JOURNAL OF INFORMATION SCIENCE AND ENGINEERING 29, 925-945 (2013)

DOI:10.1688/JISE.2013.29.5.8

Software Effort Estimation with Multiple Linear Regression: Review and Practical Application

OLGA FEDOTOVA[1], LEONOR TEIXEIRA[1,2,3] AND HELENA ALVELOS[1,2]
[1]*Department of Economics, Management and Industrial Engineering*
[2]*Governance, Competitiveness and Public Politics*
University of Aveiro
3810-193 Aveiro, Portugal
[3]*Institute of Electronics and Telematics Engineering of Aveiro*
Campus Universitário de Santiago
3810-193 Aveiro, Portugal

Software development effort estimation is the basis for the effective project planning and scheduling as well as for the project's budget definition. This article describes the most common methods used in the software effort estimation (SEE) and presents the study performed in a software development organization (SDO) that is implementing the software development process improvement framework Capability Maturity Model Integrated (CMMI). Currently SDO estimates the software effort based on the opinion of one area expert. The disadvantages of this method and the willingness to incorporate the best practices of CMMI encouraged the SDO to replace the existing effort estimation method by a formal one. The stepwise Multiple Linear Regression (MLR) technique was selected and used for the software development and software testing processes. The results achieved with MLR were compared with the estimates provided by the area expert. The model obtained for the testing team performed better results than the expert judgments, while for the development team no satisfactory model was found and a proposal for collecting data from new variables is presented.

Keywords: software development, effort estimation, multiple linear regression, practical case, CMMI

1. INTRODUCTION

Software effort estimation (SEE) is the prediction about the amount of effort required to make a software system and its duration [1]. SEE first appeared in 1950s, and since then continued to attract attention of software community specialists having the objective of developing useful models that constructively explain the development life-cycle and accurately predict the cost of developing a software product [2, 3]. Since then, there were developed a lot of models for the effort and cost estimation. The diversity of these models reported in the literature can be considered an indicator of the problem complexity, since there is no unique model that completely satisfies the need for objective, fast and accurate predictions in all circumstances.

Galorath and Evan [4] summarize the steps that are generally followed to obtain the project's effort estimation as: (i) establishment of estimation scope; (ii) establishment of technical baseline and assumptions; (iii) collection of data; (iv) software sizing; (v) preparing of baseline estimates; (vi) quantification of risks and their analysis; (vii) valida-

Received June 28, 2011; revised January 20, 2012; accepted March 29, 2012.
Communicated by Chih-Ping Chu.

tion and review of estimate; (viii) creation of project plan; (ix) documentation of estimate and lessons learned, and (x) tracking of project throughout development.

The ability to deliver the software on time, within the budget and with the expected functionalities and quality is a challenge for all software development organizations. Inaccurate estimations in software development industry is one of the most serious problems that cause the software projects failure. Both under and over estimations have negative impact on projects' results. While underestimation causes schedule delays and cost overruns that subsequently reduce the quality of end products, overestimations may lead to the loss of potential customers and partners, as well as to the inefficient distribution of the resources.

The quality of the estimates is one of the factors that determine the success of the project and helps to avoid the risks related to costs and schedule overruns. SEE is usually required in the beginning of the development phase, making the task of effort estimation more complex. According to [5], the error of estimation decreases as the project progresses because each subsequent project milestone brings new information that complements the existing one. In this way, it is possible to reduce the variability of effort estimation and make more accurate predictions.

This article presents a study carried out on a specific software development organization (SDO) where the effort required to prototype, develop, test and document software products was estimated by the respective area expert. The disadvantages associated to this method (like lack of experts and objective criteria for the estimation performance, difficulty to reproduce and use the knowledge and experience of an expert and questionable reliability of estimates [6]) and willingness to incorporate the best practices of the software development process improvement framework – Capability Maturity Model Integrated (CMMI) – encouraged the SDO to replace the existing effort estimation method by a formal one.

The aim of this paper is to present the main software effort estimation methods, their advantages and disadvantages and to apply a formal method based on the Multiple Linear Regression technique to the historical data of a medium-sized multinational software company. Besides, the study performs predictive accuracy's comparison between the method used in the company before the study (expert judgment) and the MLR results.

The authors believe that the contribution of the present article is valuable for both academic and practical purposes as it provides a literature based comparison and evaluation of different effort estimation techniques, associated advantages and disadvantages and main motivation and obstacles of their application, after which a practical case is described. The study may be of particular relevance for software development organizations that aim to improve the quality of their software effort estimates. Besides, the adoption of formal effort estimation methods is a requirement that has to be fulfilled by companies that intend to be CMMI appraised.

2. THEORETICAL BACKGROUND

This section provides literature overview of the existing classifications of software effort estimation techniques, characterizes the most popular classification, performs the motivations and reasons for failure of the effort estimations, and finally describes the model used to perform effort estimations in the studied SDO.

2.1 Classification of Software Effort Estimation Techniques

The literature reports a great variety of classifications of the SEE methods. Li, Ruhe *et al*. [7] and Shepperd, Schofield *et al*. [8] provide common classification of effort estimation techniques, categorizing them into expert judgment, analogy based or machine learning and algorithmic methods:

- Expert judgment effort estimation techniques are based on the person's experience and intuition [7];
- Analogy based or machine learning techniques predict the estimate from the analysis of projects with similar characteristics [7, 9];
- Algorithmic techniques are based on mathematical models and produce effort estimations as function of a number of variables [7, 10].

Singh, Bhatia *et al*. [1] give a more detailed classification of effort estimation techniques than the previous one and divide them in empirical techniques, model/theory techniques, expertise techniques, regression techniques, composite techniques and machine learning techniques.

- Empirical techniques correspond to the analogy-based techniques which estimations are based on the practice and previous experience.
- Model/Theory based techniques are the algorithm based techniques that include Function Point Analysis, SLIM, Checkpoints and COCOMO model.
- Expertise techniques are equivalent to the expert judgment when a person carries out estimation based on non-explicit and non-recoverable reasoning [1];
- Regression based models are used to infer how the Y-variables are related to X-variable(s), requiring data from previous projects;
- Composite techniques combine both approaches – expert judgment and project data – in a consistent way in order to obtain the effort estimation [2, 11].

Attarzadeh and Ow [12] and Leung and Fan [10] give more generalized classification of effort estimation techniques dividing them into algorithmic and non-algorithmic ones. Algorithmic techniques are based on mathematical models that are categorized as analytical and empirical ones [10]. Empirical models establish the formula for the current project using data available from previous projects, while the analytical models' formula is based on a set of global assumptions, such as the rate at which the developer solves problems and the number of available problems [10, 13].

