criticism is about a low interrater reliability of FP counts, that is, whether two individuals measuring the size of the same system using FPA would generate the same FP count. However, the function points measurement interrater reliability appears to be sufficiently high, posing no practical barrier to their adoption in the industry [69].

2.4 COSMIC

COSMIC [40] [124] is a method for measuring COSMIC Function Points (CFP) that is considered the new generation of functional sizing [127]. It was developed in the COSMIC (Common Software Measurement International Consortium) project and it is an attempt to solve the problems of its main predecessor, FPA.

Its official documentation [40] includes the types of software for which the method has been designed to measure functional size (domain of applicability), the software models used for measurement and the COSMIC measurement process. COSMIC Functional Points Analysis is also an ISO certified functional sizing method [60].

Using COSMIC, we measure the functional size of a piece of software in three distinct and related phases:

1. Setting the measurement strategy using the principles of the Software Context Model (Table 2.5), stabilishing:

- The purpose of the measurement.
- The scope of each piece of software to be measured;
- The functional users and the boundary of each piece of software.
- The required level of granularity for the measurements.

2. Mapping the artifacts of the software to be measured onto the Generic Software Model (Table 2.6), which includes the following steps:
   - Identify the events of the functional users that the software must respond and then the functional processes.
   - Identify the data movements (Entries, Exits, Reads and Writes) of each functional process and data groups that are moved.
3. Measuring the specific elements of this model using a defined set of rules and processes.

Basically, the COSMIC size measurement is based on the following principle:

> "The functional size of a piece of software is directly proportional to the number of its data movements." (COSMIC Method, Version 3.0 [40])

Also, the measurement rules and process follows the characteristics listed in Table 2.7.

In addition to the same advantages of FPA, the COSMIC measurement method of functional size does not classify data using an ordinal scale. Also, it does not include weights to represent software complexity. For these reasons, it does not have most of the problems reported against of FPA. However, we did not found any published work that demonstrated the validity of the measure with respect to measurement theory.

2.5 Test Points Analysis (TPA)

Test Point Analysis (TPA) is method to measure the volume of testing work to be undertaken in a software development project [101]. This volume of work is expressed in a unit-of-work called Test Points (TP). The TPA measurement method is an extension of Function Point Analysis for the testing development phase.

The total number of test points of an information system is calculated by adding the number of test points necessary for testing the dynamic and the static measurable (testable) quality characteristics of the system.

Dynamic Test Points

The number of test points necessary for testing the dynamic measurable quality characteristics of the system assigned to each function of the system includes:

- The number of function points assigned to it using an adaptation of FPA or other two alternative models (FP).

- Influential factors divided into two categories:
  - Function-dependent (D_f): user-importance (Ue), usage-intensity (Uy), interfacing (I), complexity (C) and uniformity (U).
  - Quality requirements related to the dynamic quality characteristics to be tested (Q_d): suitability, security, usability and efficiency.

All function-dependent factors are rated in Low, Average or High according to some guidelines. Different weigths are associated for each factor. The following formula is used to calculate D_f:

$$D_f = \frac{Ue + Uy + I + C}{20} * U \tag{2.2}$$

where 20 is the sum of all nominal rate values of Ue, Uy, C and I.

The factors related to quality requirements are rated in 0, 3, 4, 5 or 6, according to is importance in the project. Q_d is calculated as follows. The rating for each factor is divided by four (the nominal rating), then multiplied by a weighting factor. Finally, Q_d is the sum of all these values added together.

Finally, the number of dynamic points of a function (TP_f) is obtained by the following formula:

$$TP_f = FP_f * D_f * Q_d \tag{2.3}$$

where FP_f is the number of function points assigned to the function f of the information system.

Static Test Points

The static measurable quality characteristics (Q_i) are based on the ISO 9126 quality characteristics [61]. For each quality characteristic to be observed during the test, the value sixteen is added to the Q_i factor rating.

Total Test Points

Finally, the total number of test points of an information system is calculated by the following formula:

$$TP = \Sigma TP_f + \frac{FP * Q_i}{500} \tag{2.4}$$

TPA includes an effort estimation model similar to the used in Function Point Analysis. It defines environment factors (test tools, test environment, development environment, etc.) that should be rated according some guidelines. The sum of these environment factors is then multiplied by the total number of test points. This adjusted counting is then multiplied by the

current test productivity (number of test hours required per test point). The author suggested that the productivity is generally a value between 0.7 and 2.0.

The result of a TPA is an estimate for the complete test process, excluding test planning. An effort distribution between phases can be done using historical data. According to his experience, the author reported the following effort distribution:

Although the author makes not clear how this model was built and calibrated, this model appears to have the same problems of FPA cited in 4.2.3.4. Also, we did not found any published work reporting the use of this model and presenting results from empirical studies, achieved accuracy, etc. For instance, there is no reference about in which situations this model was applied. For instance, the effort distribution shown in Table 2.8 may vary significatively when using MBT [111].

2.6 Other Measurement Methods

There are other extensions of function points analysis. MkII FPA [62] and NESMA FSM [103] are variations of FPA that are also certified by ISO as international functional size measurement standards. FAST Function Points [120] is an extension that tries to optimize the counting steps in order to reduce counting effort.

The Use Case Point Analysis (UCP) [97] is an extension of FPA and estimates the size of a system based on use case specifications. Object-Counts and Object-Points are object-based output measures [22].

2.7 Final Considerations

This chapter presented the most used software size measurement methods, which includes SLOC Counting, Function Points Analysis, COSMIC, Test Point Analysis and others. Each metric has its specifics advantages and limitations. For instance, SLOC is difficult to estimate in an early stage of development and FPA has several critiques about its construction process. Table 6.1 summarizes the main characteristics of these measurement methods.

In the next chapter, we review the existing estimation models and after that we present our proposed test size measure and its measurement method in Chapter 4.

Table 2.5 List of principles of the software context model defined in COSMIC.

Principles of the Software Context Model
• Software is bounded by hardware. • Software is typically structured into layers. • A layer may contain one or more separate 'peer' pieces of software and any one piece of software may further consist of separate peer components. • Any piece of software to be measured, shall be defined by its measurement scope, which shall be confined wholly within a single layer. • The scope of a piece of software to be measured shall depend on the purpose of the measurement. • The functional users of a piece of software shall be identified from the functional user requirements of the piece of software to be measured as the senders and/or intended recipients of data. • A piece of software interacts with its functional users via data movements across a boundary and the piece of software may move data to and from persistent storage within the boundary. • The functional user requirement (FUR) of software may be expressed at different levels of granularity. • The level of granularity at which measurements should normally be made is that of the functional processes. • If it is not possible to measure at the level of granularity of the functional processes, then the FUR of the software should be measured by an approximation approach and scaled to the level of granularity of the functional processes.

Table 2.6 List of principles of the generic software model defined in COSMIC.

Principles of the Generic Software Model
• The application receives input and produces output (or another outcome) for the functional users. • Functional user requirements can be mapped into unique functional processes. • Each functional process consists of sub-processes. • Sub-processes are data movements or a data manipulations. • Events cause functional users to trigger a functional process by an Entry data movement. • A data movement moves a single data group. • A data group consists of a unique set of data attributes that describe a single object of interest. • There are four types of data movement: – An Entry moves a data group into the software. – An Exit moves a data group out of the software. – A Write moves a data group from the software to persistent storage. – A Read moves a data group from persistent storage to the software. • A functional process shall include a minimum of two data movements, including at least one Entry data movement and either a Write or an Exit data movement. • For measurement purposes, data manipulation subprocesses are not separately measured. It is assumed to be accounted for by the data movement with which it is associated.

Table 2.7 Characteristics of the rules and processes used in the COSMIC measurement method.

Characteristics
• 1 CFP (COSMIC Function Point)is equivalent to a single data movement. • The functional size of a functional process is defined as the arithmetic sum of the number of its constituent data movements . • The functional size of any required functional change(s) to a piece of software is by convention the arithmetic sum of the number of its data movements that must be added, modified and deleted as a consequence of the required change(s). • The minimum functional size for a single functional process is 2 CFP, because the smallest functional process must have at least one Entry (as input), and either one Exit (as output) or one Write (as an alternative useful outcome). As a change may affect only one data movement, it follows that the minimum size of a change to a functional process is 1 CFP.

Table 2.8 Test effort distribution according to the author of Test Point Analysis.

PHASE	EFFORT DISTRIBUTION
Preparation	10%
Specification	40%
Execution	45%
Completion	5%
TOTAL	100%

Table 2.9 Summary of the software size measurement methods reviewed in this chapter.

	SLOC	FPA	COSMIC	TPA	No. of Tests	No. of steps
Artifact	Source code	Requirements	Requirements	Requirements	Test suite	Test specification
Attribute	Application size and complexity	Application size and complexity	Application size and complexity	Volume of test	Test suite size	Test size
Restriction	Language specific			Requirements written as use cases		
Availability	After implementation	After requirement specification	After requirement specification	After requirement specification	After test specification	After test specification
Standardized	There is some work in progress	Yes	Yes	Yes	Yes	Yes
Deterministic	Yes	No, result depends on the expertise in the measurement method	No, result depends on the expertise in the measurement method	No, result depends on the expertise in the measurement method	Yes	Yes
Measurement cost	None for languages supported by tools	High, since there is a lot of manual effort	Very high, since there is a lot of manual effort and it is more abstract method	High, since there is a lot of manual effort	None when supported by tools	None when supported by tools
Calibration cost	Not required	Significant, but not required	Significant, but not required	Significant, but not required	Not required	Not required
Required backgound	None when supported by tools	Knowledge in the measurement method and measured application	Knowledge in the measurement method and measured application	Knowledge in the measurement method and measured application	None	None
Automation	Yes	No	No	No	Yes	Yes
Validity	Same application can be implemented with significant different SLOC, depending on the language, developer, etc.	There are several critiques (see Section 4.2.3.4)	Result depends on who is measuring	Problems similar to FPA	Tests can have significantly different sizes and complexity	Steps can have significantly different execution complexities. Same test can have different measures depending on the level of detail used by the test designer

CHAPTER 3

State of Art in Effort Estimation

As presented in Chapter 1, we aim to develop and evaluate models that can be used for estimating test execution effort. This chapter presents the state of art in estimation models, including a general classification for the existing models. Due to the large number of models, we detail here only the models more related to our context.

3.1 A General Classification for Effort Estimation Models

There are several papers and books in the literature classifying and describing existing estimation models and techniques, such as [65], [26], [118], [106] and [98]. As discussed in [26], no single estimation technique can be considered the best for all situations. For this reason, it is important to know the strengths and limitations of these techniques to better select them to produce more realistic estimates.

We present in Table 3.1 a general classification that we considered appropriated for presenting the existing software estimation models. Then, we overview the main related estimation models in the next sections.

3.2 Productivity-Based Models

A simple way to estimate effort is to consider the productivity of historical databases, which is calculated observing the relation between the outputs (what is produced) and inputs (demanded effort). Although very limited, this method for making estimates still being used in the practice, probably due to its simplicity and lack of viable alternatives.

This estimation approach has a significant limitation. Estimates are accurate only the variance of the test productivity is small, that is, the productivity is constant. In other words, there is no other variables, such as the environment factors, influencing the effort.

3.2.1 Average Effort or Conversion Factor

This method uses historical average effort for doing activities (usually mean effort) as the basis for new estimates. For instance, we can estimate the time for executing tests based on the test productivity in previous projects using the following equation:

$$Effort = N_{current} * \frac{N_{previous}}{ExecutionTime_{previous}}$$

Table 3.1 General classification of existing estimation models.

Classification	Description	Example of Models
Productivity-Based	Simple models that are practically based only on the historical relation between output and input (e.g., SLOCs written per hour)	Average effort Conversion factor
Parametric	Models with mathematical algorithms or parametric equations. The parameters are usually related to the project under estimation.	COCOMO 81 SLIM
Statistical	Models created by using statistical techniques.	Regression models
Probabilistic	Models created by using probabilistic techniques.	Bayesian networks
Learning-Oriented	Models create by using machine learning techniques	Decision tree learning Case-Based Reasoning Rule Induction Neural Networks SVM
Expert-Based	Uses expert judgement for making estimates.	Delphi Wideband Delphi Planning Game Planning Poker WBS
Combined Techniques	Models created by a combination of techniques.	COCOMO II MP5 algorithm

where:

– $Effort$ is the estimated effort to execute the tests of the current project.

– $N_{current}$ is the number of tests of the current project.

– $N_{previous}$ is the number of tests of the previous project.

– $ExecutionTime_{previous}$ is the effort spent on executing tests of the previous project.

In this case, the effort is estimated by regarding the mean effort to execute a test. The generated estimates can be accurate only if productivity is almost constant.

This is also the approach used by Test Point Analysis (TPA) [101] and some other function points-based models. A conversion factor is calculated based on the inverse of productivity and it is used according to the following equations:

$$Effort = Points * CF$$

$$CF = \frac{Effort_{previous}}{Points_{previous}}$$

where:

- $Effort$ is the estimated effort to run the project.
- $Points$ is the number of points calculated for the current project.
- $Points_{previous}$ is the number of points calculated for the previous project.
- $Effort_{previous}$ is the effort spent on running the previous project.

3.3 Parametric Models

3.3.1 Putnam's Software LIfe-cycle Model (SLIM)

In the late 1970s, Larry Putnam of Quantitative Software Measurement [113] developed the Software Life-cycle Model (SLIM) [112]. SLIM describes the time and effort required to finish a software project of a specified size. Putnam noticed in their projects that software staffing profiles followed the well-known Rayleigh distribution.

In the practice, the effort is calculated by the following equation:

$$Effort_{total} = \left(\frac{Size}{Productivity}\right)^3 \frac{1}{Time^4}$$

where:

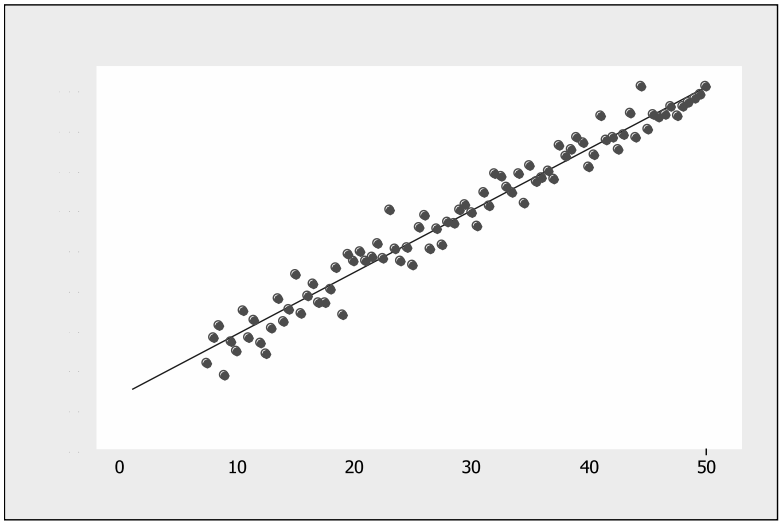
- $Effort_{total}$ is the estimated effort to run the project.
- $Size$ is a sizing measure for the project.
- $Productivity$ is the team productivity.
- $Time$ is the schedule for complete the project.

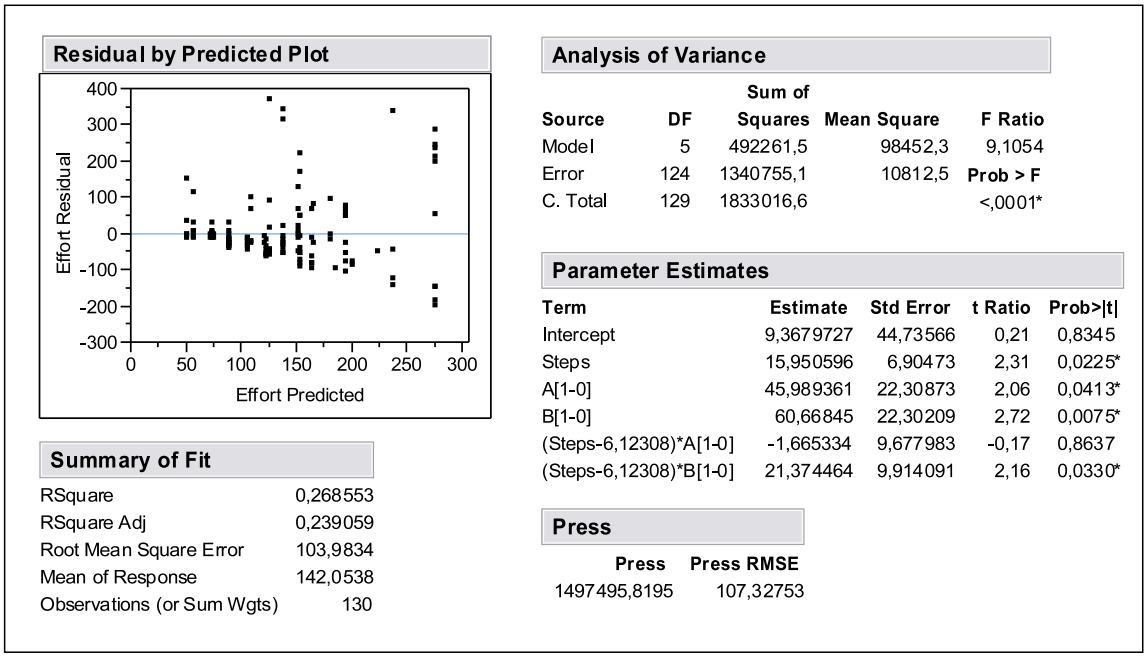
This method is highly sensitive to uncertainty in both project size and team productivity estimates. Also, we need to define the schedule to estimate the project effort.

3.4 Statistical Models

3.4.1 Regression Models

Regression analysis [51] is a statistical technique that investigates and models the relationship between dependent variables (response variables) and independent variables (predictor variables). This relationship is defined as mathematical model called regression equation. The


0
10
20
30
40
50



Analysis of Variance

Source	DF	Sum of Squares	Mean Square	F Ratio
Model	5	492261,5	98452,3	9,1054
Error	124	1340755,1	10812,5	Prob > F
C. Total	129	1833016,6		<,0001*

Summary of Fit

RSquare	0,268553
RSquare Adj	0,239059
Root Mean Square Error	103,9834
Mean of Response	142,0538
Observations (or Sum Wgts)	130

Parameter Estimates

Term	Estimate	Std Error	t Ratio	Prob>|t|
Intercept	9,3679727	44,73566	0,21	0,8345
Steps	15,950596	6,90473	2,31	0,0225*
A[1-0]	45,989361	22,30873	2,06	0,0413*
B[1-0]	60,66845	22,30209	2,72	0,0075*
(Steps-6,12308)*A[1-0]	-1,665334	9,677983	-0,17	0,8637
(Steps-6,12308)*B[1-0]	21,374464	9,914091	2,16	0,0330*

Press

Press	Press RMSE
1497495,8195	107,32753

Figure A.9 Regression analysis for initial model and raw data using M3 (Steps).

A.4 Estimation Accuracy Achieved During the Montecarlo Experiment.

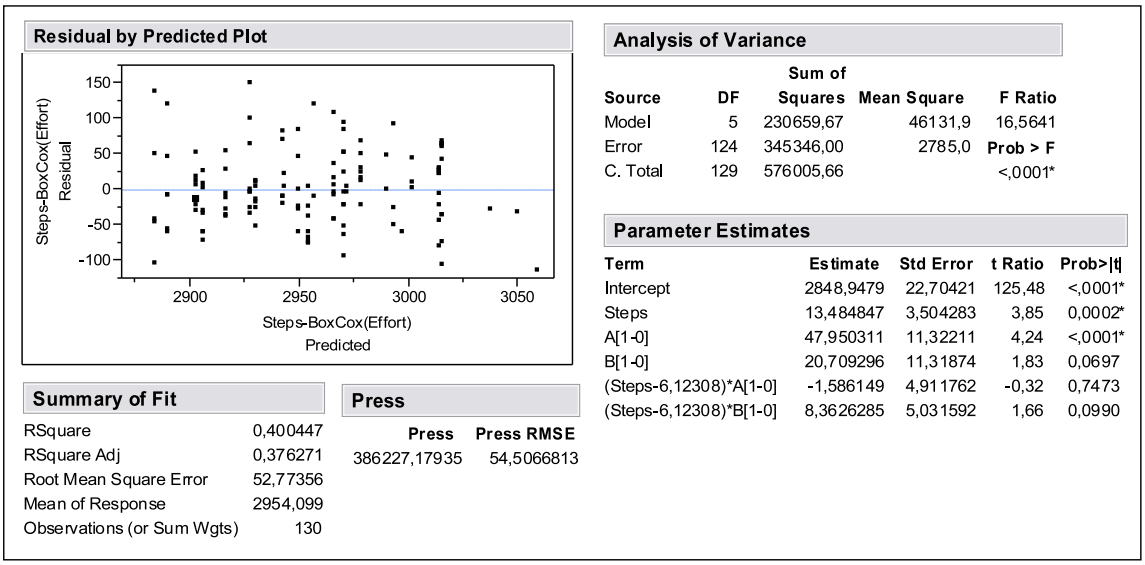


Figure A.10 Regression analysis for transformed data using M3 (Steps).

Table A.1 Person correlations between variables Effort, Keys, Screen, Delay and ListManip.

