Information and Software Technology 55 (2013) 1796–1809

Contents lists available at SciVerse ScienceDirect

Information and Software Technology

journal homepage: www.elsevier.com/locate/infsof

Towards a simplified definition of Function Points

Luigi Lavazza [a,*], Sandro Morasca [a], Gabriela Robiolo [b]

[a] Università degli Studi dell'Insubria, Department of Theoretical and Applied Sciences, Italy
[b] Universidad Austral, Facultad de Ingeniería, Argentina

ARTICLE INFO

Article history:
Received 28 September 2012
Received in revised form 8 March 2013
Accepted 16 April 2013
Available online 3 May 2013

Keywords:
Functional size measurement
Function Points
Effort prediction

ABSTRACT

Background: The measurement of Function Points is based on Base Functional Components. The process of identifying and weighting Base Functional Components is hardly automatable, due to the informality of both the Function Point method and the requirements documents being measured. So, Function Point measurement generally requires a lengthy and costly process.
Objectives: We investigate whether it is possible to take into account only subsets of Base Functional Components so as to obtain functional size measures that simplify Function Points with the same effort estimation accuracy as the original Function Points measure. Simplifying the definition of Function Points would imply a reduction of measurement costs and may help spread the adoption of this type of measurement practices. Specifically, we empirically investigate the following issues: whether available data provide evidence that simplified software functionality measures can be defined in a way that is consistent with Function Point Analysis; whether simplified functional size measures by themselves can be used without any appreciable loss in software development effort prediction accuracy; whether simplified functional size measures can be used as software development effort predictors in models that also use other software requirements measures.
Method: We analyze the relationships between Function Points and their Base Functional Components. We also analyze the relationships between Base Functional Components and development effort. Finally, we built effort prediction models that contain both the simplified functional measures and additional requirements measures.
Results: Significant statistical models correlate Function Points with Base Functional Components. Basic Functional Components can be used to build models of effort that are equivalent, in terms of accuracy, to those based on Function Points. Finally, simplified Function Points measures can be used as software development effort predictors in models that also use other requirements measures.
Conclusion: The definition and measurement processes of Function Points can be dramatically simplified by taking into account a subset of the Base Functional Components used in the original definition of the measure, thus allowing for substantial savings in measurement effort, without sacrificing the accuracy of software development effort estimates.

1. Introduction

Correct and timely prediction of software development cost may provide software companies with a very valuable competitive advantage. In software development, effort is the main cost driver [1], so the ability to predict it early in the life cycle may help software companies decide whether it is economically sensible for them to develop a software product and, if they decide to go ahead with development, price the product right and allocate resources adequately.

Software size is one of the main factors that are believed to influence software development effort. Intuitively, it makes sense to presume that the greater the size of a software product, the greater the effort needed to develop it, and, therefore, the greater the cost. Such intuition was confirmed and formalized by models that established a relationship between functional software size and software effort (see for instance [2,3]).

A number of measures have been proposed to quantify the size of a software product early on in software development, based on Functional User Requirements (FUR). These proposals are collectively known as Functional Size Measurement (FSM) methods. The introduction of FSM has also led to the evolution of effort prediction models (a recent overview of FSM-based effort prediction tools and techniques can be found in [4]).

Function Points Analysis (FPA) [5] was the first proposal for FSM. Among other elements, FPA takes into account the so-called "elementary processes" of a software application, which are sized

* Corresponding author. Tel.: +39 0332218930; fax: +39 0223954458.
E-mail address: luigi.lavazza@uninsubria.it (L. Lavazza).

0950-5849/$ - see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.infsof.2013.04.003

based on the amount of data exchanged across the boundaries of the application and the number of data groups accessed within the application. FPA provides a measure of size in Function Points (FP). FPA is carried out during the early development phases, so it requires manual analysis of informal documents such as software requirements, and the sizing (called "counting") of an application well before it is built, as we concisely explain in Section 2.

Following the introduction of FPA, other FSM methods have been proposed, e.g., MarkII Function Points [6], NESMA [7], FiSMA [8], and the Common Software Measurement International Consortium (COSMIC) method [9]. All of these methods have provided important contributions towards the measurement of the functional size of software. However, they share a number of issues that need to be dealt with and resolved to make FSM a more reliable and widely accepted method.

1. A first issue with FSM methods is that they are generally lengthy and costly, as they require that measurers read requirements specifications made of collections of heterogeneous documents, often totaling several hundred pages. In fact, FP measurement speed can be as low as 200 Function Points per work day [10]. Accordingly, if the size measure is needed for bidding purposes in three days, an application having size around 1000 FP could not be measured in time. In addition, we should consider that the cost of five work days of a professional FP counter could be considered too high in some organizations.
2. A second issue is that some degree of subjectivity in the results is to be expected, since the definitions of FSM are informal and prone to different interpretations by different subjects. Moreover, requirements documents themselves are generally written in natural language, thus allowing for different interpretations, which, in turn, may lead different measurers to give different values for the same size measures, as has already been shown [11–14].
   To address these first two issues, a few requirements modeling methodologies and measurement tools have been introduced to partly automate the measurement process [15–17]. However, the use of any automated tool on a set of informal requirements suffers from the typical problems of any automatic analysis of natural language texts, so it is not possible to guarantee that the values of the measures obtained automatically are close enough to those that would be provided by expert measurers, which are the reference ones. As a result, these proposals are not widely used.
3. A third, more fundamental issue is related to the very definitions of FSM methods (mainly FPA), which have been subject to critical scrutiny, because of the potential problems they may generate. For instance, it has been observed that the components upon which the FP counting is based have strong inter-correlations, which could make the measures somewhat unreliable [18].

In this paper, we focus on FPA, which is by far the most widely used FSM method. Specifically, we address the three issues mentioned above by simplifying the definition of Function Points, without reducing the accuracy of estimation models built based on the simplified measures. Our simplified measure definitions benefit from the strengths of consolidated FP counting procedures, which have been assessed and refined over the last few years. Our simplification is obtained and validated via an empirical approach based on real-world data to this end.

Our basic idea comes from the observation that FP measurement is based on the measure of Basic Functional Components (BFC) and the subsequent weighting and aggregation of BFC measures. As we show in the review of the literature in Section 8, some studies have provided some initial evidence suggesting that the measures of BFC[1] are statistically associated with Unadjusted Function Points (UFP), which are the most basic part of FP measure, and the standardized one, as we explain in Section 2. So, it may not really be necessary to consider *all* of the elements taken into consideration by FPA to get a representative measure for the functional size of software. Being able to exclude a few BFC from the measure definition would have important practical consequences towards the solution of the three issues outlined above:

1. Needing to identify and evaluate fewer BFC provides two important benefits: first, the duration and cost of measurement can be reduced; second, the evaluation of size can start earlier, as soon as user requirements specifications concerning the considered BFC are complete, while the specification of requirements concerning the BFC not included in the measure can proceed in parallel with the measurement. These effects can enable measurers to provide the functional measures "on time," i.e., within the given deadlines.
2. The degree of subjectivity could be reduced. In fact, the fewer BFC have to be identified and evaluated, the less room there is for different interpretations of requirements.
3. Theoretical benefits could be achieved as well. In fact, if the measure were based on a single BFC, the problems due to inter-correlations among BFC would simply disappear.

The achievement of the expected enefits described above is discussed in Section 9.2.

Though functional size measures can be used for different purposes, we here concentrate on the most typical one, i.e., effort estimation. Thus, the final goal of the research described in this paper is to propose and empirically validate a method that:

(a) simplifies FP definitions, by retaining FPA measurement criteria and practices, but using only a subset of FPA BFC,
(b) can predict software development effort with accuracy comparable to that of FPA, and
(c) can be used, possibly in combination with measures of other requirements attributes, to predict software development effort with better accuracy than FP measures alone.

In our empirical approach, we use data from the public ISBSG dataset for points (a) and (b) above, because it contains data on a relatively high number of projects. As for point (c), we used another dataset that we have independently collected on a different set of projects, as this dataset also contains the values of a requirements complexity measure that is not included in the ISBSG dataset.