The classification of effort estimation techniques presented by Boehm, Abts *et al*. [2] is close to the Singh, Bhatia *et al*. [1] classification approach, adding the dynamics based techniques. Dynamics based models emphasize the dynamic character of the software project effort data and consist on the application of a continuous simulation modeling methodology that detects the changes of the effort data over the duration of the project [2].

Laird and Brennan [14] enrich the Li, Ruhe *et al*. [7] and Shepperd, Schofield *et al*. [8] classification by adding methods using benchmark data, proxy points and custom models. Models based on the use of a benchmark data allow organizations that do not

have their own historical database to elaborate effort estimation based on existing data offered by another organization. Proxy point method decomposes the development task in components (proxies) and estimates size of each element, based on the historical data [15, 16]. Custom models opposite to all referred techniques do not impose any standard model for effort estimation, allowing modifications of formal models in order to adapt to the specific reality and needs of organization.

Table 1. Categories and respective equivalent of the most popular techniques.

Categories		Equivalent Technique from another Classification (Author)
Non – Algorithmic Techniques	Expert Judgement	– Expertise Technique (Singh, Bhatia *et al.* [1]) – Expert Opinion (Laird and Brennan [14]) – Expertise Based Technique (Boehm, Abts *et al.* [2]) – Expert Judgment (Li, Ruhe *et al.* [7]; Shepperd, Schofield *et al.* [8])
	Analogy base or machine learning	– Empirical and Machine Learning techniques (Singh, Bhatia *et al.* [1]) – Analogy base or machine learning (Li, Ruhe *et al.* [7]) – Analogy (Laird and Brennan [14]) – Learning oriented techniques (Boehm, Abts *et al.* [2])
	Composite techniques	– Composite techniques (Boehm, Abts *et al.* [2]; Singh, Bhatia *et al.* [1])
Algorithmic Techniques	Algo-rithmic Model	– Algorithmic effort estimation (Li, Ruhe *et al.* [7]; Shepperd, Schofield *et al.* [8]) – Algorithmic Model (Attarzadeh and Ow [12]; Leung and Fan [10] Laird and Brennan [14]) – Dynamics Based Techniques (Boehm, Abts *et al.* [2]) – Regression Techniques (Singh, Bhatia *et al.* [1]; Boehm, Abts *et al.* [2]) – Model/Theory Technique (Singh, Bhatia *et al.* [1]) – Model Based Technique (Boehm, Abts *et al.* [2])

All mentioned classifications have a common set of techniques, which name may vary from classification to classification but the meaning maintains the same, besides some categories are particularizations of the more generic classifications. From our point of view, the presented techniques fall into one of the following categories: (i) expert judgment; (ii) algorithmic; (iii) analogy based or machine learning; and (iv) composite techniques. These categories can be characterized as algorithmic or non-algorithmic and in this way can be structured in a hierarchical way. Table 1 presents the categories and gives the respective equivalents from the earlier presented classifications. Algorithmic models group all the techniques that have mathematical basis, such as dynamics based, regression based and model/theory based techniques, while the non-algorithmic models are based on expert judgments and analogy/machine learning techniques. Composite techniques consist of a combination of both algorithmic and non-algorithmic models aggregating the advantages of both approaches.

2.2 Advantages and Disadvantages of the Most Common Software Effort Estimation Approach

The most popular classification described in section 2.1 is the one provided by Li, Ruhe *et al*. [7] and Shepperd, Schofield *et al*. [8], grouping the techniques in three main categories: (i) expert judgment; (ii) analogy based or machine-learning approach; and (iii) algorithmic effort estimation. The main advantages and disadvantages of each of these categories are presented in Table 2. This information can help in choosing which one shall be used in each specific situation.

Table 2. Advantages and disadvantages of the most popular classification of software effort estimation.

	Advantages	Disadvantages
Expert judgment	– Provides fast estimation [17]; – Is useful when organization does not have any historical data in database [1]; – Provides estimates which are adjusted and calibrated to the past of organization by means of expert experience; – Does not require any historical data; is good for unique or new projects [4].	– Provides estimations that are relied on the experts experience and intuition that sometimes are questionable; factors that influence the estimation are hard to be documented [1, 6, 17]; – May not provide consistent estimation [4].
Analogy based or machine-learning approach	– Is low cost, simple and relatively accurate [18]; – Can employ a wide range of metrics [19]; – Is not sensitive to the outliers; deals with poorly understood domains; can be made in the early phase of the project [7].	– Is unable to handle missing and non-quantitative data; quality of estimates relies on quality of historical data [7]; – Requires database of appropriate projects [19]; – Does not include adjustments related to extreme analogues and inaccurate estimations [9]; – Needs analogies that match the new project characteristics [18].
Algorithmic effort estimation	– Is objective, fast and easy to use [4]; – Provides relatively accurate results in the case of existence of historical data [8]; – Provides more objective results and can be iterated in different lifecycles [17].	– Needs to be adjusted or calibrated to the local circumstances [8]; – Uses size variables that are difficult to obtain in the early stages of the project; – Has difficulty in modelling inherent complex relationships between contributing factors; – Is unable to support categorical data [12]; – The data analysis can be complex [20]; – Is sensitive to outliers [21].

Expert judgment is usually used by organizations that do not have any database with historical data [1], providing rather fast estimations adjusted to the past of the organization. Galorath and Evan [4] recommend this method of effort estimation for new or unique projects whose characteristics do not fall into the pattern of the past projects. Among the main disadvantages of this method, Singh, Bhatia *et al*. [1] and Bajwa [17] mention difficulty of extraction of factors that influence the estimation and total dependency of results accuracy on the expert experience and intuition that sometimes are questionable.

Analogy based or machine learning approach is distinguished by its low cost, simplicity and relative accuracy when there exists a reliable database [18]. This method can be applied on the early phases of the project's lifecycle [7], employing a great variety of metrics [19]. Besides, analogy based or machine learning approaches are not sensitive to the outliers' presence and can deal with poorly understood domains [7]. The main weakness of this model lies in the need of a database with appropriate projects similar to the new one to perform the effort prediction of high quality [18, 19]. Analogy based or machine learning techniques are unable to handle missing and non-quantitative data [7] and do not make adjustments related to extreme analogues and inaccurate estimations [9].

Galorath and Evan [4] characterize algorithmic effort estimation as objective, fast and easy to use. Estimations based on this method are relatively accurate in the case of existence of historical data [8]. Algorithmic effort estimation, opposite to the analogy or machine learning technique, is sensitive to the outliers which may influence the quality of final results [20]. As algorithmic effort estimation is based on the software size measure variables (such as lines of code – LOC, function points, number of functions, modules or program features required) that are normally available only in the end of the project, this type of estimation is difficult to be applied in the project's early stages [12]. Algorithmic effort estimation methods, opposite to the expert judgment, need to be calibrated or adjusted to the local circumstances [8] and are unable to support categorical data [12].