	Effort	Keys	Screen	Delay
Keys	0,338			
	0,000			
Screen	0,347	0,676		
	0,000	0,000		
Delay	0,639	0,259	0,406	
	0,000	0,003	0,000	
ListManip	0,300	0,778	0,137	0,256
	0,001	0,000	0,119	0,003

Cell Contents: Pearson correlation
P-Value

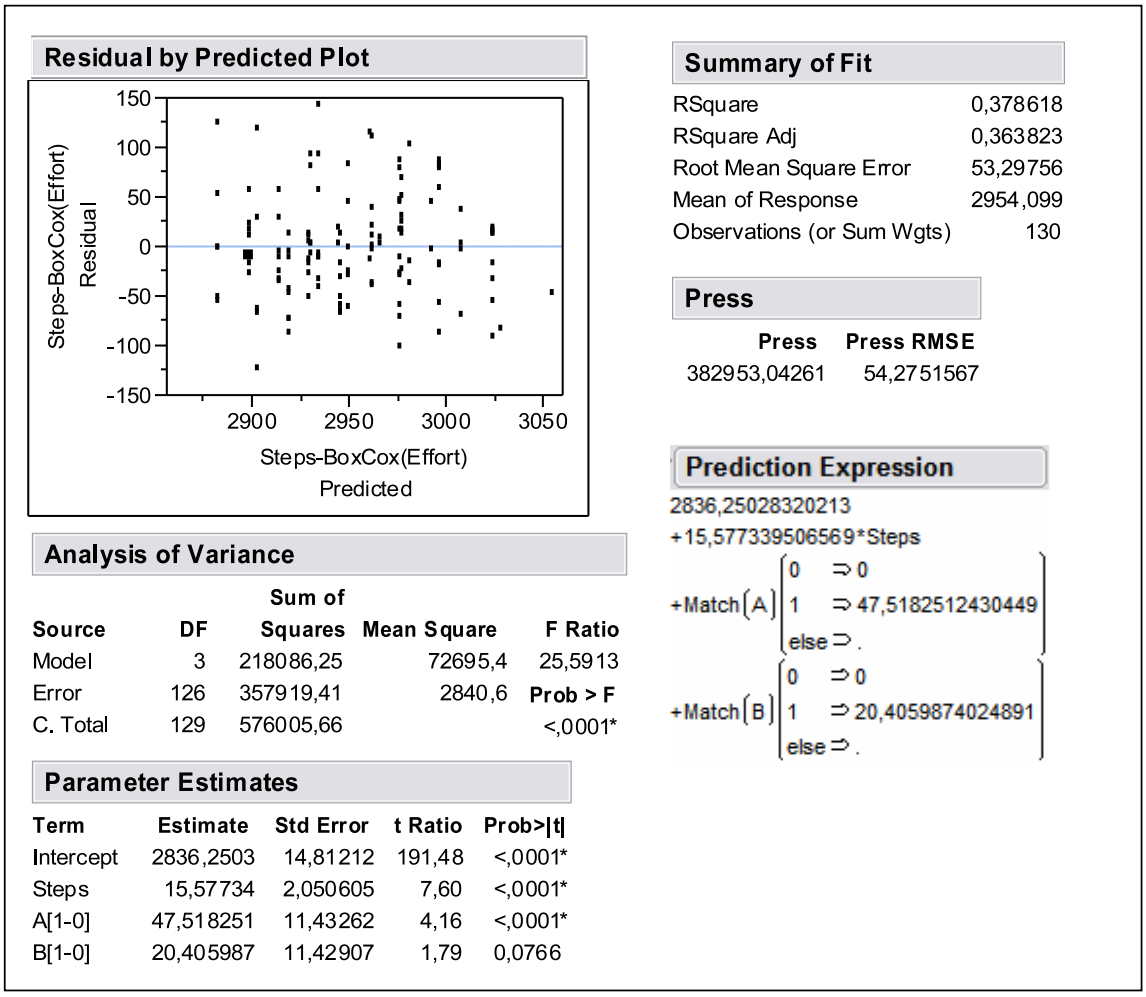
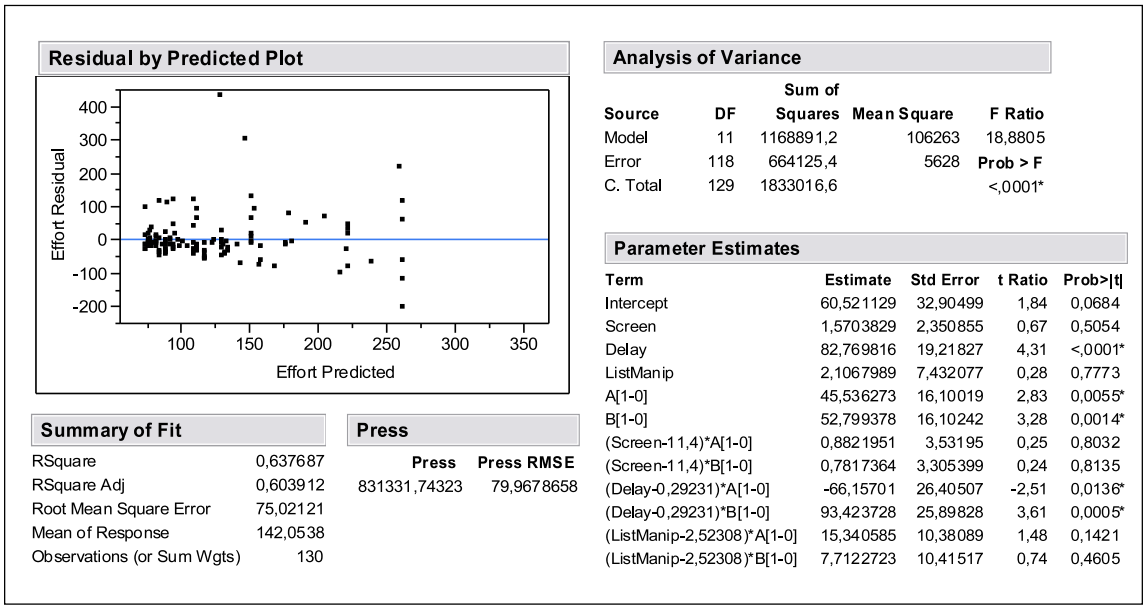


Figure A.11 Final regression analysis for model with measure M3 (Steps).



Analysis of Variance

Source	DF	Sum of Squares	Mean Square	F Ratio
Model	11	1168891,2	106263	18,8805
Error	118	664125,4	5628	Prob > F
C. Total	129	1833016,6		<,0001*

Summary of Fit

RSquare	0,637687
RSquare Adj	0,603912
Root Mean Square Error	75,02121
Mean of Response	142,0538
Observations (or Sum Wgts)	130

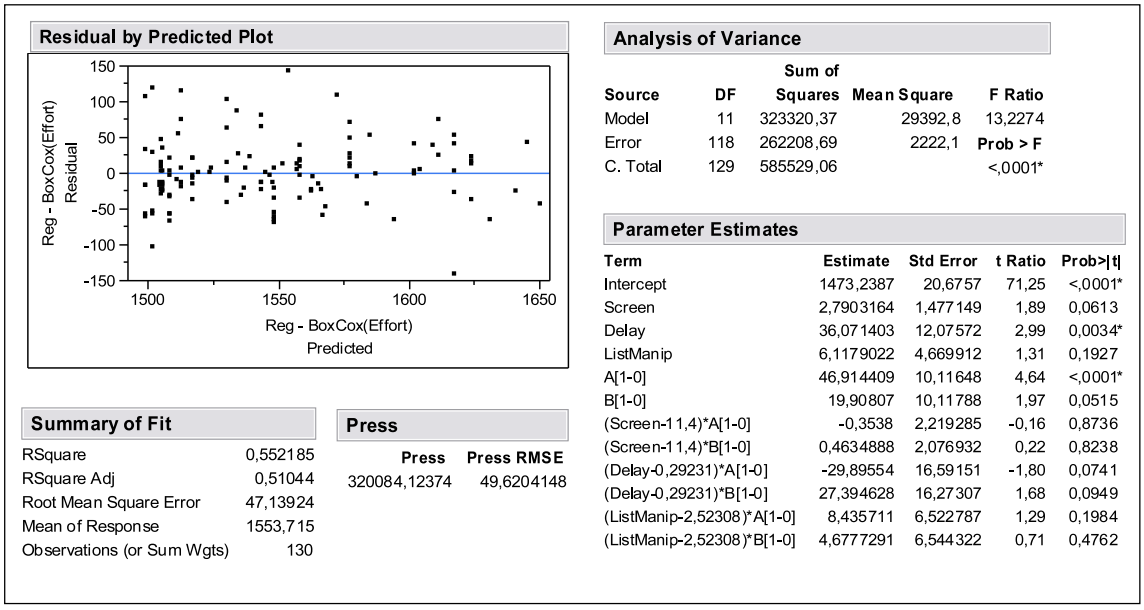
Press

Press	Press RMSE
831331,74323	79,9678658

Parameter Estimates

Term	Estimate	Std Error	t Ratio	Prob>|t|
Intercept	60,521129	32,90499	1,84	0,0684
Screen	1,5703829	2,350855	0,67	0,5054
Delay	82,769816	19,21827	4,31	<,0001*
ListManip	2,1067989	7,432077	0,28	0,7773
A[1-0]	45,536273	16,10019	2,83	0,0055*
B[1-0]	52,799378	16,10242	3,28	0,0014*
(Screen-11,4)*A[1-0]	0,8821951	3,53195	0,25	0,8032
(Screen-11,4)*B[1-0]	0,7817364	3,305399	0,24	0,8135
(Delay-0,29231)*A[1-0]	-66,15701	26,40507	-2,51	0,0136*
(Delay-0,29231)*B[1-0]	93,423728	25,89828	3,61	0,0005*
(ListManip-2,52308)*A[1-0]	15,340585	10,38089	1,48	0,1421
(ListManip-2,52308)*B[1-0]	7,7122723	10,41517	0,74	0,4605

Figure A.12 Regression analysis for initial model and raw data using M4 (Screen, Delay and ListMap).



Analysis of Variance

Source	DF	Sum of Squares	Mean Square	F Ratio
Model	11	323320,37	29392,8	13,2274
Error	118	262208,69	2222,1	Prob > F
C. Total	129	585529,06		<,0001*

Summary of Fit

RSquare	0,552185
RSquare Adj	0,51044
Root Mean Square Error	47,13924
Mean of Response	1553,715
Observations (or Sum Wgts)	130

Press

Press	Press RMSE
320084,12374	49,6204148

Parameter Estimates

Term	Estimate	Std Error	t Ratio	Prob>|t|
Intercept	1473,2387	20,6757	71,25	<,0001*
Screen	2,7903164	1,477149	1,89	0,0613
Delay	36,071403	12,07572	2,99	0,0034*
ListManip	6,1179022	4,669912	1,31	0,1927
A[1-0]	46,914409	10,11648	4,64	<,0001*
B[1-0]	19,90807	10,11788	1,97	0,0515
(Screen-11,4)*A[1-0]	-0,3538	2,219285	-0,16	0,8736
(Screen-11,4)*B[1-0]	0,4634888	2,076932	0,22	0,8238
(Delay-0,29231)*A[1-0]	-29,89554	16,59151	-1,80	0,0741
(Delay-0,29231)*B[1-0]	27,394628	16,27307	1,68	0,0949
(ListManip-2,52308)*A[1-0]	8,435711	6,522787	1,29	0,1984
(ListManip-2,52308)*B[1-0]	4,6777291	6,544322	0,71	0,4762

Figure A.13 Regression analysis with transformed data and using M4 (Screen, Delay and ListMap).

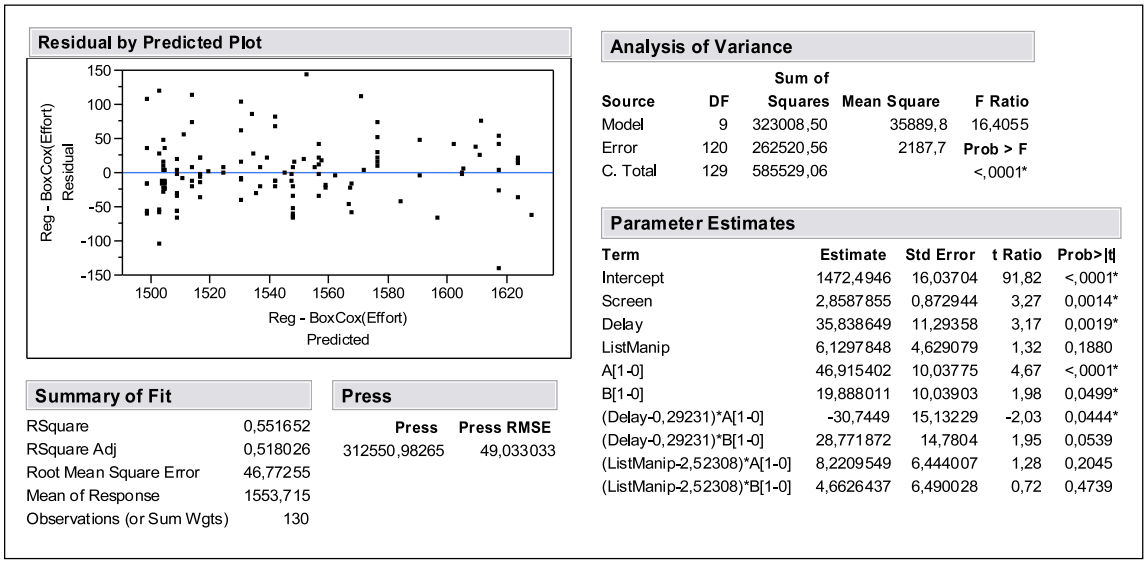


Figure A.14 Regression analysis after removing Screen and Tester interaction.

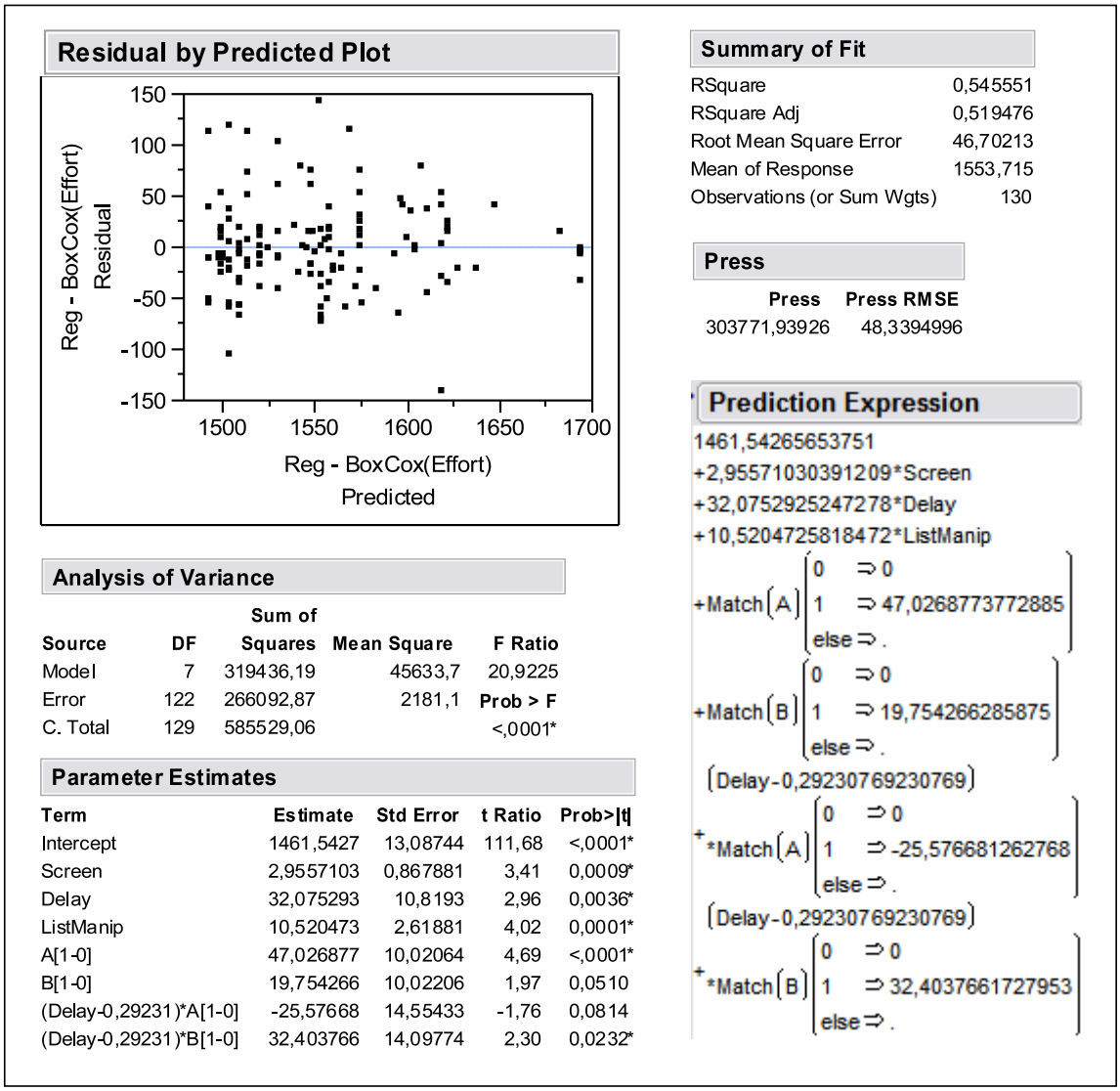


Figure A.15 Final regression analysis for model with the multiple measure M4 (Screen, Delay and ListMap).

Table A.2 Achieved estimation accuracy during the Montecarlo experiment.

Run	Fold	$MMRE_{M1}$	$MMRE_{M2}$	$MMRE_{M3}$	$MMRE_{M4}$	$MdMRE_{M1}$	$MdMRE_{M2}$	$MdMRE_{M3}$	$MdMRE_{M4}$	$PRED(25)_{M1}$	$PRED(25)_{M2}$	$PRED(25)_{M3}$	$PRED(25)_{M4}$
1	1	0.3231531	0.3222408	0.3879816	0.3302809	0.2200194	0.1979572	0.3521780	0.2263055	53.1914900	55.3191500	46.8085100	55.3191500
	2	0.3687061	0.2981505	0.3658667	0.2889180	0.3465122	0.2048523	0.3292313	0.2017730	33.3333300	54.7619000	35.7142900	52.3809500
	3	0.4055633	0.3379296	0.3699111	0.3416559	0.2116998	0.2276806	0.2326314	0.1611436	56.0975600	58.5365900	51.2195100	63.4146300
2	1	0.8082085	0.4755509	0.5601816	0.5659554	0.5032686	0.3914574	0.5579014	0.4838891	31.1111100	31.1111100	28.8888900	31.1111100
	2	0.3199700	0.2371512	0.2826592	0.2432853	0.2583015	0.1717762	0.2700247	0.1867208	46.6666700	64.4444400	48.8888900	64.4444400
	3	0.3356722	0.3321568	0.3161393	0.3310696	0.2510364	0.2045042	0.2303129	0.1914859	50.0000000	55.0000000	52.5000000	57.5000000
3	1	0.3831275	0.3329893	0.3941076	0.3067447	0.3269603	0.2508045	0.3459371	0.2489294	34.8837200	48.8372100	39.5348800	51.1627900
	2	0.4583230	0.3012411	0.3698476	0.4388989	0.2437702	0.2442298	0.2515116	0.2886696	50.0000000	50.0000000	50.0000000	45.6521700
	3	0.3795359	0.3676712	0.3772175	0.4019098	0.2693037	0.2354573	0.3393405	0.1941013	48.7804900	56.0975600	43.9024400	56.0975600
4	1	0.4031048	0.3676349	0.5026704	0.3501078	0.3893295	0.3319917	0.4402304	0.3077768	34.8837200	44.1860500	32.5581400	41.8604700
	2	0.2995103	0.3192169	0.3293302	0.4511147	0.2101507	0.1616278	0.2727008	0.1965387	56.8181800	61.3636400	45.4545500	61.3636400
	3	0.3827815	0.2694322	0.3518752	0.3119563	0.2814537	0.2255025	0.2350464	0.2067006	44.1860500	51.1627900	51.1627900	53.4883700
5	1	0.3424800	0.3139264	0.3552702	0.3135339	0.3292214	0.2657454	0.3435031	0.2706394	36.9565200	45.6521700	43.4782600	47.8260900
	2	1.2735840	0.3852773	0.5311836	0.5503391	0.2307896	0.2488098	0.3001957	0.2486581	51.2195100	51.2195100	41.4634100	51.2195100
	3	0.3532152	0.2818783	0.3545528	0.2671516	0.3106591	0.2212512	0.2848413	0.1788499	41.8604700	53.4883700	44.1860500	62.7907000
6	1	0.3447737	0.4228141	0.4718681	0.4021450	0.2750306	0.2550669	0.2977026	0.2075058	45.0000000	47.5000000	37.5000000	55.0000000
	2	0.3647691	0.3230157	0.3711690	0.3372393	0.3053829	0.2860536	0.2663031	0.2501254	44.4444400	48.8888900	44.4444400	48.8888900
	3	0.3878517	0.3227557	0.3900215	0.3434537	0.3464986	0.2398275	0.3363814	0.2487052	37.7777800	51.1111100	40.0000000	51.1111100
7	1	0.4608072	0.3619990	0.4216753	0.3633618	0.3921776	0.2508791	0.3764264	0.2126513	22.2222200	48.8888900	31.1111100	57.7777800
	2	0.6385491	0.3091850	0.4230127	0.3925942	0.2606970	0.2398228	0.3359569	0.2050851	44.1860500	51.1627900	44.1860500	51.1627900
	3	0.3411599	0.3127663	0.3504575	0.3353045	0.2417284	0.2376973	0.3268247	0.3280628	52.3809500	52.3809500	40.4761900	45.2381000
8	1	0.3695658	0.2587222	0.3295230	0.2906511	0.2771141	0.2011871	0.2310858	0.1933677	45.6521700	56.5217400	50.0000000	56.5217400
	2	0.3857831	0.3501629	0.4563051	0.3436118	0.2823556	0.2575570	0.3877016	0.2876184	40.9090900	47.7272700	36.3636400	47.7272700
	3	0.3267374	0.3459486	0.3473023	0.3361639	0.3431156	0.2687971	0.2817649	0.2494405	45.0000000	42.5000000	42.5000000	52.5000000
9	1	0.4301671	0.3028943	0.3894596	0.3312418	0.3558362	0.1837366	0.3083495	0.1903318	43.4782600	60.8695700	43.4782600	60.8695700
	2	0.3226473	0.3311261	0.3390044	0.3077798	0.2312563	0.2619608	0.2673120	0.2151440	55.8139500	48.8372100	46.5116300	53.4883700
	3	0.3144915	0.3095553	0.3894399	0.3119565	0.2561495	0.2484808	0.3495004	0.2929986	46.3414600	51.2195100	41.4634100	41.4634100
10	1	0.3782253	0.3664297	0.3812491	0.3597830	0.3231096	0.2229840	0.2447773	0.2167557	38.6363600	54.5454500	52.2727300	56.8181800
	2	0.3127602	0.2565278	0.4169600	0.2630931	0.3004206	0.1401137	0.3449308	0.1804112	44.1860500	58.1395300	41.8604700	58.1395300
	3	0.4101570	0.3468671	0.4223514	0.4198185	0.2979135	0.2759639	0.3597334	0.3319871	44.1860500	44.1860500	37.2093000	37.2093000
11	1	0.3160376	0.2793473	0.3033297	0.2610053	0.2313745	0.2377623	0.2185913	0.1983536	55.3191500	53.1914900	51.0638300	63.8297900
	2	0.3389720	0.4102436	0.4980558	0.4674766	0.2851647	0.2782390	0.3478130	0.3072740	48.7804900	46.3414600	43.9024400	43.9024400
	3	0.3936770	0.3627783	0.4049295	0.3330514	0.3146490	0.2562198	0.3129560	0.2243989	45.2381000	47.6190500	40.4761900	52.3809500
12	1	0.3816348	0.3171687	0.3983212	0.3064865	0.2831230	0.2482651	0.3224443	0.2284614	47.6190500	50.0000000	42.8571400	54.7619000
	2	0.2983577	0.2963749	0.2972389	0.2926312	0.2438710	0.2061782	0.2751800	0.2455689	52.1739100	56.5217400	43.4782600	52.1739100
	3	0.8484145	0.4034218	0.5098526	0.5961846	0.4151261	0.1941184	0.3734938	0.3124658	40.4761900	54.7619000	40.4761900	45.2381000
13	1	0.7571444	0.4241337	0.5973104	0.6566125	0.4910010	0.3104128	0.5666175	0.3904472	31.8181800	43.1818200	29.5454500	36.3636400
	2	0.3678053	0.2644300	0.3406024	0.2659470	0.3150985	0.2120615	0.3216466	0.2010791	35.5555600	57.7777800	37.7777800	62.2222200
	3	0.2801965	0.3021304	0.2680231	0.3132029	0.1874391	0.1762225	0.1981086	0.2047955	63.4146300	58.5365900	58.5365900	56.0975600
14	1	0.3924630	0.3583864	0.5338854	0.3678526	0.2589884	0.2561219	0.3697831	0.2356765	47.5000000	50.0000000	37.5000000	52.5000000
	2	0.3548780	0.3645307	0.3597169	0.3579502	0.2952041	0.3177460	0.3359592	0.3107713	44.4444400	37.7777800	42.2222200	44.4444400
	3	0.3304880	0.2630977	0.3303476	0.2619533	0.3003557	0.1950516	0.2379633	0.1610393	46.6666700	64.4444400	55.5555600	62.2222200

