Exploiting Prior-phase Effort Data to Estimate the Effort for the Subsequent Phases: a Further Assessment

Filomena Ferrucci
Department of Management and Information Technology
University of Salerno
Italy
fferrucci@unisa.it

Carmine Gravino
Department of Management and Information Technology
University of Salerno
Italy
gravino@unisa.it

Federica Sarro
CREST Department of Computer Science
University College London
United Kingdom
f.sarro@ucl.ac.uk

ABSTRACT

Context. Development effort estimation is a managerial activity that takes place throughout the life-cycle of the software so that it may benefit from information that becomes available as the project progresses. Researchers investigated the use of prior-phase effort data to estimate the effort in subsequent phases, as well as early phase effort data to estimate the total development effort. *Objective.* We assessed the usefulness of the effort spent for each phase in order to predict the effort required during the subsequent phase and until the end of the development process. We compared the use of effort data against the use of Function Points (i.e., a functional size measure widely used for effort estimation) and verified whether it is useful to combine them. *Method.* We performed an empirical study employing 25 applications from a single software company. The company collected effort from 3 different phases (i.e., specification and analysis, system and object design, and implementation and testing). Linear regression was used to build the estimation models. *Results.* Our analysis revealed that we obtained more accurate estimations by using prior-phase effort data to estimate the effort of subsequent phases. The combination of the prior-phase efforts and Function Points allowed us to improve the estimations in some cases. *Conclusion.* The effort spent in the prior-phase of a project is a good predictor for the effort that will be required later. In a continuous estimation process project managers can benefit from this effort data to obtain more accurate estimations for the subsequent phase(s).

Categories and Subject Descriptors

D.2.8 [**Software Engineering**]: Metrics; D.2.9 [**Software Engineering**]: Management

General Terms

Management, measurement, experimentation

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from Permissions@acm.org.
PROMISE '14, September 17, 2014, Turin, Italy
Copyright 2014 ACM 978-1-4503-2898-2/14/09 ...$15.00.
http://dx.doi.org/10.1145/2639490.2639509

Keywords

Effort estimation; Functional Size measures; Function Points Analysis; Prior-phase effort

1. INTRODUCTION

When developing software systems, the estimation of the effort required is a crucial activity to make a bid, plan the development activities, allocate resources adequately, and so on. Development effort, meant as the work carried out by software practitioners, is the most dominant project cost, and also the most difficult to estimate and control. Significant over or underestimates can be very expensive and deleterious for a company [32] [39].

Several approaches have been proposed to estimate software development effort. Many of them rely on an algorithmic method that takes in input some project factors influencing the development effort (such as software size) and produce an effort estimate (e.g., [7], [33], [26]). Functional Size Measurement (FSM) methods, such as Function Points Analysis (FPA), have been widely applied in software engineering for sizing software systems. The obtained functional size can then be used as independent variable in effort estimation models [19]. A few works have also investigated the possibility of exploiting as independent variable the effort of the activities already completed [27], [41]. The research described in this paper is placed in this context and is based on the following considerations.

It is widely recognized that effort estimation is a managerial activity that takes place throughout the whole software life-cycle [6] [27] and can benefit from the information that become available as the project progresses. A continuos estimation (i.e., reviewing the project estimates and planning on an ongoing basis) is required in all projects to assess whether the initial estimate has been fairly good or it has to be adjusted according to the actual status of the project [6]. Moreover, during the software process, project managers can benefit from updated estimations not only of the total project effort but also of the effort devoted to each phase (e.g., design, implementation, testing) that has still to be carried out. Several works have analyzed the distribution of effort spent for different phases identifying possible patterns that depend on several features (e.g., employed development activities, employed technologies, characteristics of software company) [35] [43] [12].

To the best of our knowledge the first work addressing the problem of estimating the effort of a development phase exploiting the effort of a previous phase was done by Mac-

Donell and Shepperd [27]. They built an estimation model to predict the effort of the design phase by exploiting the effort of the planning phase (i.e., MS1 in Figure 1(a)). Similarly, the efforts of the implementation and testing phases were estimated using the effort of the design phase, i.e., by employing the models MS2 and MS3 in Figure 1(a). Furthermore, the model MS4 was built by exploiting the effort of the implementation phase and used to predict the effort for the testing phase. The results of their study suggested that prior-phase effort data can be useful to predict the effort of a subsequent phase during the software development process thus improving the subsequent management activities.

Recently Tsunoda et al. [41] have investigated if the use of early phase effort (e.g., related to planning and requirements analysis) is useful to estimate the total development effort. To this end they compared the resulting estimation model with the ones built by using the software size, expressed in terms of Function Points, and a combination of early effort data and software size. They built four models to predict the total development effort exploiting (i) the effort of the planning phase (i.e., model T2 in Figure 1(b)), (ii) the sum of both efforts for planning and requirements analysis phases (i.e., model T3 in Figure 1(b)), (iii) the combination of Function Points and planning effort, (iv) the combination of Function Points and effort for planning and requirements analysis. The model exploiting Function Points alone was also considered. The results of the empirical analysis showed the usefulness of early effort data which, alone or combined with Function Points, were able to provide estimation more accurate than the models based only on Function Points.

Based on the above encouraging results, we carried out an empirical analysis by generalizing the MacDonell and Shepperd idea [27] and taking into account the efforts of any prior-phase to predict the efforts of the subsequent phase and of all the remaining phases in the development process (see Figure 1(c)). Moreover, as made by Tsunoda et al. [41] we also compared the obtained accuracy with respect to the ones given by the corresponding estimation models based on Function Points. In particular, we defined the following research question:

RQ_1 Can prior-phase efforts provide more accurate estimates of the subsequent phase(s) effort than the corresponding models based on the use of Function Points alone?

