

## 2 Web Effort Estimation

Emilia Mendes, Nile Mosley, Steve Counsell

**Abstract:** Software effort models and effort estimates help project managers allocate resources, control costs, and schedule and improve current practices, leading to projects that are finished on time and within budget. In the context of Web development and maintenance, these issues are also crucial, and very challenging, given that Web projects have short schedules and a highly fluidic scope. Therefore this chapter has two main objectives. The first is to introduce the concepts related to effort estimation and in particular Web effort estimation. The second is to present a case study where a real effort prediction model based on data from completed industrial Web projects is constructed step by step.

**Keywords:** Web effort estimation, Manual stepwise regression, Effort models, Web size measures, Prediction accuracy, Data analysis.

### 2.1 Introduction

The Web is used as a delivery platform for numerous types of Web applications, ranging from complex e-commerce solutions with back-end databases to on-line personal static Web pages. With the sheer diversity of Web application types and technologies employed, there exists a growing number of Web companies bidding for as many Web projects as they can accommodate. As usual, in order to win the bid, companies estimate unrealistic schedules, leading to applications that are rarely developed within time and budget.

Realistic effort estimates are fundamental for the successful management of software projects; the Web is no exception. Having realistic estimates at an early stage in a project's life cycle allows project managers and development organisations to manage their resources effectively.

To this end, prediction is a necessary part of an effective process, whether it be authoring, design, testing, or Web development as a whole. A prediction process involves:

- The identification of measures (e.g. number of new Web pages, number of new images) that are believed to influence the effort required to develop a new Web application.
- The formulation of theories about the relationship between the selected measures and effort (e.g. the greater the number of new static Web pages, the greater the development effort for a new application).

- The capturing of historical data (e.g. size and actual effort) about past Web projects or even past development phases within the same project.
- The use of this historical data to develop effort estimation models for use in predicting effort for new Web projects.
- The assessment of how effective those effort estimation models are, i.e. the assessment of their prediction accuracy.

Cost and effort are often used interchangeably within the context of effort estimation (prediction) since effort is taken as the main component of project costs. However, given that project costs also take into account other factors such as contingency and profit [20] we will use the word “effort” and not “cost” throughout this chapter.

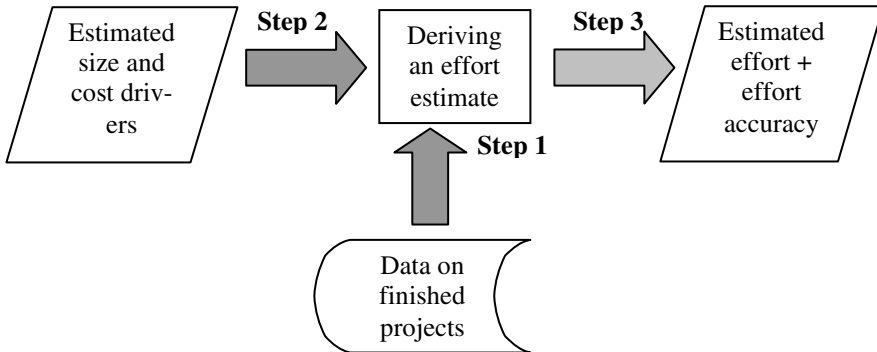
Numerous effort estimation techniques have been proposed and compared over the last 20 years. A classification and description of such techniques is introduced in Sect. 2.2 to help provide readers with a broader overview. To be useful, an effort estimation technique must provide an effort estimate for a new project that is not widely dissimilar from the *actual* effort this project will need to be finished. The effectiveness of effort estimation techniques to provide accurate effort estimates is called prediction power. Section 2.3 presents the four most commonly used measures of prediction power and, in Section 2.4, the associated prediction accuracy. Finally, Sect. 2.5 details a case study building an effort estimation model using data from world-wide industrial Web projects.

## 2.2 Effort Estimation Techniques

The purpose of estimating effort is to predict the amount of effort to accomplish a given task, based on knowledge of other project characteristics that are believed to be related to effort. Project characteristics (independent variables) are the input, and effort (dependent variable) is the output we wish to predict (see Fig. 2.1). For example, a given Web company may find that to predict the effort necessary to implement a new Web application, it will require the following input: estimated number of new Web pages, total number of developers who will help develop the new Web application, developers’ average number of years of experience with the development tools employed, and the number of functions/features (e.g. shopping cart) to be offered by the new Web application.

A task to be estimated can be as simple as developing a single function (e.g. creating a table on the database) or as complex as developing a large application, and in general the one input (independent variable) assumed to have the strongest influence on effort is size. Other independent variables may also be influential (e.g. developers’ average experience, number of

tools employed) and these are often identified as cost drivers. Depending on the techniques employed, we can also use data on past finished projects to help estimate effort for new projects.



**Fig. 2.1.** Components of a cost model

Several techniques for effort estimation have been proposed over the past 30 years in software engineering. These fall into three general categories [37]: expert opinion, algorithmic models and artificial intelligence techniques.

### 2.2.1 Expert Opinion

Expert opinion represents the process of estimating effort by subjective means, and is often based on previous experience from developing/managing similar projects. It has been and still is widely used in software and Web development.

The drawback of this technique is that it is very difficult to quantify and to determine those factors that have been used to derive an estimate, making it difficult to repeat. However, studies show that this technique can be an effective estimating tool when used in combination with other less subjective techniques (e.g. algorithmic models) [11,30,31].

In terms of the diagram presented in Fig. 2.1, the sequence occurs as follows:

- An expert looks at the estimated size and cost drivers related to a new project for which effort needs to be estimated.
- Based on the data obtained in a) (s)he remembers or retrieves data on past finished projects for which actual effort is known.
- Based on the data from a) and b) (s)he subjectively estimates effort for the new project. Deriving an accurate effort estimate is more likely to occur when there are completed projects similar to the one having its

effort estimated. The sequence described corresponds to steps 2, 1, and 3 in Fig. 2.1. The knowledge regarding the characteristics of a new project is necessary to retrieve, from either memory or a database, knowledge on finished similar projects. Once this knowledge is retrieved, effort can be estimated.

### 2.2.2 Algorithmic Techniques

To date, the most popular techniques described in the effort estimation literature are algorithmic techniques. Such techniques attempt to formalise the relationship between effort and one or more project characteristics. The result is an algorithmic model. The central project characteristic used in such a model is usually taken to be some notion of software size (e.g. the number of lines of source code, number of Web pages, number of links). This formalisation is often translated as an equation such as that shown by Eq. 2.1, where  $a$  and  $b$  are parameters that also need to be estimated. Equation 2.1 shows that size is the main factor contributing to effort, and can be adjusted according to an Effort Adjustment Factor ( $EAF$ ), calculated from cost drivers (e.g. developers, experience, tools). An example of an algorithmic model that uses Eq. 2.1 is the CONstructive COSt Model (COCOMO) model [2], where parameters  $a$  and  $b$  are based on the type of project under construction, and the  $EAF$  is based on 15 cost drivers that are calculated and then summed.

$$Estimated\ Effort = a\ EstSizeNewproj^b\ EAF \quad (2.1)$$

where:

$a$ ,  $b$  are parameters chosen based on certain criteria, such as the type of software project being developed.  $EstSizeNewproj$  is the estimated size for the new project.  $EAF$  is the Effort Adjustment Factor.

Equations 2.2 and 2.3 are different examples of algorithmic equations (models), where both are obtained by applying regression analysis techniques [33] on data sets of past completed projects. Equation 2.2 assumes a linear relationship between effort and its size/cost drivers whereas Equation 2.3 assumes a non-linear relationship. In Equation 2.3, when the exponent is  $< 1$  we have economies of scale, i.e., larger projects use less effort comparatively than smaller projects. The opposite situation (exponent  $> 1$ ) gives diseconomies of scale, i.e. larger projects use more effort comparatively than smaller projects.

$$EstimatedEffort = C + a_0 EstSizeNewproj + a_1 CD_1 + \dots + a_n CD_n \quad (2.2)$$

$$EstimatedEffort = C\ EstSizeNewproj^{a_0}\ CD_1^{a_1} \dots CD_n^{a_n} \quad (2.3)$$

where:

$C$  is a constant denoting the initial estimated effort (assuming size and cost drivers to be zero) derived from past data.

$a_0 \dots a_n$  are parameters derived from past data.

$CD_1 \dots CD_n$  are other project characteristics, other than size, that have an impact on effort.

The COCOMO model is an example of a *generic algorithmic model*, believed to be applicable to any type of software project, with suitable calibration or adjustment to local circumstances. In terms of the diagram presented in Fig. 2.1, the model uses parameter values that are based on past project data; however, for anyone wishing to use this model, the steps to use are 1, 2, and 3. Step 1 is used only once to calculate the initial values for its parameters, which are then fixed from that point onwards. The single use of step 1 makes this model a *generic algorithmic model*.

Regression-based algorithmic models are most suitable to local circumstances such as “in-house” analysis as they are derived from past data that often represents projects from the company itself. Regression analysis, used to generate regression-based algorithmic models, provides a procedure for determining the “best” straight-line fit to a set of project data that represents the relationship between effort (the response or dependent variable) and project characteristics (e.g. size, experience, tools, the predictor or independent variables) [33]. The regression line is represented as an equation, such as those given by Eqs. 2.1 and 2.2. The effort estimation models we will create in Sect. 2.5 fall into this category.

Regarding the regression analysis itself, two of the most widely used techniques are multiple regression (MR) and stepwise regression (SWR). The difference between both is that MR obtains a regression line using all the independent variables at the same time, whereas SWR is a technique that examines different combinations of independent variables, looking for the best grouping to explain the greatest amount of variation in effort. Both use least squares regression, where the regression line selected is the one that reflects the minimum values of the sum of the squared errors. Errors are calculated as the difference between actual and estimated effort and are known as the residuals [33].

The sequence followed here is as follows:

- a) Past data is used to generate a cost model.
- b) This model then receives, as input, values for the new project characteristics.
- c) The model generates estimated effort. The sequence described herein corresponds to steps 1, 2, and 3 from Fig. 2.1, in contrast to that for expert opinion.

A description of regression analysis is presented in Chap. 12.

### 2.2.3 Artificial Intelligence Techniques

Artificial intelligence techniques have, in the last decade, been used as a complement to, or as an alternative to, the previous two categories. Examples include fuzzy logic [22], regression trees [34], neural networks [38], and case-based reasoning [37]. We will cover case-based reasoning (CBR) and regression trees (CART) in more detail as they are currently the most popular machine learning techniques employed for Web cost estimation. A useful summary of numerous machine learning techniques can also be found in [10].

#### *Case-Based Reasoning*

Case-based reasoning (CBR) provides estimates by comparing the current problem to be estimated against a library of historical information from completed projects with a known effort (case base). It involves [1]:

- i. Characterising a new project  $p$ , for which an estimate is required, with attributes (features) common to those completed projects stored in the case base. In terms of software cost estimation, features represent size measures and cost drivers which have a bearing on effort. Feature values are normally standardized (between 0 and 1) such that they have the same degree of influence on the result.
- ii. Use of this characterisation as a basis for finding similar (analogous) completed projects, for which effort is known. This process can be achieved by measuring the “distance” between two projects, based on the values of the number of features ( $k$ ) for these projects. Although numerous techniques can be used to measure similarity, nearest neighbour algorithms using the unweighted Euclidean distance measure have been the most widely used to date in software and Web engineering.
- iii. Generation of a predicted value of effort for project  $p$  based on the effort for those completed projects that are similar to  $p$ . The number of similar projects will depend on the size of the case base. For small case bases (e.g. up to 90 cases), typical values are 1, 2, and 3 closest neighbours (analogies). For larger case bases no conclusions have been reached regarding the best number of similar projects to use. The calculation of estimated effort is obtained using the same effort value as the closest neighbour, or the mean of effort for two or more analogies. This is the common choice in Web and software engineering.

The sequence of steps used with CBR is as follows:

- a) The estimated size and cost drivers relating to a new project are used to retrieve similar projects from the case base, for which actual effort is known.
- b) Using the data from a) a suitable CBR tool retrieves similar projects and calculates estimated effort for the new project. The sequence just described corresponds to steps 2, 1, and 3 in Fig. 2.1, similar to that employed for expert opinion. The characteristics of a new project must be known in order to retrieve finished similar projects. Once similar projects are retrieved, then effort can be estimated.