We would like to note that our paper is about the simplification of the *definition* of Function Points while other proposals have addressed the simplification of the *process* of functional size measurement while keeping the definitions of Function Points measures unchanged. For instance, the NESMA indicative [19] methods provide ways to estimate the size in FP of a software application based exclusively on the knowledge of data functions; the NESMA estimated method requires the identification and classification of all data and transaction functions, but does not require the assessment and weighting of each function. Similarly, with the Early & Quick Function Points (EQFP) [20], different parts of the system can be measured at different detail levels: a part of the system can be measured following the IFPUG manual rules [21], while other parts can be measured on the basis of coarser-grained

[1] For simplicity, we use the same form for the singular and the plural of acronyms in this paper. So, for instance, we use "BFC" for both "Base Functional Component" and "Base Functional Components."

information. However, all these methods provide estimates of size in Function Points. This is a fundamental difference with respect to the method that we propose in this paper, whose result is the definition and use of FSM measures that (1) are not Function Points (i.e., the value of the measure according to the simplified definition is different from the value obtained via traditional FPA), (2) support effort estimation, and (3) are clearly correlated with UFP. In our approach, the simplification of the measurement process comes as a consequence of the simplification of the measure definition.

The paper is organized as follows. Section 2 gives a concise introduction to FPA. Section 3 details the goals of our empirical study, the research questions, and the statistical techniques used. Section 4 addresses goal (a) above, by empirically investigating the statistical relationships between BFC and functional software measures. Section 5 addresses goal (b) above and shows that the measures of functional size based on BFC may be used as a replacement of traditional size measures in effort prediction. In addition, Section 6 addresses goal (c) above and shows how effort prediction models can be built by using BFC and other software requirements measures. Threats to the validity of our study are discussed in Section 7. Section 8 accounts for related work, and Section 9 draws the conclusions of the paper and provides an outline for future work.

2. A concise introduction to Function Points Analysis

The Function Point method was originally introduced by Albrecht to measure the size of data-processing systems from the end-user's point of view, with the goal of estimating the development effort [5].

The initial interest sparked by FPA along with the recognition of the need for improvement in its counting practices led to founding the International Function Points User Group (http://www.ifpug.org/), which provides guidelines for carrying out FPA, makes FPA counting rules evolve along with the evolution in software technologies, and oversees FPA's standardization.

IFPUG FPA is now an ISO standard [21] in its "unadjusted" version. So, throughout the paper, unless otherwise explicitly stated, we refer exclusively to Unadjusted Function Points, which are generally referred to as "UFP."

The basic idea of FPA is that the "amount of functionality" released to the user can be evaluated by taking into account the data used by the application to provide the required functions, and the transactions (i.e., operations that involve data crossing the boundaries of the application) through which the functionality is delivered to the user. Both data and transactions are evaluated at the conceptual level, i.e., they represent data and operations that are relevant to the user. Therefore, Function Points (FP) are counted on the basis of the user requirements specification. The boundary indicates the border between the application being measured and the external applications and user domain.

In Function Point Analysis, functional user requirements are modeled as a set of Basic Functional Components (BFC), which are considered the elementary unit of functional user requirements. Each of the identified BFC is then measured; finally, the size of the whole application is obtained as the sum of the sizes of BFC.

FPA BFC are data functions (DF), which are classified into internal logical files (ILF) and external interface files (EIF), and transactional functions, which are classified into external inputs (EI), external outputs (EO), and external inquiries (EQ) according to the main intent of the process. Each function, whether a data or transactional one, contributes a number of FP that depends on its "complexity." Each function is weighted on the basis of its complexity according to given tables. Finally, the number of so-called Unadjusted Function Points (UFP) is obtained by summing the contribution of the function types. Details about FP measurement can be found in the manual [21].

Throughout the paper we use the notation $M(x)$ to indicate the measure of the size of BFC x, obtained according to IFPUG measurement rules. So, for instance, $M(\mathrm{EI})$ is the measure of the size of external inputs.

The size of a software application – expressed as the number of Unadjusted Function Points (UFP) – is described in Formula (1).

$$\mathrm{UFP} = M(\mathrm{EI}) + M(\mathrm{EO}) + M(\mathrm{EQ}) + M(\mathrm{ILF}) + M(\mathrm{EIF}) \quad (1)$$

3. Overview of the empirical study

We here provide an overview of the goals, research questions, and statistical techniques of our study, whose empirical analyses are described in Sections 4–6.

3.1. Goals of the analysis

Our purpose is to assess if it is possible to simplify the definition of Function Points, by investigating whether some BFC can be safely ignored. Thus, we stuck to the framework of FPA, where UFP is computed as sum of BFC sizes, with the underlying implicit idea that the single contributions related to each individual BFC are perfectly comparable and can be summed. In fact, the purpose of complexity weights in FPA is to make the contributions of all BFC comparable so they can be summed to obtain UFP.

Thus, we consider sums of the sizes of subsets of BFC as candidate simplification measures. For notational convenience, SBFC indicates a measure computed as the sum of the contributions of a subset of FPA BFC. For instance, SBFC could indicate $M(\mathrm{EI}) + M(\mathrm{EO}) + M(\mathrm{EQ})$, or just $M(\mathrm{EI})$.

Conversely, it is not the goal of this paper to study whether it is possible to build multivariate regression models based on (subsets of) BFC for estimating software development effort, like some of the related papers in the literature instead do (see for instance [22,23]), and even though we studied the use of multivariate models in estimation elsewhere [24]. This would mean introducing a new layer of weights on top of FPA complexity weights.

The goals of our analyses are detailed in the following subsections, which refine the three goals mentioned in the introduction.

3.1.1. Verification of the role of BFC in functional size measures

In Section 4 we perform an initial investigation to corroborate the idea that sums of BFC size measures could be used as simplifications of UFP.

The influence of at least some SBFC on UFP is to be expected. For instance, it is quite clear that $M(\mathrm{EI}) + M(\mathrm{EO}) + M(\mathrm{EQ})$ is an important part of UFP. However, we would like to empirically check how accurately the various SBFC are correlated with UFP to decide which SBFC may be used as sensible simplifications. If, for instance, $M(\mathrm{EI})$ and $M(\mathrm{EI}) + M(\mathrm{EO}) + M(\mathrm{EQ})$ were correlated with UFP with the same accuracy, it would make sense to select $M(\mathrm{EI})$ as the better simplification because it would be less effort-consuming to compute $M(\mathrm{EI})$ than $M(\mathrm{EI}) + M(\mathrm{EO}) + M(\mathrm{EQ})$. This actually turns out to be the case in our study (see Section 4).

In addition, there are other, less obvious, reasons for the correlations found. For instance, ILF and EIF are bound to act as FTR in several transaction functions. Thus, even if ILF and EIF were not counted, they would anyway contribute to the size measure of transaction functions that use them (as they act as FTR and determine the weight of the transaction).

We do not strive to explain the nature and causes of the mentioned correlations here. The important point for our purposes is that size can be measured based on just a subset of the FPA BFC,

so this subset of BFC can be considered to build a true simplification of UFP. The existence of such models provides necessary support for the rest of the work.

3.1.2. Building predictive models of software development effort with accuracy comparable to that of UFP

After performing the check described above, we need to empirically investigate whether the sums of individual BFC that are good simplifications for UFP are also good effort predictors. Just because, say, *M*(EI) is well correlated with UFP and UFP is well correlated with software development effort does not mean that *M*(EI) too is well correlated with effort.

Note that it is *not* our goal to estimate the measure of size in UFP on the basis of a few BFC and *then* use the estimated size in UFP for estimating the development effort. Such a two-step prediction is exemplified in Formula (2), where we assume a COCOMO-like relationship between effort and size.

$$\begin{cases} \text{EstimatedUFP} = a \times \text{SBFC} + b \\ \text{Effort} = k \times \text{EstimatedUFP}^c \end{cases} \quad (2)$$

Estimating effort via the equations in Formula (2) would compose two prediction errors. This would result in errors greater than those of regression models based on SBFC (or UFP). The dangers of using two-step prediction procedures are one of the most important reasons why it is not advised that one use UFP (or FP) to predict LOC and then use the estimate of LOC to predict effort.