Our empirical study employed 25 applications from a single software company while linear regression was used to build the estimation models. The company collected effort from 3 different phases (i.e., specification and analysis, system and object design, and implementation and testing). So, we built an estimation model to predict the effort of the system and object design phases exploiting the effort data of the specification and analysis phases (i.e., Eff1 in Figure 1(c)). Similarly, we used the effort data of the specification and analysis phase to estimate the effort spent for both the system and object design and the implementation and testing phases i.e., by employing model Eff2 in Figure 1(c). We also exploited both effort data of the specification and analysis and system and object design phases in order to predict the effort of the implementation and testing phase (i.e., model Eff3 in Figure 1(c)). Moreover, we built the corresponding models based on Function Points alone (i.e., FP1, FP2, and FP3). We also considered the estimation models based on the combination of Function Points and phases effort data (i.e., EffFP1, EffFP2, EffFP3), as made by Tsunoda et al. [41].

This leads to the following research question:

RQ_2 Can the combination of Function Points and prior-phase efforts provide more accurate estimates of the subsequent phase(s) efforts than the corresponding models based on the use of prior-phase efforts alone?

To address it we built three estimation models, i.e., Eff1, Eff2, and Eff3, using as exploratory variables the effort data and the size measure, expressed in terms of Functions Points.

Structure of the paper. Section 2 describes the design of the performed empirical study, while the results and their discussion are reported in Section 3. Section 4 summarizes the related work. Conclusion and future work conclude the paper.

2. EMPIRICAL STUDY DESIGN

2.1 Employed Functional Size Measure

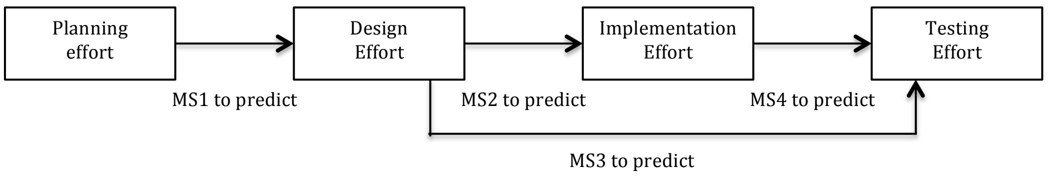
Functional Size Measurement (FSM) methods have obtained worldwide acceptance and allow measuring software size in terms of the functionality provided to the users. The success of FSMs can be mainly due to their early applicability and independence from the adopted programming languages. Function Point Analysis (FPA) was the first FSM method [4][5] and since then several variants have been defined (e.g., MarkII and NESMA [11]) aiming at improving size measurement or extending the applicability domain. IFPUG Function Points (FP, for short) represents the version of the Function Point Analysis (FPA) managed by the International Function Point Users Group (IFPUG) [19]. The FP method sizes an application using its Functional User Requirements (FURs) or other software artifacts that can be abstracted in terms of FURs.

To accomplish the measurement, each FUR is functionally decomposed into Base Functional Components (BFC), and each BFC is categorized into one of five Data or Transactional BFC Types. The Data functions are Internal Logical Files and External Interface Files, while the Transactional ones are External Inputs, External Outputs and External Inquires.

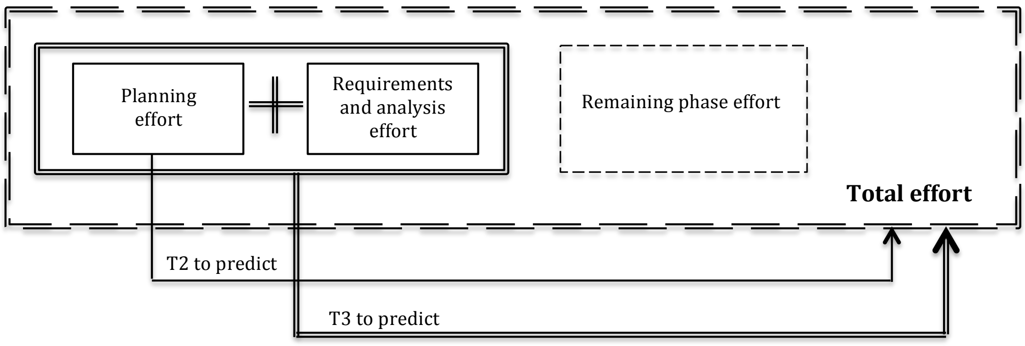
Then, the "complexity" of each BFC is assessed. This step depends on the kind of function type and requires the identification of further attributes (such as the number of data fields to be processed). Once derived this information, a table provided in the IFPUG approach method [19] specifies the complexity of each function, in terms of Unadjusted Function Points (UFP).

The sum of all these UFPs gives the functional size of the application. Subsequently, a Value Adjustment Factor (VAF), can be computed to take into account non-functional requirements, such as Performances, Reusability, and so on. The final size of the application in terms of Function Points is given by $FP = UFP \cdot VAF$.

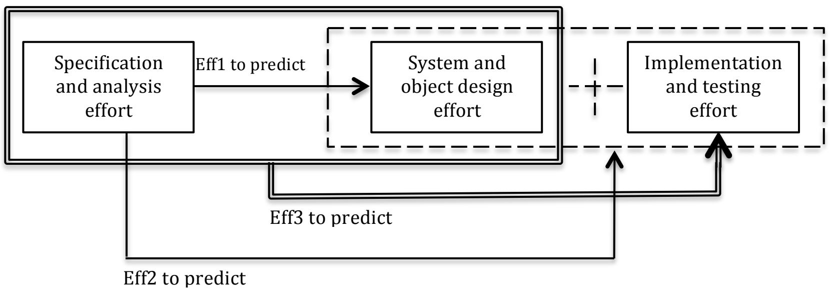
FP is widely exploited to estimate effort but also productivity (i.e., Function Points per person-month), and quality (i.e., number of defects with respect to requirements, design, coding and user documentation phases). For more details about the application of the IFPUG method, readers are referred to the counting manual [1] [19].



(a) MacDonell and Shepperd analysis [27]



(b) Tsnonoda et al. Analysis [41]


Specification and analysis effort
Eff1 to predict
System and object design effort
Implementation and testing effort
Eff3 to predict
Eff2 to predict

(c) our analysis

Figure 1: Exploiting development phase efforts

2.2 Data Set

The data for our empirical study were provided by a medium-sized software company, whose core business is the development of enterprise information systems, mainly for local and central government. Among its clients, there are health organizations, research centers, industries, and other public institutions. The company is specialized in the design, development, and management of solutions for Web portals, enterprise intranet/extranet applications (such as Content Management Systems, e-commerce, work-flow managers, etc.), and Geographical Information Systems. It has about fifty employees, it is certified ISO 9001:2000, and it is also a certified partner of Microsoft, Oracle, and ESRI.