15	1	0.2904830	0.2710063	0.5005408	0.2726495	0.2135658	0.1400226	0.2869511	0.1578536	59.5238100	64.2857100	45.2381000	61.9047600
	2	0.4488122	0.3861752	0.4501162	0.4212459	0.3441601	0.2998069	0.4246570	0.2803793	31.9148900	42.5531900	31.9148900	46.8085100
	3	0.3759861	0.3410088	0.3687985	0.3158255	0.3190396	0.2304074	0.3407867	0.2108731	31.7073200	53.6585400	43.9024400	58.5365900
16	1	0.4171941	0.2782483	0.4411465	0.2641473	0.3083049	0.1846931	0.3473898	0.1697331	44.1860500	55.8139500	44.1860500	60.4651200
	2	0.4193076	0.4086050	0.4668451	0.4715510	0.3136406	0.2756592	0.3387610	0.3171086	34.0909100	45.4545500	40.9090900	43.1818200
	3	0.3740837	0.3850148	0.4059280	0.3796023	0.3225798	0.3383976	0.3372809	0.2882041	39.5348800	41.8604700	37.2093000	41.8604700
17	1	0.4033106	0.4042740	0.4196828	0.4092889	0.3050726	0.3298018	0.3670797	0.3641608	40.9090900	38.6363600	29.5454500	40.9090900
	2	0.4042971	0.2817891	0.3549501	0.2915481	0.3527203	0.1968417	0.2485510	0.1957725	43.1818200	56.8181800	50.0000000	54.5454500
	3	0.2720138	0.2507760	0.4042051	0.2282906	0.2235242	0.1522805	0.2639264	0.1581760	54.7619000	61.9047600	47.6190500	66.6666700
18	1	0.3740371	0.3521402	0.3761406	0.4383943	0.3241792	0.2455112	0.2903017	0.2509793	42.2222200	51.1111100	40.0000000	48.8888900
	2	0.3718238	0.3710872	0.3695392	0.3559424	0.3864646	0.2969627	0.3327533	0.2650990	43.1818200	43.1818200	43.1818200	47.7272700
	3	0.3015659	0.2864805	0.4564183	0.2952485	0.2292078	0.2195339	0.3297123	0.2334276	53.6585400	51.2195100	41.4634100	53.6585400
19	1	0.5374282	0.2989806	0.4302508	0.4579440	0.3352734	0.2237530	0.3344979	0.2678835	37.5000000	56.2500000	39.5833300	45.8333300
	2	0.3680821	0.3415138	0.4016134	0.3182518	0.2928886	0.2844749	0.2994710	0.1957172	45.6521700	47.8260900	41.3043500	54.3478300
	3	0.3795666	0.4095046	0.4342311	0.4025095	0.2548819	0.2586800	0.3387307	0.2724784	50.0000000	50.0000000	47.2222200	50.0000000
20	1	0.3689718	0.3083550	0.3653394	0.3087586	0.3209625	0.2453803	0.3267531	0.2279828	39.1304300	50.0000000	43.4782600	54.3478300
	2	0.6234981	0.3996726	0.4742701	0.4823132	0.4657378	0.3510290	0.4612770	0.4098534	36.1702100	42.5531900	36.1702100	38.2978700
	3	0.3005221	0.2604501	0.2931436	0.2350128	0.2192784	0.1759679	0.2123798	0.1682139	54.0540500	64.8648600	54.0540500	64.8648600
21	1	0.2512398	0.2579160	0.2765426	0.2697048	0.1711904	0.2195263	0.1744131	0.1764700	65.1162800	58.1395300	60.4651200	58.1395300
	2	0.3818387	0.4030685	0.3904774	0.5397877	0.3647787	0.2463772	0.3479498	0.2954381	40.9090900	50.0000000	40.9090900	45.4545500
	3	0.7020670	0.3423741	0.5197481	0.6846622	0.3415775	0.2783690	0.3670599	0.3978615	32.5581400	44.1860500	25.5814000	39.5348800
22	1	0.9147913	0.3443745	0.4297557	0.4273250	0.2816322	0.2872796	0.3077031	0.3060680	38.6363600	40.9090900	45.4545500	45.4545500
	2	0.4247661	0.3865697	0.4301702	0.3958360	0.3728182	0.2685263	0.3714420	0.2403840	36.3636400	47.7272700	36.3636400	52.2727300
	3	0.3095269	0.2660507	0.3072073	0.2692346	0.3098379	0.2302718	0.2796263	0.2307205	47.6190500	54.7619000	47.6190500	54.7619000
23	1	0.3087700	0.2855006	0.2928306	0.2806263	0.2168805	0.1917723	0.1912730	0.1930079	53.3333300	55.5555600	55.5555600	57.7777800
	2	0.3896682	0.3739602	0.4276404	0.3960767	0.3456354	0.2613371	0.3174709	0.2687629	41.3043500	50.0000000	41.3043500	47.8260900
	3	0.3799771	0.4246405	0.5135088	0.3606956	0.3396388	0.3408349	0.3613756	0.3091225	41.0256400	38.4615400	33.3333300	46.1538500
24	1	0.3054624	0.2665792	0.3127117	0.2723393	0.2258112	0.1948328	0.2442640	0.1866395	54.7619000	57.1428600	57.1428600	57.1428600
	2	1.3330650	0.3485674	0.6396040	0.5030265	0.3873722	0.3045933	0.4262326	0.3565426	31.1111100	44.4444400	35.5555600	40.0000000
	3	0.3583825	0.3403032	0.3323760	0.3452216	0.2374548	0.2457892	0.2337940	0.1609647	51.1627900	53.4883700	51.1627900	58.1395300
25	1	0.4923840	0.3962998	0.5084618	0.6437915	0.3911838	0.2830203	0.3577838	0.2650202	36.3636400	45.4545500	38.6363600	43.1818200
	2	0.3430444	0.3316746	0.3968287	0.3609079	0.2399020	0.2178125	0.2921784	0.2549675	50.0000000	52.1739100	39.1304300	50.0000000
	3	0.3186181	0.2631613	0.2897880	0.2682154	0.2684474	0.2261512	0.2617005	0.1943262	47.5000000	55.0000000	47.5000000	60.0000000
26	1	0.3441864	0.2963088	0.3687368	0.2755318	0.2839783	0.2252772	0.3056884	0.2171104	41.3043500	52.1739100	43.4782600	56.5217400
	2	0.3801342	0.3600816	0.3960429	0.3517967	0.2965076	0.2766621	0.2722739	0.2544760	37.7777800	44.4444400	46.6666700	48.8888900
	3	0.3786089	0.4934357	0.4116495	0.5697831	0.2578853	0.3171148	0.3096966	0.3139776	46.1538500	46.1538500	46.1538500	46.1538500
27	1	0.3585842	0.3038143	0.4223971	0.3170740	0.2908019	0.2166593	0.3673678	0.3025103	43.4782600	54.3478300	39.1304300	43.4782600
	2	0.3936629	0.2985101	0.3556246	0.3280930	0.3255773	0.1815890	0.2870713	0.1819533	43.4782600	63.0434800	41.3043500	60.8695700
	3	0.3295887	0.3680235	0.3598696	0.3507570	0.2153556	0.2845943	0.2360606	0.2600249	57.8947400	47.3684200	50.0000000	50.0000000
28	1	0.3362577	0.3058933	0.3649527	0.2682416	0.2682499	0.1832973	0.2392332	0.1633367	47.8260900	58.6956500	52.1739100	60.8695700
	2	0.3727300	0.2892695	0.3726712	0.2974676	0.3019123	0.2434624	0.3440066	0.2355123	45.2381000	50.0000000	42.8571400	50.0000000
	3	0.3913811	0.3417571	0.4048732	0.3463225	0.3458093	0.2266235	0.3463727	0.2464200	38.0952400	52.3809500	38.0952400	52.3809500
29	1	0.5000447	0.3425452	0.4802927	0.4932172	0.3999501	0.2558706	0.4429714	0.3587928	38.6363600	45.4545500	38.6363600	43.1818200
	2	0.3435460	0.3060727	0.3137854	0.3234526	0.2361788	0.1748290	0.2158636	0.1553292	52.2727300	65.9090900	52.2727300	65.9090900
	3	0.3096423	0.2722647	0.3054111	0.2520151	0.2502888	0.2095254	0.2975098	0.1870984	50.0000000	57.1428600	42.8571400	64.2857100

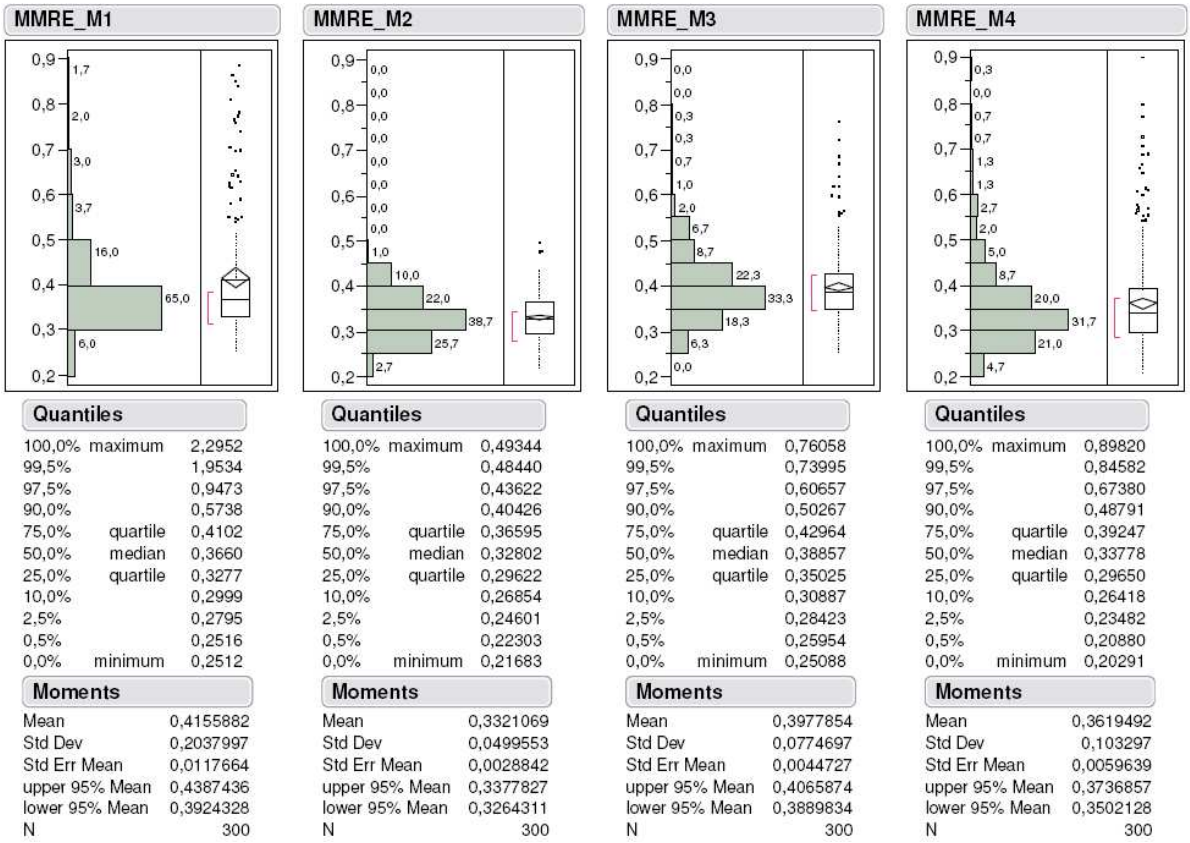
30	1	0.2833690	0.2832454	0.2812119	0.2765234	0.1875372	0.1657786	0.2235453	0.1989869	55.5555600	60.0000000	53.3333300	64.4444400
	2	0.6930475	0.4409406	0.6833792	0.5560549	0.4715069	0.3635104	0.5213627	0.3538105	30.2325600	39.5348800	32.5581400	39.5348800
	3	0.3031468	0.2801734	0.3238490	0.2850277	0.2302785	0.2050848	0.2132957	0.1649786	52.3809500	59.5238100	52.3809500	61.9047600
31	1	0.3134348	0.3692380	0.3344984	0.3598269	0.2613027	0.3110798	0.2705707	0.2827104	48.8888900	35.5555600	44.4444400	40.0000000
	2	0.3734707	0.3064326	0.3939990	0.3671665	0.2777410	0.2053310	0.3665319	0.2342816	47.6190500	59.5238100	40.4761900	52.3809500
	3	0.3780099	0.2936234	0.4477804	0.2952292	0.3582080	0.2317931	0.3775016	0.2224544	34.8837200	53.4883700	32.5581400	53.4883700
32	1	0.3206003	0.3365997	0.3589137	0.3617232	0.2839640	0.2745082	0.2596079	0.2686170	45.6521700	45.6521700	45.6521700	47.8260900
	2	0.4866637	0.3061124	0.4813852	0.4675444	0.2802344	0.2082908	0.3315499	0.2398751	46.6666700	60.0000000	40.0000000	51.1111100
	3	0.4083581	0.3430425	0.4044617	0.3360041	0.3389951	0.2386452	0.3343528	0.2160797	43.5897400	53.8461500	46.1538500	53.8461500
33	1	0.3639443	0.2906365	0.3185624	0.3233284	0.2893718	0.1709287	0.2819027	0.1881776	40.9090900	59.0909100	47.7272700	68.1818200
	2	0.5092185	0.3216261	0.4577393	0.4012983	0.3457513	0.2214786	0.3649441	0.3085567	41.3043500	54.3478300	36.9565200	45.6521700
	3	0.3531944	0.4062154	0.3948630	0.3907978	0.2787786	0.2866121	0.3230154	0.2707934	47.5000000	47.5000000	47.5000000	50.0000000
34	1	0.3872245	0.2927672	0.3828613	0.2961784	0.3588772	0.2736898	0.3249266	0.2312848	35.5555600	48.8888900	40.0000000	55.5555600
	2	0.2980375	0.2901695	0.3773043	0.2673053	0.2283433	0.2298492	0.2572913	0.2490905	55.5555600	55.5555600	48.8888900	51.1111100
	3	0.3994787	0.4178776	0.4123786	0.4237660	0.2538480	0.3458772	0.3783578	0.3378922	47.5000000	37.5000000	37.5000000	40.0000000
35	1	0.3485101	0.3198515	0.3830275	0.3147973	0.2635711	0.2682711	0.3274967	0.1963317	48.7804900	48.7804900	43.9024400	58.5365900
	2	0.3585062	0.3018215	0.3386403	0.3009672	0.2382187	0.2155205	0.2818950	0.2248350	52.1739100	54.3478300	45.6521700	52.1739100
	3	0.3512021	0.3664276	0.4323005	0.4082586	0.2856521	0.2333269	0.3144980	0.2743557	44.1860500	53.4883700	34.8837200	46.5116300
36	1	0.3494392	0.3175181	0.4165073	0.3043716	0.2679499	0.2180013	0.2952186	0.2394557	48.8372100	53.4883700	44.1860500	51.1627900
	2	0.4289240	0.3737027	0.4044823	0.3868890	0.3848649	0.2898872	0.3039190	0.2501850	36.9565200	47.8260900	30.4347800	50.0000000
	3	0.3277001	0.2820631	0.3539027	0.2806643	0.2432227	0.2324233	0.2979814	0.2124138	51.2195100	53.6585400	43.9024400	53.6585400
37	1	0.3423939	0.3182573	0.3861947	0.3440409	0.3122389	0.2400186	0.2883325	0.3292488	37.7777800	53.3333300	40.0000000	44.4444400
	2	0.3361501	0.3854553	0.3791369	0.3754219	0.2670313	0.2878582	0.3431034	0.3189152	48.8372100	37.2093000	37.2093000	37.2093000
	3	0.4080500	0.2977754	0.3667399	0.3315998	0.2444853	0.1909441	0.2495883	0.1597293	50.0000000	71.4285700	50.0000000	66.6666700
38	1	0.4094632	0.3564929	0.3953017	0.3450629	0.3345304	0.2864089	0.3764993	0.3050956	33.3333300	42.2222200	37.7777800	48.8888900
	2	0.2519506	0.3446561	0.3294185	0.4006743	0.1838864	0.1953442	0.1890097	0.2089161	65.2173900	60.8695700	58.6956500	56.5217400
	3	0.3662047	0.2977964	0.3921851	0.2947717	0.3288188	0.2132341	0.3142106	0.2126861	38.4615400	51.2820500	38.4615400	58.9743600
39	1	0.4177722	0.3446110	0.4239742	0.3364388	0.3378390	0.2712184	0.3832618	0.2749812	35.5555600	46.6666700	35.5555600	44.4444400
	2	0.2985433	0.2785681	0.2870263	0.3034691	0.2581886	0.2155529	0.2230703	0.2712917	48.8888900	57.7777800	51.1111100	48.8888900
	3	0.3428025	0.3986164	0.4891319	0.3968313	0.2559416	0.2448331	0.3497236	0.2516973	47.5000000	52.5000000	35.0000000	50.0000000
40	1	0.3495400	0.3858762	0.4241322	0.3788236	0.2564210	0.2744419	0.3581022	0.2316018	48.8888900	44.4444400	44.4444400	57.7777800
	2	0.2820709	0.2795667	0.2508831	0.2492312	0.2328352	0.2113983	0.1921624	0.1914922	52.3809500	54.7619000	57.1428600	59.5238100
	3	0.4409121	0.3167561	0.4375662	0.3390313	0.3910239	0.3005511	0.3840969	0.3039090	32.5581400	46.5116300	25.5814000	41.8604700
41	1	0.8370299	0.4238575	0.5546240	0.5210420	0.3744060	0.4036022	0.3535961	0.3708013	36.3636400	36.3636400	38.6363600	43.1818200
	2	0.3196623	0.2421846	0.3076349	0.2339196	0.2548419	0.1675777	0.2152227	0.1759846	48.8372100	60.4651200	51.1627900	60.4651200
	3	0.3422110	0.2837287	0.3511695	0.3001181	0.2817927	0.1818347	0.3023532	0.2161937	48.8372100	58.1395300	46.5116300	58.1395300
42	1	0.3461391	0.2701948	0.3105463	0.2485185	0.2770662	0.2057036	0.3002912	0.1751378	43.4782600	60.8695700	43.4782600	65.2173900
	2	0.4232825	0.3968237	0.4302298	0.3921070	0.3025430	0.3269384	0.3834751	0.2982242	38.6363600	43.1818200	40.9090900	47.7272700
	3	0.3283335	0.3004384	0.3987532	0.3160236	0.3035561	0.2011893	0.3553197	0.2800800	45.0000000	60.0000000	45.0000000	47.5000000
43	1	0.3744972	0.3474159	0.4052111	0.3458709	0.2991157	0.2875880	0.3510954	0.2388446	44.4444400	46.6666700	44.4444400	51.1111100
	2	0.3046342	0.2929095	0.2938761	0.2841256	0.2643946	0.2101754	0.2857507	0.2110668	45.4545500	56.8181800	43.1818200	54.5454500
	3	0.8835826	0.3303985	0.5127742	0.8981970	0.4034160	0.2487695	0.3290634	0.3825366	26.8292700	51.2195100	39.0243900	29.2682900
44	1	0.3180588	0.2546016	0.3041562	0.2606906	0.2329550	0.1623470	0.2042192	0.1969433	52.0833300	56.2500000	58.3333300	60.4166700
	2	0.3736585	0.3958080	0.4709125	0.3909272	0.3249426	0.2553936	0.4439730	0.2347462	41.8604700	46.5116300	32.5581400	53.4883700
	3	0.3924350	0.3141357	0.3467153	0.2849552	0.2721457	0.2698391	0.3002115	0.2497884	46.1538500	48.7179500	43.5897400	51.2820500