Instead, after showing that the sums of some BFC size measures are strongly correlated with functional size, our goal is to investigate the possibility of directly using the same sums in the prediction of effort, as in Formula (3)

$$\text{Effort} = a \times \text{SBFC}^b \quad (3)$$

This is done in Section 5 to address the second goal of the paper (building predictive models of software development effort with accuracy comparable to that of UFP). Throughout the paper, the term "accuracy" indicates the closeness of an effort estimate to the actual value. Actually, we are interested in evaluating the *difference* of accuracy of models that use different types of size measures to estimate the development effort. In general, the accuracy of an effort estimate depends on many factors (including the accuracy of the measures used in the estimation process). Since all the pairs of models being compared in this paper use data from the same repository, we can safely assume that the *difference* in accuracy of the models is not affected by the accuracy of the data. Therefore, accuracy is a property of the models, which summarize a relationship between size and effort. So, by comparing the accuracy of models based on different size measures, we actually compare the correlation to effort of these different size measures.

3.1.3. Building predictive models of software development effort, possibly in combination with measures of other requirements attributes, with better accuracy than FPA

We investigate whether SBFC may also be used in more sophisticated models that use additional factors that can affect the development effort and can be measured early in the software development lifecycle. Showing that these models turn out to have better accuracy than the effort prediction models obtained by using SBFC or UFP alone would support the usefulness of the simplified approach and the possibility of building effort prediction models that are usable in practice.

3.2. Research questions

The research questions addressed in the paper are thus the following ones:

RQ1 Is it possible to build a statistically significant model for the relationship between UFP and a *SBFC*, so as to suggest that *SBFC* can be used as a replacement of UFP?

RQ2 Are there any statistically significant models of Effort vs. *SBFC* whose residuals are not significantly greater than those of Effort vs. UFP models, so as to suggest that *SBFC* can be used as a replacement of UFP for effort estimation?

RQ3 Is it possible to build statistically significant software development effort models that use *SBFC* and other requirements measures, whose accuracy is sufficient to make them suitable for use in practice?

Table 1 summarizes the analyses that were carried out, what statistical techniques were used and in which sections they are described.

3.3. Statistical techniques

We applied a variety of statistical techniques in the analyses documented here, depending on their usefulness and appropriateness for the specific datasets we have used.

Specifically, we used Ordinary Least Square (OLS); whenever OLS linear regression could not be applied, we carried out a log–log transformation and checked whether the assumptions underlying OLS with these transformed variables are supported by evidence. When applying OLS linear regressions (with or without log–log transformations) we use Cook's distance [25] to identify outliers.

When the assumptions for OLS linear regression are not satisfied, even after the log–log transformation, we use a specific kind of robust regression based on the Least Median of Squares (LMS) [26], whose goal is to produce linear models that are not biased by overinfluential outliers and for which fewer and weaker assumptions need to be satisfied than for OLS linear regression. LMS regression has already been successfully used in several empirical software engineering studies (see for instance [27–30]).

We have also checked the existence of statistical dependence between variables by using so-called nonparametric association indicators such as Spearman's ρ and Kendall's τ. The presence of a nonparametric association does not necessarily imply that an accurate predictive model can be built. However, it is an indication that there is a statistical dependence between two variables, which is a necessary condition for building a statistically significant and accurate predictive model. In what follows, we use the term "correlation" when dealing with linear relationships such as the ones identifiable via OLS (with or without log–log transformation) or LMS. We use the term "association" to denote the (non-necessarily linear) nonparametric relationship between two variables.

In addition, to get further evidence that simplified size measures can be effectively used to estimate development effort, we used them also in estimations based on analogy criteria. To this

Table 1
Summary of the analyses reported in the paper.

Goal	Analysis performed	Statistical technique used	Paper section
RQ1	UFP vs. SBFC	Kendall's and Spearman's tests	4.2.1
RQ1	UFP vs. SBFC	LMS linear regression	4.2.2
RQ2	Effort vs. SBFC	Kendall's and Spearman's tests	5.1.1
RQ2	Effort vs. SBFC	log–log OLS regression	5.1.2
RQ2	Effort vs. SBFC	Analogy-based estimation	5.3
RQ3	Effort vs. SBFC and complexity	log–log OLS regression	6

end, projects were considered analogous when their sizes (evaluated with the proposed simplified measures) where similar.

To assess the accuracy of the regression models, we used a few goodness-of-fit indicators that are commonly found in the Empirical Software Engineering literature, namely R^2, the Mean Magnitude of Relative Error (MMRE), Pred(25), and the error range. When we carried out OLS linear regression, we used the usual R^2 defined for OLS linear regression, which we denote in this paper as R^2_{OLS}. When we carried out LMS linear regression, we evaluated the quality of the fit by R^2_{LMS}, which is defined in [30]. Even though it is better known and therefore more immediately interpretable than R^2_{LMS}, R^2_{OLS} may be safely used only if a model is obtained through OLS regression. If this is not the case, R^2_{OLS} is no longer guaranteed to range between 0 and 1, so it is difficult to interpret R^2_{OLS} as the proportion of explained variance of the dependent variable provided by a linear model.

Also, criticisms have been cast on the usefulness and meaning of MMRE and Pred(25), even though MMRE is often quoted as the *de facto* current accuracy indicator used in Empirical Software Engineering. The interested readers can refer to [31] for a discussion of these issues. At any rate, though we may somehow caution the readers about them, we have used MMRE and Pred(25) here to provide some additional pieces of information about the accuracy of models. Boxplots representing the distributions of relative residuals, or relative error ranges are also reported to complement MMRE and Pred(25).

We set a 0.05 statistical significance threshold throughout the paper, as is customary in Empirical Software Engineering studies. All the results reported in Sections 4–6 are characterized by p-value < 0.05, unless otherwise stated.

When building LMS regression models, the significance of the model was evaluated by applying the sign test to the difference of the absolute values of the residuals with respect to constant LMS regression and the absolute values of the residuals with respect to univariate LMS regression (the detailed definition of this test is in [30]). Summarizing, the sign test is used to verify if the univariate LMS regression provides a significantly better model than constant LMS regression, which is a model that, instead, would predict that the value of the dependent variable (say, UFP) is constant as the independent variable (say, a BFC) varies, i.e., it would predict that the dependent variable does not depend on the independent variable.

4. Functional sizing based on BFC

Release 11 of the ISBSG dataset [32] includes data from over five thousand projects. For each project, the following measures are reported, among others:

- year of project;
- count approach (IFPUG, COSMIC, NESMA, LOC, etc.);
- functional size in UFP, together with M(EI), M(EO), M(EQ), M(ILF), M(EIF);
- development effort in Person-Hours (mean = 5502, median = 1867, max = 645,694);
- development type (there are 1970 new developments and 2969 enhancements);
- miscellaneous information about the organization, application type, etc.;
- implementation language, hardware platform, and operating systems;
- several other types of information, not relevant for our purposes.

4.1. ISBSG data selection criteria

The ISBSG Repository contains a large amount of data on a very wide range of projects. For reasons that are discussed below, it is not possible to use the ISBSG data as a whole for our purposes. We selected samples of data to be used in the analysis as follows.

- The quality of the project data provided is classified on an ordinal scale. As in most studies that use the ISBSG data (see for instance [33–35]), we selected only the data points with the highest quality, namely those having Data Quality Rating and UFP Rating = 'A' or 'B'.
- We discarded all of the projects for which all of the BFC size measures were missing, since they are essential to our study. As for the specific individual BFC data, it is not uncommon that a project has no EIF or no EQ, i.e., that it does not rely on any read-only files or no inquiries are made to it. On the other hand, it would be quite unusual if even one among M(ILF), M(EI), or M(EO) were null, so project data with even one of those BFC size measures null are very likely to be erroneous. We therefore selected projects having M(ILF), M(EI), and M(EO) non-null.