When using CBR there are six parameters to consider [35]:

- Feature Subset Selection
- Similarity Measure
- Scaling
- Number of Analogies
- Analogy Adaptation
- Adaptation Rules

#### *Feature Subset Selection*

Feature subset selection involves determining the optimum subset of features that yield the most accurate estimation. Some existing CBR tools, e.g. ANGEL [36] optionally offer this functionality using a brute force algorithm, searching for all possible feature subsets. Other CBR tools (e.g. CBR-Works) have no such functionality, and therefore to obtain estimated effort, we must use all of the known features of a project to retrieve the most similar cases.

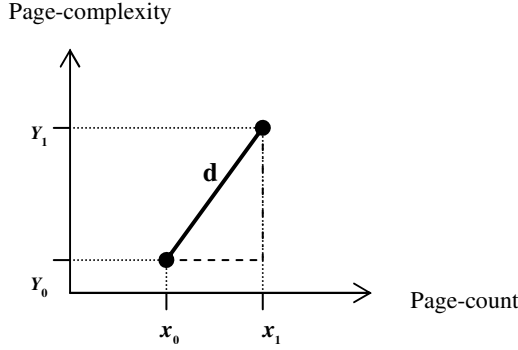
#### *Similarity Measure*

The similarity measure measures the level of similarity between different cases, with several similarity measures proposed in the literature. The most popular in the current Web/software engineering literature [1,24,35] are the unweighted Euclidean distance, the weighted Euclidean distance, and the maximum distance. Other similarity measures are presented in [1].

Unweighted Euclidean distance: The unweighted Euclidean distance measures the Euclidean (straight-line) distance  $d$  between the points  $(x_0, y_0)$  and  $(x_1, y_1)$ , given by the equation:

$$d = \sqrt{(x_0 - x_1)^2 + (y_0 - y_1)^2} \quad (2.4)$$

This measure has a geometrical meaning as the shortest distance between two points in an  $n$ -dimensional Euclidean space [1].



**Fig. 2.2.** Euclidean distance using two size attributes

Figure 2.2 illustrates this distance by representing coordinates in a two-dimensional space, E2. The number of features employed determines the number of dimensions,  $E_n$ .

**Weighted Euclidean distance:** The weighted Euclidean distance is used when features vectors are given weights that reflect the relative importance of each feature. The weighted Euclidean distance  $d$  between the points  $(x_0, y_0)$  and  $(x_1, y_1)$  is given by the following equation:

$$d = \sqrt{w_x(x_0 - x_1)^2 + w_y(y_0 - y_1)^2} \quad (2.5)$$

where  $w_x$  and  $w_y$  are the weights of  $x$  and  $y$  respectively.

**Maximum distance:** The maximum distance computes the highest feature similarity, i.e. the one to define the closest analogy. For two points  $(x_0, y_0)$  and  $(x_1, y_1)$ , the maximum measure  $d$  is equivalent to the formula:

$$d = \sqrt{\max((x_0 - x_1)^2, (y_0 - y_1)^2)} \quad (2.6)$$

This effectively reduces the similarity measure down to a single feature, although this feature may differ for each retrieval episode. So, for a given “new” project  $P_{new}$ , the closest project in the case base will be the one that has at least one size feature with the most similar value to the same feature in project  $P_{new}$ .



### *Scaling*

Scaling (also known as standardisation) represents the transformation of attribute values according to a defined rule, such that all attributes present values within the same range and hence have the same degree of influence on the results [1]. A common method of scaling is to assign zero to the minimum observed value and one to the maximum observed value [15]. This is the strategy used by ANGEL.

### *Number of Analogies*

The number of analogies refers to the number of most similar cases that will be used to generate the estimation. With small sets of data it is reasonable to consider only a small number of analogies [1]. Several studies in software engineering have restricted their analysis to the closest analogy ( $k=1.0$ ) [3,30], while others have used two and three analogies [1,13,14,24,25,27,32].

### *Analogy Adaptation*

Once the similar cases have been selected the next step is to decide how to generate the estimation for project  $P_{new}$ . Choices of analogy adaptation techniques presented in the literature vary from the nearest neighbour [3,14], the mean of the closest analogies [36], the median [1], inverse distance weighted mean and inverse rank weighted mean [15], to illustrate just a few. The adaptations used to date for Web engineering are the nearest neighbour, mean of the closest analogies [24,25], and the inverse rank weighted mean [26,27].

Each adaptation is explained below:

*Mean:* The average of  $k$  analogies, when  $k > 1$ . This is a typical measure of central tendency, often used in the software and Web engineering literature. It treats all analogies as being equally influential on estimated effort.

*Median:* The median of  $k$  analogies, when  $k > 2$ . This is also a measure of central tendency, and has been used in the literature when the number of closest projects increases [1].

*Inverse rank weighted mean:* Allows higher ranked analogies to have more influence than lower ones. If we use three analogies, for example, the closest analogy (CA) would have weight = 3, the second closest (SC) weight = 2, and the third closest (LA) weight = 1. The estimation would then be calculated as:

$$InverseRankWeighedMean = \frac{3CA + 2SC + LA}{6} \quad (2.7)$$

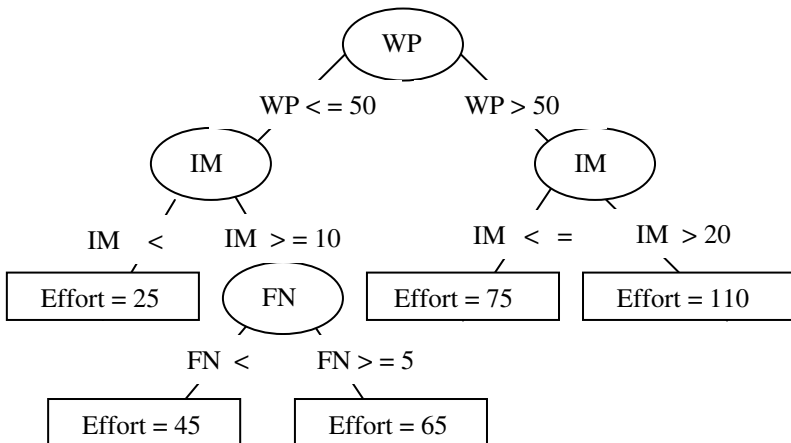
### Adaptation Rules

Adaptation rules are used to adapt estimated effort, according to a given criterion, such that it reflects the characteristics of the target project more closely. For example, in the context of effort prediction, the estimated effort to develop an application would be adapted such that it would also take into consideration the application's size values.

### Classification and Regression Trees

The objective of a Classification and Regression Tree (CART) model is to develop a simple tree-structured decision process for describing the distribution of a variable  $r$  given a vector of predictors  $vp$  [5]. A CART model represents a binary tree where the trees' leaves suggest values for  $r$  based on existing values of  $vp$ . For example, assume the estimated effort to develop a Web application can be determined by an estimated number of pages (WP), number of images (IM), and number of functions (FN). A regression tree such as the one shown in Fig. 2.3 is generated from data obtained from past finished Web applications, taking into account their existing values of effort, WP, IM, and FN. These are the predictors that make up the vector  $vp$ . Once the tree has been built it is used to estimate effort for a new project. So, to estimate effort for a new project where  $WP = 25$ ,  $IM = 10$ , and  $FN = 4$  we would navigate down the tree structure to find the estimated effort. In this case, 45 person hours.

Whenever predictors are numerical the CART tree is called a *regression tree* and whenever predictors are categorical the CART tree is called a *classification tree*.



**Fig. 2.3.** Example of a regression tree for Web cost estimation

A CART model constructs a binary tree by recursively partitioning the predictor space (set of values of each of the predictors in vector  $vp$ ) into subsets where the distribution of values for the response variable (effort) is successively more uniform. The partition itself is determined by splitting rules associated with each of the non-leaf nodes.

A “purity” function calculated from the predictor data is employed to split each node. There are numerous types of “purity” functions where the choice is determined by the software tool used to build the CART model, the type of predictors employed, and the goals for using a CART model (e.g. using it for cost estimation). The sequence used with CART is as follows:

- a) Past data is used to generate a CART model.
- b) This model is then traversed manually in order to obtain estimated effort, using as input values for the new project characteristics.
- c) The sequence described corresponds to steps 1, 2, and 3 from Fig. 2.1, in contrast to that for expert opinion and CBR.

## 2.3 Measuring Effort Prediction Power and Accuracy

An effort estimation model  $m$  uses historical data of finished projects to predict the effort of a new project. Some believe this is enough to provide accurate effort estimates. However, to gauge the accuracy of this model we need to measure its predictive accuracy.

To measure a model’s predictive accuracy first calculate the predictive power for each of a set of new projects  $p_1$  to  $p_n$  that used the effort estimation model  $m$ . Once predictive power for  $p_1$  to  $p_n$  has been obtained, their values are aggregated, which gives the predictive power of model  $m$  and hence its corresponding predictive accuracy.

This section describes how to measure the predictive power of a model, and how to measure a model’s predictive accuracy.

### 2.3.1 Measuring Predictive Power

The most common approaches to date for measuring predictive power of effort estimation models are:

- The Mean Magnitude of Relative Error (MMRE) [37]
- The Median Magnitude of Relative Error (MdMRE) [30]
- The Prediction at level  $n$  (Pred( $n$ )) [36]

The basis for calculating MMRE and MdMRE is to use the Magnitude of Relative Error (MRE) [16], defined as:

$$MRE = \frac{|e - \hat{e}|}{e} \quad (2.8)$$

where  $e$  is the actual effort and  $\hat{e}$  is the estimated effort.

The mean of all MREs is the MMRE, calculated as:

$$MMRE = \frac{1}{n} \sum_{i=1}^{i=n} \frac{|e_i - \hat{e}_i|}{e_i} \quad (2.9)$$

As the mean is calculated by taking into account the value of every estimated and actual effort from the data set employed, the result may give a biased assessment of a model's predictive power when there are several projects with large MREs.

An alternative to the mean is the median, which also represents a measure of central tendency, as it is less sensitive to the existence of several large MREs. The median of MRE values for the number  $i$  of observations (data values) is called the MdMRE.

Another indicator which is commonly used is the prediction at level  $l$ , also known as  $\text{Pred}(l)$ . This measures the percentage of effort estimates that are within  $l\%$  of their actual values.

MMRE, MdMRE, and  $\text{Pred}(l)$  are taken as the *de facto* standard evaluation criteria to measure the predictive power of effort estimation models [39].

### 2.3.2 Measuring Predictive Accuracy

In order to calculate the predictive accuracy of a given effort estimation model  $m$ , based on a given data set of finished projects  $d$ , we do the following:

1. Divide the data set  $d$  into a training set  $t$  and a validation set  $v$ . It is common to create training sets that use 66% of the projects from the complete data set, leaving 34% for the validation set.
2. Using  $t$ , produce an effort estimation model  $m$  (if applicable).
3. Using  $m$ , predict the effort for each of the projects in  $v$ , simulating new projects for which effort is unknown.

Once done, we will have, for each project in  $v$ , an *estimated* effort  $\hat{e}$ , calculated using the model  $m$ , and also the *actual* effort  $e$  that the project actually used. We are now able to calculate the predictive power (MRE) for each project in the validation set  $v$ . The final step, once we have obtained the

predictive power for each project, is to aggregate these values to obtain MMRE, MdMRE, and Pred(25) for  $v$ , which is taken to be the same for  $m$ .

Calculated MMREs and MdMREs with values up to 0.25, and Pred(25) at 75% or above, indicate good prediction models [6].

This splitting of a data set into training and validation sets is also known as cross-validation. An  $n$ -fold cross-validation means the original data set is divided into  $n$  subsets of training and validation sets. When the validation set has only one project the cross-validation is called “leave-one-out” cross-validation. This is an approach commonly used when assessing prediction accuracy using CBR.

## 2.4 Which Is the Most Accurate Prediction Technique?

Section 2.2 introduced numerous techniques for obtaining effort estimates for a new project, and all have been used, each with a varying degree of success. Therefore the question that is often asked is: Which of the techniques provides the most accurate prediction?

To date, the answer to this question has been simply “it depends”.

Algorithmic models have some advantages over machine learning and expert opinion, such as:

1. Allowing users to see how a model derives its conclusions, an important factor for verification as well as theory building and understanding of the process being modelled [10].
2. The need to be specialised relative to the local environment in which they are used [21,7].

Despite these advantages, no convergence on which effort estimation technique has the best predictive power has yet been reached, even though comparative studies have been carried out over the last 15 years (e.g. [1,3,4,8–10,12–16,30,32,35–37]).