This company provided us information on 25 applications they developed. All the projects have been developed by exploiting SUN J2EE or Microsoft .NET technologies. Oracle has been the most commonly adopted DBMS, but also SQL Server, Access, and MySQL have been employed in some of these projects. As for the employed development methodology, the company applied the waterfall model to develop the applications considered in our empirical study.

As for the collection of the information, the software company used timesheets to keep track of the application development effort. Each team member annotated the information about his/her development effort every day, and weekly each project manager stored the sum of the efforts for the team. Furthermore, to collect the information needed to calculate the values of the size measure in terms of FPA, the company usually uses template to be filled in by the project managers.

Table 1 reports on some summary statistics related to the 25 applications employed in our study[1]. The variables are EFF_A, i.e., the effort spent for the specification and analysis phases, EFF_D, i.e., the effort spent for the system

[1]Raw data cannot be revealed because of a Non Disclosure Agreement with the software company.

Table 1: Descriptive statistics of the considered variables

Var	Obs	Min	Max	Mean	Median	Std. Dev.
EFF$_A$	25	208	1496	853.2	864	332.04
EFF$_D$	25	264	1528	905	960	352.38
EFF$_I$	25	310	1513	817.8	738	319.43
EFF$_{D\&I}$	25	574	3041	1724	1748	664.24
FP	25	89	915	366.76	303.94	208.65

and object design phases, EFF$_I$, i.e., the effort spent for the implementation and testing phases, and EFF$_{D\&I}$, i.e., the effort obtained by summing EFF$_D$ and EFF$_I$, that are expressed in terms of person-hours, and FP, expressed in terms of number of Function Points obtained with IFPUG FPA.

2.3 Estimation Method

Linear regression is a statistical technique that explores the relationship between a dependent variable and one or more independent variables [34], providing a prediction model described by an equation

$$y = b_1x_1 + b_2x_2 + ... + b_nx_n + c \quad (1)$$

where y is the dependent variable, $x_1, x_2, ..., x_n$ are the independent variables, b_i is the coefficient that represents the amount the variable y changes when the variables x_i changes 1 unit, and c is the intercept.

In our empirical study, we employed simple linear regression to obtain an estimation model that uses the variable representing the effort of a prior-phase (e.g., EFF$_D$) as dependent and the variable denoting the employed size measure (i.e., FP) or the effort of a subsequent phase (e.g., EFF$_A$) as independent. Once the prediction model was constructed, the effort estimation of the subsequent development phase for a new application was obtained by sizing the application in terms of FPA (or considering the effort of the prior-phase, respectively) and using this value in the obtained model.

Moreover, we applied SWR (StepWise linear Regression) to build estimation models that exploit more variables as independent variables (i.e., FP, EFF$_A$, or EFF$_D$), and EFF$_D$, EFF$_I$, or EFF$_{D\&I}$ as dependent variable. This allowed us to consider models that combine the use of the size measure or the efforts of prior-phases to estimate the effort of the subsequent phase(s). SWR allowed us to compute linear regression in stages [15]. The model is built by adding, at each stage, the independent variable with the highest association to the dependent variable, taking into account all the variables currently in the model. It aims to find the set of independent variables (predictors) that best explains the variation in the dependent variable (response)[2].

To evaluate the goodness of fit of a regression model we considered several indicators. In particular, to determine the goodness of fit of the regression models we can use the coefficient of determination, R^2, which measures the percentage of variation in the dependent variable explained by the independent variable. A high R^2 value is an indication of the goodness of the prediction model. Other useful indicators are the F value and the corresponding p-value (denoted by Sign F), whose high and low values, respectively, denote a high degree of confidence for the prediction. Moreover, a t statistic can be performed to determine the p-value and the t-value of the coefficient and the intercept for each model in order to evaluate its statistical significance. p-value provides the probability that a coefficient of a variable is zero, while t-value can be used to evaluate the importance of the variable for the generated model. A p-value less than 0.05 indicates that we can reject the null hypothesis and the variable is a significant predictor with a confidence of 95%. As for the t-value, a variable is significant if the corresponding t-value is greater than 1.5.

2.4 Validation Method and Evaluation Criteria

We carried out a cross-validation to assess whether or not the obtained effort predictions are useful estimations of the actual development effort. In particular, we exploited a leave-one-out cross validation, which means that the original data set is divided into *n=25* different pairs (25 is the size of the original data set) of training and validation sets, where each tranining and validation set has *n-1* and one project, respectively. Training sets are used to build models with linear regression and validation sets are used to validate the obtained models. A recent study have shown the advantages of leave-one-out cross validation with respect to K-fold cross validation to assess software effort estimation models [25].

Regarding the evaluation criteria, we used some statistics of absolute residuals (i.e., $|Actual$ - $Predicted|$), namely:

- Median of Absolute Residuals (MdAR);
- Mean of Absolute Residuals (MAR);

These measures are unbiased since they are not based on ratios, as it is the case for other summary measures such as MMRE [14] which has undesirable properties, e.g. asymmetry [38]. Observe that we also report other summary measures, namely MMRE, MdMRE, Pred(25) (see [14] for their definitions), to allow a comparison with previous researches published in this context and they are not used for the assessment of the achieved effort estimations.

Moreover, we tested the statistical significance of the obtained results to compare predictions obtained with different approaches (e.g., to establish if the estimation models based on the use of the functional size measure provided significantly better absolute residuals than effort based estimation models [23] [40]). In particular, we performed the T-test to verify the following null hypothesis "the two considered population of absolute residuals have identical distributions". Observe that we used the Wilcoxon signed rank test when absolute residuals were not normally distributed [13].