As a result of the selection criteria described above, we obtained a sample of over 600 projects measured in FP.

4.2. UFP vs. BFC size measures

We now show the results of our association and correlation analyses on the possible relationships between SBFC and UFP and between the size measures of individual BFC.

4.2.1. Non-parametric analysis

The results are reported in Table 2, where Transaction Functions (TF) denote the set of all transactions and Data Functions (DF) denote the set of all data functions. Accordingly, M(TF) = M(EI) + M(EO) + M(EQ) and M(DF) = M(ILF) + M(EIF). It is easy to see that the measure of TF is extremely well associated with UFP. Also, M(EI) alone is very well associated with UFP. The association with M(TF) was expected – being M(TF) an important component of UFP – but our results show that the association is extremely strong. The association between M(EI) and UFP is a bit more surprising: it seems that even just one out of five BFC can provide the essential sizing information. Actually, there are also fairly strong associations between the BFC size measures, except for M(EIF), as shown in Table 3. There is also a fairly high association between M(TF) and M(DF): Spearman's rank correlation ρ is 0.643, Kendall's τ = 0.462.

4.2.2. Regression analysis

Even though there is a strong association between SBFC and UFP, OLS linear regression does not provide valid quantitative models of these relationships, as its assumptions are not satisfied, even after a log–log transformation. Therefore, we used LMS robust regression, and we obtained the following statistically valid and significant model:

Table 2
Associations between SBFC and UFP: non-parametric analysis.

Measure	Kendall's τ	Spearman's ρ
M(EI)	0.658	0.839
M(EO)	0.597	0.776
M(EQ)	0.528	0.692
M(ILF)	0.619	0.804
M(EIF)	0.264	0.363
M(TF)	0.828	0.953
M(DF)	0.635	0.824

Table 3
Associations between SBFC: non-parametric analysis.

	Kendall's τ				Spearman's ρ			
	M(EO)	M(EQ)	M(ILF)	M(EIF)	M(EO)	M(EQ)	M(ILF)	M(EIF)
M(EI)	0.438	0.448	0.449	0.072	0.593	0.602	0.616	0.101
M(EO)		0.288	0.417	0.194		0.408	0.587	0.268
M(EQ)			0.327	0.097			0.454	0.136
M(ILF)				0.195				0.266

$$\mathrm{UFP} = 28 + 1.137 \times M(\mathrm{TF}) \quad (4)$$

The quality of the fit, on the entire dataset, was evaluated by $R^2_{LMS} = 0.74$.

All the indicators of the precision of fit appear fairly good: MMRE = 19.7%, Pred(25) = 71%. The error range is −82% to 156%.

The analysis of the relationship between M(EI) and UFP carried out by using LMS regression provided the following model:

$$\mathrm{UFP} = 79 + 1.9 \times M(\mathrm{EI}) \quad (5)$$

The model has $R^2_{LMS} = 0.41$. The residuals are characterized by MMRE = 40.5%, Pred(25) = 38.4% and error range = −91% to 503%.

The distribution of relative residuals is reported in Fig. 1.

No other models could be found by using LMS, i.e., M(EI) and M(TF) are the only measures that support a statistically significant model for UFP.

4.3. Remarks on size analysis

The usage of LMS regression allowed us to derive statistically valid and significant models showing that UFP are linearly correlated with size measures of subsets of their BFC.

It is worthwhile noticing that, while previous studies suggested the existence of associations between SBFC and FP [36], the usage of LMS regression allowed us to derive *quantitative* models of the correlation linking SBFC and FP. To the best of our knowledge, no such models were ever proposed before.

These results suggest that, in principle, one could think of using the size measure – carried out according to standard FPA measurement process [21] – of a subset of the BFC as replacements of the size of the whole application in UFP. This hypothesis is tested in the rest of the paper.

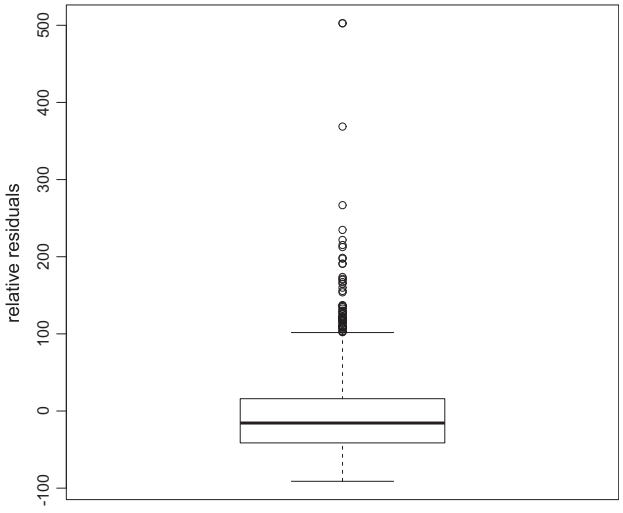

Fig. 1. ISBSG UFP vs. M(EI): boxplot of the relative residuals of the LMS regression.

5. Effort prediction based on BFC

Since effort is usually predicted on the basis of functional size, and we have shown that functional size expressed in UFP is correlated to the size of subsets of the BFC, it is reasonable to further investigate the idea that effort can also be estimated on the basis of a subset of the BFC size measures. To support this hypothesis, we built different types of the traditional Effort = f(functional size) models by analyzing the ISBSG dataset.

First of all, we need to notice that the ISBSG dataset contains data concerning both new developments and enhancements. When dealing with enhancements, some authors correlate the required effort with just the size of the enhancement, i.e., they build models of Effort = f (functional size of the enhancement). This choice may create a number of problems, since the enhancement effort in general depends not only on the size of the enhancement, but on the total size of the program that is being enhanced, because a few development activities involve the whole program, even when just a small fraction is modified. Since the ISBSG database reports only the size of changes, an accurate analysis of enhancements is not possible. Therefore, in the rest of the paper we deal exclusively with new development projects data from the ISBSG database.

It is also generally recognized that development effort depends on several other parameters, in addition to the size of the software to be developed. Actually, the ISBSG database includes several indications concerning variables that affect development effort and in principle could be useful to build effort models. Among these are: organization type, application type, type of architecture, development technique (life cycle), process maturity (e.g., CMM, CMMI, SPICE), development method, etc. However, the values of these variables are not given for the majority of the ISBSG projects: only 19.6% of the projects in the ISBSG dataset specify the application type, architecture and development techniques used. So, these variables cannot be included in the analysis: in fact, selecting the projects for which the mentioned measures are given would result in subsets too small to support statistically sound analyses.

Other potentially useful data – like the skills of developers or the complexity of the application – are not present in the ISBSG database.

To analyze reasonably homogeneous sets of data, we selected from the ISBSG dataset the sets of projects that were

- carried out not earlier than year 2000, to increase the probability that they were developed using similar technologies, methodologies and tools and maximize the usefulness of our results for effort estimation in future projects;
- implemented using the same "primary programming language": thus, we group projects that are more likely to belong to the same application areas and were supported by similar toolsets. Stratifying by language is a fairly consolidated practice in the FPA community. Previous research [37,38] found that the "primary programming language" used is an important element affecting the development effort.

Of course, the filtering described above does not guarantee that we get completely homogeneous sets of data, but it surely decreases the fraction of development effort due to unknown variables.

The filtering described above produced three sets of data: namely, the sets of projects written in Cobol, Java, and Visual Basic. Other sets were discarded because they did not contain enough projects to support reliable statistical analysis (only datasets containing at least 20 projects were considered in the analysis

reported in this Section). The sets obtained were then analyzed separately.

5.1. Predicting effort based on SBFC

5.1.1. Non-parametric evaluation of the association between SBFC and effort

The existence of associations between effort and SBFC is supported by the results of Spearman's and Kendall's indicators reported in Table 4.

The data in Table 4 indicate that several SBFC appear to be associated with effort as strongly as UFP is. It is noticeable that for Java projects even the measure of a simple BFC like EI appears to be better associated with effort than UFP is.