45	1	0.3646964	0.3466661	0.3502764	0.3047249	0.2803691	0.2092272	0.2373079	0.1654402	45.2381000	57.1428600	54.7619000	59.5238100
	2	0.2954406	0.2438265	0.2849636	0.2387744	0.2525128	0.1800699	0.2012093	0.1937823	48.9361700	63.8297900	51.0638300	65.9574500
	3	1.2138920	0.4043676	0.6674945	0.5415898	0.3557955	0.3760702	0.4572442	0.3835527	34.1463400	41.4634100	36.5853700	36.5853700
46	1	0.3573765	0.3667912	0.4844711	0.3497609	0.2519710	0.2205379	0.4164761	0.2753149	48.8372100	53.4883700	37.2093000	48.8372100
	2	0.3870369	0.2980134	0.3764762	0.2948557	0.3163812	0.2445686	0.2864887	0.2453954	40.9090900	52.2727300	40.9090900	50.0000000
	3	0.3431302	0.3307701	0.3403515	0.3706898	0.3004839	0.1988265	0.2918909	0.2168029	46.5116300	51.1627900	44.1860500	51.1627900
47	1	0.3480073	0.3239897	0.3494652	0.3367563	0.3393435	0.2751529	0.2723313	0.2670792	45.6521700	47.8260900	47.8260900	45.6521700
	2	0.2818555	0.2853387	0.3502404	0.2575568	0.2145042	0.2144474	0.2307708	0.1932026	60.0000000	57.7777800	51.1111100	62.2222200
	3	0.4168519	0.3211166	0.4355479	0.3703156	0.3806636	0.2181770	0.3752339	0.2108869	41.0256400	56.4102600	33.3333300	53.8461500
48	1	0.4776570	0.4262623	0.5000144	0.4115953	0.3971374	0.3531788	0.3626891	0.3251758	38.0952400	35.7142900	38.0952400	42.8571400
	2	0.3493451	0.4040998	0.3753392	0.3380636	0.2766208	0.3032733	0.2864593	0.2520937	45.4545500	43.1818200	43.1818200	50.0000000
	3	0.2841693	0.2792160	0.3308060	0.2956085	0.1795146	0.1437683	0.2125015	0.1867457	59.0909100	63.6363600	54.5454500	63.6363600
49	1	0.3823250	0.3783556	0.4012924	0.3698800	0.2955928	0.2652477	0.3169985	0.2846088	39.0243900	46.3414600	41.4634100	46.3414600
	2	0.3897459	0.3222902	0.4018754	0.3250870	0.3944305	0.2705539	0.2946955	0.2513523	31.2500000	47.9166700	37.5000000	47.9166700
	3	0.6273910	0.3115725	0.4903514	0.7670410	0.2045839	0.2209000	0.2807858	0.3523317	60.9756100	53.6585400	43.9024400	41.4634100
50	1	0.3011898	0.2671293	0.2911238	0.2577343	0.2035402	0.2115738	0.2284272	0.1764882	58.1395300	60.4651200	53.4883700	62.7907000
	2	0.7629380	0.3787962	0.5253933	0.4831045	0.4764328	0.2531188	0.3469107	0.2758163	34.7826100	50.0000000	34.7826100	45.6521700
	3	0.3307482	0.3103517	0.3561616	0.3015692	0.2972877	0.2691661	0.3466939	0.3043237	46.3414600	46.3414600	36.5853700	43.9024400
51	1	0.3325419	0.3182673	0.3413560	0.3109597	0.2522373	0.2108389	0.2490843	0.1870237	48.8888900	53.3333300	51.1111100	62.2222200
	2	0.8623596	0.4363446	0.6167958	0.4884441	0.4723434	0.3853933	0.5013774	0.3767512	32.6087000	30.4347800	23.9130400	34.7826100
	3	0.3777739	0.2578787	0.3195976	0.2346020	0.2967699	0.1794138	0.2424723	0.1829162	43.5897400	56.4102600	51.2820500	58.9743600
52	1	0.3379470	0.2684425	0.3044442	0.2741645	0.3146326	0.1521768	0.2553727	0.1852021	43.1818200	56.8181800	50.0000000	59.0909100
	2	2.2952330	0.3280128	0.4522834	0.5751354	0.3997054	0.2721249	0.3465319	0.2824563	33.3333300	46.6666700	37.7777800	44.4444400
	3	0.3968939	0.3815512	0.4022771	0.3855837	0.3114373	0.2564582	0.2939588	0.2385948	41.4634100	48.7804900	48.7804900	51.2195100
53	1	0.6168577	0.3052824	0.4069238	0.3115733	0.3057302	0.2513273	0.3001500	0.2508241	41.3043500	50.0000000	36.9565200	50.0000000
	2	0.4156757	0.4057802	0.3970704	0.3699827	0.3819782	0.2193539	0.3279420	0.2396545	44.1860500	51.1627900	44.1860500	51.1627900
	3	0.3371090	0.3311884	0.3732708	0.3318758	0.2501578	0.2065605	0.2278501	0.1947848	48.7804900	53.6585400	53.6585400	53.6585400
54	1	0.4177208	0.3173788	0.3647455	0.3246763	0.2833268	0.2866402	0.2884213	0.2767458	45.4545500	47.7272700	45.4545500	47.7272700
	2	0.3004346	0.3808072	0.3721697	0.3674183	0.1956311	0.2272725	0.2525537	0.2559421	60.8695700	52.1739100	47.8260900	50.0000000
	3	0.3780430	0.2921516	0.3891560	0.2747169	0.3572323	0.2226886	0.3307580	0.1943553	35.0000000	52.5000000	32.5000000	62.5000000
55	1	0.3617110	0.3025318	0.3587986	0.3204869	0.3396498	0.2292748	0.3106527	0.2032021	33.3333300	57.7777800	40.0000000	60.0000000
	2	0.5485970	0.3685521	0.5942407	0.5630981	0.2782342	0.3063868	0.3218867	0.2872325	45.6521700	45.6521700	41.3043500	45.6521700
	3	0.3532838	0.3385601	0.3615898	0.3320067	0.2702162	0.2180326	0.3023577	0.2628983	46.1538500	51.2820500	46.1538500	48.7179500
56	1	0.4831650	0.4083559	0.4721307	0.4179023	0.3843412	0.3508630	0.4014507	0.3476362	37.2093000	41.8604700	30.2325600	34.8837200
	2	0.3213175	0.2911832	0.3628796	0.3161809	0.2559081	0.2119930	0.2646545	0.2108083	48.8888900	53.3333300	44.4444400	55.5555600
	3	0.2838518	0.2604604	0.3623545	0.2145703	0.2349044	0.1985978	0.2542373	0.1477393	52.3809500	61.9047600	50.0000000	69.0476200
57	1	0.4005793	0.2861437	0.3987691	0.2930363	0.3592903	0.2599099	0.3600205	0.2421992	37.7777800	48.8888900	40.0000000	51.1111100
	2	0.2916518	0.3617607	0.4765699	0.3499719	0.2334459	0.2710598	0.3929733	0.3032751	50.0000000	47.6190500	42.8571400	42.8571400
	3	0.3491349	0.3187589	0.3300827	0.3310285	0.3085647	0.2218095	0.2660726	0.1977956	44.1860500	58.1395300	48.8372100	60.4651200
58	1	0.4018602	0.3139289	0.3623776	0.3573823	0.3662520	0.1965252	0.2607690	0.2164089	42.5531900	55.3191500	46.8085100	55.3191500
	2	0.3337689	0.3041560	0.4020999	0.2981016	0.2783665	0.2097132	0.3493247	0.2479333	46.5116300	58.1395300	41.8604700	51.1627900
	3	0.3391564	0.3402499	0.3663555	0.3486315	0.2590295	0.2949167	0.3070869	0.3135866	47.5000000	42.5000000	42.5000000	40.0000000
59	1	0.4498700	0.3317841	0.3917154	0.3325495	0.3480761	0.2650102	0.2687261	0.2243865	35.7142900	47.6190500	45.2381000	52.3809500
	2	0.3831006	0.2964351	0.3740240	0.3226434	0.3687319	0.2556685	0.3166982	0.2765100	35.4166700	50.0000000	37.5000000	39.5833300
	3	0.3038043	0.3459165	0.4026401	0.3456502	0.1904594	0.1597200	0.3027627	0.2169758	52.5000000	55.0000000	45.0000000	55.0000000

60	1	0.3589263	0.2871739	0.3517363	0.2947723	0.3028370	0.2291126	0.2973947	0.2333354	41.3043500	52.1739100	45.6521700	52.1739100
	2	0.2991415	0.3431870	0.3293367	0.3236304	0.2343984	0.2459972	0.2996721	0.2545594	53.4883700	51.1627900	46.5116300	46.5116300
	3	0.6134123	0.4395802	0.5226206	0.7944776	0.3404141	0.2485714	0.3638122	0.2645668	39.0243900	51.2195100	36.5853700	48.7804900
61	1	0.3396187	0.3620345	0.4138367	0.3815478	0.3415026	0.2553550	0.3204835	0.2348563	40.0000000	48.8888900	35.5555600	53.3333300
	2	0.3753746	0.3268695	0.3908199	0.3102815	0.2618977	0.2265235	0.3445931	0.2576401	50.0000000	52.1739100	43.4782600	47.8260900
	3	0.3783412	0.3068600	0.3275114	0.2852289	0.3103977	0.2631821	0.2612191	0.2096248	35.8974400	48.7179500	48.7179500	51.2820500
62	1	0.2975033	0.2445695	0.3351699	0.2516936	0.2647822	0.1520297	0.2411121	0.1504936	47.8260900	67.3913000	52.1739100	63.0434800
	2	0.3702619	0.3261278	0.3597973	0.3182472	0.2878155	0.2729122	0.2942035	0.2462102	44.4444400	46.6666700	48.8888900	51.1111100
	3	0.3814497	0.4396652	0.4509899	0.4451516	0.2933439	0.3573586	0.3782803	0.3175787	46.1538500	38.4615400	33.3333300	43.5897400
63	1	0.2781554	0.2802427	0.3087087	0.2794998	0.2254738	0.2253318	0.2382803	0.1745132	52.3809500	57.1428600	52.3809500	54.7619000
	2	0.4493243	0.3667502	0.4605924	0.5323613	0.3908908	0.3087118	0.3422935	0.3549524	28.2608700	43.4782600	34.7826100	32.6087000
	3	0.3607106	0.4133715	0.3935899	0.4461057	0.2620063	0.2156865	0.2710175	0.2186045	47.6190500	57.1428600	45.2381000	59.5238100
64	1	0.3208683	0.3263453	0.4072629	0.3060768	0.2396316	0.2167882	0.3528603	0.2445015	52.2727300	56.8181800	38.6363600	50.0000000
	2	0.2992704	0.2825270	0.3065466	0.2830754	0.2349761	0.2535385	0.2366836	0.2303322	51.1627900	48.8372100	51.1627900	55.8139500
	3	0.4101745	0.3532221	0.4040225	0.4389836	0.2897701	0.2217101	0.2831424	0.2365824	41.8604700	53.4883700	44.1860500	51.1627900
65	1	0.6409660	0.3287646	0.4062270	0.4685041	0.3083567	0.2336524	0.3451475	0.2225224	41.3043500	54.3478300	43.4782600	52.1739100
	2	0.3528045	0.3301349	0.3747853	0.3380706	0.2685175	0.2391098	0.2945604	0.2328995	47.8260900	52.1739100	43.4782600	54.3478300
	3	0.3795107	0.2647317	0.3387536	0.2498443	0.3063557	0.2244121	0.3139166	0.1643284	34.2105300	60.5263200	47.3684200	55.2631600
66	1	0.4292615	0.3329837	0.3754952	0.3017607	0.4004586	0.2525470	0.3166909	0.2453448	30.9523800	50.0000000	35.7142900	52.3809500
	2	0.3693333	0.3081351	0.4034973	0.2889771	0.2703260	0.2616951	0.2903629	0.2041599	48.8888900	46.6666700	40.0000000	57.7777800
	3	0.3372249	0.3525165	0.3803914	0.4554702	0.2847138	0.2256834	0.2967768	0.2948771	44.1860500	51.1627900	37.2093000	44.1860500
67	1	0.7796990	0.4361010	0.5634064	0.6639696	0.4025065	0.3062011	0.4801849	0.2965684	41.8604700	44.1860500	37.2093000	41.8604700
	2	0.3681397	0.3032667	0.3231127	0.2916153	0.2737978	0.2494775	0.2501962	0.2304561	47.7272700	50.0000000	50.0000000	54.5454500
	3	0.3111440	0.2789157	0.3442807	0.2748028	0.2501114	0.1796696	0.2455565	0.1624417	48.8372100	65.1162800	51.1627900	62.7907000
68	1	0.4472970	0.3848026	0.4234863	0.3814446	0.4232903	0.2241720	0.3440248	0.2149737	25.5814000	51.1627900	34.8837200	55.8139500
	2	0.2778869	0.2860347	0.3235853	0.2632478	0.2203392	0.2063015	0.2583811	0.1665342	54.3478300	54.3478300	47.8260900	63.0434800
	3	0.3615526	0.2975240	0.3803815	0.3308861	0.2236040	0.2444038	0.2692001	0.2231530	53.6585400	53.6585400	43.9024400	53.6585400
69	1	0.3722997	0.2573874	0.3388939	0.2581483	0.2739082	0.1789773	0.2837586	0.1946706	38.6363600	63.6363600	45.4545500	56.8181800
	2	0.3595004	0.4045980	0.3904144	0.4428626	0.3225070	0.3287225	0.3234238	0.3472849	40.0000000	40.0000000	35.5555600	42.2222200
	3	1.3797940	0.3144600	0.5026606	0.4625831	0.2815145	0.2322296	0.2826371	0.2674876	41.4634100	51.2195100	48.7804900	48.7804900
70	1	0.3748779	0.3254920	0.4638268	0.3160359	0.3262693	0.2135878	0.3306665	0.1590473	39.1304300	54.3478300	39.1304300	54.3478300
	2	0.4103353	0.3403302	0.4278954	0.3636980	0.2983943	0.2847131	0.3542297	0.3201655	47.6190500	45.2381000	42.8571400	47.6190500
	3	0.3231707	0.3139042	0.3321167	0.3098944	0.2575099	0.2288988	0.2796486	0.2406760	47.6190500	52.3809500	45.2381000	52.3809500
71	1	0.3859624	0.3670697	0.4070047	0.4449743	0.3273867	0.2739938	0.3222382	0.2913048	34.8837200	48.8372100	30.2325600	46.5116300
	2	0.5765127	0.2668863	0.4157799	0.6114376	0.2304649	0.2027045	0.2359255	0.2637420	55.5555600	60.0000000	51.1111100	48.8888900
	3	0.3483999	0.3586299	0.3530073	0.3422202	0.2337044	0.2499269	0.2581167	0.2034260	52.3809500	50.0000000	50.0000000	52.3809500
72	1	0.3901118	0.2473078	0.3716073	0.2529911	0.2676198	0.1806375	0.2427873	0.1361679	44.4444400	60.0000000	51.1111100	64.4444400
	2	0.4451124	0.4211556	0.4292926	0.3984997	0.3545732	0.2754781	0.3082434	0.2157195	32.5000000	45.0000000	42.5000000	57.5000000
	3	0.3180987	0.3395709	0.3782553	0.3467012	0.2324962	0.2315946	0.2898166	0.2549304	55.5555600	51.1111100	48.8888900	48.8888900
73	1	0.3893728	0.3596497	0.3856463	0.3915228	0.3222361	0.2620475	0.3118680	0.2283765	37.7777800	48.8888900	37.7777800	51.1111100
	2	0.3150695	0.3628259	0.4395888	0.3550312	0.2287800	0.2430316	0.3806946	0.2951521	53.4883700	51.1627900	44.1860500	46.5116300
	3	0.3758090	0.2758224	0.3262157	0.2760445	0.3210766	0.2125765	0.2511349	0.1814537	42.8571400	59.5238100	50.0000000	54.7619000
74	1	0.4408192	0.4275484	0.5080685	0.4353627	0.4675687	0.3718559	0.4787676	0.3543726	37.7777800	40.0000000	26.6666700	40.0000000
	2	0.3276963	0.2527155	0.3103334	0.2644848	0.2437359	0.2086773	0.1990446	0.1834132	53.3333300	57.7777800	55.5555600	60.0000000
	3	0.3789825	0.3669943	0.4334715	0.3781782	0.3123766	0.1869044	0.2717526	0.2140066	37.5000000	55.0000000	47.5000000	55.0000000

75	1	0.3518443	0.3267643	0.3700717	0.3384420	0.2128322	0.2703933	0.2717296	0.2715327	52.1739100	45.6521700	50.0000000	47.8260900
	2	0.3441817	0.3595755	0.4672426	0.3621040	0.2946620	0.2782576	0.3766045	0.2497635	38.0952400	47.6190500	33.3333300	50.0000000
	3	0.3451528	0.2678763	0.3196665	0.2578086	0.2723640	0.1829943	0.2115323	0.1640147	47.6190500	61.9047600	54.7619000	64.2857100
76	1	0.3168950	0.2719494	0.3003691	0.2621978	0.2544241	0.2386344	0.2576079	0.2381368	50.0000000	52.1739100	50.0000000	56.5217400
	2	0.5098264	0.4139428	0.4996717	0.4109360	0.4442790	0.3193911	0.4143888	0.2635226	26.8292700	43.9024400	29.2682900	46.3414600
	3	0.2786612	0.3001802	0.3662923	0.3195228	0.2425002	0.2157418	0.2353085	0.2354185	53.4883700	53.4883700	51.1627900	58.1395300
77	1	0.3705594	0.3768175	0.4328096	0.3544802	0.3098273	0.2944988	0.3369948	0.2312887	48.8888900	48.8888900	42.2222200	51.1111100
	2	0.2982211	0.2792717	0.3049353	0.2771756	0.1848281	0.1784835	0.2203430	0.1654296	56.8181800	61.3636400	54.5454500	61.3636400
	3	0.5489652	0.3200751	0.4804607	0.3309194	0.4206593	0.2319440	0.4025407	0.2391441	26.8292700	51.2195100	29.2682900	51.2195100
78	1	0.4103070	0.2961961	0.3734318	0.3078541	0.3649047	0.2846688	0.3514701	0.2617636	32.6087000	47.8260900	36.9565200	43.4782600
	2	0.2969537	0.3152579	0.3070875	0.3073901	0.1976189	0.2417746	0.2364946	0.2545917	56.0975600	51.2195100	53.6585400	48.7804900
	3	0.6507405	0.3583787	0.5069428	0.4990380	0.3110879	0.1967235	0.3743607	0.2609798	46.5116300	51.1627900	41.8604700	48.8372100
79	1	0.3652011	0.3218576	0.3808787	0.4080897	0.2644668	0.2295662	0.3199696	0.3256644	46.6666700	57.7777800	37.7777800	42.2222200
	2	0.3656974	0.2838384	0.4022473	0.2741104	0.2741083	0.1919828	0.3134740	0.1732542	50.0000000	56.8181800	47.7272700	56.8181800
	3	0.3489550	0.3495649	0.3658232	0.3270024	0.3112977	0.2400223	0.2862741	0.2315561	36.5853700	51.2195100	46.3414600	56.0975600
80	1	0.3350087	0.3540168	0.3243074	0.4271601	0.2125236	0.1901688	0.2469283	0.2132167	54.7619000	59.5238100	50.0000000	52.3809500
	2	0.4162316	0.3225048	0.4414844	0.3418157	0.3620774	0.2408391	0.3625847	0.2362723	36.9565200	54.3478300	41.3043500	52.1739100
	3	0.3238401	0.3518221	0.4047847	0.3201529	0.2773167	0.2796759	0.2879156	0.2474613	47.6190500	47.6190500	45.2381000	50.0000000
81	1	0.3132461	0.3219281	0.4072988	0.3068295	0.2572532	0.2590808	0.3131026	0.2477491	48.8372100	44.1860500	39.5348800	51.1627900
	2	0.3338442	0.3416417	0.3601557	0.3251565	0.2622921	0.2688364	0.2979329	0.2195781	45.6521700	47.8260900	43.4782600	54.3478300
	3	0.4416975	0.3351405	0.4088000	0.3574613	0.2779434	0.1972192	0.2322468	0.1879061	43.9024400	58.5365900	53.6585400	58.5365900
82	1	0.5156373	0.3177855	0.4837145	0.3574559	0.4287993	0.2506738	0.3887741	0.2559183	21.4285700	50.0000000	33.3333300	50.0000000
	2	0.2760978	0.2775517	0.3133152	0.2602771	0.2013365	0.1958442	0.2555416	0.2028712	55.5555600	53.3333300	48.8888900	62.2222200
	3	0.3449104	0.3755705	0.3914767	0.4291842	0.2552617	0.2693277	0.2966026	0.2385236	48.8372100	48.8372100	44.1860500	53.4883700
83	1	0.3833680	0.3332808	0.3504952	0.3099113	0.3305595	0.2765275	0.2892102	0.2231634	35.7142900	47.6190500	47.6190500	57.1428600
	2	0.3299765	0.3179602	0.3877605	0.3059258	0.3040851	0.2403707	0.3673302	0.2290136	40.4255300	53.1914900	40.4255300	53.1914900
	3	0.3436131	0.3712486	0.3639196	0.5347240	0.1707582	0.1747488	0.2418403	0.2320838	58.5365900	58.5365900	51.2195100	51.2195100
84	1	0.7386648	0.3889671	0.5580965	0.6041432	0.4130162	0.2890679	0.5065037	0.2980317	36.9565200	43.4782600	30.4347800	41.3043500
	2	0.3312184	0.2291037	0.2956699	0.2438644	0.2652925	0.1696607	0.1843328	0.1775113	45.0000000	70.0000000	60.0000000	62.5000000
	3	0.3503513	0.3525747	0.3741310	0.3928504	0.3131674	0.2487421	0.3465060	0.2414775	40.9090900	50.0000000	36.3636400	52.2727300
85	1	0.3768612	0.3616867	0.3325100	0.3847385	0.3111535	0.2505156	0.2833472	0.2370604	43.4782600	50.0000000	43.4782600	54.3478300
	2	0.3256865	0.3278298	0.4303042	0.3087094	0.2327500	0.1849687	0.3252923	0.1857343	54.3478300	56.5217400	45.6521700	63.0434800
	3	0.3940904	0.3299992	0.4083663	0.3426210	0.3108716	0.2861796	0.3739267	0.2837205	39.4736800	42.1052600	39.4736800	42.1052600
86	1	0.3167974	0.2928361	0.2975346	0.3026221	0.2636116	0.2110499	0.2146039	0.1830404	46.6666700	60.0000000	51.1111100	60.0000000
	2	0.4104827	0.3322111	0.4076245	0.3113143	0.4092717	0.3030238	0.3469314	0.2534439	37.7777800	44.4444400	35.5555600	48.8888900
	3	0.3099783	0.3280262	0.4508528	0.3374917	0.2126703	0.2544846	0.3560249	0.2780511	52.5000000	47.5000000	42.5000000	45.0000000
87	1	0.4183568	0.3411251	0.4204802	0.3411233	0.4089016	0.2541263	0.4030705	0.2411782	32.6087000	47.8260900	34.7826100	50.0000000
	2	0.3028325	0.3149739	0.3159096	0.3506093	0.1688538	0.2151799	0.2249206	0.2155911	60.0000000	57.7777800	53.3333300	62.2222200
	3	0.3166065	0.2866915	0.5132950	0.2943289	0.2439448	0.2252775	0.3380711	0.2425137	51.2820500	56.4102600	43.5897400	51.2820500
88	1	0.7718368	0.4235806	0.6187970	0.6861628	0.3708225	0.2793652	0.4249438	0.2687733	35.5555600	37.7777800	35.5555600	46.6666700
	2	0.3506788	0.3211887	0.3709625	0.3019963	0.3132216	0.2386384	0.2913132	0.2082914	43.1818200	52.2727300	43.1818200	54.5454500
	3	0.3115945	0.3308167	0.2834091	0.2955990	0.2425241	0.2077583	0.2108555	0.1746010	51.2195100	56.0975600	58.5365900	56.0975600
89	1	0.3410640	0.2432120	0.3390348	0.2548133	0.2358209	0.1711340	0.2315940	0.1601417	52.1739100	65.2173900	52.1739100	67.3913000
	2	0.3142235	0.3310879	0.4020273	0.3274192	0.2602367	0.2577539	0.3317733	0.2703573	47.7272700	47.7272700	40.9090900	45.4545500
	3	0.4323974	0.3871653	0.3934658	0.3717334	0.3816186	0.3328582	0.3439345	0.2744407	37.5000000	42.5000000	37.5000000	47.5000000