2.3 Main Motivation and Obstacles of Formal Effort Estimation Models Application

In spite of the existence of a great variety of models for SEE there is no unique method that presents more accurate and precise results in every situation and for all projects [1, 20]. There is no consensus in literature about the effectiveness of one or another SEE method. For example, Li, Ruhe *et al*. [7] state that in 60% of published studies, analogy-based effort estimation shows better results than the other two methods (expert judgment and analogy-based model).

Galorath and Evan [4] resume the main reasons for the software estimation failures to: (i) the lack of or misuse of historical data; (ii) overoptimistic leadership or management; (iii) failure to use the estimate or (iv) failure to keep the estimate current.

According to Jorgensen [21] low popularity of software development effort models may be explained by the discomfort performed by the software development organizations during the use of models that they do not fully understand.

Some of the reasons for a software development organization moving from the method based on expert judgments to a formal one are concerned with: (i) the better performance shown by models in the case of less predictive environments [22]; (ii) inde-

pendence from the experts presence and experience; and finally; (iii) satisfaction of the requirements imposed by the frameworks for software development process improvement (such as CMMI and SPICE BPG) for adoption of rationale method of effort and cost estimation in order to guarantee the evolution to higher capability/maturity levels.

2.4 Assessment of Models' Accuracy

The major challenge of an effort estimation model consists in its capacity to produce accurate predictions. Among the main causes of inaccuracy of estimates, Jorgensen and Molokken-Ostvold [23] refer unexpected events and overbooked tasks, change requests from the clients, problems with resource allocation, poor requirements specification and too little time spent on effort estimation work. On the other hand, such factors as enlargement of the buffer in order to deal with unexpected events and requirements specification changes, experience from previous projects, high degree of flexibility, knowledge in how to implement requirements specification, good cost control and much time spent on effort estimations positively contribute to the accuracy of estimates.

The most common accuracy predictive statistics are the mean magnitude relative error (MMRE) and the percentage relative error deviation within x (PRED(x)) [24]. Both these measures are based on the value of magnitude relative error (MRE).

MRE is a normalized measure of the discrepancy between the actual data values (in this case effort values) and the estimated values [25] in Eq. (1).

$$MRE = \frac{|\, Effort_{actual} - Effort_{estimated} \,|}{Effort_{actual}} \tag{1}$$

MMRE is the mean value of MRE of all observations (n) in the sample: Eq. (2).

$$MMRE = \frac{1}{n} \times \sum_{i=1}^{n} (MRE_i) \tag{2}$$

PRED(x) is defined as the average fraction of the MRE's values that are off by no more than x [24], and is calculated in the following way in Eq. (3).

$$PRED(x) = \frac{1}{n} \times \sum_{i=1}^{n} \begin{cases} 1, & \text{if } MRE \le x \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

In this paper, x value is considered to be 0,25 as recommended by most authors. PRED(0,25) is used to give the percentage of estimates that were found to be within the tolerance of 25% of their actual value. Some studies also use PRED(0,20) and PRED (0,30) with little differences in results. Conte, Dunsmore *et al*. [26] consider the values of MMRE $\le$ 0,25 and PRED(0,25) $\ge$ 0,75 as desirable for accurate effort model.

2.5 Multiple Linear Regression

Multiple Linear Regression belongs to the algorithmic group of techniques. According to Leung and Fan [10] it is an empirical model that requires data from the past pro-

jects in order to evaluate the current projects. Boehm, *et al*. [2] and Singh, *et al*. [1] distinguish MLR as one of the categories of effort estimation techniques that is used to find out how the dependent variable (Y) is related to the independent variables (X_i) [27]. MLR model is defined as in Eq. (4):

$$Y = \beta_0 + \beta_1 X_1 + \beta_1 X_1 + \ldots + \beta_n X_n + \varepsilon \tag{4}$$

where X_1, X_2, ..., X_n are regressors; β_0 is the intercept parameter; β_1, β_2, ..., β_n are the regression coefficients; and ε is the error component.

To assess the adequacy of the model, the coefficient of determination R^2 is used [28]. It measures the proportion of the total variability of the independent variable about the mean that results from the fitting of the multiple regression model [29].

As was already mentioned, each technique has its own specific characteristics that make it suitable to solve a particular problem. According to [30], MLR technique is usually employed when: (i) the number of cases is significantly higher than the number of parameters to be estimated; (ii) the data has a stable behaviour; (iii) there is a small number of missing data; (iv) a small number of independent variables are sufficient (after transformations if necessary) to linearly predict output variables (also transformed if necessary), so as to enable an interpretable representation. Regression may be used when there is a need for a simple model and analysis tool of effort estimation to support the preliminary attempts [31]. Application of MLR method requires verification of the associated assumptions. The major assumptions to be considered are [32, 33]:

- Linearity – the relationship between each X_i and Y is linear, thus the model adequately describes the behaviour of data;
- The error component is an independent and normally distributed variable with constant variance and mean value zero.

Problems in data set or use of incorrect model may result in violation of these assumptions [34].

There are several possible procedures for the selection of the independent variables to be included in the MLR model. One of them consists in the inclusion of all the independent variables that are considered relevant, while others use stepwise procedures – forward regression, backward regression and stepwise regression [35, 36]. This study uses the stepwise model, which is more popular than the other ones. This method includes the independent variables one at a time (X_i), beginning with the one that has highest correlation with Y. In each step, R^2 value is evaluated and it is verified if each of the previously included variables contribute to the R^2 increase, if not it is excluded.

The next section describes the approach followed by a medium-sized multinational software development company using a stepwise MLR method for the estimation of the effort for testing and development teams.

3. FORMAL EFFORT ESTIMATION APPROACH: APPLICATION IN A SOFTWARE DEVELOPMENT ORGANIZATION

This part of the study presents the approach followed by a specific multinational SDO

in order to implement a formal software effort estimation method. This need emerged as a result of the adoption, by organization, of the framework for software development process improvement – CMMI. In order to stimulate the software development process maturity in the *Project Planning* process area CMMI requires the establishment of estimates for work products and tasks based on estimation rationale [37].

Before the implementation of CMMI practices organization's estimates were based on the judgement of only one expert. Since it is not considered to be a valid method, there were identified two possible solutions to meet the requirements of CMMI. One of them consisted in adoption of the formal Delphi method [10] with participation of at least 3 area experts for the effort prediction. Another solution was the implementation of the algorithmic model for the effort estimation based on the historical data of organization. Due to the more favourable costs/benefits relationship associated to the second proposal, the organization decided to proceed with it.