5.1.2. Models of the correlation between SBFC and effort

We built quantitative models of effort vs. size by adopting a widely used procedure in data analysis, i.e., we used OLS linear regression after log–log transformations, which led to equations of the kind provided by COCOMO [1], i.e., $Effort = a \times Size^b$. The log–log transformation is applied to "normalize" the data as suggested in [39].

The models are summarized in Table 5. Column a and b correspond to the coefficients in the model $Effort = a \times Size^b$.

To compare the models found, we studied relative residuals. When talking about effort prediction, the relative (i.e., percentage) residuals need to be considered, rather than plain residuals. For instance, a 2 Person-Month error is probably acceptable for a 2 Person-Year project, while it is surely not acceptable for a 4 Person-Month project: in the former case, the error is around 8%; in the latter case, it is 50%.

To evaluate the equivalence of models, Kitchenham et al. suggest the use of the paired t-test of absolute residuals [40]. This indication follows the proposal by Stensrud and Myrveit, who used paired t-tests of the MRE [41]. We agree with these indications in principle, but since in our case the usage of t-test is often prevented by non-normality of the distribution, we use the more robust Wilcoxon signed rank test and the Wilcoxon rank sum test (also known as Mann–Whitney test) [42].

Table 4
Associations between SBFC and effort for new developments: language-specific non-parametric analysis.

Language	n	Variable	Kendall's τ	Spearman's ρ
Cobol	20	M(EI)	0.537	0.704
Cobol	20	M(EO)	0.417	0.596
Cobol	20	M(EQ)	0.381	0.490
Cobol	20	M(ILF)	0.345	0.470
Cobol	20	M(EIF)	0.525	0.722
Cobol	20	M(TF)	0.589	0.743
Cobol	20	M(DF)	0.466	0.669
Cobol	20	UFP	0.621	0.777
Java	29	M(EI)	0.601	0.794
Java	29	M(EO)	0.400	0.520
Java	29	M(EQ)	0.372	0.517
Java	29	M(ILF)	0.499	0.659
Java	29	M(TF)	0.611	0.799
Java	29	M(DF)	0.463	0.633
Java	29	UFP	0.591	0.791
VB	56	M(EI)	0.505	0.651
VB	56	M(EO)	0.505	0.687
VB	56	M(EQ)	0.446	0.624
VB	56	M(ILF)	0.439	0.609
VB	56	M(TF)	0.558	0.742
VB	56	M(DF)	0.399	0.554
VB	56	UFP	0.561	0.728

We applied the Wilcoxon signed rank test and the Wilcoxon rank sum test to the distributions of absolute relative residuals: the results are reported in column "equiv." of Table 5. The tests indicate that, for all the considered implementation languages, we cannot reject the null hypothesis that the models based on M(EI), M(EO), or M(TF) are equivalent to the model based on UFP in terms of absolute relative residuals.

In practice, the analysis described above seems to indicate that using M(EI) (or M(EO), or M(TF)) instead of UFP to estimate the effort does not cause the accuracy of the estimate to decrease.

5.2. Exploration of alternative data grouping criteria

The analysis described above was carried out on data samples obtained by grouping project data by main programming language. Other criteria could be used to partition the dataset to obtain reasonably homogeneous samples. Other authors have grouped project data by application area (see for instance [34]). We tried to apply the same grouping concept, to check whether the findings described above also hold for data samples characterized according to different principles.

We found that only one application area, namely "Financial transaction process/accounting," occurs often enough to form a sufficiently large data sample (73 data points). For this dataset, we found only models with rather low R^2_{OLS} and low accuracy.

We also tried to form data samples that were homogeneous with respect to both the main programming language and the application type. In principle, these samples could be sufficiently homogeneous to provide models that are precise enough to be usable in practice. Unfortunately, even the largest obtained sample (Financial transaction process/accounting applications written in Visual Basic) was quite small (15 data points) and provided no significant model.

On the contrary, we performed the analysis of all new software developments carried out since year 2000 with no distinction with respect to the implementation language. As expected, the resulting models, based on a data subgroup including 536 data points, are worse than those reported in Table 5 as far as R^2, MMRE, and Pred (25) are concerned. The same models are equivalent to those reported in Table 5 as far as the accuracy of SBFC based models is concerned.

5.3. Effort estimation by analogy

To get further evidence that simplified size measures can be effectively used to estimate the development effort, we used them in estimations based on analogy criteria, rather than on statistical models. To this end, we used the same data subsets used in Section 5.1, i.e., a set of 20 Cobol applications, a set of 29 Java applications, and a set of 56 Visual Basic applications.

Estimation by analogy was carried out according to the following criteria. For each project, the K closest analog projects were selected among those implemented with the same programming language and developed before the considered project, but not more than ten years older. The mean productivity (size/effort) of the K selected analogs was computed. Finally, the estimated effort was computed as the size of the project to be estimated divided by the mean productivity of the analogs.

This process was applied for M(EI), M(EO), M(TF), M(DF) and UFP. M(EQ) and M(EIF) were excluded because several projects do not include any EQ or EIF, and establishing a similarity on the basis of the absence of a BFC does not seem to make much sense.

We also used different values of K, from 1 up to 15, so we could select the value of K that optimizes the estimates. Probably due to the fact that the ISBSG dataset contains different numbers of projects for each language, we got different values of K for each

Table 5
Language-specific log–log OLS regression models for SBFC vs. effort for new developments.

Lang.	Var.	b	a	R^2_{OLS}	Outl.	n	MMRE	Pred (25)	%err. range	Equiv.
COBOL	EI	0.904	96.8	0.516	0	20	120	15	−85 to 797	**
COBOL	EO	0.768	154.1	0.382	0	20	165	25	−88 to 1319	**
COBOL	EQ	1.009	73.6	0.702	1	13	66	23	−89 to 205	*
COBOL	ILF	0.719	184.2	0.289	0	20	171	5	−91 to 1303	**
COBOL	EIF	1.634	12.5	0.645	4	19	121	16	−96 to 599	*
COBOL	TF	0.964	27.4	0.582	0	20	118	15	−80 to 831	**
COBOL	DF	0.981	38.8	0.478	0	20	136	15	−90 to 1049	**
COBOL	UFP	1.057	9.6	0.591	0	20	119	0	−79 to 872	
Java	EI	0.614	232.2	0.686	8	29	50	38	−75 to 245	**
Java	EO	0.642	221.1	0.428	5	29	61	28	−94 to 316	**
Java	EQ	0.298	984.0	0.335	8	27	86	30	−88 to 731	*
Java	ILF	0.385	640.3	0.425	4	29	82	24	−82 to 923	x
Java	TF	0.592	151.2	0.614	4	29	58	41	−75 to 340	**
Java	DF	0.536	238.0	0.585	5	29	63	34	−83 to 503	x
Java	UFP	0.674	63.5	0.717	5	29	49	48	−77 to 284	
VB	EI	0.888	51.9	0.801	13	56	92	38	−97 to 2240	**
VB	EO	0.679	138.0	0.626	8	56	129	27	−89 to 3290	**
VB	EQ	0.632	202.2	0.593	7	54	137	30	−91 to 3826	*
VB	ILF	0.764	66.7	0.511	7	56	185	14	−94 to 6179	x
VB	TF	0.895	17.7	0.852	13	56	91	32	−93 to 2430	**
VB	DF	0.665	94.8	0.404	5	56	217	14	−94 to 6786	x
VB	UFP	1.033	4.6	0.821	12	56	130	34	−94 to 4117	

** = no evidence of non-equivalence according to both Wilcoxon and Mann–Whitney tests; * = no evidence of non-equivalence according to only one test; x = evidence that UFP model is better, according to at least one test.

language. Thus, for each language we selected the value of K that provides the most accurate estimate.