One justification is that these studies have used data sets with differing characteristics (e.g. number of outliers,<sup>1</sup> amount of collinearity,<sup>2</sup> number of variables, number of projects) and different comparative designs.

Shepperd and Kadoda [35] presented evidence for the relationship between the success of a particular technique and training set size, nature of the “effort estimation” function (e.g. continuous<sup>3</sup> or discontinuous<sup>4</sup>), and

---

<sup>1</sup> An outlier is a value which is far from the others.

<sup>2</sup> Collinearity represents the existence of a linear relationship between two or more independent variables.

<sup>3</sup> A continuous function is one in which “small changes in the input produce small changes in the output” ([http://e.wikipedia.org/wiki/Continuous\\_function](http://e.wikipedia.org/wiki/Continuous_function)).

characteristics of the data set. They concluded that the “best” prediction technique that can work on any type of data set may be impossible to obtain.

Mendes et al. [28] investigated three techniques for Web effort estimation (stepwise regression, case-based reasoning, and regression trees) by comparing the prediction accuracy of their respective models. Stepwise regression provided the best results overall. This trend has also been confirmed using a different data set of Web projects [29]. This is therefore the technique to be used in Sect. 2.5 to build an effort estimation model for estimating effort for Web projects.

## 2.5 Case Study

The case study we present here describes the construction and further validation of a Web effort estimation model using data on industrial Web projects, developed by Web companies worldwide, from the Tukutuku database [29].<sup>5</sup> This database is part of the ongoing Tukutuku project,<sup>6</sup> which collects data on Web projects, for the development of effort estimation models and to benchmark productivity across and within Web companies.

The database contains data on 87 Web projects: 34 and 13 come from 2 single Web companies respectively and the remaining 40 projects come from another 23 companies. The Tukutuku database uses 6 variables to store specifics about each company that volunteered projects, 10 variables to store particulars about each project, and 13 variables to store data about each Web application (see Table 2.1). Company data is obtained once and both project and application data are gathered for each volunteered project.

All results presented were obtained using the statistical software SPSS 10.1.3 for Windows. Further details on the statistical methods used throughout this case study are given in Chap. 12. Finally, all the statistical tests set the significance level at 95% ( $\alpha = 0.05$ ).

---

<sup>4</sup> “If small changes in the input can produce a broken jump in the changes of the output, the function is said to be discontinuous (or to have a discontinuity)” ([http://e.wikipedia.org/wiki/Continuous\\_function](http://e.wikipedia.org/wiki/Continuous_function)).

<sup>5</sup> The raw data cannot be displayed here due to a confidentiality agreement with those companies that have volunteered data on their projects.

<sup>6</sup> <http://www.cs.auckland.ac.nz/tukutuku>.

**Table 2.1.** Variables for the Tuketuku database

NAME	SCALE <sup>7</sup>	DESCRIPTION
<b>COMPANY DATA</b>		
COUNTRY	Categorical	Country company belongs to.
ESTABLISHED	Ordinal	Year when company was established.
SERVICES	Categorical	Type of services company provides.
NPEOPLEWD	Ratio	Number of people who work on Web design and development.
CLIENTIND	Categorical	Industry representative of those clients to whom applications are provided.
ESTPRACT	Categorical	Accuracy of a company's own effort estimation practices.
<b>PROJECT DATA</b>		
TYPEPROJ	Categorical	Type of project (new or enhancement).
LANGS	Categorical	Implementation languages used.
DOCPROC	Categorical	If project followed defined and documented process.
PROIMPR	Categorical	If project team involved in a process improvement programme.
METRICS	Categorical	If project team part of a software metrics programme.
DEVTEAM	Ratio	Size of project's development team.
TEAMEXP	Ratio	Average team experience with the development language(s) employed.
TOTEFF	Ratio	Actual total effort used to develop the Web application.
ESTEFF	Ratio	Estimated total effort necessary to develop the Web application.
ACCURACY	Categorical	Procedure used to record effort data.
<b>WEB APPLICATION</b>		
TYPEAPP	Categorical	Type of Web application developed.
TOTWP	Ratio	Total number of Web pages (new and reused).
NEWWP	Ratio	Total number of new Web pages.
TOTIMG	Ratio	Total number of images (new and reused).
NEWIMG	Ratio	Total number of new images created.
HEFFDEV	Ratio	Minimum number of hours to develop a single function/feature by one experienced developer that is considered high (above average). <sup>8</sup>
HEFFADPT	Ratio	Minimum number of hours to adapt a single function/feature by one experienced developer that is considered high (above average). <sup>9</sup>

<sup>7</sup> The different types of measurement scale are described in Chap. 12.

<sup>8</sup> This number is currently set to 15 hours based on the collected data.

NAME	SCALE <sup>7</sup>	DESCRIPTION
HFOTS	Ratio	Number of reused high-effort features/functions without adaptation.
HFOTSA	Ratio	Number of reused high-effort features/functions adapted.
HNEW	Ratio	Number of new high-effort features/functions.
FOTS	Ratio	Number of reused low-effort features without adaptation.
FOTSA	Ratio	Number of reused low-effort features adapted.
NEW	Ratio	Number of new low-effort features/functions.

The following sections describe our data analysis procedure, adapted from [23], which consists of:

1. Data validation
2. Variables and model selection
3. Model inspection
4. Extraction of effort equation
5. Model validation

### 2.5.1 Data Validation

Data validation (DV) performs the first screening of the collected data. It generally involves understanding what the variables are (e.g. purpose, scale type, see Table 2.1) and also uses descriptive statistics (e.g. mean, median, minimum, maximum) to help identify any missing or unusual cases.

Table 2.2 presents summary statistics for numerical variables. None of the numerical variables seem to exhibit unusual or missing values, although this requires careful examination. For example, one would find it strange to see *zero* as minimum value for Total Images (TOTIMG) or *one* as minimum value for Total Web Pages (TOTWP). However, it is possible to have either a Web application without any images or a Web application that provides all its content and functionality within a single Web page. Another example relates to the maximum number of Web pages, which is 2000 Web pages. Although it does not seem possible at first to have such large number of pages we cannot simply assume this has been a data entry error. We were unable to obtain confirmation from the source company. However, further investigation revealed that 1980 pages were developed from scratch, and numerous new functions/features (five high-effort and seven low-effort) were also implemented. In addition, the development team consisted of two people who had very little experience with the six

---

<sup>9</sup> This number is currently set to 4 hours based on the collected data.



programming languages used. The total effort was 947 person hours, which can correspond to a three-month project assuming both developers worked at the same time. If we only consider number of pages and effort, the ratio of number of minutes per page is 27:1, which seems reasonable given the lack of experience of the development team and the number of different languages they had to use.

**Table 2.2.** Descriptive statistics for numerical variables

Variables	N	Min.	Max.	Mean	Median	Std. dev.
DEVTEAM	87	1	8	2.37	2	1.35
TEAMEXP	87	1	10	3.40	2	1.93
TOTWP	87	1	2000	92.40	25	273.09
NEWWP	87	0	1980	82.92	7	262.98
TOTIMG	87	0	1820	122.54	40	284.48
NEWIMG	87	0	800	51.90	0	143.25
HEFFDEV	87	5	800	62.02	15	141.25
HEFFADPT	87	0	200	10.61	4	28.48
HFOTS	87	0	3	.08	0	.41
HFOTSA	87	0	4	.29	0	.75
HNEW	87	0	10	1.24	0	2.35
FOTS	87	0	15	1.07	0	2.57
FOTSA	87	0	10	1.89	1	2.41
NEW	87	0	13	1.87	0	2.84
TOTEFF	87	1	5000	261.73	43	670.36
ESTEFF	34	1	108	14.45	7.08	20.61

Once we have checked the numerical variables our next step is to check the categorical variables using their frequency tables as a tool (see Tables 2.4 to 2.7).

Tables 2.4 to 2.6 show that most projects followed a defined and documented process, and that development teams were involved in a process improvement programme and/or part of a software metrics programme. These positive trends are mainly due to the two single companies that together volunteered data on 47 projects (54% of our data set). They have answered “yes” to all three categories. No unusual trends seem to exist.

Table 2.7 shows that the majority of projects (83%) had the actual effort recorded on a daily basis, for each project and/or project task. These numbers are inflated by the two single companies where one chose category “good” (11 projects) and the other chose category “very good” (34 projects). The actual effort recording procedure is not an adequate effort estimator per

se, being used here simply to show that the effort data gathered seems to be reliable overall.

**Table 2.3.** Frequency table for type of project

Type of project	Frequency	%	Cumulative %
New	39	44.8	44.8
Enhancement	48	55.2	100.0
Total	87	100.0	

**Table 2.4.** Frequency table for documented process

Documented process	Frequency	%	Cumulative %
no	23	26.4	26.4
yes	64	73.6	100.0
Total	87	100.0	

**Table 2.5.** Frequency table for process improvement

Process improvement	Frequency	%	Cumulative %
no	28	32.2	32.2
yes	59	67.8	100.0
Total	87	100.0	

**Table 2.6.** Frequency table for metrics programme

Metrics programme	Frequency	%	Cumulative %
no	36	41.4	41.4
yes	51	58.6	100.0
Total	87	100.0	

**Table 2.7.** Frequency table for companies' effort recording procedure

Actual effort recording procedure	Frequency	%	Cumulative %
Poor	12	13.8	13.8
Medium	3	3.4	17.2
Good	24	27.6	44.8
Very good	48	55.2	100
Total	87	100.0	

Once the data validation is complete, we are ready to move on to the next step, namely variables and model selection.

## 2.5.2 Variables and Model Selection

The second step in our data analysis methodology is sub-divided into two separate and distinct phases: preliminary analysis and model building.

Preliminary analysis allows us to choose which variables to use, discard, modify, and, where necessary, sometimes create. Model building determines an effort estimation model based on our data set and variables.

### *Preliminary Analysis*

This important phase is used to create variables based on existing variables, discard unnecessary variables, and modify existing variables (e.g. joining categories). The net result of this phase is to obtain a set of variables that are ready to use in the next phase, model building. Since this phase will construct an effort model using stepwise regression we need to ensure that the variables comply with the assumptions underlying regression analysis, which are:

1. The input variables (independent variables) are measured without error. If this cannot be guaranteed then these variables need to be normalised.
2. The relationship between dependent and independent variables is linear.
3. No important input variables have been omitted. This ensures that there is no specification error associated with the data set. The use of a prior theory-based model justifying the choice of input variables ensures this assumption is not violated.
4. The variance of the residuals is the same for all combinations of input variables (i.e. the residuals are homoscedastic rather than heteroscedastic)<sup>10</sup>.
5. The residuals must be normally distributed.
6. The residuals must be independent, i.e. not correlated.<sup>11</sup>
7. The independent variables are not linearly dependent, i.e. there are no linear dependencies among the independent variables.

The first task within the preliminary analysis phase is to examine the entire set of variables and check if there is a significant amount of missing values (> 60%). If yes, they should be automatically discarded as they prohibit the use of imputation methods<sup>12</sup> and will further prevent the identification of useful trends in the data. Table 2.2 shows that only ESTEFF presented missing values greater than 60%. ESTEFF was gathered to give

<sup>10</sup> Further details are provided in Chap. 12.

<sup>11</sup> Further details are provided in Chap. 12.

<sup>12</sup> Imputation methods are methods used to replace missing values with estimated values.

an idea of each company's own prediction accuracy; however, it will not be included in our analysis since it is not an effort predictor per se. Note that a large number of zero values on certain size variables do not represent missing or rounded values.

Next we present the analyses for numerical variables first, followed by the analyses for categorical variables.

### *Numerical Variables: Looking for Symptoms*

Our next step is to look for symptoms (e.g. skewness<sup>13</sup>, heteroscedasticity<sup>14</sup>, and outliers<sup>15</sup>) that may suggest the need for variables to be normalised, i.e. having their values transformed such that they resemble more closely a normal distribution. This step uses histograms, boxplots, and scatter plots.

Histograms, or bar charts, provide a graphical display, where each bar summarises the frequency of a single value or range of values for a given variable. They are often used to check if a variable is normally distributed, in which case the bars are displayed in the shape of a bell-shaped curve. Histograms for the numerical variables (see Figs. 2.4 to 2.6) suggest that all variables present skewed distributions, i.e. values not symmetrical about a central value.