90	1	0.3489064	0.2614509	0.3389931	0.2824408	0.3145974	0.2027398	0.2924216	0.2032923	38.2978700	61.7021300	42.5531900	61.7021300
	2	0.3587134	0.3950344	0.5049008	0.4656816	0.2478583	0.2953421	0.2704891	0.2994968	51.1627900	48.8372100	48.8372100	39.5348800
	3	0.3472310	0.3739562	0.3639682	0.3510876	0.2502139	0.2366591	0.2344232	0.2334771	50.0000000	52.5000000	52.5000000	52.5000000
91	1	0.5459279	0.2770252	0.5053487	0.5523080	0.3605594	0.2174157	0.3633588	0.2616532	42.2222200	60.0000000	44.4444400	48.8888900
	2	0.3256194	0.3507663	0.3796206	0.4032296	0.2779118	0.2547676	0.2549110	0.3854722	48.9361700	46.8085100	48.9361700	40.4255300
	3	0.3746793	0.4169853	0.4003693	0.3556700	0.2407592	0.2658369	0.3353677	0.1792421	50.0000000	47.3684200	39.4736800	52.6315800
92	1	0.3599760	0.3360941	0.3751943	0.3523008	0.3295788	0.3245900	0.3715006	0.2887550	36.9565200	43.4782600	39.1304300	43.4782600
	2	0.3184263	0.3120139	0.4067715	0.2934151	0.2419332	0.1885242	0.2399954	0.1795101	51.1627900	58.1395300	51.1627900	60.4651200
	3	0.4246668	0.3559542	0.4267487	0.3964185	0.3492239	0.2547337	0.3560764	0.2734448	31.7073200	48.7804900	36.5853700	46.3414600
93	1	0.5871281	0.2787247	0.4045150	0.3706311	0.2258212	0.1934794	0.2969777	0.1890692	52.1739100	60.8695700	45.6521700	58.6956500
	2	0.3277498	0.3389597	0.3255038	0.3216296	0.2556537	0.2443670	0.2683391	0.2416849	48.8888900	53.3333300	48.8888900	51.1111100
	3	0.4227246	0.3809746	0.4203892	0.3625769	0.4002227	0.2617689	0.3917751	0.2561860	35.8974400	48.7179500	35.8974400	48.7179500
94	1	0.3083997	0.2168323	0.2754460	0.2029147	0.2337619	0.1875625	0.1935236	0.1443843	55.5555600	64.4444400	55.5555600	68.8888900
	2	1.6183180	0.4741985	0.7197203	0.7235415	0.4834169	0.3583806	0.5375341	0.4061069	28.5714300	45.2381000	26.1904800	38.0952400
	3	0.3177171	0.3237089	0.3462156	0.3381957	0.2986992	0.2493960	0.2923990	0.2525051	44.1860500	51.1627900	46.5116300	48.8372100
95	1	0.3786423	0.3987485	0.4467908	0.3612250	0.3237319	0.3355805	0.3251528	0.2310552	42.2222200	37.7777800	28.8888900	53.3333300
	2	0.2982706	0.2584363	0.2913406	0.2315553	0.2014879	0.1850492	0.1876637	0.1549019	57.1428600	64.2857100	59.5238100	71.4285700
	3	0.4385959	0.4319274	0.5063706	0.4673583	0.3233762	0.3805828	0.4608949	0.4109027	41.8604700	39.5348800	37.2093000	39.5348800
96	1	0.3124963	0.2799691	0.4074673	0.2251768	0.2211005	0.1913427	0.3528121	0.1747627	53.3333300	62.2222200	46.6666700	68.8888900
	2	0.3733452	0.3038699	0.3844355	0.3514599	0.2593387	0.2076535	0.2923772	0.2088785	48.8372100	55.8139500	41.8604700	55.8139500
	3	0.3764826	0.3848851	0.3905337	0.3734660	0.3466340	0.3545853	0.3851914	0.3192598	38.0952400	35.7142900	30.9523800	33.3333300
97	1	0.3867470	0.3992236	0.3858684	0.3804548	0.3486876	0.3196042	0.3146273	0.2534807	30.2325600	41.8604700	39.5348800	48.8372100
	2	0.2998694	0.3427810	0.3741658	0.3579449	0.2176048	0.2496498	0.2726929	0.2838181	53.1914900	51.0638300	44.6808500	48.9361700
	3	0.6951454	0.2959008	0.7605825	0.7025348	0.3983250	0.2672535	0.3922007	0.3303522	30.0000000	47.5000000	35.0000000	42.5000000
98	1	0.4334657	0.3254603	0.4207962	0.3351549	0.3429166	0.2828291	0.3567806	0.2478828	34.0425500	44.6808500	29.7872300	51.0638300
	2	0.3296130	0.3320504	0.3732084	0.3363737	0.2213248	0.1600684	0.2466010	0.1551253	51.1627900	58.1395300	51.1627900	58.1395300
	3	0.2851279	0.3423681	0.3679259	0.3744569	0.2405277	0.1998190	0.2309230	0.2716161	52.5000000	52.5000000	52.5000000	45.0000000
99	1	0.5419061	0.3829216	0.4937181	0.3785928	0.4231820	0.2516885	0.3857705	0.2384920	23.2558100	48.8372100	34.8837200	53.4883700
	2	0.3479886	0.3337721	0.3907765	0.3018272	0.2428926	0.2155485	0.3287908	0.2304648	52.2727300	54.5454500	38.6363600	50.0000000
	3	0.2955894	0.2868188	0.3207125	0.2922147	0.2091871	0.2118695	0.2001772	0.1879010	58.1395300	53.4883700	53.4883700	60.4651200
100	1	0.9832362	0.3449878	0.5071198	0.6074663	0.3754474	0.2490674	0.3652907	0.3445351	34.0909100	50.0000000	34.0909100	38.6363600
	2	0.3645273	0.3116204	0.3539551	0.3073810	0.3291429	0.2567671	0.3011672	0.2380785	32.6087000	47.8260900	39.1304300	52.1739100
	3	0.2960865	0.3526035	0.2981843	0.3785261	0.1899727	0.1718822	0.1952693	0.1570130	62.5000000	65.0000000	57.5000000	65.0000000



MMRE_M1 Quantiles		
100,0%	maximum	2,2952
99,5%		1,9534
97,5%		0,9473
90,0%		0,5738
75,0%	quartile	0,4102
50,0%	median	0,3660
25,0%	quartile	0,3277
10,0%		0,2999
2,5%		0,2795
0,5%		0,2516
0,0%	minimum	0,2512

MMRE_M1 Moments	
Mean	0,4155882
Std Dev	0,2037997
Std Err Mean	0,0117664
upper 95% Mean	0,4387436
lower 95% Mean	0,3924328
N	300

MMRE_M2 Quantiles		
100,0%	maximum	0,49344
99,5%		0,48440
97,5%		0,43622
90,0%		0,40426
75,0%	quartile	0,36595
50,0%	median	0,32802
25,0%	quartile	0,29622
10,0%		0,26854
2,5%		0,24601
0,5%		0,22303
0,0%	minimum	0,21683

MMRE_M2 Moments	
Mean	0,3321069
Std Dev	0,0499553
Std Err Mean	0,0028842
upper 95% Mean	0,3377827
lower 95% Mean	0,3264311
N	300

MMRE_M3 Quantiles		
100,0%	maximum	0,76058
99,5%		0,73995
97,5%		0,60657
90,0%		0,50267
75,0%	quartile	0,42964
50,0%	median	0,38857
25,0%	quartile	0,35025
10,0%		0,30887
2,5%		0,28423
0,5%		0,25954
0,0%	minimum	0,25088

MMRE_M3 Moments	
Mean	0,3977854
Std Dev	0,0774697
Std Err Mean	0,0044727
upper 95% Mean	0,4065874
lower 95% Mean	0,3889834
N	300

MMRE_M4 Quantiles		
100,0%	maximum	0,89820
99,5%		0,84582
97,5%		0,67380
90,0%		0,48791
75,0%	quartile	0,39247
50,0%	median	0,33778
25,0%	quartile	0,29650
10,0%		0,26418
2,5%		0,23482
0,5%		0,20880
0,0%	minimum	0,20291

MMRE_M4 Moments	
Mean	0,3619492
Std Dev	0,103297
Std Err Mean	0,0059639
upper 95% Mean	0,3736857
lower 95% Mean	0,3502128
N	300

Figure A.16 Analysis of MMRE distribution according to estimation models using measures M1, M2, M3 and M4.

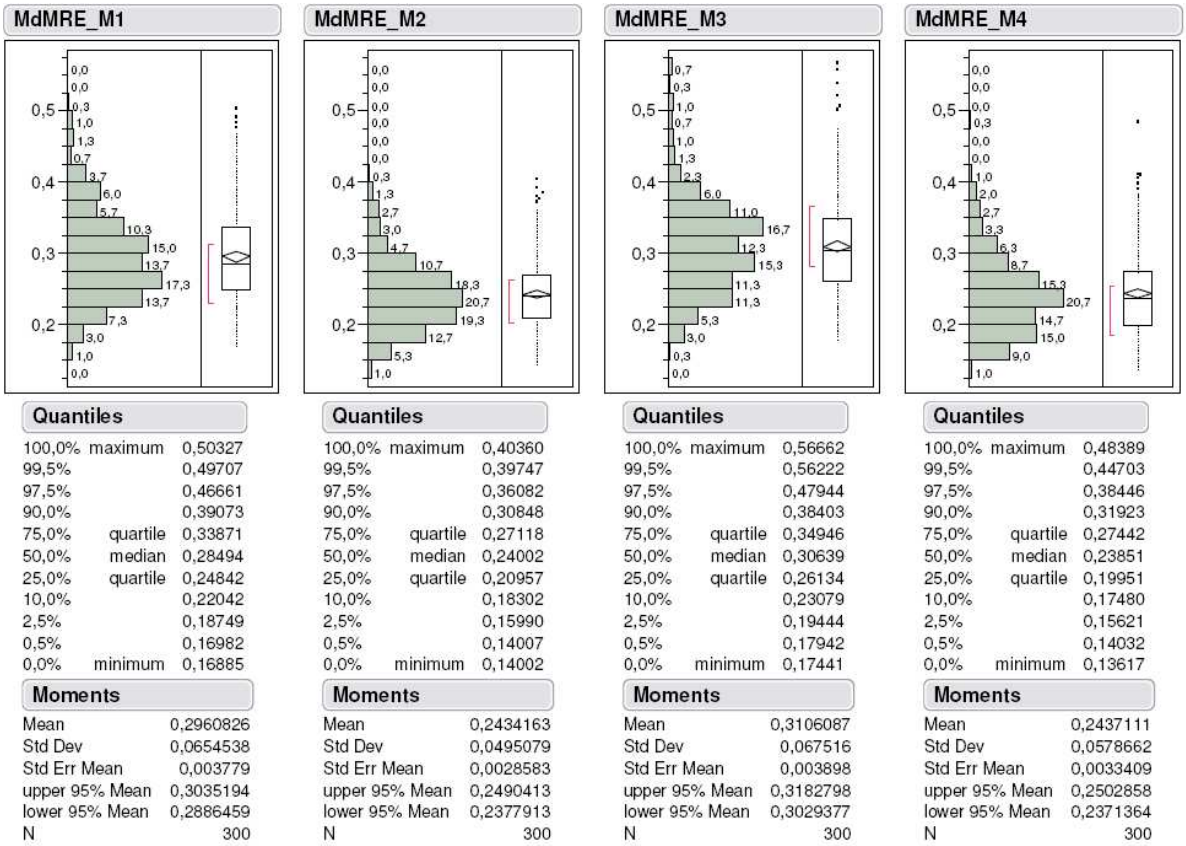
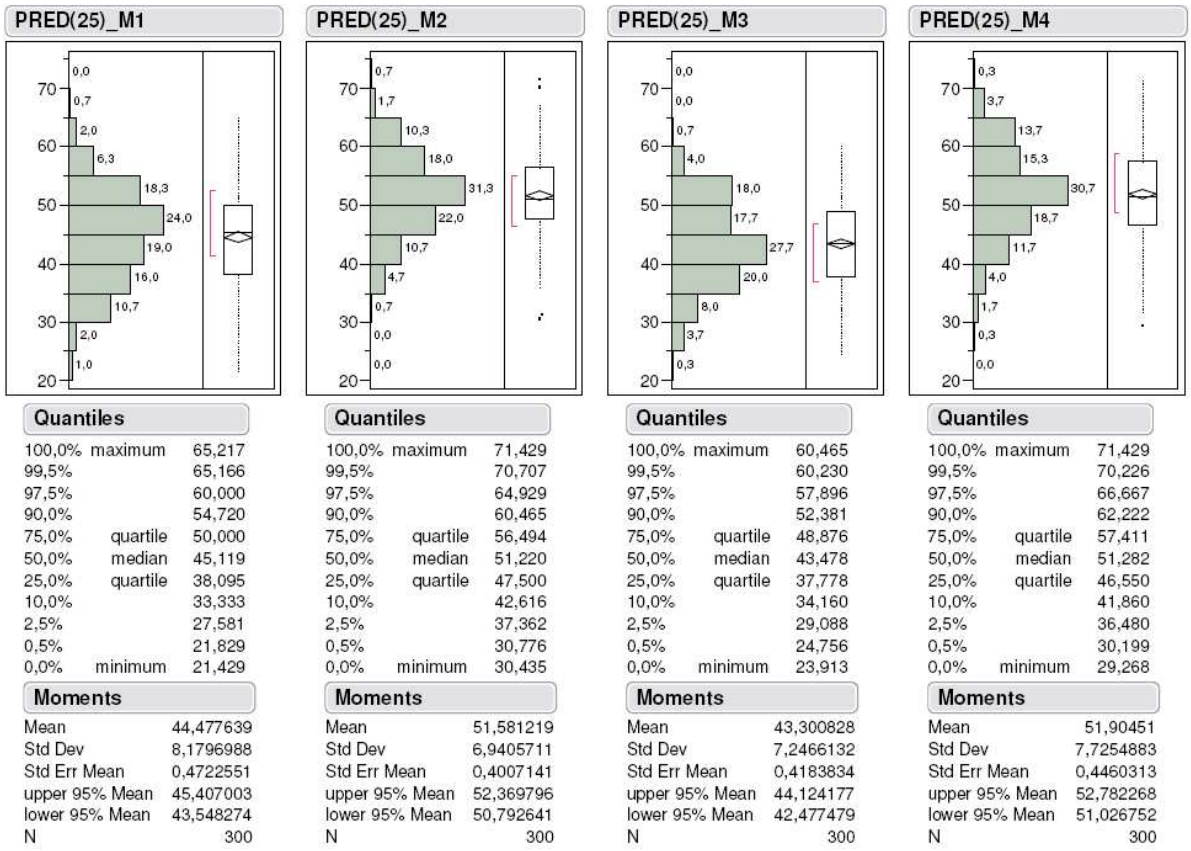


Figure A.17 Analysis of MdMRE distribution according to estimation models using measures M1, M2, M3 and M4.



PRED(25)_M1 Quantiles		
100,0%	maximum	65,217
99,5%		65,166
97,5%		60,000
90,0%		54,720
75,0%	quartile	50,000
50,0%	median	45,119
25,0%	quartile	38,095
10,0%		33,333
2,5%		27,581
0,5%		21,829
0,0%	minimum	21,429

PRED(25)_M1 Moments	
Mean	44,477639
Std Dev	8,1796988
Std Err Mean	0,4722551
upper 95% Mean	45,407003
lower 95% Mean	43,548274
N	300

PRED(25)_M2 Quantiles		
100,0%	maximum	71,429
99,5%		70,707
97,5%		64,929
90,0%		60,465
75,0%	quartile	56,494
50,0%	median	51,220
25,0%	quartile	47,500
10,0%		42,616
2,5%		37,362
0,5%		30,776
0,0%	minimum	30,435

PRED(25)_M2 Moments	
Mean	51,581219
Std Dev	6,9405711
Std Err Mean	0,4007141
upper 95% Mean	52,369796
lower 95% Mean	50,792641
N	300

PRED(25)_M3 Quantiles		
100,0%	maximum	60,465
99,5%		60,230
97,5%		57,896
90,0%		52,381
75,0%	quartile	48,876
50,0%	median	43,478
25,0%	quartile	37,778
10,0%		34,160
2,5%		29,088
0,5%		24,756
0,0%	minimum	23,913

PRED(25)_M3 Moments	
Mean	43,300828
Std Dev	7,2466132
Std Err Mean	0,4183834
upper 95% Mean	44,124177
lower 95% Mean	42,477479
N	300

PRED(25)_M4 Quantiles		
100,0%	maximum	71,429
99,5%		70,226
97,5%		66,667
90,0%		62,222
75,0%	quartile	57,411
50,0%	median	51,282
25,0%	quartile	46,550
10,0%		41,860
2,5%		36,480
0,5%		30,199
0,0%	minimum	29,268

PRED(25)_M4 Moments	
Mean	51,90451
Std Dev	7,7254883
Std Err Mean	0,4460313
upper 95% Mean	52,782268
lower 95% Mean	51,026752
N	300

Figure A.18 Analysis of PRED(25) distribution according to estimation models using measures M1, M2, M3 and M4.

APPENDIX B

Attempt to Validate Cost Drivers Using Historical Data

After identifying cost drivers that may significantly impact test execution effort, we run a case study to evaluate the use of the cost drivers to estimate test execution effort. The details of this study are presented next.

B.1 Planning

We planned this case study using the Goal/Question/Metric approach [23] and defining the procedures to be executed, as presented next.

B.1.1 Goals, Questions, Metrics and Hypotheses

This study has the following main goals:

G1: Verify if the identified cost drivers can improve test execution effort estimation accuracy.

G2: Compare the performance of a test execution effort estimation model when regarding and disregarding cost drivers.

The assessment of these goals are performed by answering the following research questions:

Q1: Which cost drivers significantly improve the test execution effort estimation accuracy?

Q2: How more accurate are the estimates when regarding cost drivers?

As described in the next section, we build and analyze estimation models using statistical techniques. One of these statistical techniques (stepwise regression) identifies the cost drivers that impact test execution effort with statistical significance, answering the research question Q1.

For answering the research question Q2, we analyze the estimation accuracy of the created estimation models using standard accuracy measures [121]:

MMRE: Mean magnitude of the relative error.

MdMRE: Median magnitude of the relative error.

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

The statistical significance of the observed estimation accuracy is analyzed through the test of the following statistical hypotheses:

1) The mean magnitude of the relative error is smaller when regarding cost drivers (*reg*) than when disregarding them (*disreg*).

$$H_{01} : MMRE_{reg} \geq MMRE_{disreg}$$

$$H_{11} : MMRE_{reg} < MMRE_{disreg}$$

2) The median magnitude of the relative error is smaller when regarding cost drivers than when disregarding them.

$$H_{02} : MdMRE_{reg} \geq MdMRE_{disreg}$$

$$H_{12} : MdMRE_{reg} < MdMRE_{disreg}$$

3) The percentage of estimates that are within 25% of the actual values is larger when regarding cost drivers than when disregarding them.

$$H_{03} : PRED(25)_{reg} \leq PRED(25)_{disreg}$$

$$H_{13} : PRED(25)_{reg} > PRED(25)_{disreg}$$

B.1.2 Historical Data Analysis

In this study, we analyze the historical data of the industry on the mobile application domain that sponsored the survey described in Section 5.6. The analyzed database had approximately six-month data containing information about the execution of more than 300 test suites, which means the execution of thousands of different test cases by more than 30 testers of different test teams.

To achieve our goals, this study is organized in three parts. In the first part, we analyze and prepare the data, removing inconsistencies and some outliers, as well using expert opinion to gather missing values. Then, the second part of this study uses stepwise regression (SWR) [51] to find the best set of cost drivers (predictors) to explain the variation in the test execution effort (response).

SWR is a statistical technique that builds a regression model, which is a mathematical equation relating independent (predictors) and dependent (response) variables. We use the forward and backward method, that is, the regression model is created by adding and possibly removing independent variables one-at-a-time until some stopping rule is satisfied. In each step of this analysis, a variable can be added to the model if it is considered statistically significant.

Also, variables can be removed from the model if they become nonsignificant after the inclusion of other variables.

After that, the last part of this study performs a cross-validation analysis to evaluate and compare the accuracy to estimate test execution effort when using the set of selected cost drivers and when disregarding it. 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 and enable us to test our statistical hypotheses.

B.2 Execution

In the next subsections, we detail the main activities executed during this study and their results. To avoid bias during the presentation of these results, our interpretation of them are presented later, in Section B.3.1.

B.2.1 Gathering Historical Data

In the historical database, each record stored information about the executed tests, tested product, testers and related information. However, the information about most of the identified cost drivers were not stored, since they were not being used at that time. For this reason, we had to gather information about the cost drivers in the analyzed period of time.

We analyzed the possibility to obtain past information for each cost driver described in Table 5.18. After talking to test experts, managers and people that maintain the infrastructure (test environment, test tools, etc.), we verified that we were able to obtain reliable past information only for the cost drivers presented in Table B.1. Information about other cost drivers were not completely available, too costly to be obtained or only indirectly available through some of the variables presented in Table B.2. We used these variables related to some cost drivers in an attempt to capture some effect of them.

Our first task to obtain past information about cost drivers was to organize and consolidate the data to identify which tests were executed during the considered period, who executed the tests, what phones were tested and so forth. After that, we analyzed each cost driver individually, as shown next.

Since we did not have access to all phone performance specifications, we invited seven experienced testers to assess phone performance using their previous experience in testing them. We adapted the Delphi [79] and Card Sorting [45] techniques to reach a consensus about the performance of the phones tested during the analyzed period.

We asked seven participants to classify the phone performance into three levels (low, average and high). They received some slips of paper, each one containing the model of one phone. Then, we run some trials in which they had to group the phones according to the three levels we defined. At the end of each trial, we removed the slips of the phones which performance achieved a consensus. We verified that this process was very fast and not boring for the participants. After three trials, we achieved the consensus about the performance of the phones tested

Table B.1 Cost drivers obtained by analyzing past projects information and expert opinion.