Project planning and further monitoring and control within the organization are made by means of change set's management. Change set (CS) is the element of work breakdown structure that is considered to be the work unit grouping a set of requirements. For this reason, project's effort and cost estimations are performed in the CS level, providing possibility for more detailed cost and effort control.

The existence of two-years old data in the SDO' historical database where the critical variables of finished projects were saved and the need for a simple, objective, fast and accurate model of effort estimation to support the preliminary attempts [31] led to explore the possibility of using a software effort estimation method based on MLR.

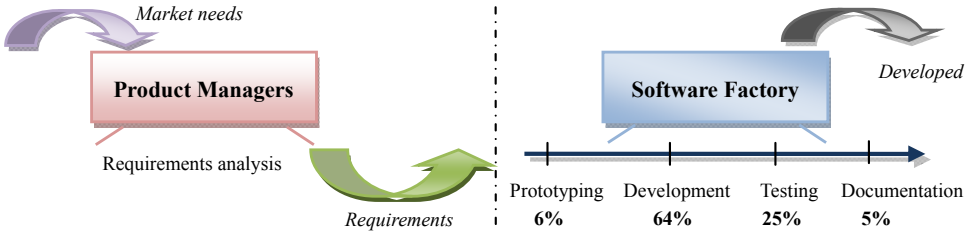

Fig. 1. SDO's project workflow.

3.1 Problem Contextualization

SDO has a matrix organizational structure that is typical of the medium and large software development companies [38]. Thus, SDO's work is organized by projects, involving specialists from different departments. Fig. 1 presents the workflow for the typical project within the SDO. The requirements specification is developed by the elements from the Product Management team and delivered to the Software Factory that consists of 4 main independent units responsible for the CS prototyping, development, testing and documentation. Each of these teams provides its estimations for the project's CSs, resulting in the overall effort estimation.

The sample that was used as the basis of the effort estimation method considered all closed CSs, which were carried out along the prototyping, development, testing and

documentation phases within the Software Factory. The requirements analysis phase was excluded from the study, since Product Management team is not included in the Software Factory, where the productive process begins after the requirements delivery. Thus, there were considered 106 CSs from 13 projects of different sizes. From the tasks associated with the referred CS, 6% of the effort was dedicated to the CS prototyping, 64% was spent on CS development (*i.e.* programming and realizing the unitary tests), while 25% and 5% were spent on CS testing and documenting, respectively. Due to the low weight of the prototyping and documentation efforts, when compared to the total project effort, it was decided, for the initial phase, not to include these tasks in the scope of the formal effort estimation method and continue with the effort prediction based on the opinion of the area expert. Nevertheless, in a later phase, estimations for the prototyping and documentation tasks should also be formalized.

The variables used to characterize each CS included in the sample are presented in the Table 3.

Table 3. Characterization of the variables included in the study.

Variable Acronym	Variable Description
Dev_Eff	Effective hours spent on programming of CS and unitary tests.
Dev_Frc	Number of hours forecasted by the expert to program the CS and effectuate unitary tests.
QA_Eff	Number of effective hours spent on testing the CS.
QA_Frc	Number of hours forecasted by the expert to test the CS.
Nr_Req	Number of requirements of the CS.
Nr_CRs	Number of change requests – development tasks – of the CS.
Nr_Modules	Number of modules – logic units of code – in which the CS had impact.
Prot	Variable that indicates if CS will (Prot=1) or will not (Prot=0) be prototyped.
Code Complexity	Ordinal variable that presents the complexity of CS programming, which vary from low to high (1-3).

Table 4. Descriptive statistics of CSs.

All CSs	Development Team		Testing Team	
	Dev_Eff	*Dev_Frc*	*QA_Eff*	*QA_Frc*
Mean	67,64	57,45	21,68	20,22
Stand. Deviation	72,49	58,46	21,55	20,02
Minimum	3,00	3,00	1,00	2,00
Maximum	363,50	280,00	120,00	105,00
Median	38,25	35,00	14,00	14,00

Table 4 presents some descriptive statistics for the variables *Dev_Eff* and *Dev_Frc* (development team) and *QA_Eff* and *QA_Frc* (testing team). As can be observed in Table 4, on average, estimated values are less than the effective values for development and testing teams, showing that there is a tendency to the effort underestimation on both processes (difference of 10,19 for development and 1,47 for testing). The high values

obtained for the standard deviation for the four variables are due to the diversity of the nature of the projects, being some of them simple (minimum of 3 hours of development) and others more complex (maximum of 363,5 hours of development). It can also be noted that the distributions of the variables' values are right-asymmetric, as the median values are, in all the cases, lower than the mean values. This indicates that the majority of the CSs are small ones, consuming much less development and testing time than the big CSs (50% of the CSs took less than 40 hours to be developed, while the more complex CSs took 363,5 hours and the average length of their development was 67,64h).

The described scenario indicates that the effort estimation task is complex, and that it may cause delays on project delivery and cost overruns.

3.2 Effort Estimation Model for the Development Team

The effort estimation of the software development team was performed considering the already mentioned sample of 106 CSs. Table 5 contains some information about 10 MLR models obtained using the stepwise method. They differ one from another by the sample size, by the set of independent variables included in the model and by the variable transformation used in the dependent variable.

Table 5. Summary of the regression models for the development team.

Model Number	Sample Size	R^2	Dependent variable (Y)		Independent variables (X_n)				
			Dev_Eff	ln(*Dev_Eff*)	*Prot*	*Nr_Req*	*Code Complexity*	*Nr_CRs*	*Nr_Modules*
1	106	0,493	*x*		*x*		*x*	*x*	
2	106	0,382		*x*		*x*	*x*	*x*	*x*
3	89	0,508	*x*		*x*			*x*	*x*
4	89	0,547		*x*	*x*		*x*	*x*	
5	84	0,537	*x*		*x*	*x*		*x*	
6	84	0,524	*x*		*x*	*x*	*x*	*x*	
7	84	0,524		*x*	*x*	*x*	*x*		
8	84	0,358		*x*		*x*	*x*	*x*	
9	76	0,429		*x*	*x*	*x*	*x*		
10	69	0,092	*x*					*x*	

The second column of the Table 5 refers to the sample size considered in each model, which variation is justified by the successive elimination of outliers. The third column represents the R^2 value, which meaning was already referred. The fourth and fifth columns contain the information about the dependent variable considered in the model – either the original one (*Dev_Eff*) or the transformed one (ln(*Dev_Eff*)). Transformation was performed in order to try to obtain better results. The last five columns refer to the inclusion, or not, of the independent variables in the models.