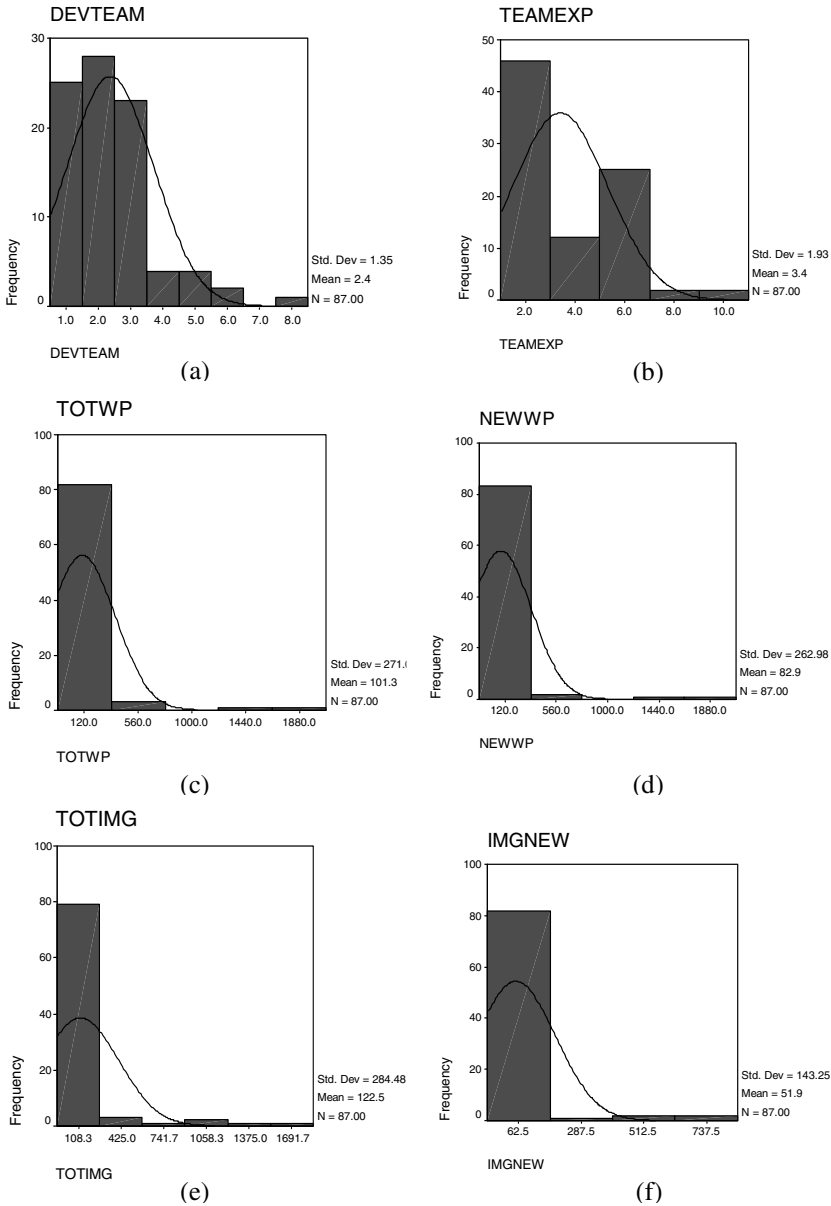
Next we use boxplots to check the existence of outliers. Boxplots (see Fig. 2.7) use the median, represented by the horizontal line in the middle of the box, as the central value for the distribution. The box's height is the inter-quartile range, and contains 50% of the values. The vertical (whiskers) lines up or down from the edges contain observations which are less than 1.5 times inter-quartile range. Outliers are taken as values greater than 1.5 times the height of the box. Values greater than 3 times the box's height are called extreme outliers [19].

---

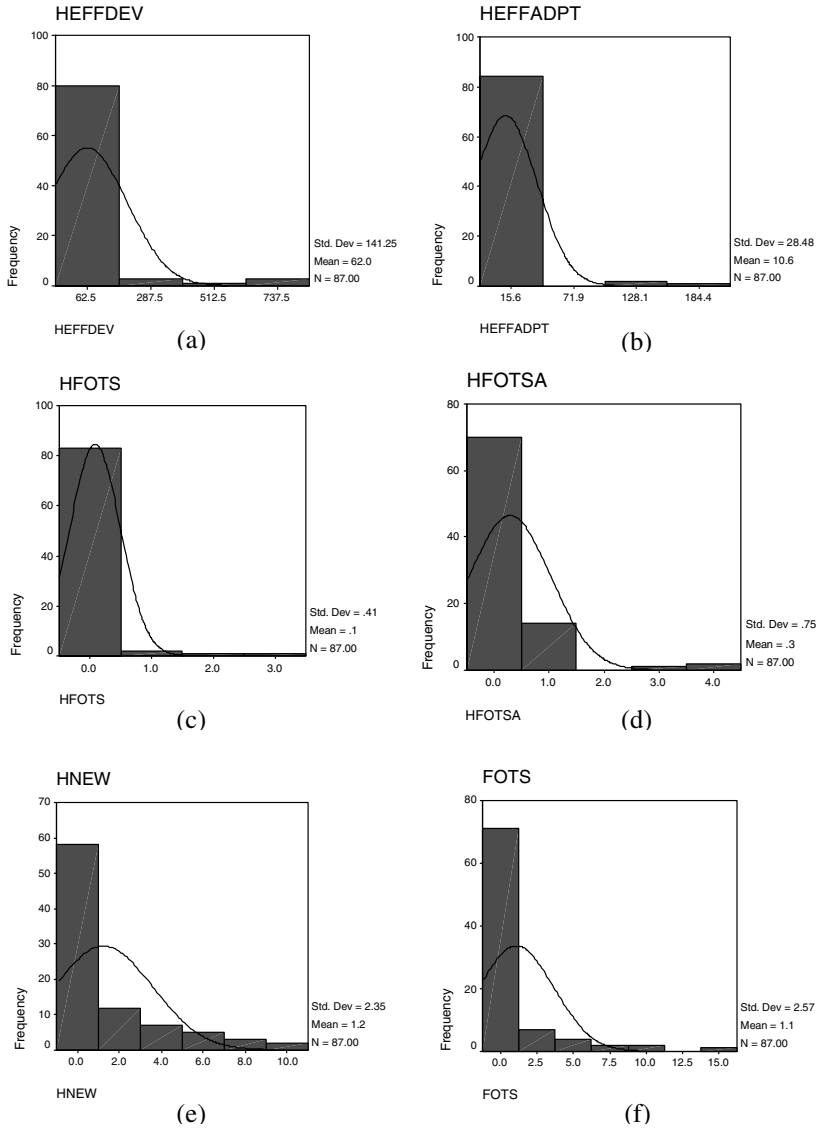
<sup>13</sup> Skewness measures to what extent the distribution of data values is symmetrical about a central value.

<sup>14</sup> Heteroscedasticity represents unstable variance of values.

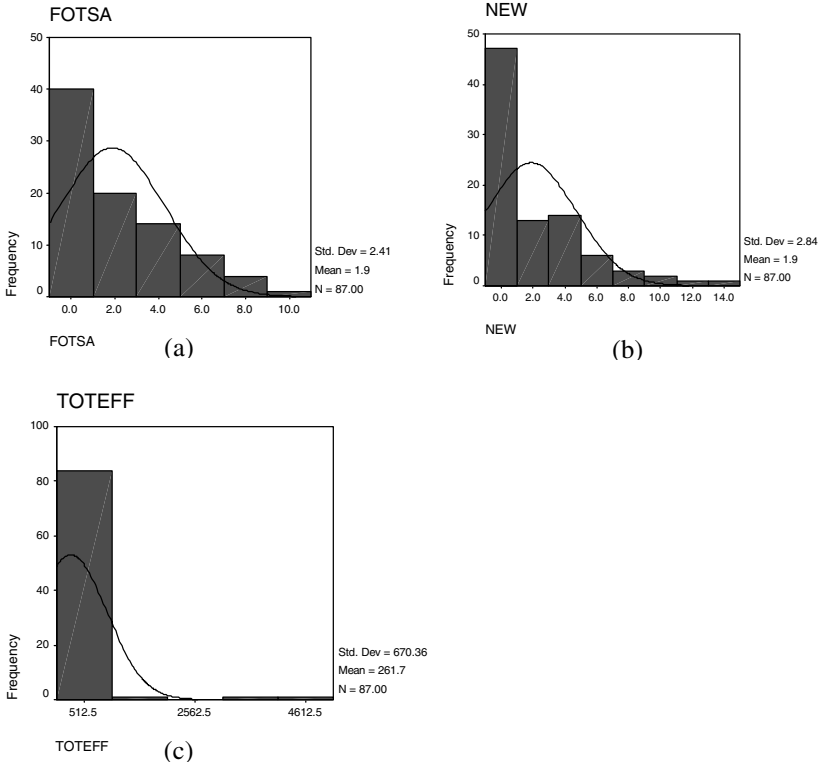
<sup>15</sup> Outliers are unusual values.



**Fig. 2.4.** Distribution of values for six numerical variables



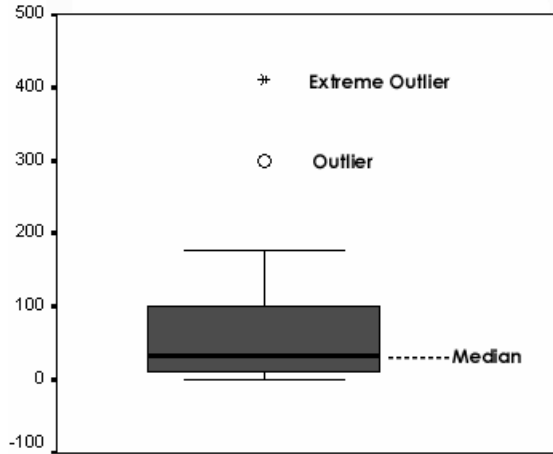
**Fig. 2.5.** Distribution of values for another six numerical variables



**Fig. 2.6.** Distribution of values for three numerical variables

When upper and lower tails are approximately equal and the median is in the centre of the box, the distribution is symmetric. If the distribution is not symmetric the relative lengths of the tails and the position of the median in the box indicate the nature of the skewness. The length of the box relative to the length of the tails gives an indication of the shape of the distribution. So, a boxplot with a small box and long tails represents a very peaked distribution, whereas a boxplot with a long box represents a flatter distribution [19].

The boxplots for numerical variables (see Fig. 2.8) indicate that they present a large number of outliers and peaked distributions that are not symmetric.



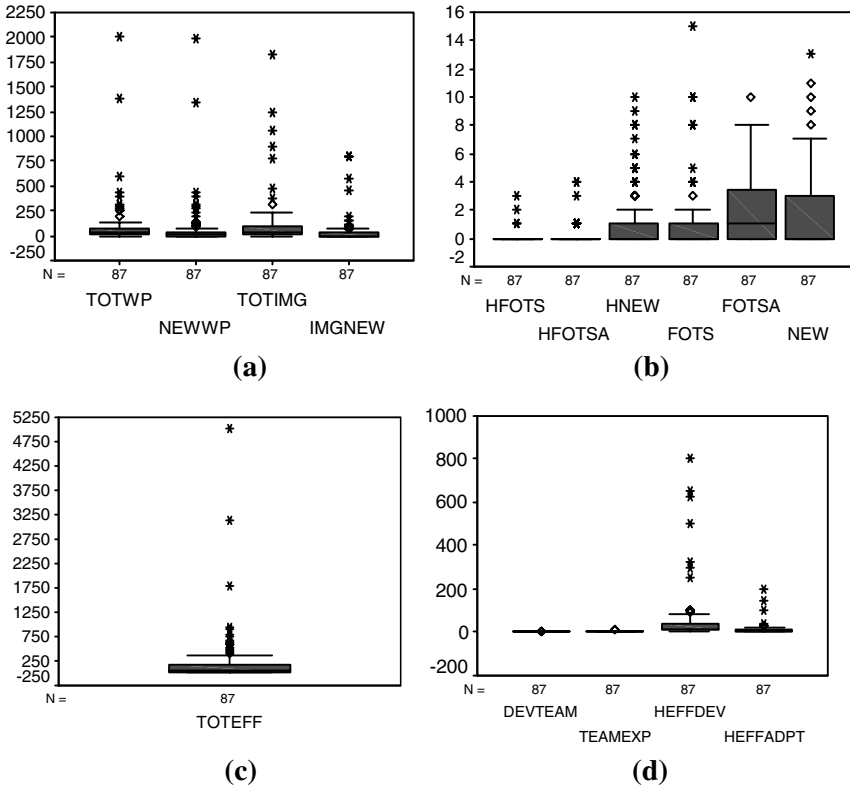
**Fig. 2.7.** Main components of a boxplot

Whenever outliers are present they should be investigated further, since they may be a result of data entry error. In our study we looked at all cases, in particular in relation to projects that exhibited very large effort values, but did not find anything in the data to suggest they should be removed from the data set. Note that when there are doubts about the correctness of the data, the best solution is to contact the data source for confirmation. Only if the source is not available should an assessment be based on consistency with other variables.