Description	Abbreviation	Scale	Range or Scale points	Source of information
Phone performance	PhonePerf	ordinal	Low, Avg, High	Expert assessment.
Quality of the test cases	QualityTC	ordinal	Low, Avg, High	Expert assessment.
% of Testers with Average experience	ExpAvgPerc	ratio	[0, 100]	Historical records of training and team allocation.
% of Testers with High experience	ExpHighPerc	ratio	[0, 100]	Historical records of training and team allocation.
% of Testers with Average English skills	EngAvgPerc	ratio	[0, 100]	English class level and team allocation.
% of Testers with High English skills	EngHighPerc	ratio	[0, 100]	English class level and team allocation.

in the past.

We used the same approach to reach a consensus about the quality of the test cases. As the tests related to each family of products have similar quality, the participants classified the tests of each family of products in low, average or high. Also, the english skill of the testers were assessed based on their current level of english class. Regarding the tester experience in testing, we determined it based on their job start date.

B.2.2 Stepwise Regression

The stepwise regression analysis have some assumptions that must be satisfied. First, the relation between independent and dependent variables is expected to be linear. Also, it is assumed that the residuals (predicted minus actual values) are normally distributed. We verified that the numerical variables used in this study were not normally distributed. For instance, a simple linear regression analysis between test execution effort and execution points built a model with residuals not normally distributed. For this reason, we transformed these numerical variables into a natural logarithmic scale using the box-cox transformation technique [51] [85] in order to approximate the data to normal distributions.

For each numerical variable that required data transformation we created a new variable using the letter *L* and the original variable name. For instance, we created the variable $LEffort$ for storing the transformed data of the variable $Effort$. After all this process, we verified that the linear correlation between the transformed effort data ($LEffort$) and the other transformed data from independent variables were improved.

In this study, some variables considered have ordinal scales, such as the tested applica-

Table B.2 Other related variables available in historical databases.

Description	Abbreviation	Scale	Range or Scale points	Related cost drivers
Effort to execute the tests	Effort	ratio	Positive numbers	-
Number of execution points	EP	ratio	Positive numbers	-
Number of tests	NumOfTests	interval	Positive integers	-
Name of the tested product	Product	nominal	A, B, C, ..., M, N, O	Stability
Product Family of the tested product	ProductFamily	nominal	F1, F2, F3, F4, F5, F6, F7	Performance
Technology of the tested product	Tech	nominal	T1, T2	Feature Experience

tion, the model of the tested hardware and the tester's language skill. For including them in the regression model, we created $(n-1)$ dummy variables [51] for each of these categorical variables, where n is the number of different values (scale points) that the variable can take. For instance, the variable *Product* requires the creation of 14 dummy variables (*Product_A* to *Product_N*) for representing 15 different products (from letter A to O). The dummy variables can take the values 0 (not present) and 1 (present). For example, the variable *Product_A* takes value 1 only when A is the tested product. When variables *Product_A* to *Product_N* take the value 0, it means that the tested product is O.

After that, we run the stepwise regression with α = 0.15 to add and to remove variables, the most common used values in the practice. We verified that some of the selected independent variables were high correlated (multicollinearity problem). For instance, the independent variables number of tests and execution points are highly correlated. However, a high degree of multicollinearity produces large variance in the regression coefficient estimates.

To avoid multicolinearity, we calculated the Pearson's correlation coefficient for measuring the linear relationship between the independent variables and also between independent and dependent variables. Hence, when the SWR selected two independent variables with high linear correlation, we discarded the one less correlated with the dependent variable (effort) and repeated the stepwise regression analysis. In this way, we run the stepwise regression until all selected independent variables were not highly correlated between them.

The SWR selected 5 variables with α at 0.15. However, after running a multiple regression analysis (independent variables are fixed), we verified that one of these variables (*LEngSkillAvPerc*) were not significative with α at 0.05. We discarded this variable and then used the multiple regression analysis to build the following equation:

$$LEffort = -3.91 + 1.06LEP - 0.0586LExpHighPerc$$

$$-0.0989 ProductFamilyD + 0.08 ProductB \quad \text{(B.1)}$$

The output of the multiple regression analysis is presented next.

```
Predictor            Coef   SE Coef       T      P
Constant          -3,9000    0,2249  -17,34  0,000
LEP                1,05855   0,02390   44,29  0,000
LExpHighPerc    -0,057588  0,009053   -6,36  0,000
ProductFamilyD   -0,09927   0,02318   -4,28  0,000
ProductB          0,08038   0,01915    4,20  0,000


S = 0,139850   R-Sq = 86,8%   R-Sq(adj) = 86,6%


Analysis of Variance

Source           DF      SS      MS       F      P
Regression        4  40,361  10,090  515,91  0,000
Residual Error  314   6,141   0,020
Total           318  46,502


Source            DF  Seq SS
LEP                1  38,831
LExpHighPerc       1   0,622
ProductFamilyD     1   0,564
ProductB           1   0,345
```

As we can see, the adjusted R^2 achieved was 86.6%. We confirmed the regression analysis assumptions and the model stability by:

- Using the normal probability plot in Figure B.1, the residual plot to in Figure B.2 and other graphs to verify if the residuals were random and normal.
- Calculating the Cook´s distance values [85] to identify influential data points. Twelve test project with distance higher than 3 x (4/n) were removed to test the model stability. After running the regression analysis with the new data set, the coefficients remaining stable and the adjusted R^2 was improved in almost 1%.

Then, we transformed back the regression equation to the following equation:

$$Effort = EP^{1.06}(ExpHighPerc + 1)^{-0.0586} \\ e^{(-3.91 + 0.0989 ProductFamilyD + 0.08 ProductB)} \quad \text{(B.2)}$$

B.2.3 Cross-validation Analysis

The last part of this study is the cross-validation analysis. We run a 10-fold cross-validation, that is, we partition the data into 10 folds. Then, each of these folds are used to test the models (test set) built using the others folds (training set). In each iteration of the cross-validation analysis, we run the multiple regression analysis to build two models based on the training data set:

Figure B.1 Normal probability plot for the regression model.

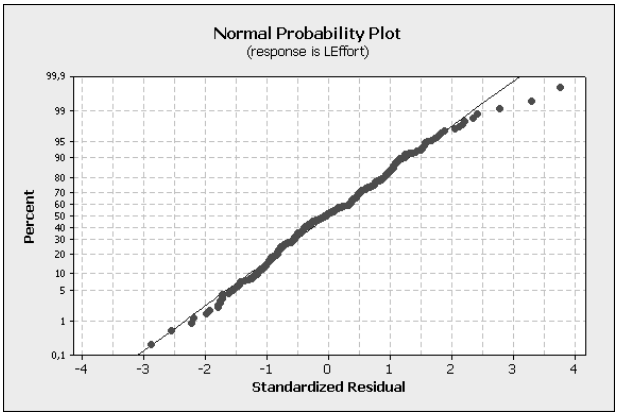
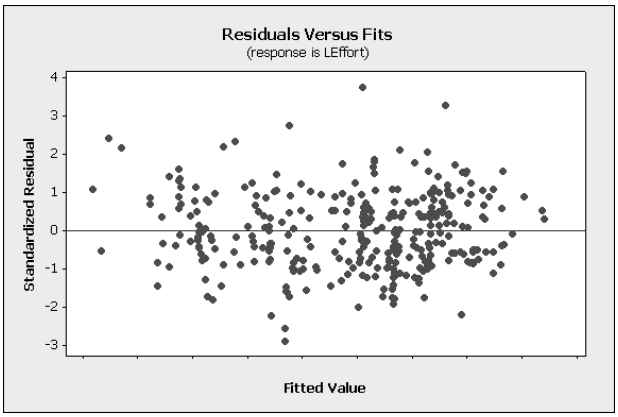


Figure B.2 Residual plot for the regression model.



Model-1: Model that regards cost drivers. This model consider all the variables selected by the stepwise regression analysis.

Model-2: Model that does not regard cost drivers. This model only consider the independent variable *LEP*, since it only represents the size and complexity of the tests to be executed.

After that, we use the test set to evaluate the accuracy of both models using the measures defined in Section B.1.1. We repeated the cross-validation analysis three times with different partitioning. Table B.3 summarizes the mean accuracy achieved by each model in the 30 iterations.

Table B.3 Mean of the MMRE, MdMRE and PRED(25) observed in the cross-validation analysis.

Estimation model	MMRE	MdMRE	PRED(25)
With cost drivers	0,1169	0,1015	92,48%
No cost drivers	0,1171	0,0906	90,59%

Finally, we tested our statistical hypotheses using the paired t-test with α at 0.05. In all cases, the test did not found enough evidence to reject the null hypotheses (all p-values were larger than 0.05).

B.3 Final Considerations

B.3.1 Interpretation of Results

During the stepwise regression, the achieved adjusted R^2 was greater than 70%, which is considered good, but yet smaller than the ideal (close to 95%). This means that there are missing cost drivers in the model. Indeed, this result was expected due the lack of historical data about them. The normal probability and the residual plots shows that the residuals are normally distributed and with constant variance, as assumed by the regression analysis.

Observing the equation generated by the regression, we can see that *LEP* is the most important variable due to its highest coefficient and range of values that it can take. This result was also expected, since the project size is usually the variable with more impact on project effort. However, the coefficient of the other variables are too small when compared to the *LEP* coefficient. Also, the range of values of the others variables are also smaller, leading to a small impact on the effort estimates. This fact means that the variables did not have strong relationship with the cost drivers that are affecting the effort.

On the other hand, the signs used in the coefficients of each variable are intuitively consistent. Basically, the effort increases with the size of the test project (*LEP*) and if the tested product is B (a newer product with low stability). Also, the effort decreases with the percentage of high experienced testers and when the tested product is from the family D (high performance).

With respect to the cross-validation, the accuracy achieved by both models were similar. This fact was confirmed by the statistical hypotheses tests, which did not found any statistical significant difference between the accuracy of the two models.

B.3.2 Threats to Validity

Construct validity: In this case study, we faced with the lack of historical data about the identified cost drivers. We used measures that can be indirectly related to the cost drivers under investigation. However, some of these variables may not have the expected relation with the cost drivers.

Internal validity: The use of the testers to create some missing values based on expert opinion may bias the results, since they may feel that their productivity are under evaluation too. To control this threat, we carefully explained the goals of the study.

Some variables in the historical database were used as indirect measures for cost drivers. These variables may not represent a causal relationship with the response variable (effort). To control this threat, we reviewed the variables selected by the stepwise regression analysis.

Conclusion validity: We were not able to support our hypothesis that the use of cost drivers related to testing improves the estimation accuracy of test execution effort. One of the possible causes is the missing of the relationships between the response (effort) and cost drivers which impact were not captured by none of the independent variables.

External validity: The findings of this study may not be generalized due to the lack of information about the cost drivers. Others studies are required to achieve more representative results.

APPENDIX C

Test Execution Effort Estimation Tool

This appendix describes a tool that we developed to support an automated estimation of functional test execution effort based on the measurement of size and execution complexity of tests or test suites. This tool is also discussed in [19].

C.1 Functionalities

This section presents the main functionalities provided by our tool, which uses the model described in [11] and [15] to estimate the execution effort of a given test suite based on the test specifications.

Natural Language Processing. To automatically estimate the effort to execute test suites, the tool must be able to process test specifications, which are usually written in natural language. Figure C.1 presents an example of test specification written in natural language.

Our tool has a natural language processing mechanism that identifies the test action represented by each test step (i.e., sentence in the test specification) based on the main verb of the sentence. Figure 1 shows some examples of test actions on the mobile application domain, such as the selection of a menu option (cell B2) and the insertion of content in form fields (cells B3 and B4).

Test Size and Execution Complexity Measurement. To estimate test execution effort, the tool measures the size and execution complexity of the test cases in execution points. The number of execution points of a test or test suite shows how big and complex it is. This measure is calculated by analyzing the test actions found in test specifications according to specific system characteristics. On the mobile application domain, examples of these characteristics

	A	B	C
1	**Step**	**Description**	**Expected results**
2	1	Start the message center.	The phone is in message center.
3	2	Select the new message option.	The phone is in message composer.
4	3	Insert a recipient address into the recipients field.	The recipients field is filled.
5	4	Insert a SMS content into the message body.	The message body is populated.
6	5	Send the message.	The send message transient is displayed. The message is sent.
7			

Figure C.1 Sample spreadsheet with test specification written in natural language.

are the number of screens to navigate the number of keys to press and delays in the application response time. More details on execution points and its measurement method are presented in [11].

These characteristics represent some general functional and non-functional requirements of the system under test that are exercised when the test is executed. The number of execution points assigned to a test action depends on its impact on the test execution complexity (Low, Average or High). Finally, the number of execution points of a test suite is the sum of execution points assigned to each test action found in the test specifications.

Test Execution Effort Estimation. Several models can be used to estimate test execution effort based on the number of execution points and cost drivers [4]. These cost drivers are factors usually related to the testing team and environment, such as team experience and tool support. Our tool allows the use of different estimation models, such as regression models or productivity-based estimation models. Regression models are equations that can regard execution points and the effect of the cost drivers to estimate the test execution effort. A productivity-based estimation is based on the average time per execution point spent during the tests execution (execution time / execution points).

Model Configuration. Before estimating test execution effort, it is necessary that the user configures the tool with the characteristics that should be used to measure the size and execution complexity of test specifications. In addition, the user needs to indicate which cost drivers should be used when estimating test execution effort. For both characteristics and cost drivers, the user has to indicate their impact (influence levels and weights) with respect to test execution complexity and effort. All this information may be specific to some application domains and it can be determined through Delphi panels [79]. Figure C.2 shows an example of the configuration of system characteristics for the mobile application domain.

C.2 Tool Architecture

We developed the tool in Java using the architecture shown in Figure C.3. In summary, the tool consists of the following components: Reader, Natural Language Processing, Model Configuration, Effort Estimation and Persistence Manager. These components are detailed next.

Reader: It is responsible for reading the files containing test specifications, extracting the test steps from each test specification. For that, this component has to manipulate the internal structure of the files.

Natural Language Processing: It is responsible for identifying the test action represented by each test step. First, the tool parses the sentence and identifies its main verb. Then, the tool converts the verb to its infinitive form, which is used to identify the test action. Next, the verb in the infinitive form is validated against a database that stores thousands of existing verbs of the English language. If the verb is not recognized, we consider that either the parsing or the conversion to infinitive form has failed. Since the English language is dynamic, the user is always enabled to add new verbs to the database. Validated verbs are then considered as test actions. When the tool finds a test action, it verifies if the test action is already stored in the test action database. This database contains all test actions previously processed by the tool. If the test action is not found in the database, it is stored.

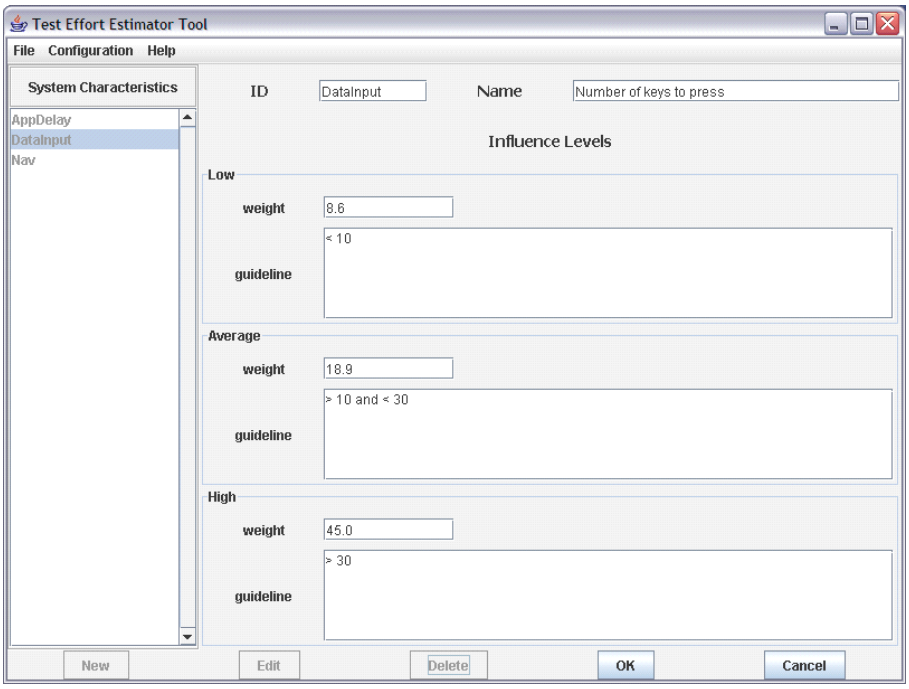


Figure C.2 Configuration of the characteristics used to measure test size and execution complexity.

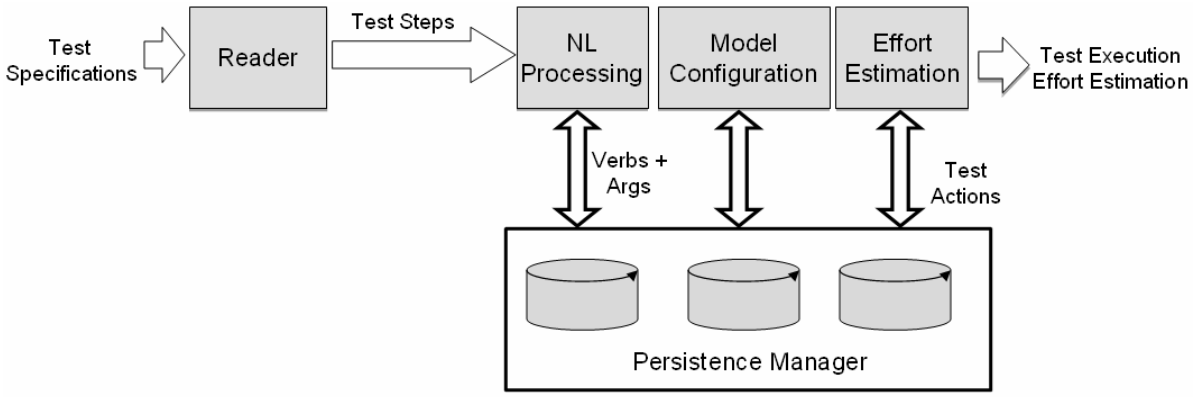


Figure C.3 Architecture of the automated test execution effort estimation based on test specifications.

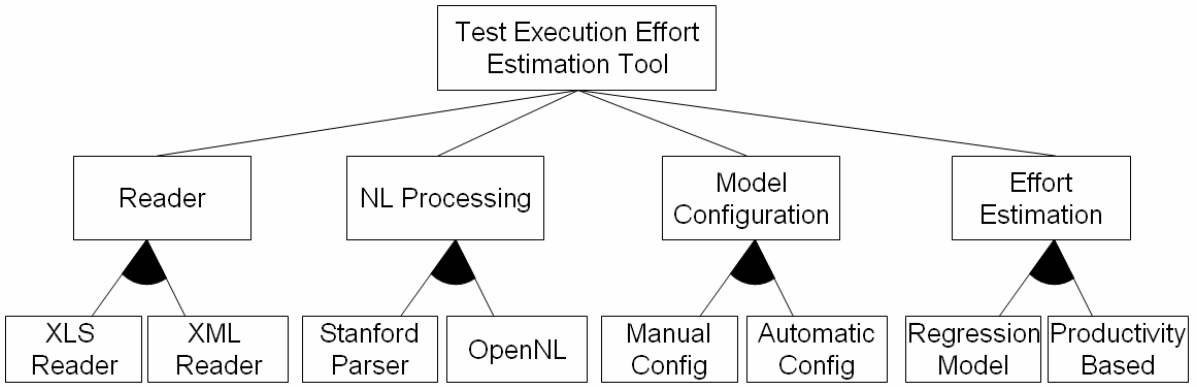


Figure C.4 Architecture of the automated test execution effort estimation based on test specifications.

Model Configuration: It is responsible for supporting the configuration of all information required to estimate the test execution effort, such as information about cost drivers, system characteristics, influence levels, weights and guidelines. In addition, the user has to evaluate the new test actions found in the test specifications, according to the system characteristics previously defined.

Effort Estimation: It is responsible for estimating the execution effort of a given suite of tests processed by the tool. For that, this component computes the number of execution points of the test suite and uses this measure to estimate the required test execution effort. All the information to compute execution points (characteristics, weights, etc.) and to estimate test execution effort (estimation model parameters, cost drivers, etc.) are retrieved by the Persistence Manager.

Persistence Manager: It is responsible for the persistence of data, storing information about the model configuration (system characteristics, cost drivers, influence levels, weights) and the test actions found in the analyzed test cases.

C.2.1 Feature Model

The presented architecture makes easy the customization of our tool, according to the customer needs. The feature model [115] presented in Fig. C.4 shows the possible extensions of each component of the tool architecture.

We implemented this as follows. The Reader component uses the Apache POI API [1] to read test specifications from Microsoft Excel files. The Natural Language Processing uses the Stanford Parser API [2] to parse test steps. The Model Configuration requires a manual configuration process based on expert opinion, although the tool can be extended by future works to automatically calibrate the model (define weights of characteristics and cost drivers) based on historical data. The Effort Estimation component supports both the regression and the productivity-based models.

C.3 Final Considerations

This automation support is essential to make feasible the test execution effort estimation based on the analysis of a high number of test specifications (may be hundreds) written in natural language. We discussed the main functionalities and architecture components of the tool. Currently, the tool allows the processing of test specifications written in English to estimate test size and execution complexity, as well as test execution effort. This tool was used to support our empirical studies, suggesting the viability of this tool.

Some of the drawbacks of this tool are the necessity to manually configure the measurement method of execution points and the estimation model based on regression analysis, for instance. In addition, it requires a manual evaluation of complexity for the first time a test action is found in test specifications. However, previous empirical studies indicated that little effort is required for the manual configuration of the tool, since the vocabulary for writing test specifications is reduced (few verbs for hundreds of tests) [15].

There are some improvements in the tool that can be done in future works. One example is an automatic or semi-automatic model configuration, supporting the calibration of the estimation models through an automatic analysis of historical databases. The automation support of effort estimation provided by this tool can be integrated with model-based testing tools. This integration can support the use of execution effort as an additional criterion for limiting the number of test cases to be generated.

APPENDIX D

ManualTEST: A Tool for Collecting Manual Test Execution Data

In this appendix, we present ManualTEST [17], a tool that we developed to improve the accuracy of the data collected in our empirical studies and to provide information to locate and identify sources of problems occurred during the execution of these studies.

In addition, ManualTEST can improve the accuracy of the collected data, providing information to locate and identify sources of problems occurred during the execution of these studies. ManualTEST was successfully used in several studies run in two real test sites that provide testing services for a major mobile phone manufacturer.

D.1 Manual Test Execution

In empirical studies involving manual test execution, it is necessary to have at least one or more of the following components: tester, product under tested, test specification and test environment. In a manual test execution, a tester reads and executes each step of a test specification using a product under test and a test environment.