In order to have an indication of the practical/managerial significance of the results we verified the effect size [21]. Effect size is a simple way of quantifying the standardized difference between two groups. It has many advantages over using only the tests of statistical significance, since "whereas p-values reveal whether a finding is statistically significant, effect size indicates practical significance" [21]. In particular, we employed the point-biserial correlation r because it is suitable to compute the magnitude of the difference when a non parametric test is used [16]. In the empirical software engineering field [21], the magnitude of the effect sizes

[2]We employed the automatic procedure provided by the R environment.

measured using the point-biserial correlation is classified as follows: small (0 to 0.193), medium (0.193 to 0.456), and large (0.456 to 0.868).

2.5 Threats to Validity

It is widely recognized that several factors can bias the construct, internal, external, and conclusion validity of empirical studies [24] [29] [42].

As for the construct validity, the choice of the size measures and how to collect the information needed to determine size measures and actual effort represents a crucial aspect. Regarding the selection of the approach to size the applications, we employed (IFPUG) FPA [1] that is a widely employed FSM method and has been applied for years by the company involved in our study. Often, the collection of information about the size measures and actual effort represents the main difficulty of this kind of study [22]. We checked the procedure used by the involved software company to carefully collect the information we need for the empirical analysis. In particular, we verified that the data collection task was done in a controlled and uniform fashion. Of course we are aware that controlled experiments ensure a higer level of confidence.

Concerning internal validity [29], the data considered were provided by the company and there was no selection neither on the applications nor the managers who provided the related information. Thus, no bias has apparently been introduced. Other concerns could regard data reliability and lack of standardization. However, the projects managers employed the same templates for all the applications and they were instructed on how to use the questionnaires, to correctly provide the required information. Furthermore, instrumentation effects in general did not occur in this kind of studies.

As for the conclusion validity, we applied the estimation method and the statistical tests by verifying all the required assumptions.

With regards to the external validity, a possible threat could be related to the fact that we considered applications from one company. It is recognized that the results obtained in an industrial context might not hold in other contexts [10]. However, in our analysis we were interested in analyzing the experience of a single company also based on the consideration that the identification of the activities of the process development and the distribution of the effort spent for each phase can depend on the specific software company [12].

3. RESULTS AND DISCUSSION

We performed the linear regression analysis to build the effort estimation models by using the data set of 25 applications (Table 1). Some statistics about the obtained linear regression models are shown in Table 2. It is worth noting that when we applied SWR using EFF$_A$ and EFF$_D$ as independent variables and EFF$_I$ as dependent variable, i.e., model Eff3 in the case of RQ1, the procedure in stages produced as the best predicting model the one employing only EFF$_D$ as independent variable. In the case of RQ2, when combining FP with the effort of prior-phase(s), SWR selected only EFF$_A$ for model EffFP1 which is equal to Eff1 (see Table 2). Similarly, only EFF$_D$ and FP were selected by SWR when considering EFF$_A$, EFF$_D$, and FP as independent variables and EFF$_I$ as dependent variable (i.e., model EffFP3 in Table 2).

Before applying linear regression we tested for normality the dependent and independent variables, employing the Shapiro-Wilk Test [37]. No transformation was required since all the variables were normally distributed, i.e., the test provided p-values greater than 0.05 in all the cases (see Table 3). In the application of linear regression we first verified the underlying assumptions, i.e., the existence of a linear relationship between the independent variable and the dependent variable (*linearity*), the average value of the error term is 0 (*zero mean*), the constant variance of the error terms for all the values of the independent variable (*homoscedasticity*), the normal distribution of the error terms (*normality*) [28]. In the following, we report on the analysis carried out to verify these assumptions. Note that the results are intended as statistically significant at α=0.05 (i.e., 95% confidence level):

- linearity. The linear relationship between each independent variable and the corresponding dependent variable was verified by using the (two-tailed) Pearson's correlation test [17]. The results of the performed tests suggested that there was a high (statistic $>$0.9) and significant (p-value $<$0.001) correlation between EFF$_A$ and EFF$_D$, EFF$_A$ and EFF$_{D\&I}$, and EFF$_D$ and EFF$_I$ (see Table 4). On the other hand, there was a significant correlation between FP and EFF$_D$, FP and EFF$_{D\&I}$, FP and EFF$_I$, with a statistic lower (0.775, 0.816, and 0.841, respectively) than the one characterizing the other models.

- zero mean. We performed the Student t-test to verify that the average value of the error term is not significantly different from 0. The results reported in Table 5 reveal that this assumption can be considered to be verified for all the built models (t-statistic = 0 and p-value = 1).

- homoscedasticity. We investigated the homoscedasticity assumption by performing the Breush-Pagan Test [9], with the homoscedasticity of the error terms as null hypothesis. As we can see from Table 5, the p-values obtained for all the models are greater than 0.05 and thus we can considered the assumption to be verified.

- normality. We used the Shapiro-Wilk Test [37], by considering as null hypothesis the normality of error terms. Again, we can considered the assumption to be verified since the p-values are greater than 0.05 for all the models (see Table 5).

Moreover, we verified the *lack of multicollinearity* in the independent variables, since it is an undesirable situation in regression analysis when two independent variables are highly correlated. To this end, we decided to check multicollinearity for the models EffFP2 and EffFP3 by first analyzing the correlation among the involved independent variables. The results of the Pearson correlation test performed on the variables and reported in Table 4 suggest that for the two independent variables EFF$_A$ and FP of model EffFP2 the correlation statistic is less than 0.8. Similarly, the two independent variables EFF$_D$ and FP of model EffFP3 are correlated with a statistic less than 0.8. Thus, this criterion holds for both the models having more than one independent variable. Moreover, we computed the tolerance that is calculated as 1-R$_j^2$, where R$_j^2$ is the coefficient of determination

Table 2: Linear regression results grouped for research question (RQ)