Table 6 reports the results of the analysis. For each SBFC and language, the values of MdMRE (the Median Magnitude of Relative Error), MMRE, Pred(25) and the relative error ranges are given, along with the number of projects that it was possible to estimate. The boxplots of relative errors obtained using each measure are illustrated in Fig. 2 (mean values are shown as diamonds). It is possible to see that estimates based on M(EI) do not appear substantially worse than those based on UFP. This is confirmed by both Wilcoxon signed rank test and the Wilcoxon rank sum test (Mann–Whitney test) [42]. Tests indicate that we cannot reject the null hypothesis that the residuals of estimates based on M(EI), M(EO), M(TF) or UFP are equivalent to those based on UFP in terms of relative residuals.

These results do not depend on the selected value of K, i.e., M(EI), M(EO), M(TF) and UFP appear equivalent (according to Wilcoxon and Mann–Whitney tests) for all values of K. Only M(DF) yields greater residuals for some values of K and some languages.

Table 6
Estimation by analogy using UFP and SBFC: accuracy indicators.

Language	Measure	K	MdMRE	MMRE	Pred (25)	N	%err. range
COBOL	M(EI)	5	67.5	195.6	21.3	94	−91 to 2224
COBOL	M(EO)	5	76.6	360	19.1	94	−88 to 7405
COBOL	M(TF)	5	54.9	180.9	27.7	94	−81 to 5502
COBOL	M(DF)	5	71.1	216.7	19.1	94	−82 to 6138
COBOL	UFP	5	54.7	176.3	26.6	94	−82 to 5293
Java	M(EI)	5	78.9	94.7	26.9	26	−78 to 260
Java	M(EO)	5	60	86.1	30.8	26	−95 to 389
Java	M(TF)	5	37.3	90.1	26.9	26	−56 to 616
Java	M(DF)	5	56.1	141.9	30.8	26	−80 to 678
Java	UFP	5	44.3	90.5	26.9	26	−76 to 469
VB	M(EI)	2	58.7	218.8	26.4	87	−98 to 4672
VB	M(EO)	2	52.1	196.4	28.7	87	−94 to 4686
VB	M(TF)	2	61	136.3	25.3	87	−95 to 2599
VB	M(DF)	2	69.4	212.6	17.2	87	−98 to 3156
VB	UFP	2	59	188.8	21.8	87	−92 to 4271

6. Towards models that use additional requirements measures

As already mentioned, the goal of this paper is not to produce readily usable models, but rather to show that simplified measures of functional size can be used in effort prediction models without loss in accuracy. Nevertheless, it is quite clear that defining models that are able to perform estimates with acceptable errors is fundamental for the practical usability of the models.

Of course, the acceptability of estimate accuracy is intrinsically subjective: for instance, a practitioner could be satisfied with MMRE = 20%, while another could require MMRE $\leqslant$ 15%. However, to stay on the safe side, we take COCOMO [1] as a reference. In this section, we derive models that base effort estimation on both simplified size measures and complexity measures: these models' accuracy is fairly good; actually it is better than COCOMO's, so we can consider these models usable in practice.

Since the definition of COCOMO, it is widely recognized that practically usable effort prediction models should consider not only the size of the application to be developed, but other factors that affect the required effort. Accordingly, we expect that statistically significant effort models based on a simplified measure of size *and* other parameters can have relatively low error levels.

To support this idea, we propose a model of effort based on two variables: a measure of functional size and a measure of the functional complexity of user requirements, along the lines described in [43,44]. As a complexity measure we used the density of Paths. The measure of Paths [45] is obtained by applying the principles of McCabe's complexity measure to the descriptions of elementary processes. Accordingly, the density of Paths is defined as the average number of Paths per transaction function. When M(EI) is used in place of UFP, Path density is redefined accordingly: only EI processes are considered.

The dataset used in this analysis is reported in Table 7. The data are about a set of small business projects, most of which were web applications. They were all new developments.

Since the dataset contains only 22 data points, it does not support conclusive evidence, but rather a sort of "proof of concept" of the viability of using simplified functional size measures in models meant to provide sufficiently precise estimates.

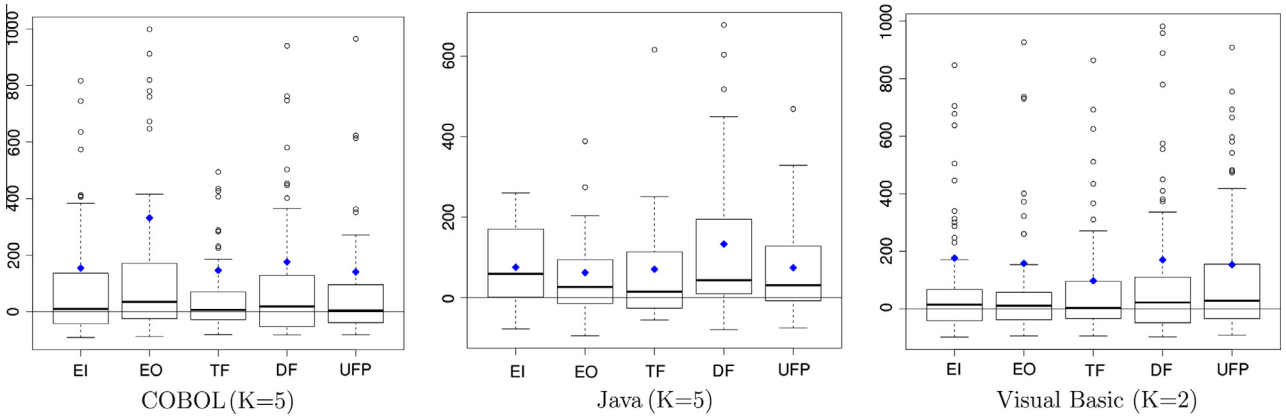

Fig. 2. Boxplots of relative residuals of estimates by analogy using different SBFC.

Table 7
Dataset used to derive models that use additional requirements measures.

Project ID	Actual effort	UFP	Num. Trans.	*M*(EI)	Num. EI	Paths
P1	410	185	39	78	24	71
P2	474	269	58	97	29	73
P3	382	171	19	35	11	60
P4	285	113	15	40	12	49
P5	328	110	14	45	9	34
P6	198	86	9	18	4	35
P7	442	75	10	21	6	50
P8	723	214	33	83	21	97
P9	392	340	47	79	17	83
P10	272	179	27	36	9	42
P11	131	115	17	24	6	18
P12	1042	168	26	63	20	118
P13	348	107	16	46	12	32
P14	243	111	12	36	10	68
P15	300	40	4	12	4	33
P16	147	59	10	31	10	20
P17	169	61	5	19	5	17
P18	121	72	13	41	13	21
P19	16,809	623	100	128	32	693
P20	5221	201	23	38	10	288
P21	342	41	27	24	8	24
P22	268	46	27	12	4	16

The models described in Table 8 were found. These results provide evidence that models based on simplified functional size measures and other relevant parameters can provide a level of precision that can be considered acceptable for practical effort prediction. In any case, the models reported in Table 8 can be possibly improved by considering other relevant factors, like the expertise of developers, the usage of tools, and the volatility of requirements, for instance.

7. Threats to validity

Like with any other correlational study, the threats to the validity of our study need to be assessed, along with the actions that have been undertaken to mitigate them.

7.1. Threats to internal validity

The limited size of the datasets used in some of the effort prediction models may be a first threat to internal validity. Despite the relatively small numbers of data points, we still filtered out outliers, to make sure that the results are not unduly influenced by a very small number of high-leverage points, even though this further reduced the cardinality of the samples. We also used non-parametric and robust techniques whenever the preconditions of parametric techniques were not supported by evidence to be on the safe side, even at the expense of sacrificing some statistical power. We also tried to identify homogeneous samples whenever possible. However, due to a variety of reasons, even a large dataset like ISBSG does not really support subsampling based on the values for several factors.

7.2. Threats to external validity

The ISBSG dataset contains a substantial number of projects in several application domains. Nevertheless, it may not be representative of the entire universe of software applications. We partitioned the dataset in subsamples according to the main language used in each project to obtain effort prediction models that may be more representative for each of those languages. At any rate, the relatively small size of some of the samples may make the models we found of limited external validity. We also tried to mitigate the potential threats due to the changes in the development practices over time by selecting only the projects from year 2000 onward. This should make the models we obtained more applicable to current projects.