As can be observed, the higher value of R^2 corresponds to the model 4, elaborated on the basis of *Prot*, *Code Complexity* and *Nr_CRs* variables and explains 54,7% of the effective software development effort variation.

Table 6. Summary of accuracy predictive statistics for the development team.

Model Number	MMRE Expert	MMRE Regression	PRED(0,25) Expert	PRED(0,25) Regression
1	0,36	1,44	44%	22%
2	0,36	1,06	44%	18%
3	0,59	1,04	45%	26%
4	0,59	0,69	45%	27%
5	0,52	1,12	44%	40%
6	0,52	1,13	44%	42%
7	0,52	0,72	44%	26%
8	0,52	0,93	44%	26%
9	0,48	0,80	43%	34%
10	0,35	1,53	46%	26%

Table 6 presents a summary of accuracy predictive statistics used to assess the results of the regression models and of the estimates provided by the expert for the development team.

MRE values were obtained for regression and expert judgment models using Eq. (1). MMRE Regression values were calculated using Eq. (2). MMRE Expert values were calculated on the basis of the effective time spent on development (*Dev_Eff*) and time forecasted by the expert to develop the CS (*Dev_Frc*), using the same equations. PRED (0,25) Expert and PRED(0,25) Regression were calculated on the basis of MRE Expert and MRE Regression results, respectively, using Eq. (3).

It can be noticed that all the regression based models present worse results than the ones obtained with the expert opinion, either in terms of MMRE or in terms of PRED (0,25), as:

MMRE Expert < MMRE Regression and PRED(0,25) Expert > PRED(0,25) Regression.

Besides, there is a great variation of MMRE and PRED(0,25) values in the MLR approach, originated by the sample size decrease or independent variables set changes. This phenomenon revealed data instability that may be originated by the absence of a common reference scale for variables characterization.

Due to the results obtained with referred analysis it was decided not to adopt any of the presented MLR models and proceed with identification and classification of the variables that will further compose the organization's variables database and allow the creation of a formal model of effort estimation for the development team. The study made in order to propose the new set of variables will be presented in the next section. Meanwhile, effort estimation procedure will be based on the formal Delphi method to guarantee the implementation of CMMI's Specific Practice 1.4 – *Determine Estimates of Effort and Cost* based on formal model [37]. This will contribute to the achievement of Specific Goal 1 – *Establish Estimates* of Project Planning Process Area.

3.2.1 Proposal for the variables selection for the haracterization of the CS

In order to define new variables and enrich the existing set of variables that will best

describe the CS, there were carried out interviews with five developers, in addition to the literature review.

One of the principal factors that determine the accuracy of effort prediction is the size measurement. Laird and Brennan [14] argument the importance of measuring accurately size, as:

- Contracts signed with customers and employees depend upon the size;
- Size shows the volume of the software;
- Effort is calculated from the size.

Koch and Mitlöhner [19], Finnie, Wittig *et al*. [20], Lucia, Pompella *et al*. [40] and Hill, Thomas *et al*. [18], among other authors, distinguish size as the main factor that influences the algorithmic effort estimation approaches. Size of CS was referred by all interviewed developers as the important variable for effort estimation. One of the possible reasons for the failure of the regression-based models for the estimation in the case of the development team may consist in the absence of the size-measure variable. As may be observed from the set of variables used to classify the CS, none of them corresponds to the CS size. The CS Size adapted to the reality of the SDO can be calculated in the following way in Eq. (5):

$$CS\ Size = \sum_{i=1}^{n} (Requirement_i\ \ Complexity \times Number\ of\ \ Use\ Case per\ Requirement_i), \quad (5)$$

i – number of requirements that compose CS.

Requirement complexity is an ordinal variable that varies from 1 to 3 and measures the complexity of requirement in terms of functionalities to be implemented. *Number of Use Cases per requirement* corresponds to the number of Use Cases that have to be specified in order to perform the requirement in the Use Case diagram.

In addition to the size variable, Hill, Thomas *et al*. [18] refer the need to measure system complexity, personnel capabilities and experience, hardware constraints and the availability of software development tool.

According to Jones [3] there are four key factors that have impact on software estimating methodologies: (i) the experience of personnel, (ii) the technologies used (programming languages, support tools, *etc*.), (iii) the development process, and (iv) the programming environment where the developer works.

All factors named by Hill, Thomas *et al*. [18] and Jones [3] were, in one or another way mentioned by the organization's development specialists as relevant during the software effort prediction. Thus, there were considered other five ordinal variables (which scale varies from 1 to 3 and in a case of *CS Implementation Impact* variable – from 1 to 5) to complement the CS size variable and to ensure the best characterization of the effort estimation unit – Change Set:

- *Business acquaintance* – defines the degree to which team elements are familiar with the business rules, laws, *etc*.;
- *CS implementation Impact* – expresses the volume of changes to make to the impacted processes;

- *Code reuse* – indicates if there is any already existing code and in which extent it will be reused;
- *Technical experience* – measures the degree of experience of the development team in using the technology and programming language;
- *Code complexity of CS* – expresses the degree of complexity of code elaboration for the CS.

Variables classification may be seen in Appendix A.

The formal Delphi method that will be used in SDO incorporates the characterization of the earlier described variables in order to help in estimating the effort correspondent to each CS and, at the same time, ensures the collection of data required for the database creation.

3.3 Effort Estimation Model for the Testing Team

The effort estimation model for the testing team was performed taking into account 95 CSs from the original sample of 106, as there were some missing values in the *QA_Eff* variable. The stepwise MLR technique was applied to the list of independent variables characterized in Table 3, and the model that best fitted the data is summarized in Table 7 (model nr. 1). Variables *Dev_Eff* and *QA_Eff* were transformed to the logarithmic scale in order to have a residual distribution more approximated to the normal one.

Table 7. Summary of the regression models for the testing team.

Model Nr.	Sample Size	R^2	Dependent Variable (Y)	Independent Variable (X)
1.	95	62,4%	ln(*QA_Eff*)	ln(*DEV_Eff*)
2.	92	71,5%	ln(*QA_Eff*)	ln(*DEV_Eff*)

To verify the existence of outliers there was elaborated the sequence chart of the studentized deleted residuals (Fig. 2), which values should vary between '–2' and '2' [39]. There were detected three outliers, which removal originated the sample of 92 CSs for the second model (Table 7).