RQ	Dependent variable	Independent variables	Model	Variable(s)/ Intercept	Value	Std. Err	t-value	p-value	R^2	Std. Err	F	Sign. F
RQ1	EFF_D	EFF_A	Eff1	EFF_A	1.029	0.054	18.98	<0.001	0.94	88.2	360	<0.001
				Intercept	27.948	49.489	0.56	0.58				
	EFF_D	FP	FP1	FP	1.309	0.222	5.89	<0.001	0.601	227	34.7	<0.001
				Intercept	425.505	93.369	4.56	<0.001				
	$EFF_{D\&I}$	EFF_A	Eff2	EFF_A	1.927	0.112	17.230	<0.001	0.928	182	297	<0.001
				Intercept	79.452	102.124	0.780	0.440				
	$EFF_{D\&I}$	FP	FP2	FP	2.597	0.384	6.760	<0.001	0.665	392	45.800	<0.001
				Intercept	771.244	161.216	4.780	<0.001				
	EFF_I	EFF_A, EFF_D	Eff3	EFF_D	0.865	0.057	15.30	<0.01	0.91	97.6	234	<0.001
				Intercept	34.235	54.822	0.62	0.58				
	EFF_I	FP	FP3	FP	1.287	0.173	7.453	<0.001	0.707	177	55.5	<0.001
				Intercept	345.634	72.563	4.76	<0.001				
RQ2	EFF_D	EFF_A, FP	EffFP1(=Eff1)	EFF_A	1.029	0.054	18.98	<0.001	0.94	88.2	360	<0.001
				Intercept	27.948	49.489	0.56	0.58				
	$EFF_{D\&I}$	EFF_A, FP	EffFP2	EFF_A	1.627	0.155	10.53	<0.001	0.945	163	188	<0.001
				FP	0.629	0.246	2.560	0.018				
				Intercept	105.113	92.223	1.140	0.267				
	EFF_I	EFF_A, EFF_D, FP	EffFP3	EFF_D	0.687	0.077	8.87	<0.001	0.936	84.4	161	<0.001
				FP	0.387	0.131	2.96	0.007				
				Intercept	34.235	54.822	0.62	0.58				

Table 3: Results of the Shapiro-Wilk Test for normality of variables

Variables	statistic	p-value
EFF_A	0.97	0.65
EFF_D	0.973	0.731
EFF_I	0.957	0.365
EFF	0.979	0.859

Table 4: Results of the Pearson correlation test

Variables	statistic	p-value
EFF_D vs EFF_A	0.969	<0.001
EFF_D vs FP	0.775	<0.001
$EFF_{D\&I}$ vs EFF_A	0.963	<0.001
$EFF_{D\&I}$ vs FP	0.816	<0.001
EFF_I vs EFF_D	0.954	<0.001
EFF_I vs FP	0.841	<0.001

of a regression of the variable j on all the other independent variables [36]. The tolerance values we obtained for 2 and EffFP3 are 0.422 (=1-0.578) and 0.397 (=1-0.603), respectively. The criterion holds since tolerance values are greater than 0.3.

We also performed an analysis of influential observations and verified model's stability as suggested in [30]. No observations was proved to be an outlier and hence was left as per original data set.

We can observe that all the models are characterized by a very high R^2 value (greater than 0.9), a high F value, and a low Sign F (< 0.001 in all the cases), indicating that the prediction is indeed possible with a high degree of confidence (see Table 2), except for those using FP as independent variable since they have R^2 values ranging from 0.601 to 0.707 and F values ranging from 34.7 to 55.5. However, all the FP based models are characterized by a Sign F value less than <0.001. As for the performed t-statistic, for all the models the considered independent variables have a t-value and a p-value greater than 1.5 and less than 1.5, respectively, meaning that the employed variables are important for the generated models and significant. Furthermore, note that all the models using a variable denoting the effort for a development phase as independent variable have the intercept characterized by a t-value less than 1.5 and a p-value greater than 0.05, meaning that the intercept is not important and significant in the generated model. The models using FP as the only independent variable have an intercept satisfying the thresholds for t-values and p-values.

Table 6: The results of validation for each research question (RQ)

RQ	Model	MdAR	MAR	MMRE	MdMRE	Pred(25)
RQ1	Eff1	70.6	72	0.08	0.08	1
	FP1	160	195	0.30	0.15	0.72
	Eff2	117	159	0.10	0.09	1
	FP2	315	350	0.26	0.14	0.64
	Eff3	64.1	79.1	0.11	0.11	0.96
	FP3	171	159	0.23	0.17	0.64
RQ2	EffFP2	107	132	0.08	0.07	1
	EffFP3	53.2	69.7	0.04	0.03	1

To evaluate the estimation accuracy of the models built with linear regression analysis, we performed a leave-one-out cross validation, whose results are reported in Table 6.

Comparing the results achieved with the models Eff1 and FP1 (i.e., to address RQ1), we can observe that the values of MdAR and MAR are more than two times lower for the model using EFF_A, meaning that Eff1 allowed us to obtain better effort estimations. The values of MdAR and MAR characterizing the model Eff2 are also more than two times lower than those obtained with the model FP2. A similar result is achieved for model Eff3 providing estimations two time better than those achieved with FP3.

As for research question RQ2, the results of Table 6 suggest that the combination of EFF_A and FP to predict $EFF_{D\&I}$ (i.e., model EffFP2) allowed us to obtain lower MdAR and MAR values with respect to use EFF_A or FP alone (i.e., model Eff2 and FP2). Moreover, we can observe that MdAR and MAR values achieved combining the use of FP with EFF_D to predict EFF_I (i.e., model EffFP3) are lower than those obtained employing EFF_D or FP alone (i.e., model Eff3 and FP3).