The test specifications are usually described in terms of pre-conditions (initial conditions to start the test), procedure (list of test steps with inputs and expected outputs) and post-conditions [66]. In addition, they can contain more information, such as a test description, test objectives, the tested requirements and a software and hardware setup procedure. These specifications are commonly written in natural language, as shown in Figure D.1.

The data collected during manual test execution depend on the objective of the study, but the most common collected data are:

- Test id.
- Tester id.
- Date and time in which the test execution was started.
- Test execution time, which is the time spent to execute the test.
- Test result.

In general, test result indicates if the test was executed successfully (passed), if a bug was detected (failed) or if the tester could not execute the test (blocked) due to some constraint, such as lack of resource, test not applicable to the product under test, etc.

Test id: TaRGeT_0-1

Description: TaRGeT_TC_1

Objectives: Test the chronometer of the ManualTEST tool.

Requirements:
RQ_05

Setup:
None.

Initial conditions:
a) Tool should be started.
b) Tool is in the Test Selection perspective.
c) At least one test is shown in the Test Plans View.

Notes: Test case auto-generated by TaRGeT system.

Test procedure:
1) Double-click a test case X.
» The test specification is shown in the view "Selected TC". A new view X is created for the test case X.

2) Change perspective to Test Execution and inform your tester id.
» The tool is now in the Test Execution perspective. The view "Selected TC" is not opened. The view X is presented.

3) Click on button Play.
» The chronometer start counting time.

4) Click on button Pause.
» The chronometer stop counting time.

5) Click on button Play.
» The chronometer resume the counting of time.

6) Click on button Cancel.
» The counted time is discarded.

Final conditions:
None.

Cleanup:
None.

Figure D.1 Sample test specification written in natural language.

During the manual test execution in an empirical study, some problems can occur and threat the validity of the data analysis:

P1. While executing the tests, testers have to start counting test execution time just before the beginning of the test execution activity and to stop just after finishing it. The tester can forget to start or stop the chronometer, including between test executions, reducing the accuracy of the collected data.

P2. When running studies in industrial settings, participants may be interrupted by important phone calls or by other emergencies [20]. The time spent during these interruptions should not be taken into account.

P3. These interruptions should also be recorded to verify if they impacted the study [20]. In

our context, these interruptions can impact the total test execution time and the tester performance.

P4. From previous experiences [15], we observed that the effect of some confounding factors can be hard to detect, such as changes in network bandwidth or in other dynamic environmental conditions. These problems are usually hard to avoid when considering studies in industrial settings, even when running supervised experiments.

P5. For some studies, such as test prioritization based on data reuse [78], it is important to distinguish between test procedure time, test setup time, etc., making difficulty the data collection.

To solve these problems, we developed and used a tool to collect manual test execution data, as described in the next sections.

D.2 Manual Test Execution assiSTant (ManualTEST)

This section presents the main functionalities of ManualTEST, a tool developed to automate the collection of manual test execution data, avoiding the problems previously described. The tool was developed using the Eclipse Rich Client Platform (RCP) [86] and have two different perspective, one to select the tests to be executed and another one to support the test execution activity. Next, we present the details of these two perspectives and information about the data collected by the tool.

D.2.1 Test Selection

To automatically read test specifications, ManualTEST consider that these specifications are stored in spreadsheets files. In addition, the tool can be customized to read spreadsheets in different formats. Once a spreadsheet file is opened, the tests are listed to the user, as shown in Figure D.2. Then, the tester must double-click to open the tests that she is going to execute.

D.2.2 Test Execution

When the tester selects the Test Execution perspective, only the selected tests are presented. A Test Execution Controller view is also presented, as shown in Figure D.3. In this view, the play and the pause buttons are presented to the tester to control the counting of time. This integrated functionality make easy to the tester to start, pause and resume the test execution time (problem P1 and P2).

When the tester pushes the play button, the chronometer is activated and the first line of the test specification is highlighted. Using keys ↑ and ↓ from the keyboard, the tester executes the highlighted test step and go to the next one or go back to the previous one, as presented in Figure D.4. This functionality will ensure that the tester will read and execute each test step. Also, the time spent in each step is properly recorded (problem P5). For pausing the

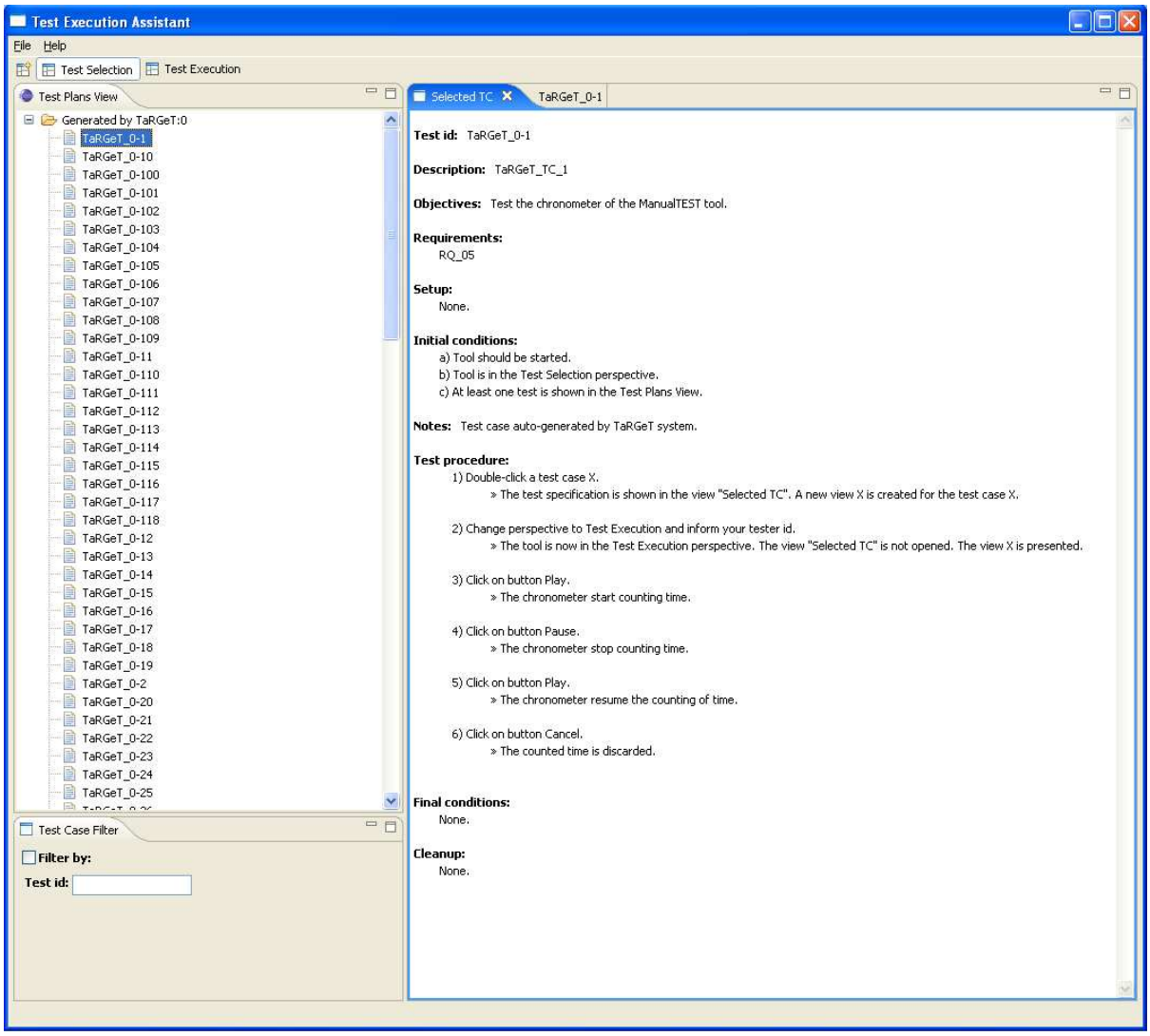


Figure D.2 Test selection perspective of ManualTEST.

chronometer, the tester should push the pause button and push the play button later to resume the time counting (problem P2).

ManualTEST counts the time based on three stages: Setup, Execution and Debug. The Setup stage is relative to the execution of steps required to build the test pre-conditions. The Execution stage is relative to the execution of the test procedure (execution of test steps and verification of expected results). Finally, the Debug stage is used for recording the time spent by the tester for confirming that a test failed. For instance, the tester may have to execute the test again to confirm the test failure or s/he may have to read the specification to confirm that the problem is a defect in the application and not a problem in the test specification. This approach is interesting when the study analyzes only one of these stages (problem P5). The total execution time is also presented by the tool.

By default, the initial test execution stage considered by the tool is Setup. To change the

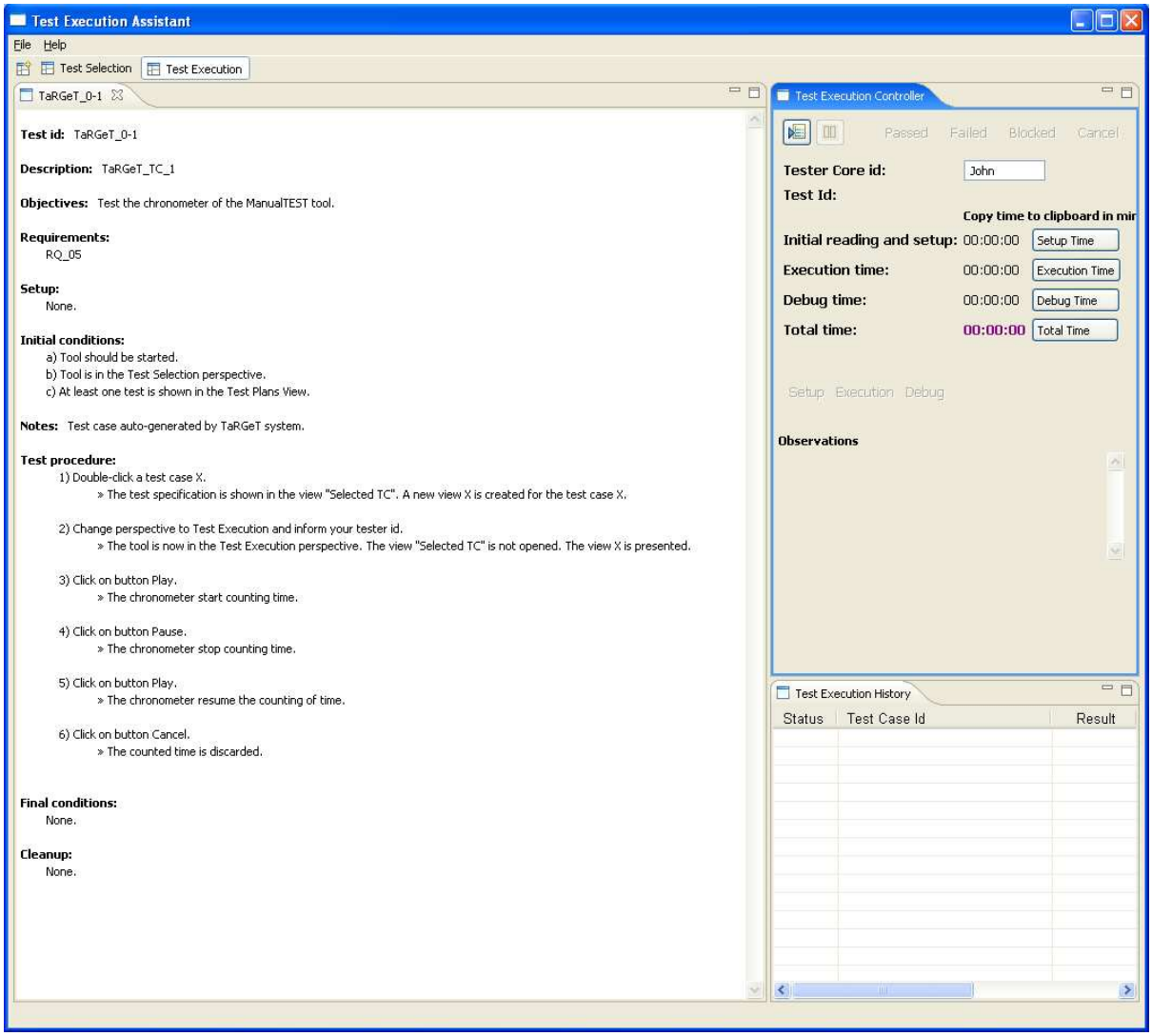


Figure D.3 Test execution perspective of ManualTEST.

stage, participants of the study can use the Setup, Execution or Debug buttons provided in the Test Execution Controller view. However, the tool automatically detects when the cursor is on the “Test procedure:” line and automatically consider the change from test setup to procedure execution, improving data collection accuracy. The test setup time, procedure execution time, debug time and total test execution time are presented to the tester in the Test Execution Controller view.

The field observations can be used to describe any problem occurred during the execution of the test. To finish the execution of a test case, the tester should press in button Passed, Failed or Blocked to indicate the correct test result. The Cancel button will discard the data collected for the current test execution.

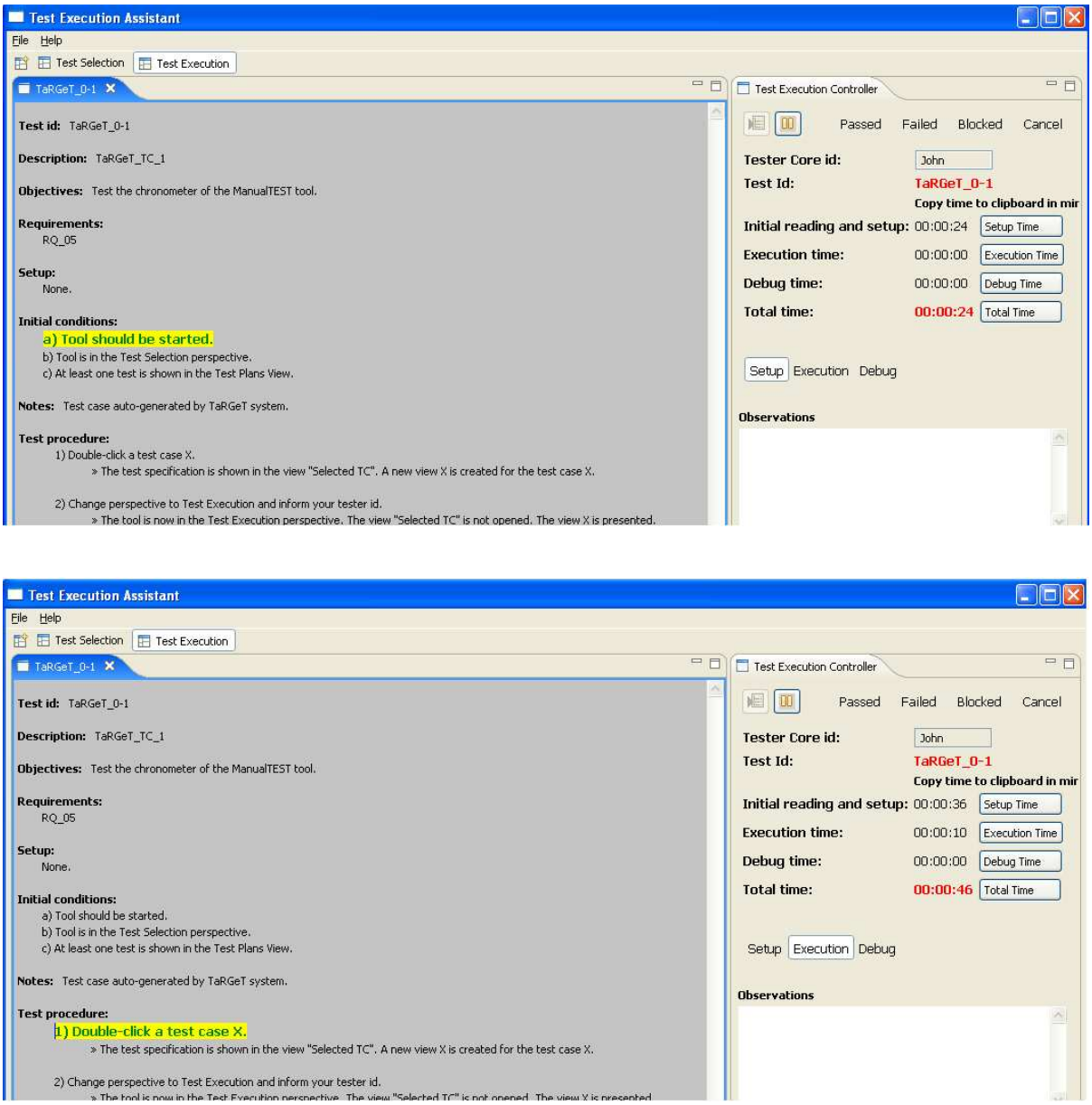


Figure D.4 ManualTEST at different moments: test step under execution is highlighted and time is automatically counted as setup time or procedure execution time.

D.2.3 Collected Data

As test execution is finished, ManualTEST stores the test result, execution times and other related information in two different spreadsheet files. The first one is presented in Figure D.5, which is basically the general information recorded for each executed test.

	A	B	C	D	E	F	G	H	I	J
1	Date	Time	Tester	Test	Result	Setup	Execution	Debug	Total time	
2	10/6/2008	19:47:44	John	TaRGeT_0-1	PASSED	00:00:29	00:00:51	00:00:00	00:01:20	

Figure D.5 Test result for a single test case.

The second spreadsheet file contains detailed information about the test execution. As presented in Figure D.6, ManualTEST stores the time spent to execute each step of the test specification. This characteristic is very important to investigate the presence of aberrant values in outlier analyses (problems P3 and P4) and to identify sources of variations on test execution time.

K	L
Step	Time
Description: TaRGeT_TC_1	00:00:03
Objectives: Test the chronometer of the ManualTEST tool.	00:00:00
Description: TaRGeT_TC_1	00:00:00
Objectives: Test the chronometer of the ManualTEST tool.	00:00:04
a) Tool should be started.	00:00:06
b) Tool is in the Test Selection perspective.	00:00:06
c) At least one test is shown in the Test Plans View.	00:00:09
Notes: Test case auto-generated by TaRGeT system.	00:00:01
Test procedure:	00:00:00
1) Double-click a test case X.	00:00:04
» The test specification is shown in the view "Selected TC". A new view X is created for the test case X.	00:00:04
2) Change perspective to Test Execution and inform your tester id.	00:00:04
» The tool is now in the Test Execution perspective. The view "Selected TC" is not opened. The view X is presented.	00:00:06
3) Click on button Play.	00:00:03
» The chronometer start counting time.	00:00:06
4) Click on button Pause.	00:00:04
» The chronometer stop counting time.	00:00:03
5) Click on button Play.	00:00:04
» The chronometer resume the counting of time.	00:00:03
6) Click on button Cancel.	00:00:03
» The counted time is discarded.	00:00:07

Figure D.6 Detailed test result includes time spent in each test step.

D.3 Advantages and Current Limitations of ManualTEST

We used ManualTEST in several studies carried out by researchers from a research group in testing [126]. These studies involved manual test execution and some of them are overviewed next.

- A case study and a controlled experiment designed to compare the time reduction for manual test setup provided by different test case prioritization techniques.

- A case study to compare functional tests generated manually or automatically by a model-based testing tool [105] with respect to their manual test execution time.
- Controlled experiments designed to analyze the relationship of different test size measures with manual test execution time.
- Controlled experiments designed to investigate the effect of cost drivers for manual test execution.
- Case studies to verify if code instrumentation has impact on manual test execution time when considering mobile phone applications.

These studies were run in industrial settings and most of them were carried out by researchers different from the authors of the tool. The researchers were asked to write the benefits and limitations of ManualTEST observed during the execution of their empirical studies. We received several feedbacks showing the advantages and limitations of ManualTEST. These feedbacks are summarized next.

- Automated test execution time collection with the click of few buttons.
- The integration of the chronometer and test specifications increased productivity and reduced the occurrence of problems (forgetting to start or pause the time, etc.). Also, this integration make easy to the researcher to monitor more than one tester at the same time.
- Detailed log files included all information required for data analysis. Due to the high level of detail, the effect of some confounding factors were detected and outliers were treated properly.
- With the detailed and automated data collection, it was also possible to study the impact of each test step on total execution time, including an analysis of the main sources of variation.
- The format of the log files was not easy to be understood by other researchers.
- There are possible improvements to implement and minor defects to correct in the tool, improving its usability.
- The available version of ManualTEST did not guide the tester through the correct sequence of tests to be executed, requiring a careful attention of testers and researchers. The implementation of this functionality can improve the benefits of the tool.
- After some interruptions, some testers forgot to resume the chronometer (push the the play button), requiring the intervention of the researcher that was monitoring the test execution. The tool can be improved by having warning messages blinking while the chronometer still paused.
- To write into the field Observations, the time should be paused by the tester. It would be interesting if the tool could do that automatically.

As we can see, ManualTEST can be used to avoid several problems during the execution of empirical studies related to manual test execution. However, its benefits depend on the correct use by the testers. For instance, the tester may still forget to push the play and pause buttons. As suggested in the feedbacks, some improvements in the tool should be done to avoid the reported problems.

D.4 Related Tools

Several testing tools are available in the marketing. For manual test authoring and execution, there is a tool that can be found in [53]. This commercial tool provides functionalities to execute and mark some steps of the test (comparisons and other verifications) as they are executed, as well as the storage of test results and test execution time. For collecting data in empirical studies, our tool presents better benefits due to characteristics such as the more detailed data collection and the highlighting of the test step being executed.

Some other researchers also have developed data collection tools for supporting their empirical studies. In [68], the authors reported a tool under development for collecting data during software engineering experiments. They were not interested in execution time, but in data concerning subjects, interactions between subjects and technology and changes in engineering artefacts. In [67], the authors proposed an unobtrusive method of collecting feedback from subjects during an experiment. They developed a tool to collect the feedbacks from experimental subjects and they identified several benefits of using the tool during the execution of four experiments, such as its use for validating the data obtained from other sources, checking process conformance and identifying problems with the experiments.

In [20], the authors developed a web-based support environment for planning and running software engineering experiments. One of its functionality is the collection of the experiment results, such as the answers of web questionnaires. The authors reported common interruptions (phone calls, lunch break, etc.) that occurred during the experiments run in real environments. In their approach, the subjects had to report the nature and time span of the interruptions. In our approach, the subject has to pause and resume the chronometer.

D.5 Final Considerations

This paper presented the main functionalities of ManualTEST, a tool developed for improving the collection of manual test execution data. This tool not only helps to collect data accurately, but it also provides information in different levels of detail, supporting the identification of problems in the collected data and analyses of outliers, sources of variability, etc. These benefits were observed during the execution of several case studies and controlled experiments in two real test sites. We intend to evolve the tool in order to overcome the current limitations and include the suggestions received during the empirical studies.