7.3. Threats to construct validity

The first main construct validity threat is due to the inherent subjectivity of counting FSM methods. We relied on the ISBSG data, but only selected the projects with the two higher categories of data quality, as is usually done in research studies that use ISBSG data. Another threat may come from interpreting MMRE, MdMRE, and Pred(25) as accuracy indicators, as pointed out by criticisms in

Table 8
Models based on size and complexity.

Effort model	R^2	Outliers	MMRE	Pred (25)	Error range
$1.422\ UFP^{0.978}\ (Path/NumTrans)^{0.76}$	0.748	7/22	37	50	−80 to 119
$7.731\ (M(EI))^{0.826}\ (Path/NumEI)^{0.395}$	0.793	8/22	33	45	−91 to 66

the previous literature. At any rate, we provided other accuracy indicators to provide a more complete picture about the accuracy of our results.

8. Related work

8.1. Work concerning the relationship between the size of BFC and the size of applications

The number and selection of the BFC upon which FSM should be based have already received some attention in the literature. Even the very first definition of FP in [46] included 4 individual BFC instead of 5 and a different way of taking into account their importance when building a functional size measure.

Having to deal with many BFC is likely to involve both practical and theoretical problems. From a practical point of view, association or correlation between BFC implies that some aspects are measured twice, with a waste of measurement effort. From the theoretical point of view, it is not clear why one should include a BFC that appears to measure some property or dimension that is already measured by another BFC.

After completing an experiment involving data from 269 projects, Lokan reported evidence of such associations [36]. In another case study involving 40 projects, Kitchenham and Känsälä [18] reported the existence of associations among BFC size measures. Specifically, the statistical analysis found that the associations indicated in Table 9 were statistically significant (in terms of Kendall's tau). Kitchenham and Känsälä also observed that FP do not have the characteristics of a valid size metric, since some elements may be counted more than once.

A paper by Jeffery and Stathis [47] empirically analyzed FP. We mention here only the parts that are most related to our work. The study uses data from 17 projects, three of which were singled out as outliers in the statistical analysis, thereby reducing the number of data points used to build the statistical models to 14; however, the entire set of 17 projects was used in the evaluation of the accuracy of the models. Among other things, the study investigates whether BFC size measures are statistically independent. The authors' idea was that if BFC size measures were actually correlated, then this would suggest that a simplified form of FP sizing would be possible across different domains, as the paper references the work by Kitchenham and Känsälä as a starting point, with data coming from a different domain. The statistical analysis found that the correlations (based on R^2_{OLS}) and the associations (in terms of Kendall's tau) illustrated in Table 10 were statistically significant. In addition, the authors found statistically significant associations and correlations between UFP and EI, EQ, ILF.

Another observed weakness of taking into account several BFC is that the variability of Function Point counting may increase. Kemerer investigated this aspect and reported that the differences in Function Point measures of the same system proveded by different counters averaged 12.2% [11]. Jeffery et al. reported even worse figures: a 30% variance within an organization, which rose to more than 30% across organizations [12]. More recent studies confirmed – at least qualitatively, if not quantitatively – the results reported by Kemerer and Jeffery. Rule [13] found that experienced software measurement specialists obtained better results than software engineers who had little measurement training. The intercounter consistency of specialists was around 5%, while that obtained by recently trained project staff was typically around 23%. In a similar study, Cuadrado-Gallego et al. [14] employed 77 trained undergraduate students in the measurement of a real world application using the IFPUG method. The obtained measures ranged from a minimum value of 57 FP to a maximum of 104 FP. The standard deviation reported was ca. 14% of the mean value.

Kitchenham and Känsälä expect that simpler counting would reduce the variability of the results [18].

The simplification of FSM methods has been the objective of several other proposals.

Symons defined Mark II Function Points [6] on the basis of only three BFC: data element types, entity-type references, and output data element types. The classification of BFC as simple, average, or complex was eliminated, because considered straightforward, but also oversimplified. Although the main characteristic of Mark II Function Points is the inclusion of a measure of complexity, Symons pointed out that the simplicity of the Mark II approach in having fewer BFC than Albrecht's method has a number of advantages, such as greater ease of calibration against measurements or estimates.

Our results concerning the associations among BFC size measures and of BFC size measures with UFP substantially agree with those mentioned above. The results of previous work were reached by analyzing datasets (like the one maintained by ISBSG) by means of nonparametric tests (typically tests on Spearman's ρ and Kendall's τ). However, no models could be derived by using Ordinary Least Squares (OLS) linear regression (not even after log–log transformations). Under this respect, our results are stronger, in that we were able to derive – via LMS linear regression – models that quantitatively represent the correlation between BFC size measures and UFP.

8.2. Work concerning the relationship between BFC and effort

Kitchenham and Känsälä also pointed out that an effort prediction model built with stepwise linear regression based on two BFC (input Function Points and output Function Points) was just as good as the effort model based on "complete" Function Points, and an effort prediction model based on the raw (i.e., not complexity-weighted) counts of files and outputs was only slightly worse than an effort model based on Function Points [18].

One of the earliest attempts to build effort prediction models based on BFC measures or even finer-grained measures is documented in [23]. The statistical study uses a dataset composed of 21 mainframe-based projects of a major Canadian financial organization. The projects were classified as major enhancements to existing transaction-based software applications. Several kinds of statistical effort prediction models were built, based on OLS regression, including models based on so-called "primary components," i.e., DET and RET and a multivariate model whose independent variables are the measures of the five BFC of FPA.

With respect to the work reported in [23], we are able to provide more reliable results, since our dataset is larger and we

Table 9
Associations between SBFC according to [18].

	M(EO)	*M*(EQ)	*M*(ILF)	*M*(EIF)
M(EI)	✓	✓	✓	
M(EO)		✓	✓	✓
M(EQ)			✓	

Table 10
Associations between SBFC according to [47].

	M(EO)	*M*(EQ)	*M*(ILF)	*M*(EIF)
M(EI)		$R^2\tau$	$R^2\tau$	τ
M(EO)				
M(EQ)			$R^2\tau$	τ

τ indicates an association (in terms of Kendall's tau); R^2 indicates a correlation according to R^2_{OLS}.

employed statistical tests such as Wilcoxon and Mann–Whitney tests. Moreover, we derived also non-linear models, while in [23] only linear models were studied.

In the already mentioned paper [47], Jeffery and Stathis empirically investigated whether the individual BFC could be used as predictors for effort, and found that *M*(EI), *M*(EQ), and *M*(ILF) are correlated with development effort. Multivariate stepwise linear regression with the individual BFC size measures as independent variables was also investigated, but did not provide a model. Finally, they investigated whether the raw (i.e., not complexity-weighted) counts of files, inputs, outputs and inquiries could be used for effort prediction and found that the total count of files, inputs, outputs and inquiries were significant predictors.

Abran et al. studied the relationship between effort and single BFC size measures by using release 8 of the ISBSG dataset [48]. They grouped projects by programming language, but they do not seem to have separated data points representing new developments from those representing enhancements. With respect to the work reported in [48], we are able to provide more complete and reliable results, since we evaluated results in terms of estimation accuracy and employed statistical tests such as Wilcoxon and Mann–Whitney tests to compare the significance of models. Moreover, we derived also non-linear models, while in [48] only linear models were studied.

A few articles report studies concerning the relationship between development effort and COSMIC BFC [49,22,50]. These articles tend to confirm that effort estimation can be carried out on the basis of a single BFC, although a few limitations of the performed analyses – namely, the usage of relatively small datasets and the fact that no accuracy comparison based on statistical tests were performed – suggest that further analysis is necessary to get reliable evidence that also the definition of COSMIC Function Point can be simplified.