Table 5: Results of the tests to verify Zero mean, Homoscedasticity, and Normality assumptions

Estimation model	Student t-test statistic/p-value	Breush-Pagan Test statistic/p-value	Shapiro-Wilk Test statistic/p-value
Eff1	0/1	1.595/0.207	0.972/0.696
FP1	0/1	1.125/0.289	0.937/0.129
Eff2	0/1	1.252/0.263	0.966/0.552
FP2	0/1	0.254/0.614	0.927/0.075
Eff3	0/1	1.923/0.165	0.955/0.324
FP3	0/1	0.340/0.560	0.929/0.082
EffFP1 (= Eff1)	0/1	1.252/0.263	0.966/0.082
EffFP2	0/1	1.265/0.531	0.960/0.406
EffFP3	0/1	3.951/0.139	0.983/0.943

To verify whether the difference highlighted with the analysis in terms of the MdAR and MAR values are significant we performed statistical tests by using absolute residuals [23] [29] [40]. The results are reported in Table 7. In particular, the T-test (or the Wilcoxon test if absolute residuals were not normally distributed) revealed that:

RQ1 The absolute residuals provided by the model Eff1 are significantly lower than those obtained with model FP1 (with a large effect size since $0.456<r<0.868$), meaning that it is better to estimate the development effort for the system and object design phases by exploiting the development effort of the previous phases, i.e., specification and analysis.

EFF_A also allowed us to obtain estimations of $\text{EFF}_{D\&I}$ (i.e., the effort for the system and object design and the implementation and testings phases) better than those achieved using FP (with a large effect size).

A similar result was achieved in the case of the effort predicted for the implementation and testing phases. Model Eff3 provided significantly less absolute residuals than the model FP3 (with a large effect size), meaning that better estimations are obtained by exploiting the development effort of the previous phases, i.e., system and object design.

RQ2 The results reported in Table 7 reveal that the model combining early effort data, i.e., EFF_A, and the functional size in terms of Function Points can provide significantly better estimations than the model based only on EFF_A (with a medium effect size since $0.193 <r<0.456$).

The results have also revealed that there is not statistically significant difference between the absolute residuals achieved with the model employing only EFF_D (i.e., the effort of system and object design phases) as independent variable and those obtained by using the model exploiting both EFF_D and FP as independent variables, to predict the effort for the implementation and testing phases (i.e., EFF_I).

The above results suggest that we can positively answer our first research question RQ1 (i.e., prior-phase efforts can provide more accurate estimates of the subsequent phase(s) effort than the corresponding models based on the use of Function Points alone).

Furthermore, the combination of FP with the effort for the specification and analysis phase (i.e., EFF_A) allowed us to significantly improve the results achieved in terms of EFF_A (and of course of FP) alone, in estimating the effort for the system and object design, and implementation and testing phases. Similarly, the combination of FP with the effort for the system and object design phases provided better estimations than using the effort for the system and object design phases alone. However, the difference in the absolute residuals is not statistically significant. Thus, we can partially positively answer our second research question RQ2.

Table 7: The results of the statistical test and effect size analysis, by considering absolute residuals, for each research question (RQ)

RQ	Comparison	Statistical test p-value	Effect size r
RQ1	Eff1 vs FP1	<0.001	0.669
	Eff2 vs FP2	0.002	0.592
	Eff3 vs FP3	0.002	0.542
RQ2	EffFP2 vs Eff2	0.042	0.342
	EffFP3 vs Eff3	0.149	0.211

Summary of results and contribution. The analysis reported above suggests that the accuracy achieved predicting the effort to accomplish a development phase with the effort of the previous phase(s) is significantly better than the accuracy obtained by exploiting a Function Points based estimation model. The results of our empirical study have also highlighted that efforts for the specification and analysis phases provided estimations of the effort for the subsequent phases (i.e., system and object design, and development and testing) better than those achieved by employing a Function Points based model.

Moreover, the combined us of specification and analysis data effort and the number of Function Points can improve the estimates. However, the improvement was not statistically significant when such a combination was used to predict the effort for the final phases (i.e., implementation and testing phases).

Thus, for the company seems to be crucial to carefully collect information on effort spent during the project in order to build estimation models able to predict the effort of the development phase(s). However, the use of a size measure, like Function Points, remain important since the combination of effort data and size measure can allow to (significantly) improve the accuracy of effort estimations. Furthermore, the use of a measurement method is also important to esitmates the effort for the first phases of the development process, i.e., when we have no effort data on prior-phases to exploit, without forgetting that a functional size measure, like Function Points, also supports the project managers for other management activities, e.g., productivity benchmarking. On the other hand, it could also be interesting to identify a less expensive method (in terms of time and cost) able to effectively predict the effort for the first phases (in our case specification and analysis).

4. RELATED WORK

Related work falls in the context of phase-based estimations concerning with both analyses and visualizations of activities and effort distribution [35] [43] [12] and in the context of the use of effort data to estimate subsequent phases [27] [20] [6] [41].

Ohlsson and Wohlin [35] used phase-based data, such as the number of requirements and flowcharts, to estimate effort for the subsequent phases. The results, however, showed that the metrics used did not correlate particularly well with effort.

Yang et al. [43] analyzed the phase effort distribution patterns and different variation sources. To this end, they carried out an empirical study on phase effort distribution data of 75 industrial projects, from the China Software Benchmarking Standard Group (CSBSG) database [18]. The results revealed some consistency in effects of software size and team size on code and test phase distribution variations, and some considerable deviations in requirements, design, and transition phases, compared with recommendations in the COCOMO model [8].

Chatzipetrou et al. [12] also studied the effort distribution over different phases of 1,500 projects from the ISBSG R11 database [3]. In particular, they studied the correlation between the effort distribution and different kinds of phase-based data, i.e., project life-cycle activities, organization type, language type, function points, and other prime project attributes. ISBSG provides information on effort for the following development phases: Planning, Specification, Design, Build, Test, and Implementation. It is worth noting that ISBSG defines Implementation as activities to do with delivering and installing the final product. In our case, the definition of Implementation refers to "implementation activities" and seems to correspond to the Build phase of ISBSG. The main findings of the Chatzipetrou et al. work are: a) the Build and Test phases are characterized by effort values greater than those of the other phases; b) the efforts are distributed among the different development phases according the organization type; c) more effort is distributed in the Design phase when high level programming language, such as 3GL, 4GL, and Java are employed, while the use of C/C++ or C produces more effort to carry out implementation activities; d) there was correlation between software size and effort data characterizing the phases.

Aroonvatanaporn et. al [6] proposed a framework to continuously monitor the software project progress and readjust the estimated effort utilizing COCOMO II [8]. A simulation of this framework, carried out on data from two academic software development projects, showed significant improvements in estimating project resources with significant reduction in estimation errors as the project progresses through its life cycle.