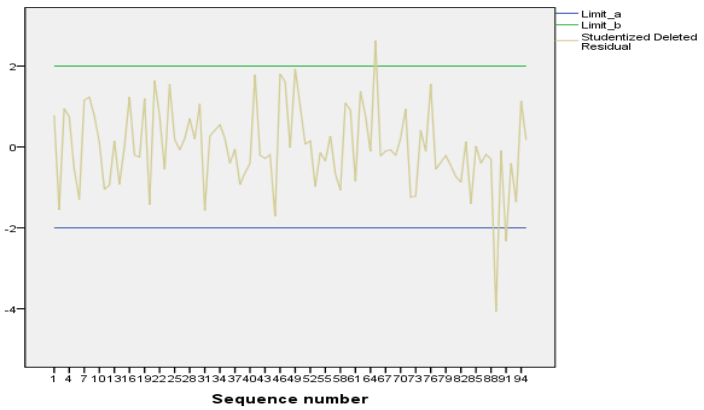

Fig. 2. Outliers analysis for the model 1.

In the second model the value of coefficient of multiple determination R^2 shows that the natural logarithm of dependent variable *Dev_Eff* explains 71,5% of the variation of the natural logarithm of the independent variable *QA_Eff*, while in the first model the same variable explains only 62,4%. This improvement can be explained by the fact of outliers' elimination that resulted in the regression line's best fitting to the existing data. Fig. 3 shows the values of the variable ln(*QA_Eff*) as a function of the values of the variable ln(*Dev_Eff*) and the correspondent regression line for the second model presented in Table 7.

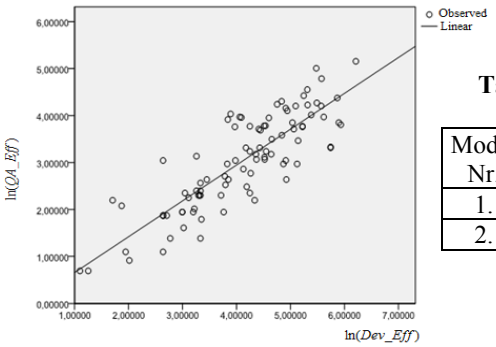

Fig. 3. Linear regression line for model 2.

Table 8. Summary of accuracy predictive statistics for testing team.

Model Nr.	MMRE Regression	MMRE Expert	PRED(0,25) Regression	PRED(0,25) Expert
1.	0,189	0,174	75%	73%
2.	0,158	0,161	79%	74%

Table 8 summarizes accuracy predictive statistics that were used to assess the results of the regression models and of the estimates provided by the expert for the testing team.

MRE, MMRE and PRED(0,25) values for the regression models and expert judgements were obtained using Eqs. (1)-(3), respectively.

Both regression models perform rather accurate results with values of PRED(0,25) ≥ 0,75 and MMRE ≤ 0,25. Nevertheless the second model presents better results than the first one. It can also be noted that in the case of Model 2 the results obtained are more accurate than the ones estimated by the expert, as MMRE Regression < MMRE Expert and PRED(0,25) Regression > PRED(0,25) Expert. Thus, the second model was analyzed in order to verify MLR assumptions.

3.3.1 Verification of regression model assumptions for model 2

MLR is based on assumptions that errors are independent, normally distributed with constant variance and mean value zero. Verification of these assumptions is fundamental to validate the developed model.

Tables 9 and 10 and Figs. 4 and 5 show the test results and the plots for Model 2 based on the outputs from the statistical tool SPSS. Kolmogorov-Smirnov and Durbin-Watson tests do not violate the assumption of the residuals' normality and independence, respectively, with the p-value of 0,489 of Kolmorov-Smirnov test (Sig>0,05) and Dublin-Watson test value of 2,139 (approximately 2). Figs. 4 and 5 are used to verify the assumption of constant variance of the residuals. As may be observed, residuals maintain approximately constant amplitude relatively to the horizontal axis and do not perform any

Table 9. Residual normality test.

		Studentized Residual
N		92
Normal Parameters[a,b]	Mean	–0,0001073
	Std. Deviation	1,005643
Most Extreme Difference	Absolute	0,087
	Positive	0,087
	Negative	–0,48
Kolmogorov-Smirnov Z		0,835
Asymp. Sig. (2-tailed)		0,489

a: Test distribution is Normal.
b: Calculated from Data.

Table 10. Test of residuals' constant variance.

Model	Dublin-Watson
2	2,139

a: Predictors: ln(*Dev_Eff*).
b: Dependent Variable: ln(*QA_Eff*)

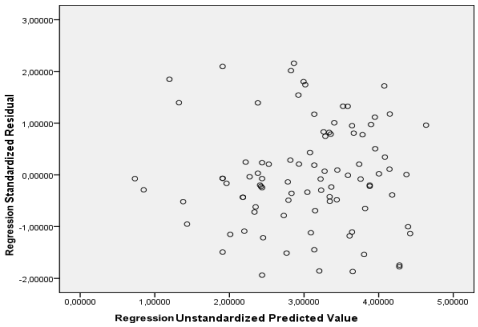

Fig. 4. Variance analysis with Y = ZRE[1] and X = UPRED[2].

[1] ZRE – Regression Standardized Residual
[2] UPRED – Regression Unstandardized Predicted Value

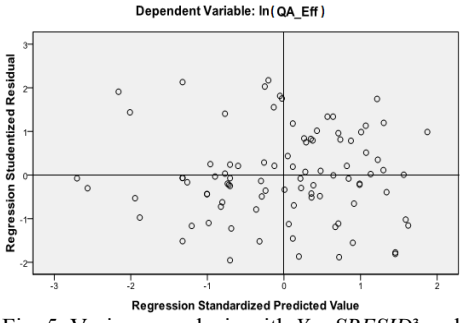

Fig. 5. Variance analysis with Y = *SRESID*[3] and X = ZPRED[4].

[3] SRESID – Regression Standardized Residual
[4] ZPRED – Regression Standardized Predicted Value

increasing or decreasing tendency. Since there is no defined pattern in the residuals location, it may be assumed that error variance remains constant [39].

As the effort estimation model for testing does not violate any of the MLR model's assumptions it may be further used by the organization for the testing tasks estimations. Nevertheless, this model requires validation with new projects in order to confirm the quality of produced results.

4. CONCLUSIONS

The study described in this paper was carried out in a medium-sized multinational software development organization that is implementing the CMMI maturity level 2. In order to answer the CMMI requirements for the Project Management Process Area, a formal effort estimation method is requested.

The project effort within the organization is divided along the prototyping, development, testing and documentation phases. Due to the relatively low weight, prototyping (6%) and documentation (5%) tasks were excluded from the scope of this study.

The practical study presented in the paper was aimed at obtaining formal models of

effort estimation for the development and testing teams, which were previously based on the expert judgement technique supported by the opinion of one expert.