The histograms and boxplots both indicate symptoms of skewness and outliers. When this situation arises it is common practice to normalise the data, i.e. to transform the data trying to approximate the values to a normal distribution. A common transformation is to take the natural log ( $\ln$ ), which makes larger values smaller and brings the data values closer to each other [23]. This is the transformation applied in our case, to all numerical variables. For consistency, all variables with a value of zero had one added to their values prior to being transformed, as there is no natural log of zero.

The Tuketuku database uses six variables to record the number of features/functions for each application. Their histograms (Fig. 2.5(c)–(f), Fig. 2.6(a)–(b)) indicate that each has a large number of zeros, reducing their likelihood of being selected by the stepwise procedure. We therefore decided to group their values by creating two new variables – TOTHIGH (summation of HFOTS, HFOTSA, and HNEW) and TOTNHIGH (summation of FOTS, FOTSA, and NEW). Their histograms are presented in Fig. 2.9(a)–(b).





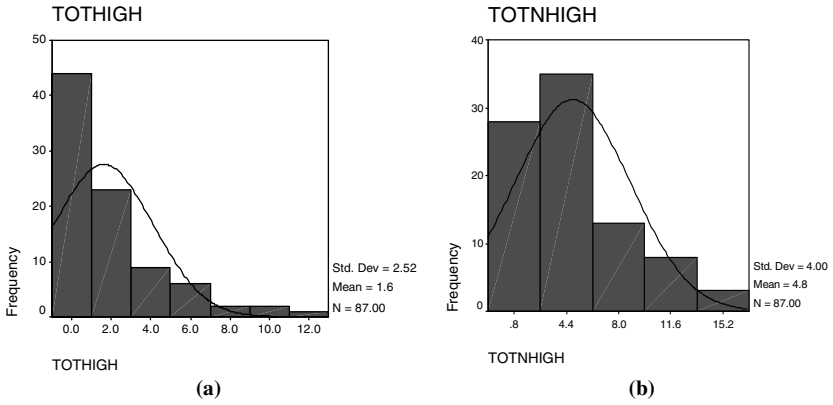
**Fig. 2.8.** Boxplots for numerical variables

Finally, we created a variable called NLANG, representing the number of different implementation languages used per project, replacing the original multi-valued variable that stored the names of the different implementation languages. The histogram is presented in Fig. 2.10.

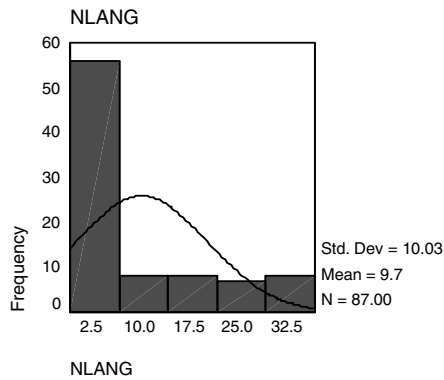
TOTHIGH, TOTNHIGH, and NLANG were also transformed since they presented skewness and outliers.

In the following sections, any variables that have been transformed have their names identified by an uppercase L, followed by the name of the variables they originated from.

The last part of the preliminary analysis is to check if the relationship between the dependent variable (LTOTEFF) and the independent variables is linear. The tool used to check such relationships is a scatter plot. Further details on scatter plots are provided in Chap. 12.



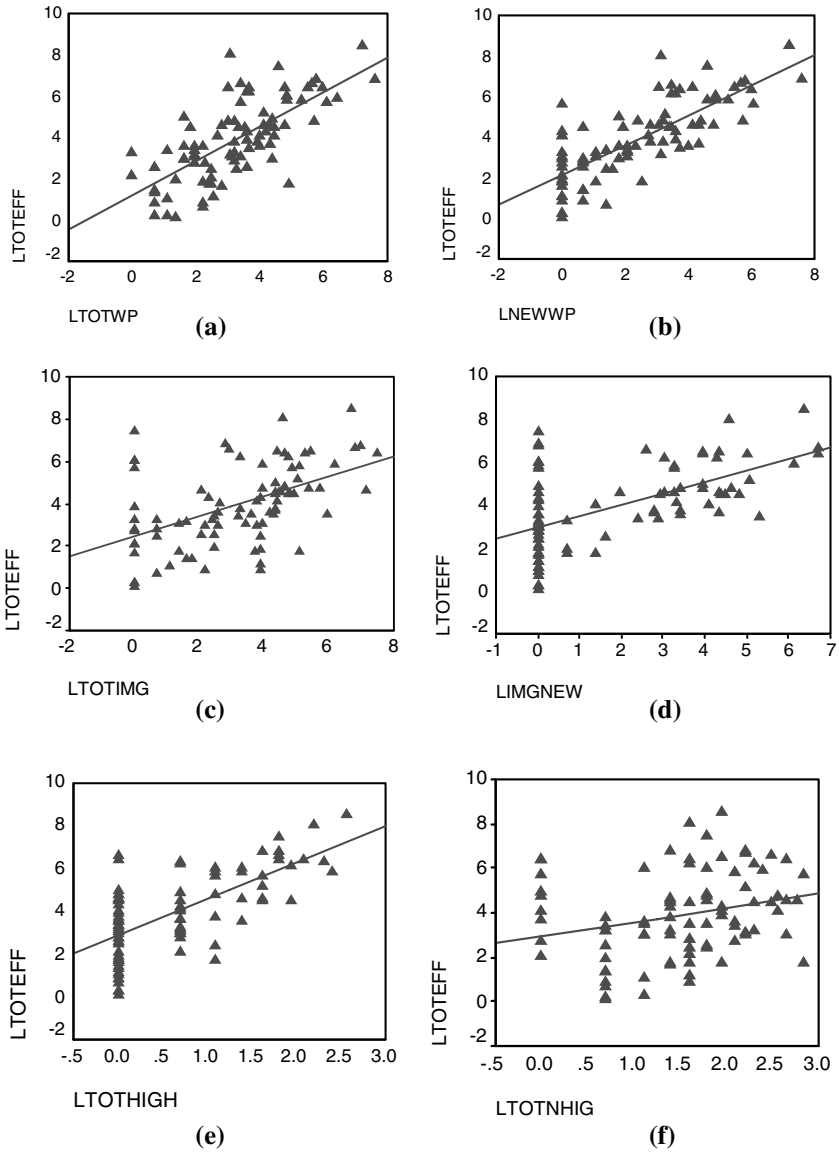
**Fig. 2.9.** Distribution of values for TOTHIGH and TOTNHIGH



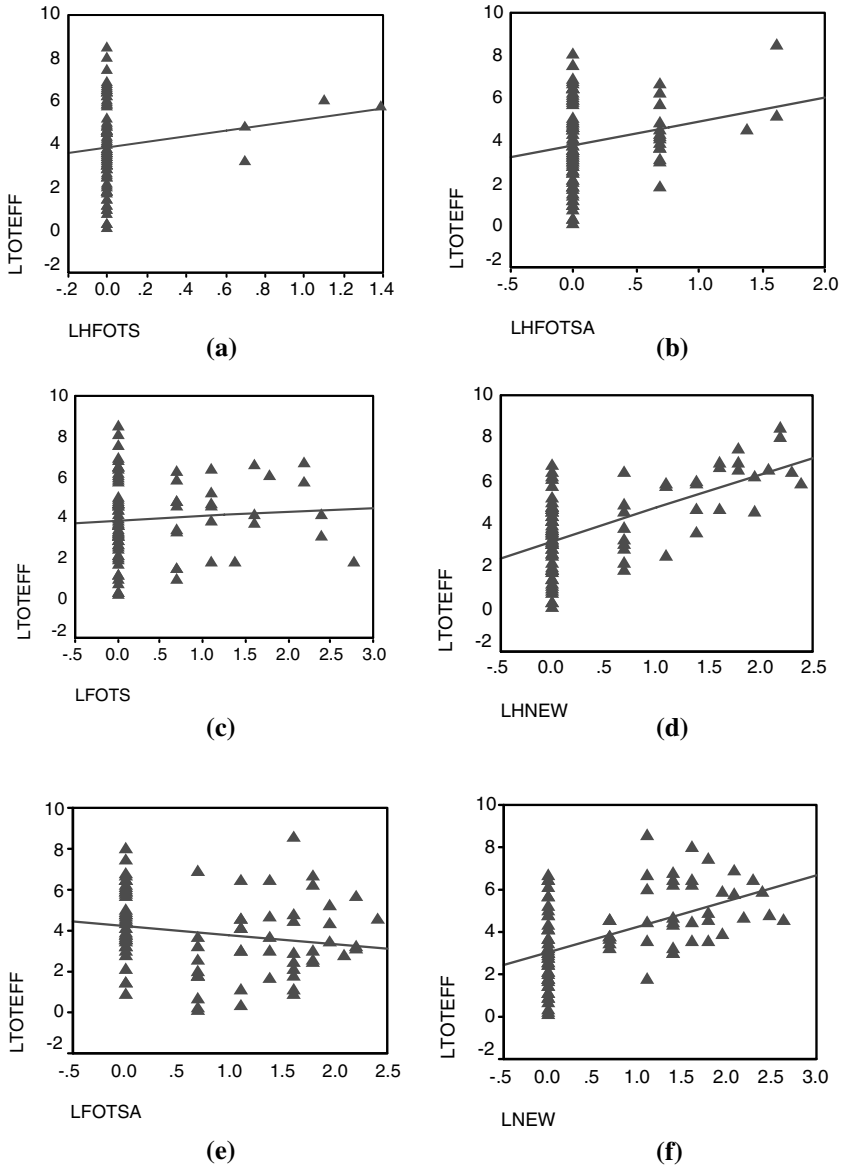
**Fig. 2.10.** Distribution of values for number of different implementation languages

*Numerical Variables: Relationship with Total Effort*

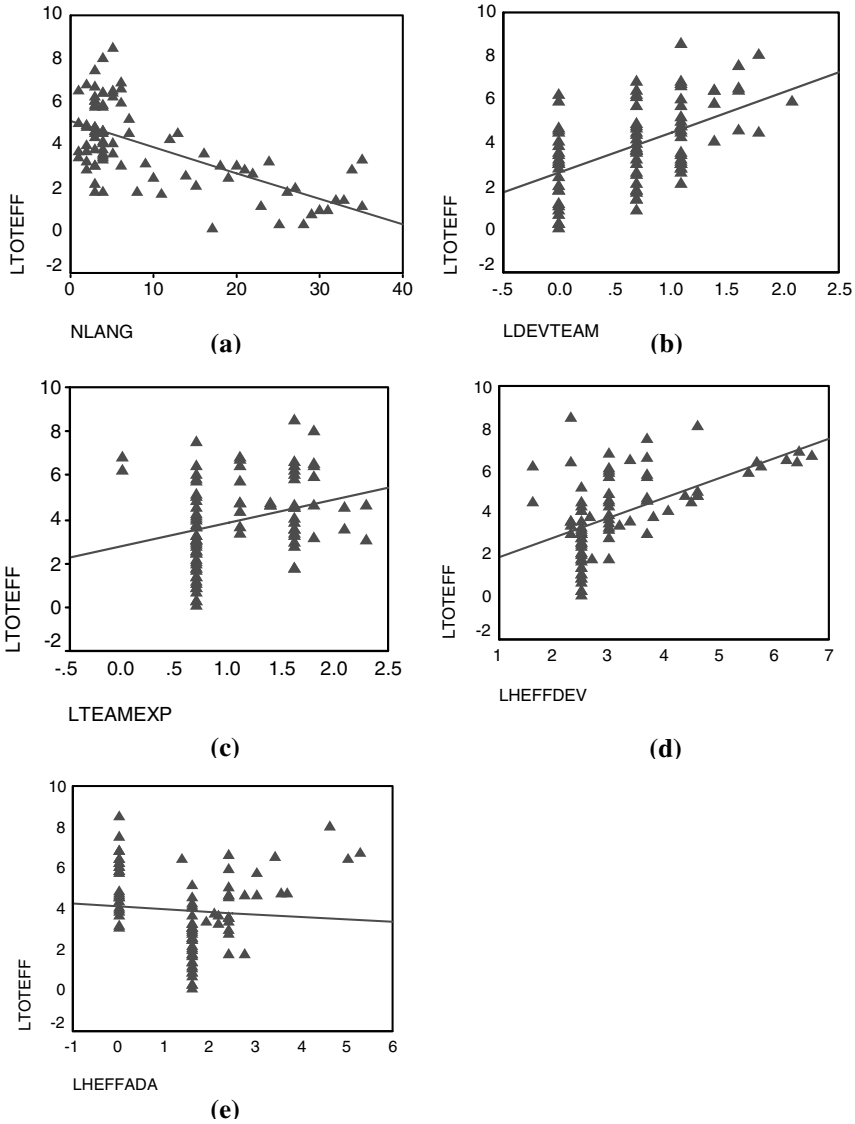
Scatter plots are used to explore possible relationships between numerical variables. They also help to identify strong and weak relationships between two numerical variables. A strong relationship is represented by observations (data points) falling very close to or on the trend line. Examples of such relationships are shown in Fig. 2.11(a)–(f), Fig. 2.12(d)–(f), and Fig. 2.13(a)–(d). A weak relationship is shown by observations that do not form a clear pattern, which in our case is a straight line. Examples of such relationships are shown in Fig. 2.12(a)–(c), and Fig. 2.13(e).