Jiang et al. [20] proposed a model to predict development effort based on the software size estimated with Function Points. The authors used the ISBSG dataset and provided estimates for the effort used in Building, Testing, and Implementation phases. The results showed that there was a strong positive relationship between the software size, given in terms of number of Function Points, and the total project development effort. Furthermore, the analysis also highlighted that there was a strong correlation between the effort of each development phase and the software size.

MacDonell and Shepperd [27] studied an approach based on the use of effort data recorded for those project tasks already completed to predict the effort needed for subsequent activities. In particular, they investigated the performance of models that use prior-phase effort as an explanatory variable and estimated next phase effort, using 16 projects collected from a single software company. The obtained results showed that prior-phase effort data can be used to augment the estimation process already in place, based on expert estimations, in order to improve the management of subsequent process tasks. However, they did not use software size as an explanatory variable nor estimate total development effort.

Tsunoda et al. [41] investigated which model shows higher estimation accuracy: a model using software size, or early phase effort, or their combination. This is the closest study to the one we have presented herein. Release 9 of ISBSG was exploited to obtain the data set used in the performed empirical study. They first applied a selection procedure on the initial 3,016 project to obtain information on 70 and 178 projects to be employed in the analysis to address their three research questions. As for the variables employed to build the models to estimate total effort, they considered the software size, given in terms of number of Function Points, and the effort for each of the following development phases: Planning, Requirement analysis, Planning and Requirement analysis. However, Planning effort and Requirement analysis effort were not considered separately as predictor of total effort since they were characterized by multicollinearity. They built estimation models that employ early phase effort as predictors, but total effort was the information to predict, while we exploit effort of prior-phases to predict the effort of the subsequent phases. It is worth noting that since we predict the effort also for all the remaining phases, in a certain sense, we also predict the total effort. They also compared the accuracy of the effort estimations obtained by using these models with the accuracy of models based only on the software size (i.e., Function Points). They also built estimation models using both early phase effort and software size. They employed linear regression as estimation techniques and applied a 5-fold cross validation to validate the estimates. As evaluation criteria they exploited several summary measures based on absolute residuals as we did in our study, namely MAR and MdAR. They also based the comparison among the different estimation methods on unbiased summary measures like MMRE and MdMRE [38]. The results showed that the models based on early effort data provided estimations better than those achieved with the models based on Function Points. Furthermore, using both software size and early phase effort improved estimation accuracy and multicollinearity did not arise. Thus, our work confirms that using prior-phase effort is more effective than using only software size, given in terms of Function Points, to estimate the effort required in subsequent phase(s), while the combination of prior-phase effort data and Function Points provide slightly improvements.

5. CONCLUSION AND FUTURE WORK

We have presented the results of an empirical study performed to analyze the possibility of applying the effort of the development phase(s) to predict the effort required for the subsequent phase(s). Furthermore, we assessed the combined use of the effort of the prior-phase(s) and the software size given in terms number of Function Points.

The study was based on 25 applications from a single software company that provided us the effort data of 3 different phases (i.e., specification and analysis, system and object design, and implementation and testing). Linear regression was used to build the estimation models.

The analysis has revealed that the effort data of prior-phase(s) can provide estimations of the effort of the subsequent development phases significantly better than those achieved by using Function Points alone, thus confirming MacDonell and Shepperd [27] results that the use of prior-phase effort data can augment the estimation process already in place thus improving the management of subsequent process tasks. Moreover, it confirms and extends the results of Tsunoda et al. [41] showing that the combination of prior-phases effort data and the number of Function Points can further improve the effort estimations.

As future work we intend to replicate the present study by employing other Functional Size Measurement methods, such as MarkII and NESMA [11] and COSMIC [2] Furthermore, we intend to replicate the study by exploiting data from publicly available data repository (like PROMISE [31]) with the aim of comparing the results we obtained on a single company data set with cross-company data sets containing different types of applications. We also intend to extend the study by exploiting other cost drivers (e.g., related to personnel and projects attributes) together with Function Points.

6. REFERENCES

[1] ISO/IEC 20926: Software Engineering - IFPUG 4.1 Unadjusted FSM Method - Counting Practices Manual, 2003.

[2] A. Abran, J. Desharnais, A. Lesterhuis, B. Londeix, R. Meli, P. Morris, S. Oligny, M. OÕNeil, T. Rollo, G. Rule, L. Santillo, C. Symons, and H. Toivonen. The COSMIC Functional Size Measurement Method - Measurement Manual, version 3.0.1, 2008.

[3] A. Abran and et al. The COSMIC Functional Size Measurement Method - Measurement Manual, version 3.0.1 - http://www.cosmicon.com/, 2009.

[4] A. Albrecht. Measuring Application Development Productivity. In *Proceedings of the Joint SHARE/GUIDE/IBM Application Development Symposium*, pages 83–92, 1979.

[5] A. J. Albrecht and J. E. Gaffney. Software Function, Source Lines of Code, and Development Effort Prediction: A Software Science Validation. *IEEE Transactions on Software Engineering*, 9(6):639–648, 1983.

[6] P. Aroonvatanaporn, C. Sinthop, and B. Boehm. Reducing estimation uncertainty with continuous assessment: Tracking the "cone of uncertainty". In *Proceedings of the IEEE/ACM International Conference on Automated Software Engineering*, ASE '10, pages 337–340, New York, NY, USA, 2010. ACM.

[7] D. Azhar, P. Riddle, E. Mendes, N. Mittas, and L. Angelis. Using ensembles for web effort estimation. In *ESEM*, pages 173–182, 2013.

[8] B. Boehm, C. Abts, A. Brown, S. Chulani, B. Clark, W. Horowitz, R. Madachy, D. Reifer, and B. Steece. *Software Cost Estimation with COCOMO II*. Prentice Hall, NJ, 2000.

[9] T. Breush and A. Pagan. A simple test for heteroscedasticity and random coefficient variation. *Econometrica*, 47:1287–1294, 1992.