In order to achieve a suitable model, there were analyzed different effort estimation techniques with the respective advantages and disadvantages. Among the variety of the studied methods and in accordance with the existing two year old organizational historical data that contained information about the amount of time spent on the development and testing of the basic elements of the work breakdown structure – Change set (CS); quantity of requirements; code complexity; and number of development tasks, it was selected the stepwise MLR technique. This method belongs to the algorithmic category of effort estimation methods and aims at establishing a linear relationship between a dependent variable and one or more independent variables. To assess the model's adequacy and accuracy, the coefficient of determination (R^2) and mean magnitude relative error (MMRE) with the percentage relative error deviation within x (PRED(x)) were used.

The stepwise MLR applied to the data of development team did not perform any viable model for the effort prediction as there were detected (i) instability of variables' behaviour during the inclusion of the new variables and the elimination of outliers and (ii) low levels of explanation of the dependent variable variation by the independent ones revealed by the low values of R^2.

Taking into consideration all mentioned factors, the organization's management decided not to adopt any of the deduced models for the development team effort estimation and to proceed with the relevant variables identification and classification to ensure the creation of the database required for the future formal method application, as well as to train the development team in variables classification to avoid the data instability. These variables were identified based on interviews carried out with the software developers and in accordance with some studies presented in the literature. As a result of this elicitation, the following set of variables to be considered for future effort estimation model emerged: (i) CS size, (ii) business acquaintance, (iii) technical experience of the development team, (iv) CS implementation impact, (v) code reuse and (vi) code complexity of CS.

While there is no sufficient data to use the regression methods for the development team, the formal Delphi method will be applied considering the referred variables, in order to satisfy the requirements of the CMMI in elaboration of project effort and cost estimations.

The stepwise MLR method applied to the testing team lead to inclusion of only one independent variable, resulting in a linear regression model. The estimates produced by the linear regression are better when compared to those of the single area expert judgements. While expert estimates presented the values of mean magnitude relative error (MMRE) and percentage relative error deviation PRED(0,25) of, respectively, 0,161 and 74%, the regression based estimates presented a MMRE value of 0,158 and a PRED(0,25) value of 79%.

Verification of the MLR assumptions did not reveal the violation of the model's hypotheses. Still the regression based model for testing team need to be validated with new projects' data in order to verify its suitability.

It is believed that the contribution of the present article is valuable for both academic and practical purposes. On the one hand, it provides literature based comparison and evaluation of different effort estimation techniques, associated advantages and dis-

advantages and main motivation and obstacles of their application. On the other hand, the paper presents a practical case where the results of the MLR and expert judgment techniques are compared in the context of the software development. This might be of special interest for the software organizations that aim to improve their effort estimates, particularly in the case of CMMI implementation, as it requires the adoption of formal effort estimation methods.

Based on the difficulties faced by the SDO in selecting the most suitable method of effort estimation among the great variety of existing ones, and as a proposal for future work, it is believed that a Decision Support System could be developed as a tool which would help on minimizing the time spent on the selection of an accurate effort estimation method and would help organizations on (i) defining the most adequate effort estimation method in different software development environments; (ii) implementing the selected technique, and finally (iii) testing of the resulting model accuracy.

REFERENCES

1. Y. Singh, P. K. Bhatia, A. Kaur, and O. Sangwan, "A review of studies on effort estimation techniques of software development," in *Proceedings of Conference Mathematical Techniques: Emerging Paradigms for Electronics and IT Industries*, 2008, pp. 188-196.
2. B. Boehm, C. Abts, and S. Chulani, "Software development cost estimation approaches – A survey," *Annals of Software Engineering*, Vol. 10, 2000, pp. 177-205.
3. C. Jones, *Estimating Software Costs: Bringing Realism to Estimating*, McGraw-Hill, NY, 2007.
4. D. D. Galorath and M. W. Evan, *Software Sizing*, *Estimation*, *and Risk Management: When Performance is Measured Performance Improves*, Taylor & Francis Group, NY, 2006.
5. S. McConnell, *Software Estimation: Demystifying the Black Art*, Microsoft Press, Washington, 2006.
6. F. Heemstra, "Software cost estimation," *Information and Software Technology*, Vol. 34, 1992, pp. 627-639.
7. J. Li, G. Ruhe, A. Al-Emran, and M. M. Richter, "A flexible method for software effort estimation by analogy," *Empirical Software Engineering*, Vol. 12, 2006, pp. 65-106.
8. M. Shepperd, C. Schofield, and B. Kitchenham, "Effort estimation using analogy," in *Proceedings of International Conference on Software Engineering*, 1996, pp. 170-178.
9. M. Jorgensen, U. Indahl, and D. Sjoberg, "Software effort estimation by analogy and regression toward the mean," *The Journal of Systems and Software*, Vol. 68, 2003, pp. 253-262.
10. H. Leung and Z. Fan, "Software cost estimation," in *Handbook of Software Engineering and Knowledge Engineering*, Hong Kong Polytechnic University, 2002.
11. L. C. Briand and I. Wieczorek, "Resource estimation in software engineering," in J. J. Marcinak, ed., *Encyclopedia of Software Engineering*, John Wiley & Sons, 2002, pp. 1160-1196.
12. I. Attarzadeh and S. H. Ow, "Software development effort estimation based on a new