Despite of the use of ManualTEST for empirical studies, the tool can also be used to support the manual test execution activities. Its use can help, for instance, to have more accurate historical data in industrial settings. Although our tool is structured for experiments related to

manual test execution, we believe that it can be extended (or similar ones can be developed) to support data collection for other types of manual activities that can have problems similar to those reported here. Finally, the benefits observed during the use of ManualTEST justified the cost to develop the tool, which took approximatelly two months of work (partial time) of one experienced developer.

Bibliography

[1] Apache POI: Java API to access microsoft format files, 2007. http://poi.apache.org/.

[2] The Stanford parser: A statistical parser, 2007. http://nlp.stanford.edu/software/lexparser. shtml.

[3] Alain Abran and Pierre N. Robillard. Function points: a study of their measurement processes and scale transformations. *Journal of Systems and Software*, 25(2):171–184, 1994.

[4] Alain Abran and Pierre N. Robillard. Function points analysis: An empirical study of its measurement processes. *IEEE Trans. Softw. Eng.*, 22(12):895–910, 1996.

[5] Mohammed Abdullah Al-Hajri, Abdul Azim Abdul Ghani, Md Nasir Sulaiman, and Mohd Hasan Selamat. Modification of standard Function Point complexity weights system. *Journal of Systems and Software*, 74(2):195–206, 2005.

[6] Allan J. Albrecht. Measuring application development productivity. *Proc. IBM Applications Development Symp.*, 1979.

[7] Allan J. Albrecht. Ad/m productivity measurement and estimate validation-draft. *IBM Corp. Information Systems and Administration, AD/M Improvement Program*, 1984.

[8] Allan J. Albrecht and John E. Gaffney. Software functions, source lines of code, and development effort prediction: A software science validation. *IEEE Trans. Software Eng.*, 9(6):639–648, November 1983.

[9] Eduardo Aranha. Estimating test execution effort based on test specification. In *Fourth edition of the International Summer School on Software Engineering*, September 2007. Student talk.

[10] Eduardo Aranha and Paulo Borba. Considering test execution complexity for estimating test execution effort. In *1st International Doctoral Symposium on Empirical Software Engineering (IDoESE 2006)*, September 2006.

[11] Eduardo Aranha and Paulo Borba. Measuring test execution complexity. In *2nd Intl. Workshop on Predictor Models in SE (PROMISE 2006), co-located with the 22nd IEEE Conference on Software Maintenance (ICSM'06)*, September 2006.

[12] Eduardo Aranha and Paulo Borba. Test execution effort and capacity estimation. In *17th IEEE International Symposium on Software Reliability Engineering (ISSRE 2006)*, November 2006.

[13] Eduardo Aranha and Paulo Borba. Test execution effort and capacity estimation. In *21st International Forum on COCOMO and Software Cost Modeling*, November 2006.

[14] Eduardo Aranha and Paulo Borba. Empirical studies of test execution effort estimation based on test characteristics and risk factors. In *2nd International Doctoral Symposium on Empirical Software Engineering (IDoESE 2007)*, September 2007.

[15] Eduardo Aranha and Paulo Borba. An estimation model for test execution effort. In *1st International Symposium on Empirical Software Engineering and Measurement (ESEM 2007)*, pages 107–116, September 2007.

[16] Eduardo Aranha and Paulo Borba. Test effort estimation models based on test specifications. In *Testing: Academic & Industrial Conference - Practice And Research Techniques (TAIC PART 2007)*, September 2007.

[17] Eduardo Aranha and Paulo Borba. Manualtest: Improving collection of manual test execution data in empirical studies. In *5th Empirical Software Engineering Latin-american Workshop (ESELAW 2008)*, Salvador, Brazil, November 2008.

[18] Eduardo Aranha, Paulo Borba, and Jose Lima. Model simulation for test execution capacity estimation. In *17th IEEE International Symposium on Software Reliability Engineering (ISSRE 2006)*, November 2006.

[19] Eduardo Aranha, Filipe de Almeida, Thiago Diniz, Vitor Fontes, and Paulo Borba. Automated test execution effort estimation based on functional test specifications. In *Brazilian Symposium on Software Engineering (SBES'08)*, 2008.

[20] Erik Arisholm, Dag I. K. Sj, Gunnar J. Carelius, and Yngve Lindsj. A web-based support environment for software engineering experiments. *Nordic J. of Computing*, 9(3):231–247, 2002.

[21] Rajiv D Banker, Hsihui Chang, and Chris F Kemerer. Evidence on economies of scale in software development. *Information and Software Technology*, 36(5):275–282, May 1994.

[22] Rajiv D. Banker, Robert J. Kauffman, and Rachna Kumar. An empirical test of object-based output measurement metrics in a computer aided software engineering (case) environment. *Journal of Management Information Systems*, 8(3):127–150, 1992.

[23] Victor Basili, Gianluigi Caldiera, and Dieter Rombach. The goal question metric approach. *Encyclopedia of Software Engineering*, 1:528–532, 1994.

[24] Kent Beck and Martin Fowler. *Planning Extreme Programming*. Addison-Wesley Professional, 2000.

[25] Barry Boehm. *Software Engineering Economics*. Prentice Hall, 1981.

[26] Barry Boehm, Chris Abts, and Sunita Chulani. Software development cost estimation approaches - a survey. *Annals of Software Engineering*, 10:177–205, 2000.

[27] Barry Boehm, Ellis Horowitz, Ray Madachy, Donald Reifer, Bradford Clark, Bert Steece, Winsor Brown, Sunita Chulani, and Chris Abts. *Software Cost Estimation with COCOMO II*. Prentice Hall, 2000.

[28] Barry Boehm and Ali Afzal Malik. System and software sizing tutorial. In *22nd International Forum on COCOMO and Systems/Software Cost Modeling*, 2007.

[29] Barry Boehm, Donald J. Reifer, and Ricardo Valerdi. Cosysmo: A systems engineering cost model. In *1st Annual Conference on Systems Integration*. Stevens Institute of Technology Campus, 2003.

[30] Bruce Bowerman and Richard O'Connell. *Linear Statistical Models, an Applied Aproach*. PWS-KENT, 2nd edition, 1990.

[31] Lionel Briand, Khaled El Emam, and Sandro Morasca. On the application of measurement theory in software engineering. *Empirical Software Engineering: An International Journal*, 1(1):61–88, 1996.

[32] Lionel C. Briand, Khaled El Emam, Dagmar Surmann, Isabella Wieczorek, and Katrina D. Maxwell. An assessment and comparison of common software cost estimation modeling techniques. In *Proceedings of the 21st international conference on Software engineering (ICSE '99)*, pages 313–322, New York, NY, USA, 1999. ACM.

[33] Lionel C. Briand, Tristen Langley, and Isabella Wieczorek. A replicated assessment and comparison of common software cost modeling techniques. In *Proceedings of the 22nd international conference on Software engineering (ICSE '00)*, pages 377–386, New York, NY, USA, 2000. ACM.

[34] Bart Broekman and Edwin Notenboom. *Testing Embedded Software*. Addison-Wesley, 2002.

[35] Christopher J. C. Burges. A tutorial on support vector machines for pattern recognition. In *Data Mining and Knowledge Discovery*, pages 121–167, 1998.

[36] John M. Chambers. *Software for Data Analysis: Programming with R*. Springer, New York, 2008.

[37] Sunita Chulani. Constructive quality modeling for defect density prediction: Coqualmo. In *International Symposium on Software Reliability Engineering (ISSRE'99)*, November 1999.

[38] Code count tools. http://sunset.usc.edu/research/CODECOUNT/.

[39] Mike Cohn. *Agile Estimating and Planning*. Prentice Hall, 2005.

[40] Common Software Measurement International Consortium (COSMIC). *The COSMIC Functional Size Measurement Method, Version 3.0*, September 2007.

[41] Gennaro Costagliola, Sergio Di Martino, Filomena Ferrucci, Carmine Gravino, Genoveffa Tortora, and Giuliana Vitiello. Effort estimation modeling techniques: a case study for web applications. In *6th international conference on Web engineering (ICWE '06)*, pages 9–16, New York, NY, USA, 2006. ACM.

[42] Hassan Diab, Marc Frappier, and Richard St-Denis. A formal definition of function points for automated measurement of b specifications. In *4th International Conference on Formal Engineering Methods (ICFEM 2002)*, pages 483–494, 2002.

[43] Harris Drucker, Chris J.C. Burges, Linda Kaufman, Alex Smola, and Vladimir Vapnik. Support vector regression machines. In *Advances in Neural Information Processing Systems*, pages 155–161. MIT Press, 1996.

[44] Ibrahim K. El-Far and James A. Whittaker. Model-based software testing. In J. Marciniak, editor, *Wiley Encyclopedia of Software Engineering*. John Wiley & Sons, 2001.

[45] Angi Faiks and Nancy Hyland. Gaining user insight: A case study illustrating the card sort technique. *College and Research Libraries journal*, 61(4):349–357, July 2000.

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

[47] David Garmus and David Herron. *Function Point Analysis, Measurement Practices for Successful Software Projects*. Addison Wesley, 2001.

[48] A. Gray and S. MacDonell. Applications of fuzzy logic to software metric models fordevelopment effort estimation. In *Fuzzy Information Processing Society*, pages 394–399, 1997.

[49] Simon Haykin. *Neural Networks: A Comprehensive Foundation*. Prentice Hall, 2nd edition, 1998.

[50] Klaus Hinkelmann and Oscar Kempthorne. *Design and Analysis of Experiments, Introduction to Experimental Design*. Wiley-Interscience, 2nd edition, 2007.

[51] Ronald Hocking. *Methods and applications of linear models: regression and the analysis of variance*. New York: Wiley-Intercience, 2003.

[52] John Hoenig and Dennis Heisey. The abuse of power: The pervasive fallacy of power calculations for data analysis. *The american statistician*, 55(1), February 2001.

[53] IBM Rational Software. Rational Manual Tester, September 2008. http://www.ibm.com/software/awdtools/tester/manual.

[54] Ali Idri, Alain Abran, and Taghi M. Khoshgoftaar. Estimating software project effort by analogy based on linguistic values. *Eighth IEEE International Symposium on Software Metrics (METRICS'02)*, 00:21, 2002.

[55] IFPUG: International Function Point Users Group. http://www.ifpug.org/.

[56] International Function Point Users Group (IFPUG). *Function Point, Counting Practices Manual, Release 1.0*, 1986.

[57] International Function Point Users Group (IFPUG). *Function Point, Counting Practices Manual, Release 2.0*, 1988.

[58] International Function Point Users Group (IFPUG). *Function Point, Counting Practices Manual, Release 3.0*, 1990.

[59] International Function Point Users Group (IFPUG). *Function Point, Counting Practices Manual, Release 4.0*, 1994.

[60] International Organization for Standardization (ISO). *ISO/IEC19761:2003, Software Engineering – COSMIC – A Functional Size Measurement Method*, 2003.

[61] International Organization of Standardization. *ISO/IEC FCD 9126-1, Information technology - Software product quality Part 1: Quality Model*, 1998.

[62] ISO. *ISO/IEC 20968: Software Engineering - MkII Function Point Analysis - Counting Practices Manual*, 2002.

[63] D. Ross Jeffery, Graham C. Low, and M. Barnes. A comparison of function point counting techniques. *IEEE Trans. Software Eng.*, 19(5):529–532, 1993.

[64] D.R. Jeffery and J. Stathis. Specification-based software sizing: an empirical investigation of function metrics. In *Proc. NASA Goddard Software Eng. Workshop*, 1993.

[65] Magne Jorgensen and Martin Shepperd. A systematic review of software development cost estimation studies. *IEEE Trans. Softw. Eng.*, 33(1):33–53, 2007.

[66] Paul Jorgensen. *Software Testing, A Craftsmans Approach*. CRC Press, second edition, 2002.

[67] Amela Karahasanoviæ, Bente Anda, Erik Arisholm, Siw Elisabeth Hove, Magne J, Dag I. Sj, and Ray Welland. Collecting feedback during software engineering experiments. *Empirical Software Engineering*, 10(2):113–147, 2005.

[68] A Karahasanovic, D I. K Sjøberg, and M Jørgensen. Data collection in software engineering experiments. In , editor, *Managing Information Technology in a Global Economy, Information Resources Management Association International Conference IRMA 2001, Software Engineering Track*, pages 1027–1028, Toronto, Ontario Canada, 2001. Idea Group Publishing.

[69] Chris F. Kemerer. Reliability of function points measurement: a field experiment. *Commun. ACM*, 36(2):85–97, 1993.

[70] Barbara Kitchenham. Counterpoint: The problem with function points. *IEEE Software*, 14(2):29,31, 1997.

[71] Barbara Kitchenham and Kari Känsälä. Inter-item correlations among function points. In *ICSE '93: Proceedings of the 15th international conference on Software Engineering*, pages 477–480, Los Alamitos, CA, USA, 1993. IEEE Computer Society Press.

[72] Barbara A. Kitchenham. The question of scale economies in software why cannot researchers agree? *Information and Software Technology*, 44(1):13–24, January 2002.

[73] Robert Kuehl. *Design of Experiments: Statistical Principles of Research Design and Analysis*. Duxbury Press, 2nd edition, 1999.

[74] Satish Kumar1, B. Ananda Krishna2, and Prem S. Satsangi3. Fuzzy systems and neural networks in software engineering project management. *Journal of Applied Intelligence*, 1994.

[75] Kevin Lano. *The B Language and Method: A Guide to Practical Formal Development*. Springer-Verlag, 1996.

[76] Daniel Leitao. Translating natural language descriptions into formal test case specifications. Master's thesis, Federal University of Pernambuco/UFPE, 2006.

[77] Ghislain Levesque, Valery Bevo, and De Tran Cao. Estimating software size with uml models. In *2008 C3S2E conference (C3S2E'08)*, pages 81–87, New York, NY, USA, 2008. ACM.

[78] Lucas Lima, Juliano Iyoda, and Augusto Sampaio. A permutation technique for test case prioritization in a black-box environment. In *2nd Brazilian Workshop on Systematic and Automated Software Testing*, Campinas-SP, Brazil, October 2008. To appear.

[79] Harold Linstone and Murray Turoff. *The Delphi Method: Techniques and Applications*. http://is.njit.edu/pubs/delphibook, 2002.

[80] David H. Longstreet. *Function Points Analysis Training Course*. Longstreet Consulting Inc, February 2005.

[81] Graham C. Low and D. R. Jeffery. Function points in the estimation and evaluation of the software process. *IEEE Trans. Softw. Eng.*, 16(1):64–71, 1990.

[82] S. MacDonell and A. Gray. Alternatives to regression models for estimating software projects. In *Proceedings of the IFPUG Fall 1996 Conference*. IFPUG, 1996.

[83] G S. Maddala. *Introduction to Econometrics*. Wiley, 3 edition, May 2001.

[84] Sergio Di Martino, Filomena Ferrucci, Carmine Gravino, and Emilia Mendes. Comparing size measures for predicting web application development effort: A case study. In *International Symposium on Empirical Software Engineering and Measurement (ESEM 2007)*, 2007.

[85] Katrina Maxwell. *Applied Statistics for Software Managers*. Prentice Hall, 2002.

[86] Jeff McAffer and Jean-Michel Lemieux. *Eclipse Rich Client Platform: Designing, Coding, and Packaging Java(TM) Applications*. Addison-Wesley Professional, 2005.

[87] Emilia Mendes. *Cost Estimation Techniques for Web Projects*. IGI Publishing, September 2007.

[88] Emilia Mendes. The use of a bayesian network for web effort estimation. In *International Conference on Web Engineering (ICWE)*, pages 90–104, 2007.

[89] Emilia Mendes and Steve Counsell. Web development effort estimation using analogy. In *Proceedings of the 2000 Australian Software Engineering Conference*, 2000.

[90] Emilia Mendes and Barbara Kitchenham. Further comparison of cross-company and within-company effort estimation models for web applications. In *METRICS '04: Proceedings of the Software Metrics, 10th International Symposium on (METRICS'04)*, pages 348–357, Washington, DC, USA, 2004. IEEE Computer Society.

[91] Emilia Mendes and Nile Mosley. Further investigation into the use of cbr and stepwise regression to predict development effort for web hypermedia applications. In *International Symposium on Empirical Software Engineering (ISESE 2002)*, 2002.

[92] Emilia Mendes, Nile Mosley, and Steve Counsell. Comparison of web size measures for predicting web design and authoring effort. *IEE Proceedings - Software*, 149(3):86–92, 2002.

[93] Emilia Mendes, Nile Mosley, and Steve Counsell. Investigating web size metrics for early web cost estimation. *J. Syst. Softw.*, 77(2):157–172, 2005.

[94] Emilia Mendes, Ian Watson, Chris Triggs, Nile Mosley, and Steve Counsell. A comparative study of cost estimation models for web hypermedia applications. *Empirical Software Engineering*, 8(2):163–196, 2003.

[95] George Milliken and Dallas Johnson. *Analysis of Messy Data: Designed Experiments*, volume I. Chapman Hall/CRC, 1993.

[96] Y. Miyazaki, M. Terakado, K. Ozaki, and H. Nozaki. Robust regression for developing software estimation models. *J. Syst. Softw.*, 27(1):3–16, 1994.

[97] Parastoo Mohagheghi, Bente Anda, and Reidar Conradi. Effort estimation of use cases for incremental large-scale software development. In *Proceedings of the 27th international conference on Software engineering (ICSE05)*, pages 303–311. ACM Press, 2005.

[98] Kjetil Molokken and Magne Jorgensen. A review of surveys on software effort estimation. *International Symposium on Empirical Software Engineering (ISESE'03)*, 00:223, 2003.

[99] Tridas Mukhopadhyay and Sunder Kekre. Software effort models for early estimation of process control applications. *IEEE Trans. Softw. Eng.*, 18(10):915–924, 1992.

[100] Glenford J. Myers. *The Art of Software Testing*. John Wiley & Sons, Inc., 2004.

[101] Suresh Nageswaran. Test effort estimation using use case points. In *14th International Internet & Software Quality Week 2001*, June 2001.

[102] Richard E. Neapolitan. *Learning Bayesian Networks*. Prentice Hall, 2003.

[103] NESMA. *Definitions and Counting Guidelines for the Application of Function Points Analysis*, version 1.0 edition, 1990.

[104] Vu Nguyen, Sophia Deeds-Rubin, and Thomas Tan. Sloc counting standards. In *22nd International Forum on COCOMO and Systems/Software Cost Modeling*, 2007.

[105] Sidney Nogueira, Emanuela Cartaxo, Dante Torres, Eduardo Aranha, and Rafael Marques. Model based test generation: A case study. In *1st Brazilian Workshop on Systematic and Automated Software Testing (SAST 2007)*, October 2007.

[106] Strategic Evolution of ESE Data Systems (SEEDS). Survey of cost estimation tools, 2001.

[107] Adriano L. I. Oliveira. Letters: Estimation of software project effort with support vector regression. *Neurocomputing*, 69(13-15):1749–1753, 2006.

[108] C. Pandian. *Software Metrics: A Guide to Planning, Analysis, and Application*. CRC Press, Inc., 2003.

[109] Robert E. Park, Robert E. Park, Thomas R. Miller, and Lt Col. Software size measurement: A framework for counting source statements. Technical report, Software Engineering Institute, 1992.

[110] http://www.crisp.se/planningpoker.

[111] Alexander Pretschner. Model-based testing. In *ICSE '05: Proceedings of the 27th international conference on Software engineering*, pages 722–723, 2005.

[112] Lawrence Putnam and Ware Myers. *Measures for Excellence*. Yourdon Press Computing Series, 1992.

[113] Quantitative software measurement, QSM Inc. http://www.qsm.com.

[114] John Ross Quinlan. Learning with continuous classes. In *Fifth Australian Joint Conf. Artificial Intelligence*, pages 343–348, 1992.

[115] Matthias Riebisch, Detlef Streitferdt, and Ilian Pashov. Modeling variability for object-oriented product lines. In *Object-Oriented Technology (LNCS)*, volume 3013, pages 165–178. Springer Berlin, 2004.

[116] Melanie Ruhe, Ross Jeffery, and Isabella Wieczorek. Using web objects for estimating software development effort for web applications. In *METRICS '03: Proceedings of the 9th International Symposium on Software Metrics*, page 30, Washington, DC, USA, 2003. IEEE Computer Society.

[117] Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 2nd edition, 2002.

[118] Chris Schofield. Non-algorithmic effort estimation techniques. Technical report, Empirical Software Engineering Research Group (ESERG) at Bournemouth University, 1998.

[119] Rolf Schwitter. English as a formal specification language. In *Proceedings of the 13th International Workshop on Database and Expert Systems Applications (DEXA02)*, pages 228–232, 2002.

[120] David Seaver. Fast function points. http://sunset.usc.edu/Activities/oct24-27-00/Presentations/Seaver_FAST Function Points.pdf.

[121] Martin Shepperd, Michelle Cartwright, and Gada Kadoda. On building prediction systems for software engineers. *Empirical Software Engineering*, 5:175–182, 2000.

[122] Sloc count. http://www.dwheeler.com/sloccount.

[123] George W. Snedecor and William G. Cochran. *Statistical Methods*. Iowa State University Press, 8 edition, 1989.

[124] Charles Symons. The cosmic functional size measurement method version 3,0. In *IWSM-Mensura 2007*, 2007.

[125] Dante Torres, Daniel Leitao, and Flavia Barros. Motorola specnl: A hybrid system to generate nl descriptions from test case specifications. *Sixth International Conference on Hybrid Intelligent Systems (HIS'06)*, page 45, 2006.

[126] Dante Torres, Sidney Nogueira, Emanuela Cartaxo, Eduardo Aranha, Paulo Borba, Flávia Barros, Patrícia Machado, Augusto Sampaio, and Alexandre Mota. Brazil test center research group. In *1st Brazilian Workshop on Systematic and Automated Software Testing (SAST 2007)*, October 2007.

[127] Frank Vogelezang. Cosmic full function points the next generation of functional sizing. In *Software Measurement European Forum - SMEF 2005*, pages 281–289, 2005.

[128] Yong Wang and Ian H. Witten. Inducing model trees for continuous classes. In *Proceedings of the poster papers of the 9th European Conference on Machine Learning (ECML 97)*, 1997.

[129] Ian Watson. *Applying Case-Based Reasoning: Techniques for Enterprise Systems*. Morgan Kaufmann, 1997.

[130] Michael Whalen, Ajitha Rajan, Mats Heimdahl, and Steven 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.

[131] C. F. Jeff Wu and Michael Hamada. *Experiments: Planning, Analysis, and Parameter Design Optimization*. Wiley-Interscience, 2000.

[132] Ales Zivkovic, Ivan Rozman, and Marjan Hericko. Automated software size estimation based on function points using uml models. *Information and Software Technology*, 47(13):881–890, 2005.