**Fig. 2.11.** Scatter plots showing strong relationships between  $ltoteff$  and several size variables



**Fig. 2.12.** Scatter plots for strong (d,e,f) and weak (a,b,c) relationships between ltoteff and several size variables



**Fig. 2.13.** Scatter plots for strong (a–d) and weak (e) relationships between ltoteff and independent variables

We can also say that a relationship is positively associated when values on the y-axis tend to increase with those on the x-axis (e.g. Fig. 2.11(a)–(f)). When values on the y-axis tend to decrease as those on the x-axis increase we say that the relationship is negatively associated (e.g. Fig. 2.12(e) and Fig. 2.13(a)).

Figures 2.11 to 2.13 show that most variables seem to present a positive relationship with LTOTEFF.

The scatter plots in Fig. 3.12(a)–(f) clearly show that the large number of zero values for the independent variables causes the dependent variable to exhibit more variability at the zero point, i.e. when independent variables have zero values, compared with non-zero values. This behaviour violates the fourth assumption underlying linear regression. Therefore within the context of this case study, we will exclude LHFOTS, LHFOSTA, LHNEW, LFOTS, LFOTSA, and LNEW from any subsequent analysis.

Our preliminary analyses for numerical variables is finished. Now we can move on and look at our categorical variables.

*Categorical Variables: Relationship with Total Effort*

This part of the analysis involves the creation of a table for each categorical variable where, for each of this variable’s category, we display the mean and median values of effort and the corresponding number of projects it is based on. The motivation is to check if there is a significant difference in effort by category. If there is, then we need to understand why.

Table 2.8 shows that on average, new projects required more effort, despite being smaller in number than enhancement projects. This should not come as a surprise since we generally know that building an application of size  $s$  from scratch takes longer than enhancing such application.

**Table 2.8.** Mean, median effort, and number of projects per type of project category

TYPEPROJ	N	Mean effort	Median effort
New	39	329.8	100.0
Enhancement	48	206.4	18.7
Total	87	261.7	43.0

Table 2.9 shows that on average, projects that did not use any documented process used higher effort, despite being smaller in number than projects that used a documented process. Further inspection of the data revealed that 70% of the 23 projects that did not use any documented process are new, and that 64% of the 64 projects that used a documented process are enhancement projects. These results are in line with those shown in Table 2.8.

**Table 2.9.** Mean, median effort, and number of projects per documented process category

DOCPROC	N	Mean effort	Median effort
no	23	307.5	50.0
yes	64	245.3	36.2
Total	87	261.7	43.0

**Table 2.10.** Mean, median effort, and number of projects per process improvement category

PROIMPR	N	Mean effort	Median effort
no	28	508.1	100.0
yes	59	144.8	25.2
Total	87	261.7	43.0

**Table 2.11.** Mean, median effort, and number of projects per metrics programme category

METRICS	N	Mean effort	Median effort
no	36	462.9	112.5
yes	51	119.7	21.0
Total	87	261.7	43.0

A similar pattern is observed in Tables 2.10 and 2.11, where, on average, projects that are not part of a process improvement or metrics programme required higher effort despite being smaller in size (61% of the 28 projects that are not part of a process improvement programme are new projects).

For projects that are not part of a metrics programme this percentage is also 61% of 36 projects. In both cases the majority of projects that are part of a process improvement or metrics programme are enhancement projects (63% of 59 and 67% of 51 respectively).

Our next step is to check the relationship between categorical variables and effort. Note that we cannot use scatter plots as categorical variables are not numerical. Therefore we use a technique called the one-way ANOVA (see Chap. 12 for details). Table 2.12 summarises the results for the one-way ANOVA.

**Table 2.12.** Results for the one-way ANOVA

Categorical variables	LTOTEFF
TYPEPROJ	Yes
DOCPROC	No
PROIMPR	Yes
METRICS	Yes

DOCPROC is the only categorical variable not significantly related to LTOTEFF; however, it will not be removed from further analysis as its relationship with LTOTEFF may be concealed at this stage [18].

Next we build the effort model using a two-step process. The first step is to use a manual stepwise regression based on residuals to select the categorical and numerical variables that jointly have a statistically significant effect on the dependent variable, LTOTEFF. The second step is to use these selected variables to build the final effort model using multivariate regression, i.e. linear regression using more than one independent variable.

The size measures used in our case study represent early Web size measures obtained from the results of a survey investigation [29], using data from 133 on-line Web forms aimed at giving quotes on Web development projects. In addition, the measures were validated by an established Web company, and a second survey involving 33 Web companies in New Zealand. Consequently it is our belief that the size measures identified are plausible effort predictors, not an ad-hoc set of variables with no underlying rationale.

#### *Building the Model Using a Two-Step Process*

This section describes the use of a manual stepwise regression based on residuals to build the effort model. This technique, proposed by Kitchenham [18], enables the use of information on residuals to handle relationships amongst independent variables. In addition, it only selects the input variables that jointly have a statistically significant effect on the dependent variable, thus avoiding any multi-collinearity problems.

The input variables to use are those selected as a result of our preliminary analyses, which are: LTOTWP, LNEWWP, LTOTIMG, LIMGNEW, LTOTHIGH, LTOTNHIG, TYPEPROJ, DOCPROC, PROIMPR, and METRICS.

Note: the distinct values of a categorical variables are called *levels*. For example, the categorical variable DOCPROC has two levels – Yes and No.

The manual stepwise technique applied to categorical variables comprises the following steps [18]:

- Step 1. Identify the categorical variable that has a statistically significant effect on LTOTEFF and gives the smallest error term (mean square within groups). This is obtained by applying simple analysis of variance (ANOVA) using each categorical variable in turn (CV1).
- Step 2. Remove the effect of the most significant categorical variable to obtain residuals (ResC1). This means that for each level of the most significant categorical variable, subtract the mean effort from the project effort values. Note that effort represents the normalised effort – LTOTEFF.



- 
- Step 3. Apply ANOVA using each remaining categorical variable in turn, this time measuring their effect on ResC1.
  - Step 4. Any categorical variables that had a statistically significant effect on LTOTEFF (in step 1), but have no statistically significant effect on ResC1, are variables related to CV1 and offer no additional information about the dependent variable. They can therefore be eliminated from the stepwise regression.
  - Step 5. Identify the next most significant categorical variable from step 4 (CV2). Again, if there are several statistically significant variables, choose the one that minimises the error term.
  - Step 6. Remove the effect of CV2 to obtain residuals (ResC2).
  - Step 7. Apply ANOVA using each remaining categorical variable in turn, this time measuring their effect on ResC2.
  - Step 8. Any categorical variables that had a statistically significant effect on ResC1, but have no statistically significant effect on ResC2, are variables related with CV2 and offer no additional information about the dependent variable. They can therefore be eliminated from the stepwise regression.
  - Step 9. Repeat the stepwise process until all statistically significant categorical variables are removed or none of the remaining variables have a statistically significant effect on the current residuals.

The initial level means for the four categorical variables to be used in our manual stepwise process are presented in Table 2.13.

Numerical variables can also be added to this stepwise procedure. Their impact on the dependent variable can be assessed using linear regression, and obtaining the mean squares for the regression model and residual. Whenever a numerical variable is the most significant, its effect has to be removed, i.e. the obtained residuals are the ones further analysed.

To construct the full regression model, apply a multivariate regression using only the variables that have been selected from the manual stepwise procedure. At each stage of the stepwise process we also need to verify the stability of the model. This involves identifying large residual and high-influence data points (i.e. projects), and also checking if residuals are homoscedastic and normally distributed. Several types of plots (e.g. residual, leverage, probability) and statistics are available in most statistics tools to accomplish such task.

**Table 2.13.** Initial level means for categorical variables

Variable/Level	No. projects	Total LTOTEFF	Mean LTOTEFF
TYPEPROJ/New	39	186.7	4.8
TYPEPROJ/Enhancement	48	154.1	3.2
DOCPROC/Yes	64	244.3	3.8
DOCPROC/No	23	96.5	4.2
PROIMPR/Yes	59	204.4	3.5
PROIMPR/No	28	136.4	4.9
METRICS/Yes	51	163.2	3.2
METRICS/No	36	177.6	4.9

The plots we have employed here are:

- A residual plot showing residuals vs. fitted values. This allows us to investigate if the residuals are random and normally distributed. For numerical variables the plotted data points should be distributed randomly about zero. They should not exhibit patterns such as linear or non-linear trends, or increasing or decreasing variance. For categorical variables the pattern of the residuals should appear “as a series of parallel, angled lines of approximately the same length” [18].
- A normal P–P plot (probability plots) for the residuals. Normal P–P plots are generally employed to verify if the distribution of a variable is consistent with the normal distribution. When the distribution is normal, the data points are close to linear.
- Cook’s D statistic to identify projects that exhibited jointly a large influence and large residual [23]. Any projects with D greater than  $4/n$ , where  $n$  represents the total number of projects, are considered to have a high influence on the results. When there are high-influence projects the stability of the model is tested by removing these projects and observing the effect their removal has on the model. If the coefficients remain stable and the adjusted  $R^2$  increases, this indicates that the high-influence projects are not destabilising the model and therefore do not need to be removed.

### *First Cycle*

Table 2.14 shows the results of applying ANOVA to categorical and numerical variables. This is the first cycle in the stepwise procedure. The numerical variable LNEWWP is the most significant, since it results in the smallest error term, represented by a within-groups mean square value of 1.47.

**Table 2.14.** ANOVA for each categorical and numerical variable for first cycle

Variable	Levels	Mean	No. proj	Between- groups MS	Within- groups MS	F test level of signifi- cance
TYPEPROJ	New	4.79	39	53.56	3.05	17.56
TYPEPROJ	Enhancement	3.20	48			$p < 0.01$
DOCPROC	Yes	3.82	64	2.44	3.65	0.42
DOCPROC	No	4.20	23			n.s.
PROIMPR	Yes	3.46	59	37.38	3.24	11.64
PROIMPR	No	4.87	28			$p = 0.001$
METRICS	Yes	3.20	51	63.54	2.93	21.67
METRICS	No	4.93	36			$p < 0.01$
LTOTWP	$LTOTEFF = 1.183 + 0.841LTOTWP$			158.22	1.82	86.97
						$p < 0.01$
<b>LNEWWP</b>	<b><math>LTOTEFF = 2.165 + 0.731LNEWWP</math></b>			<b>188.21</b>	<b>1.47</b>	<b>128.36</b>
						<b><math>p &lt; 0.01</math></b>
LTOTIMG	$LTOTEFF = 2.428 + 0.471LTOTIMG$			78.55	2.76	28.50
						$p < 0.01$
LIMGNEW	$LTOTEFF = 2.98 + 0.524LIMGNEW$			104.35	2.45	42.54
						$p < 0.01$
LTOTHIGH	$LTOTEFF = 2.84 + 1.705LTOTHIGH$			143.04	2.00	71.61
						$p < 0.01$
LTOTNHIG	$LTOTEFF = 2.954 + 0.641LTOTNHIG$			21.12	3.43	6.15
						$p = 0.015$

The single variable regression equation with LTOTEFF as the dependent/response variable and LNEWWP as the independent/predictor variable gives an adjusted  $R^2$  of 0.597. Two projects are identified with Cook's  $D > 0.045$ ; however, their removal did not seem to destabilise the model, i.e. after their removal the coefficients remained stable and the adjusted  $R^2$  increased. Furthermore, there was no indication from the residual and P-P plots that the residuals were non-normal. The residuals resulting from the linear regression are used for the second cycle in the stepwise procedure.