fuzzy logic model," *International Journal of Computer Theory and Engineering*, Vol. 1, 2009, pp. 1793-8201.
13. J. Kaur, S. Singh, and K. S. Kahlon, "Comparative analysis of the software effort estimation model," in *Proceedings of World Academy of Science*, *Engineering and Technology*, 2008, pp. 485-487.
14. L. M. Laird and M. C. Brennan, eds., *Software Measurement and Estimation: A Practical Approach*, Jonh Wiley & Sons, Inc., NJ, 2006.
15. R. Schoedel, *PROxy Based Estimation* (*PROBE*) *for Structured Query Language* (*SQL*), Technical Note CMU/SEI-2006-TN-017, Software Engineering Institute, Carnegie Mellon University, 2006, http://www.sei.cmu.edu/library/abstracts/reports/06tn017.cfm
16. G. Coleman and R. Verbruggen, "A quality software process for rapid application development," *Software Quality Journal*, Vol. 7, 1998, pp. 107-122.
17. S. S. Bajwa, *Investigating the Nature of Relationship between Software Size and Development Effort*, Department of Interaction and System Design, Blekinge Institute of Technology, Ronneby, 2009.
18. J. Hill, L. C. Thomas, and D. E. Allen, "Experts' estimates of task durations in software development projects," *International Journal of Project Management*, Vol. 18, 2000, pp. 13-21.
19. S. Koch and J. J. Mitlöhner, "Software project effort estimation with voting rules," *Decision Support Systems*, Vol. 46, 2009, pp. 895-901.
20. G. R. Finnie, G. E. Wittig, and J.-M. Desharnais, "A comparison of software effort estimation techniques: Using function points with neural networks, case-based reasoning and regression models," *Systems Software*, Vol. 39, 1997, pp. 281-289.
21. M. Jorgensen, "A review of studies on expert estimation of software development effort," *The Journal of Systems and Software*, Vol. 70, 2004, pp. 37-60.
22. M. Jordensen, G. Kirkeboen, D. I. K. Sjoberg, B. Anda, and L. Bratthall, "Human judgment in effort estimation of software projects," in *Proceedings of International Conference on Software Engineering*, Vol. 45, 2000, pp. 45-51.
23. M. Jorgensen and K. Molokken-Ostvold, "Reasons for software effort estimation error: Impact of respondent role, information collection approach, and data analysis methods," *IEEE Transactions on Software Engineering*, Vol. 30, 2004, pp. 993-1007.
24. D. Port, V. Nguyen, and T. Menzies, *Studies of Confidence in Software Cost Estimation Research Based on the Criterions MMRE and PRED*, http://menzies.us/pdf/09predmmre.pdf, 2009.
25. S. G. MacDonell and A. R. Gray, "A comparison of modeling techniques for software development effort prediction," in *Proceedings of International Conference on Neural Information Processing and Intelligent Information Systems*, 1997, pp. 869-872.
26. S. D. Conte, H. E. Dunsmore, and V. Y. Shen, *Software Engineering Metrics and Models*, Benjamin-Cummings Publishing Co., Inc., Redwood, 1986.
27. D. R. Anderson, D. J. Sweeney, and T. A. Williams, *Statistics for Business and Economics*, Thomson South-Western, Ohio, 2009.
28. D. C. Montgomery, E. A. Peck, and G. G. Vining, *Introduction to Linear Regression Analysis*, 4th ed., John Wiley & Sons, Inc., Hoboken NJ, 2006.
29. R. J. Freund, W. J. Wilson, and P. Sa, *Regression Analysis: Statistical Modeling of a*

Response Variable, Elsevier, MA, 2006.
30. A. R. Gray and S. G. MacDonell, "A comparison of techniques for developing predictive models of software metrics," *Information and Software Technology*, Vol. 39, 1997, pp. 425-437.
31. M. Jorgensen, "Regression models of software development effort estimation accuracy and bias," *Empirical Software Engineering*, Vol. 9, 2004, pp. 297-341.
32. R. J. Freund and W. J. Wilson, *Regression Analysis: Statistical Modeling of a Response Variable*, Academic Press, San Diego, 1998.
33. R. L. Ott and M. Longnecker, *An Introduction to Statistical Methods and Data Analysis*, Cengage Learning Inc., Belmont, 2010.
34. J. Cohen, P. Cohen, S. G. West, and L. S. Alken, *Applied Multiple Regression/Correlation Analysis for the Behavioral Sciences*, Lawrence Erlbaum Associates Inc., NJ, 2003.
35. K. N. Krishnaswamy, A. I. Sivakumar, and M. Mathirajan, *Management Research Methodology: Integration of Principles*, *Methods and Techniques*, Dorling Kindersley Pvt. Ltd., India, 2006.
36. A. H. Kvanli, R. J. Pavur, and K. B. Keeling, *Concise Managerial Statistics*, Thomson Learning Inc., Ohio, 2006.
37. M. B. Chrissis, M. Konrad, and S. Shrum, *CMMI®: Guidelines for Process Integration and Product Improvement*, Addison-Wesley, Canada, 2009.
38. M. Hamilton, *Software Development: Building Reliable Systems*, Prentice Hall PRT, USA, 2006.
39. M. H. Pestana and J. N. Gageiro, *Análise de Dados Para Ciências Sociais: a Complementaridade do SPSS*, Sílabo, Lisboa, Portuguese, 2003.
40. A. D. Lucia, E. Pompella, and S. Stefanucci, "Assessing effort estimation models for corrective maintenance through empirical studies," *Information and Software Technology*, Vol. 47, 2005, pp. 3-15.

APPENDIX A

Variables Characterization

Business acquaintance – if team elements are familiar with the business rules, laws, etc.:

1 – More than 3 years of experience;
2 – From 6 months to 3 years of experience;
3 – Between 0 and 6 months of experience.

CS implementation impact – volume of changes to make, number of processes to change:

1 – Alteration concerns only one file;
2 – Alteration concerns different files of unique functionality;
3 – Alteration concerns different files and different functionalities;
4 – Alteration concerns different files, functionalities and modules;
5 – Alteration results in a great amount of changes that are not known on the moment of estimation. The impact is considered to be maximal.

Code reuse – if the already existing code will be reused:

1 – With code reuse (more than 80% of code will be reused and the existing code has a high quality);

2 – With some code reuse (approximately 40-70% of code will be reused);
3 – Without any code reuse. This decision may be taken in 2 situations: when the code is absent and when the existing code has a low quality (evaluation made according to the code reviews criteria) and it costs less to rewrite it than to make alterations.

Technical experience – experience of development team in use of technology and in programming language:

1 – High (no changes of technology/programming language);
2 – Moderate (some changes in technology/programming language);
3 – Low (change of technology/ programming language).

Code complexity of CS – complexity of code elaboration for the determinate CS:

1 – Code has a low complexity;
2 – Code has a moderate complexity;
3 – Code has a high complexity.

Olga Fedotova graduated in Science of Electronics and Telecommunications Engineering and received an MSc in Management and Industrial Engineering from Aveiro University in 2010 and currently she is pursuing the Ph.D. in Information Systems area. She has worked in Business Software Solution Enterprise as a Process Engineer during the incorporation of the CMMI's best practices. During her studies, she has published in several conferences.

Leonor Teixeira graduated in Management and Industrial Engineering in 1997, received an MSc degree in Information Management in 2002 and a Ph.D. in Software Engineering Applied in Health Information Systems in 2008. She is currently an Assistant Professor in the Department of Economics, Management and Industrial Engineering at the University of Aveiro. She has published in several conferences and journals, her current research interests include software engineering, information management and software development.

Helena Alvelos graduated in Electronics and Telecommunications Engineering in 1989, received an MSc degree in Management (MBA) in 1995 and a Ph.D. in Engineering Sciences in 2002. She is currently an Assistant Professor in the Department of Economics, Management and Industrial Engineering at the University of Aveiro. She has published in several conferences and journals, and her current research interests include data analysis, quality management and statistical quality control.