[10] L. C. Briand and J. Wüst. Modeling Development Effort in Object-Oriented Systems Using Design Properties. *IEEE Transaction on Software Engineering*, 27(11):963–986, 2001.

[11] Çigdem Gencel and O. Demirörs. Functional size measurement revisited. *ACM Trans. Softw. Eng. Methodol.*, 17(3), 2008.

[12] P. Chatzipetrou, E. Papatheocharous, L. Angelis, and A. Andreou. An investigation of software effort phase distribution using compositional data analysis. In *Software Engineering and Advanced Applications (SEAA), 2012 38th EUROMICRO Conference on*, pages 367–375, Sept 2012.

[13] W. J. Conover. *Practical Nonparametric Statistics*. Wiley, 3rd edition edition, 1998.

[14] D. Conte, H. Dunsmore, and V. Shen. *Software engineering metrics and models*. The Benjamin/Cummings Publishing Company, Inc., 1986.

[15] N. Draper and H. Smith. *Applied Regression Analysis*. John Wiley & Sons, Inc., 2d Edition, New York, 1981.

[16] A. Field and G. Hole. *How to Design and Report Experiments*. Sage publications Limited, 2003.

[17] J. Freund. *Mathematical Statistics*. Prentice-Hall, Upper Saddle River, NJ, 1992.

[18] M. He, M. Li, Q. Wang, Y. Yang, and K. Ye. An investigation of software development productivity in china. In *Proceedings of the Software Process, 2008 International Conference on Making Globally Distributed Software Development a Success Story*, ICSP'08, pages 381–394, Berlin, Heidelberg, 2008. Springer-Verlag.

[19] IFPUG. International Function Point Users Group - www.ifpug.org.

[20] Z. Jiang, P. Naudé, and B. Jiang. The effects of software size on development effort and software quality, 2007.

[21] V. Kampenes, T. Dyba, J. Hannay, and I. Sjøberg. A systematic review of effect size in software engineering experiments. *Information and Software Technology*, 4(11-12):1073–1086, 2007.

[22] C. Kaner and W. Bond. Software Engineering Metrics: What Do They Measure and How Do We Know? In *Proceedings of the International Software Metrics Symposium*. IEEE press, 2004.

[23] B. Kitchenham, L. Pickard, S. MacDonell, and M. Shepperd. What accuracy statistics really measure. *IEE Proceedings Software*, 148(3):81–85, 2001.

[24] B. Kitchenham, L. Pickard, and S. Pfleeger. Case studies for method and tool evaluation. *IEEE Software*, 12(4):52–62, 1995.

[25] E. Kocaguneli and T. Menzies. Software effort models should be assessed via leave-one-out validation. *Journal of Systems and Software*, 86(7):1879–1890, 2013.

[26] E. Kocaguneli, T. Menzies, and J. W. Keung. On the value of ensemble effort estimation. *IEEE Trans. Software Eng.*, 38(6):1403–1416, 2012.

[27] S. MacDonell and M. Shepperd. Using prior-phase effort records for re-estimation during software projects. In *Software Metrics Symposium, 2003. Proceedings. Ninth International*, pages 73–86, Sept 2003.

[28] K. Maxwell. *Applied Statistics for Software Managers.* Software Quality Institute Series, Prentice Hall, 2002.

[29] E. Mendes, S. Counsell, N. Mosley, C. Triggs, and I. Watson. A Comparative Study of Cost Estimation Models for Web Hypermedia Applications. *Empirical Software Engineering*, 8(23):163–196, 2003.

[30] E. Mendes and B. Kitchenham. Further Comparison of Cross-company and Within-company Effort Estimation Models for Web Applications. In *Proceedings of International Software Metrics Symposium*, pages 348–357. IEEE press, 2004.

[31] T. Menzies, B. Caglayan, Z. He, E. Kocaguneli, J. Krall, F. Peters, and B. Turhan. The promise repository of empirical software engineering data, June 2012.

[32] T. Menzies, Z. Chen, J. Hihn, and K. Lum. Selecting Best Practices for Effort Estimation. *IEEE Transactions on Software Engineering*, 32(11):883–895, 2006.

[33] L. L. Minku and X. Yao. Ensembles and locality: Insight on improving software effort estimation. *Information & Software Technology*, 55(8):1512–1528, 2013.

[34] D. Montgomery, E. Peck, and G. Vining. *Introduction to Linear Regression Analysis.* John Wiley and Sons, Inc., 1986.

[35] M. Ohlsson and C. Wohlin. An empirical study of effort estimation during project execution. In *Software Metrics Symposium, 1999. Proceedings. Sixth International*, pages 91–98, 1999.

[36] R. OÕbrien. A caution regarding rules of thumb for variance inflation factors. *Quality & Quantity: International Journal of Methodology*, 41(5):673–690, 2007.

[37] P. Royston. An extension of Shapiro and Wilk's W test for normality to large samples. *Applied Statistics*, 31(2):115–124, 1982.

[38] M. J. Shepperd and S. G. MacDonell. Evaluating prediction systems in software project estimation. *Information & Software Technology*, 54(8):820–827, 2012.

[39] I. Sommerville. *Software Engineering (8th Edition).* Addison-Wesley, 2007.

[40] E. Stensrud and I. Myrtveit. Human performance estimating with analogy and regression models: an empirical validation. In *Proceedings of International Software Metrics Symposium*, pages 205–. IEEE press, 1996.

[41] M. Tsunoda, Y. Kamei, K. Toda, M. Nagappan, K. Fushida, and N. Ubayashi. Revisiting software development effort estimation based on early phase development activities. In *Mining Software Repositories (MSR), 2013 10th IEEE Working Conference on*, pages 429–438, May 2013.

[42] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, and B. Regnell. *Experimentation in Software Engineering.* Springer, 2012.

[43] Y. Yang, M. He, M. Li, Q. Wang, and B. Boehm. Phase distribution of software development effort. In *Proceedings of the Second ACM-IEEE International Symposium on Empirical Software Engineering and Measurement*, ESEM '08, pages 61–69, New York, NY, USA, 2008. ACM.