### *Second Cycle*

Table 2.15 shows the results of applying ANOVA to categorical and numerical variables. This is the second cycle in the stepwise procedure. The numerical variable LTOTHIGH is the most significant, since it results in the smallest error term, represented by a within-square value of 1.118. The linear regression equation with the residual as the dependent/response variable and LTOTHIGH as the independent/predictor variable gives an

adjusted  $R^2$  of 0.228. This time five projects are identified with Cook's  $D > 0.045$ ; however, their removal did not destabilise the model. In addition, the residual and P-P plots found no evidence of non-normality.

**Table 2.15.** ANOVA for each categorical and numerical variable for second cycle

Variable	Levels	Mean	No. proj	Be- tween- groups MS	Within- groups MS	F test level of signifi- cance
TYPEPROJ	New	-0.0181	39	0.023	1.466	0.016
TYPEPROJ	Enhancement	0.0147	48			n.s.
DOCPROC	Yes	0.0385	64	0.359	1.462	0.246
DOCPROC	No	-0.1072	23			n.s.
PROIMPR	Yes	-0.1654	59	5.017	1.407	3.565
PROIMPR	No	0.3486	28			n.s.
METRICS	Yes	-0.2005	51	4.954	1.408	3.519
METRICS	No	0.2840	36			n.s.
LTOTWP	LTOTEFF = $-0.474 + 0.146\text{LTOTWP}$			4.749	1.410	3.367 n.s.
LTOTIMG	LTOTEFF = $-0.417 + 0.132\text{LTOTIMG}$			6.169	1.394	4.427 p = 0.038
LIMGNEW	LTOTEFF = $-0.33 + 0.184\text{LIMGNEW}$			12.915	1.314	9.826 p = 0.002
<b>LTOTHIGH</b>	<b>LTOTEFF = <math>-0.49 + 0.775\text{LTOTHIGH}</math></b>			<b>29.585</b>	<b>1.118</b>	<b>26.457</b> <b>p &lt; 0.01</b>
LTOTNHIG	LTOTEFF = $-0.593 + 0.395\text{LTOTNHIG}$			8.015	1.372	5.842 p = 0.018

Table 2.15 also shows that TYPEPROJ, PROIMPR, METRICS, and LTOTWP have no further statistically significant effect on the residuals obtained in the previous cycle. Therefore they can all be eliminated from the stepwise procedure.

Once this cycle is complete the remaining input variables are DOCPROC, LTOTIMG, LIMGNEW, and LTOTNHIG.

*Third Cycle*

Table 2.16 shows the results of applying ANOVA to the four remaining categorical and numerical variables. This is the third cycle in the stepwise procedure. As shown in Table 2.16 none of the four remaining variables have any statistically significant effect on the current residuals, and as such the procedure finishes.

Finally, our last step is to construct the effort model using a multivariate regression analysis with only the input variables selected using the manual stepwise procedure – LNEWWP and LTOTHIGH. The coefficients for the effort model are presented in Table 2.17. Its adjusted  $R^2$  is 0.717 suggesting that LNEWWP and LTOTHIGH can explain 72% of the variation in LTOTEFF.

**Table 2.16.** ANOVA for each categorical and numerical variable for third cycle

Variable	Levels	Mean	No. proj	Be- tween- groups MS	Within- groups MS	F test level of signi- ficance
DOCPROC	Yes	0.0097	64	0.023	1.118	0.021
DOCPROC	No	-0.0272	23			n.s.
LTOTIMG	LTOTEFF = -0.109 + 0.034 LTOTIMG			0.419	1.113	0.376
LIMGNEW	LTOTEFF = -0.162 + 0.091 LIMGNEW			3.126	1.081	2.89
<b>LTOTNHIG</b>	<b>LTOTEFF = -0.192 + 0.128 LTOTNHIG</b>			<b>0.837</b>	<b>1.108</b>	<b>0.755</b>
						<b>n.s.</b>

**Table 2.17.** Coefficients for the effort model

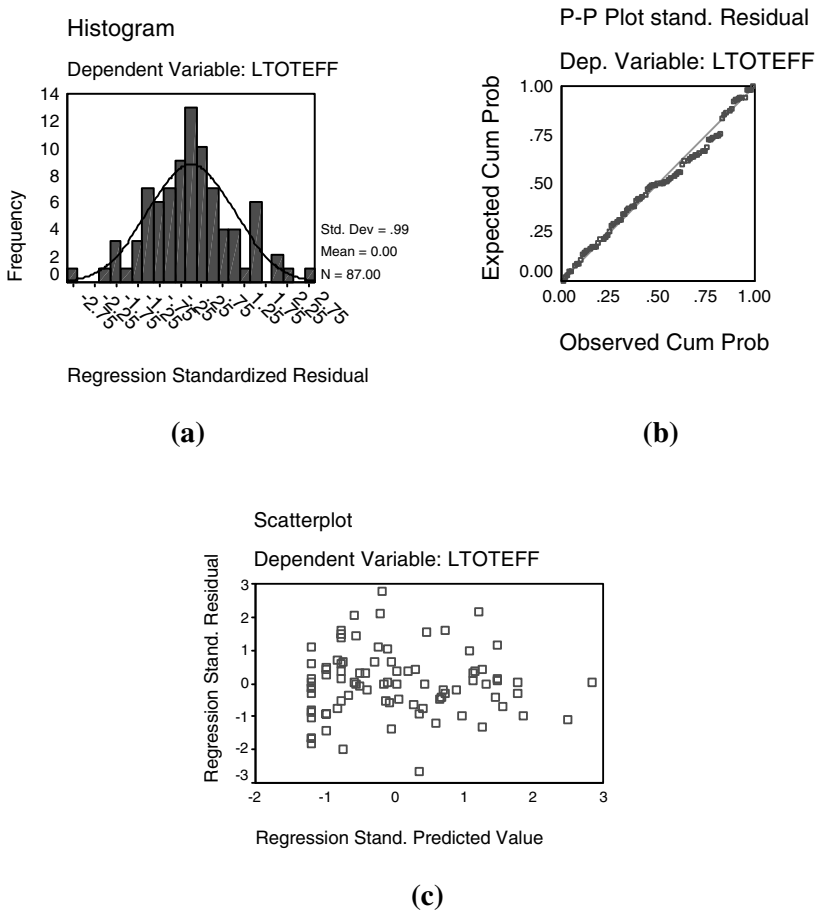
Variable	Coeff.	Std. error	t	P> t	[95% conf. interval]	
(Constant)	1.959	0.172	11.355	0.000	1.616	2.302
LNEWWP	0.553	0.061	9.003	0.000	0.431	0.675
LTOTHIGH	1.001	0.164	6.095	0.000	0.675	1.328

Four projects had Cook's  $D > 0.045$  (see Table 2.18) and so we followed the procedure adopted previously. We repeated the regression analysis after excluding these four projects from the data set. Their removal did not result in any major changes to the model coefficients and the adjusted  $R^2$  improved (0.757). Therefore we assume that the regression equation is reasonably stable for this data set and it is not necessary to omit these four projects from the data set.

**Table 2.18.** Four projects that presented high Cook's distance

ID	NEWWP	TOTHIGH	TOTEFF	Cook's D
20	20	0	625	0.073
25	0	4	300	0.138
32	22	8	3150	0.116
45	280	0	800	0.078

Figure 2.14 shows three different plots all related to residuals. The histogram (see Fig. 2.14(a)) suggests that the residuals are normally distributed, which is further corroborated by the P-P plot (see Fig. 2.14(b)). In addition, the scatter plot of standardised residuals versus standardised predicted values does not show any problematic patterns in the data.



**Fig. 2.14.** Several residual plots

Once the residuals and the stability of the regression model have been checked, we are in a position to extract the equation that represents the model.

### 2.5.3 Extraction of effort Equation

The equation that is obtained from Table 2.17 is the following:

$$LTOTEFF = 1.959 + 0.553LNEWWP + 1.001LTOTHIGH \quad (2.10)$$

This equation uses three variables that had been previously transformed, therefore we need to transform it back to its original state, which gives the following equation:

$$TOTEFF = 7.092 (NEWWP + 1)^{0.553} (TOTHIGH + 1)^{1.001} \quad (2.11)$$

In Eq. 2.11, the multiplicative value 7.092 can be interpreted as the effort required to develop one Web page.

Obtaining a model that has a good fit to the data and can alone explain a large degree of the variation in the dependent variable is not enough to assume this model will provide good effort predictions. To confirm this, it also needs to be validated. This is the procedure explained in Sect. 2.5.4.

### 2.5.4 Model Validation

As described in Sect. 2.3.2, to validate a model we need to do the following:

- Step 1. Divide data set  $d$  into a training set  $t$  and a validation set  $v$ .
- Step 2. Use  $t$  to produce an effort estimation model  $te$  (if applicable).
- Step 3. Use  $te$  to predict effort for each of the projects in  $v$ , as if these projects were new projects for which effort was unknown.

This process is known as cross-validation. For an  $n$ -fold cross-validation,  $n$  different training/validation sets are used. In this section we will show the cross-validation procedure using a one-fold cross-validation, with a 66% split. This split means that 66% of our project data will be used for model building, the remaining 34% to validate the model, i.e. the training set will have 66% of the total number of projects and the validation set will have the remaining 34%.

Our initial data set had 87 projects. At step 1 they are split into training and validation sets containing 58 and 29 projects respectively. Generally projects are selected randomly.

As part of step 2 we need to create an effort model using the 58 projects in the training set. We will create an effort model that only considers the variables that have been previously selected and presented in Eq. 2.10. These are: LNEWWP and LTOTHIGH. Here we do not perform the residual analysis or consider Cook's D since it is assumed these have also been

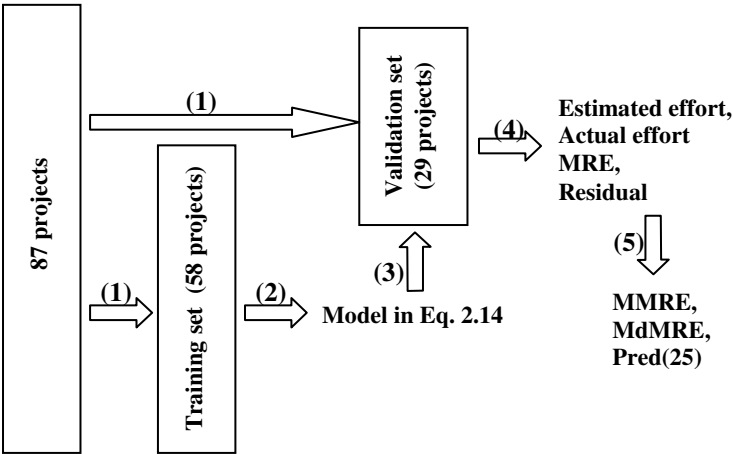
done using the generic equation, Eq. 2.10. The model’s coefficients are presented in Table 2.19, and the transformed equation is presented in Eq. 2.12. The adjusted  $R^2$  is 0.619.

**Table 2.19.** Coefficients for effort model using 58 projects

Variable	Coeff.	Std. error	t	P> t	[95% conf. interval]	
(Constant)	2.714	0.264	10.290	0.000	2.185	3.242
LNEWWP	0.420	0.073	5.749	0.000	0.273	0.566
LTOTHIGH	0.861	0.160	5.389	0.000	0.675	1.328

$$TOTEFF = 15.089 (NEWWP + 1)^{0.420} (TOTHIGH + 1)^{0.861} \quad (2.12)$$

To measure this model’s prediction accuracy we obtain the MMRE, MdMRE, and Pred(25) for the validation set. The model presented as Eq. 2.12 is applied to each of the 29 projects in the validation set to obtain estimated effort, and MRE is computed. Having the calculated estimated effort and the actual effort (provided by the Web companies), we are finally in a position to calculate MRE for each of the 29 projects, and hence MMRE, MdMRE, and Pred(25) for the entire 29 projects. This process is explained in Fig. 2.15.



**Fig. 2.15.** Steps used in the cross-validation process

Table 2.20 shows the measures of prediction accuracy, calculated from the validation set, and is assumed to represent the entire set of 87 projects.



**Table 2.20.** Prediction accuracy measures using model-based estimated effort

Measure	%
MMRE	129
MdMRE	73
Pred(25)	17.24

If we assume a good prediction model has an MMRE less than or equal to 25% and Pred(25) greater than or equal to 75% then the values presented in Table 2.20 suggest the accuracy of the effort model used is poor. However, if instead we were to use the average actual effort (average = 261) or the median actual effort for the 87 projects (median = 43) accuracy would be considerably worse. One viable approach for a Web company would be to use the effort model described above to obtain an estimated effort, and adapt the obtained values, taking into account factors such as previous experience with similar projects and the skills of the developers.

**Table 2.21.** Prediction accuracy measures based on average and median effort

	Average effort as estimated effort	Median effort as estimated effort
MMRE	4314%	663%
MdMRE	1413%	149%
Pred(25)	6.89%	3.44%

Table 2.21 presents the results for a one-fold cross-validation. However, research on effort estimation suggests that to have unbiased results for a cross-validation we should actually use at least a 20-fold cross-validation analysis [17]. This would represent for the data set presented here, the selection of 20 different training/validation sets and the aggregation of the MMREs, MdMREs, and Pred(25)s after accuracy for all 20 groups has been calculated.

## 2.6 Conclusions

This chapter introduced the concepts related to effort estimation, and described techniques for effort estimation using three general categories: expert opinion, algorithmic models and artificial intelligence (AI) techniques. In addition, it discussed how to measure effort prediction power and accuracy of effort estimation models.

This chapter also presented a case study that used data from industrial Web projects held in the Tukutuku database, to construct and validate an

effort estimation model. The size measures used in the case study represent early Web size measures obtained from the results of a survey investigation [29], using data from 133 on-line Web forms aimed at giving quotes for Web development projects. In addition, the measures were validated by an established Web company, and by a second survey involving 33 Web companies in New Zealand. Consequently we believe that the size measures identified are plausible effort predictors, not an ad-hoc set of variables with no underlying rationale.

Furthermore, a detailed analysis of the data was provided, with details of a manual stepwise procedure [18] used to build an effort estimation model. The two variables that were selected by the effort estimation model were the total number of new Web pages and the total number of high-effort features/functions in the application. Together they explained 76% of the variation in total effort. Note that the effort model constructed and the selected variables are applicable only to projects belonging to the data set on which they were constructed.

The case study details the mechanism that can be used by any Web company to construct and validate its own effort estimation models. Alternatively, Web companies that do not have a data set of past projects may be able to benefit from the cross-company effort estimation models provided within the context of the Tukutuku project, provided they are willing to volunteer data on three of their past finished projects.

## References

- 1 Angelis L, Stamelos I (2000) A Simulation Tool for Efficient Analogy Based Cost Estimation. *Empirical Software Engineering*, 5:35–68
- 2 Boehm B (1981) *Software Engineering Economics*. Prentice-Hall, Englewood Cliffs, NJ
- 3 Briand LC, El-Emam K, Surmann D, Wiecek I, Maxwell KD (1999) An Assessment and Comparison of Common Cost Estimation Modeling Techniques. In: *Proceedings of ICSE 1999*, Los Angeles, USA, pp 313–322
- 4 Briand LC, Langley T, Wiecek I (2000) A Replicated Assessment and Comparison of Common Software Cost Modeling Techniques. In: *Proceedings of ICSE 2000*, Limerick, Ireland, pp 377–386
- 5 Brieman L, Friedman J, Olshen R, Stone C (1984) *Classification and Regression Trees*. Wadsworth, Belmont, CA
- 6 Conte S, Dunsmore H, Shen V (1986) *Software Engineering Metrics and Models*. Benjamin/Cummings, Menlo Park, CA
- 7 DeMarco T (1982) *Controlling Software Projects: Management, Measurement and Estimation*. Yourdon, New York

- 8 Finnie GR, Wittig GE, Desharnais J-M (1997) A Comparison of Software Effort Estimation Techniques: Using Function Points with Neural Networks, Case-Based Reasoning and Regression Models. *Journal of Systems and Software*, 39:281–289
- 9 Gray A, MacDonell S (1997) Applications of Fuzzy Logic to Software Metric Models for Development Effort Estimation. In: *Proceedings of IEEE Annual Meeting of the North American Fuzzy Information Processing Society - NAFIPS*, Syracuse, NY, USA, pp 394–399
- 10 Gray AR, MacDonell SG (1997) A comparison of model building techniques to develop predictive equations for software metrics. *Information and Software Technology*, 39:425–437
- 11 Gray R, MacDonell SG, Shepperd MJ (1999) Factors Systematically associated with errors in subjective estimates of software development effort: the stability of expert judgement. In: *Proceedings of the 6th IEEE Metrics Symposium*
- 12 Hughes RT (1997) An Empirical investigation into the estimation of software development effort. PhD thesis, Dept. of Computing, University of Brighton
- 13 Jeffery R, Ruhe M, Wieczorek I (2000) A Comparative study of two software development cost modelling techniques using multi-organizational and company-specific data. *Information and Software Technology*, 42:1009–1016
- 14 Jeffery R, Ruhe M, Wieczorek I (2001) Using Public Domain Metrics to Estimate Software Development Effort. In: *Proceedings of the 7th IEEE Metrics Symposium*, London, UK, pp 16–27
- 15 Kadoda G, Cartwright M, Chen L, Shepperd MJ (2000) Experiences Using Case-Based Reasoning to Predict Software Project Effort. In: *Proceedings of the EASE 2000 Conference*, Keele, UK
- 16 Kemerer CF (1987) An Empirical Validation of Software Cost Estimation Models, *Communications of the ACM*, 30(5):416–429
- 17 Kirsopp C, Shepperd M (2001) Making Inferences with Small Numbers of Training Sets, January, TR02-01, Bournemouth University
- 18 Kitchenham BA (1998) A Procedure for Analyzing Unbalanced Datasets. *IEEE Transactions on Software Engineering*, April, 24(4):278–301
- 19 Kitchenham BA, MacDonell SG, Pickard LM, Shepperd MJ (2001) What accuracy statistics really measure. *IEE Proceedings Software*, June, 148(3):81–85
- 20 Kitchenham BA, Pickard LM, Linkman S, Jones P (2003) Modelling Software Bidding Risks. *IEEE Transactions on Software Engineering*, June, 29(6):54–554
- 21 Kok P, Kitchenham BA, Kirakowski J (1990) The MERMAID Approach to software cost estimation. In: *Proceedings of the ESPRIT Annual Conference*, Brussels, pp 296–314

- 22 Kumar S, Krishna BA, Satsangi PS (1994) Fuzzy systems and neural networks in software engineering project management. *Journal of Applied Intelligence*, 4:31–52
- 23 Maxwell K (2002) *Applied Statistics for Software Managers*. Prentice Hall PTR, Englewood Cliffs, NJ
- 24 Mendes E, Counsell S, Mosley N (2000) Measurement and Effort Prediction of Web Applications. In: *Proceedings of the 2nd ICSE Workshop on Web Engineering*, June, Limerick, Ireland, pp 57–74
- 25 Mendes E, Mosley N, Counsell S (2001) Web Metrics – Estimating Design and Authoring Effort. *IEEE Multimedia*, Special Issue on Web Engineering, January/March:50–57
- 26 Mendes E, Mosley N, Counsell S (2002) The Application of Case-Based Reasoning to Early Web Project Cost Estimation. In: *Proceedings of COMPSAC 2002*, Oxford, UK
- 27 Mendes E, Mosley N, Counsell S (2003) Do Adaptation Rules Improve Web Cost Estimation?. In: *Proceedings of the ACM Hypertext conference 2003*, Nottingham, UK
- 28 Mendes E, Mosley N, Counsell S (2003) A Replicated Assessment of the Use of Adaptation Rules to Improve Web Cost Estimation. In: *Proceedings of the ACM and IEEE International Symposium on Empirical Software Engineering*. Rome, Italy, pp 100–109
- 29 Mendes E, Mosley N, Counsell S (2003) Early Web Size Measures and Effort Prediction for Web Costimation. In: *Proceedings of the IEEE Metrics Symposium*. Sydney, Australia, September, pp 18–29
- 30 Myrtveit I, Stensrud E (1999) A Controlled Experiment to Assess the Benefits of Estimating with Analogy and Regression Models. *IEEE Transactions on Software Engineering*, July/August, 25(4):510–525
- 31 Ruhe M, Jeffery R, Wiczorek I (2003) Cost Estimation for Web Applications. In: *Proceedings of ICSE 2003*. Portland, USA
- 32 Schofield C (1998) An empirical investigation into software estimation by analogy. PhD thesis, Dept. of Computing, Bournemouth University
- 33 Schroeder L, Sjoquist D, Stephan P (1986) Understanding Regression Analysis: An Introductory Guide, No. 57. In: *Quantitative Applications in the Social Sciences*, Sage Publications, Newbury Park, CA
- 34 Selby RW, Porter AA (1998) Learning from examples: generation and evaluation of decision trees for software resource analysis. *IEEE Transactions on Software Engineering*, 14:1743–1757
- 35 Shepperd MJ, Kadoda G (2001) Using Simulation to Evaluate Prediction Techniques. In: *Proceedings of the IEEE 7th International Software Metrics Symposium*, London, UK, pp 349–358
- 36 Shepperd MJ, Schofield C (1997) Estimating Software Project Effort Using Analogies. *IEEE Transactions on Software Engineering*, 23(11):736–743

- 37 Shepperd MJ, Schofield C, Kitchenham B (1996) Effort Estimation Using Analogy. In: Proceedings of ICSE-18. Berlin
- 38 Srinivasan K, Fisher D (1995) Machine Learning approaches to estimating software development effort. *IEEE Transactions on Software Engineering*, 21:126–137
- 39 Stensrud E, Foss T, Kitchenham BA, Myrtveit I (2002) An Empirical validation of the relationship between the magnitude of relative error and project size. In: Proceedings of the IEEE 8th Metrics Symposium. Ottawa, pp 3–12

## Authors' Biographies

Dr. **Emilia Mendes** is a Senior Lecturer in Computer Science at the University of Auckland (New Zealand), where she leads the WETA (Web Engineering, Technology and Applications) research group. She is the principal investigator in the Tukutuku Research project,<sup>16</sup> aimed at developing and comparing Web effort models using industrial Web project data, and benchmarking productivity within and across Web companies. She has active research interests in Web measurement and metrics, and in particular Web cost estimation, Web size measures, Web productivity and quality measurement, and Web process improvement. Dr. Mendes is on the programme committee of numerous international conferences and workshops, and on the editorial board of the *International Journal of Web Engineering and Technology* and the *Journal of Web Engineering*. She has collaborated with Web companies in New Zealand and overseas on Web cost estimation and usability measurement. Dr. Mendes worked in the software industry for ten years before obtaining her PhD in Computer Science from the University of Southampton (UK), and moving to Auckland. She is a member of the Australian Software Measurement Association.

Dr. **Nile Mosley** is the Technical Director of a software development company. He has active research interests in software measurement and metrics, and object-oriented programming languages. He obtained his PhD in Pure and Applied Mathematics from Nottingham Trent University (UK).

**Steve Counsell** obtained a BSc (Hons) in Computer Studies from the University of Brighton and an MSc in Systems Analysis from the City University in 1987 and 1988, respectively. After spending some time in industry as a developer, he obtained his PhD in 2002 from the University of London and is currently a Lecturer in the Department of Information Systems and Computing at Brunel University. Prior to 2004, he was a Lecturer in the School of Computer Science and Information Systems at Birkbeck, University of London and between 1996 and 1998 was a Research Fellow at the University of Southampton. In 2002, he was a BT Short-term Research Fellow. His research interests are in software engineering, more specifically metrics and empirical studies.

---

<sup>16</sup> <http://www.cs.auckland.ac.nz/tukutuku/>.



Web Engineering

Mendes, E.; Mosley, N. (Eds.)

2006, XX, 440 p. 143 illus., Hardcover

ISBN: 978-3-540-28196-2