With respect to the research work mentioned in this section, our work provides more reliable conclusions in that (a) we used a large public dataset, and rigorously selected the data sample to be used for the analyses; (b) we used data from multiple application areas, so our results are expected to apply to a fairly wide range of software applications; (c) we used samples that are sufficiently homogeneous, thus avoiding as far as possible the influence of variables not considered in models; (d) we used sound statistical methods, including statistical tests to compare the precision of UFP based models with respect to BFC based models; (e) we used multiple types of models (e.g., LMS and OLS after log–log transformation); and (f) we showed that the simplified measures can be used in conjunction with measures of other attributes of user requirements to build models that are precise enough to be used in practice.

8.3. Work concerning the relationship between the distribution of BFC sizes and effort

Abran et al. suggested that development effort can depend on the distribution of BFC sizes [48]. Therefore, we studied the distribution of BFC size measures in our dataset. Table 11 shows the actual distributions of the average contributions of the BFC to UFP.

Table 11
Distribution of SBFC.

	M(EI)%	*M*(EO)%	*M*(EQ)%	*M*(ILF)%	*M*(EIF)%	Stdev. (%)
COBOL	21.6	23.9	15.2	29.2	10.2	7.4
Java	22.8	17.5	19.0	27.3	13.4	5.3
VB	20.0	21.0	18.8	28.1	12.0	5.7

The distributions reported in Table 11 is very similar to those reported in [48]: for instance, COBOL projects are characterized by similar *M*(EI) and *M*(EO) percentages, and by low *M*(EQ) and *M*(EIF) percentages. There does not seem to be any extremely "dominating" or "dominated" BFC.

Interestingly enough, the BFC that provide the largest contributions are not necessarily the ones whose sizes are best associated or correlated with UFP. For instance, ILF are the most frequent BFC, but they are not the BFC whose size are best associated or correlated with UFP or the development effort. So, even though an association or correlation can be expected, as we already mentioned, it is not due to the fact that the best associated or correlated BFC accounts for the absolute or even the relative majority of UFP. From a practical point of view, this implies that FSM measurers may count a reasonably small percentage of BFC without sacrificing the accuracy of predictions.

8.4. Work dealing with the simplification of the measurement process

While we explore the possibility of simplifying the measure of functional size by dropping most BFC, other researchers have explored the possibility of simplifying *the measurement process*, while retaining the original definition of the measure – that is, including all the original BFC.

The most well-known of these approaches – which are somehow orthogonal to ours – is probably the Early & Quick Function Points (EQFP) [51]. EQFP starts from the consideration that estimates are sometimes needed before the analysis of requirements is completed, when the information on the software to be developed is incomplete or not sufficiently detailed. Therefore, the EQFP measurement process considers elements that are coarser-grained than the FPA BFC, and leads to an approximate measure of size in IFPUG FP. Since several details for performing a correct measurement following the rules of the IFPUG manual (as described in [21]) are not considered, the result is a less precise measure. Reduced measurement time and costs are also a reason for adopting the method when full specifications are available, but there is the need of completing the measurement in a short time, or at little cost. An advantage of the method is that different parts of the system can be measured at different detail levels; so, for instance, a part of the system can be measured following the IFPUG manual rules, while other parts can be measured on the base of coarser-grained information.

Other methods have been proposed by NESMA [7] to simplify the process of counting FP. The Indicative NESMA method simplifies the process by only requiring the identification of logic data from a data model; the Function Point size is then computed by applying predefined weights, whose value depends on whether or not the data model is normalized in 3rd normal form:

- Nonnormalized model: Function Points = 35 × Number of ILF + 15 × Number of EIF
- Normalized model: Function Points = 25 × Number of ILF + 10 × Number of EIF

The Indicative NESMA method is quite rough in its computation: the official NESMA counting manual specifies that errors in functional size with this approach can be up to 50%.

The Estimated NESMA method requires the identification of each BFC, but does not require the assessment of the complexity of each component: Data Functions (ILF and EIF) are assumed to be of low complexity, while Transactions Functions (EI, EQ, and EO) are assumed to be of average complexity.

Another noticeable approach was proposed by Heričko and Živkovič [52]: they deal with the need of size estimates based on descriptions having variable levels of abstraction through an

iterative development lifecycle. To this end, they use the OOFP2 measurement method [53]. The core of the proposal is that, throughout the lifecycle, the application is represented at different abstraction levels, via different UML diagrams (e.g., use case diagrams at the topmost level, sequence an activity diagrams at an intermediate level, and class diagrams at the detailed level), and FP counting procedures are defined for different combinations of UML diagrams. The approach proposed by Heričko and Živkovič is in some sense complementary to ours, since they consider a software application's description at varying abstraction levels, while we consider only the level of abstraction required by the IFPUG, but limiting the scope of the measurement to a subset of the BFC.

The methods mentioned above are not meant to replace FP, but rather to provide alternative measurement processes under specific conditions. On the contrary, our analysis might bring to a complete replacement of FP. This of course would imply that also the measurement process is simplified (under any conditions), since just a subset of the full original measurement processes would be required.

An important characteristic of approaches involving the simplification of the measurement process is that, when used in the effort estimation process, they force the usage of the two-step procedure mentioned in Section 3.1.2: first you estimate the size, than you estimate effort on the basis of the estimated size, thus combining estimation and resulting in an increased error with respect to the estimates based on the "real" size. On the contrary, our approach does not cause any statistically significant loss in software development effort prediction accuracy.

8.5. Other relevant work

Gencel and Demirors critically reviewed the FSM methods [54]. They state that there exist significant improvement opportunities for FSM, both theoretically and practically. On the theoretical side, they mention a few open issues, including the additivity of the functional sizes of different BFC types, and the definition of formulas to transform different BFC types to a single type so that they can be added up.

Our proposal does not provide solutions to the problems highlighted by Gencel and Demirors, rather it directly eliminates some problems: for instance, using just one BFC to represent the functional size makes the additivity issue irrelevant.

9. Conclusions and future work

9.1. Results

The analyses reported above allow us to provide answers to the research questions stated in Section 3.2:

RQ1: Is it possible to build a statistically significant estimation model for UFP based on a SBFC, so as to suggest that SBFC can be used as a replacement of UFP?
Yes. Most SBFC appear correlated with UFP. This result is supported by both non-parametric analysis (see Table 2) and – as far as EI and TF are concerned – by linear LMS regression analysis (see the models described by Eqs. (4) and (5)).

RQ2: Are there any statistically significant models of SBFC vs. Effort that feature residuals not significantly greater than those of UFP vs. Effort models?
Yes. M(EI), M(EO), and M(TF) support models of Effort whose residuals are not significantly greater than the model based on UFP (see Table 5).

RQ3: Is it possible to build statistically significant software development effort models that use SBFC and other requirements measures, and that feature sufficient precision to make them suitable for use in practice?
Yes, as a "proof of concept." We built models of the type $Effort = a \times SBFC^b \times MC^c$, where MC is a measure of the processing complexity derived from requirements (see Table 8). The models found feature an error level that is (almost) low enough to make the models usable for effort prediction in real development environments.

9.2. Consequences

The analyses performed indicate that measuring UFP as specified by FPA does not seem necessary, since M(EI) – as well as M(EO) and M(TF) – appear as good as UFP for predicting effort. This conclusion appears reliable, in that it applies to all languages that appear frequently enough in the ISBSG dataset.

This observation has relevant implications with respect to the issues mentioned in Section 1.

9.2.1. Simplification of the measurement process

Simplifying the definition of functional measures implies simplifying the measurement process. In fact, according to the IFPUG manual [21], the core of the counting process involves the following activities:

1. Identify the ILFs and EIFs.
2. Determine the ILF complexity and their contribution to the unadjusted function point count.
3. Determine the EIF complexity and their contribution to the unadjusted function point count.
4. Identify the elementary processes.
5. Determine the primary intent of the identified elementary processes, and classify it as an EI, EO, or EQ.
6. Validate elementary processes against the transaction identification rules.
7. Determine EI complexity and their contribution to the unadjusted function point count.
8. Determine EO complexity and their contribution to the unadjusted function point count.
9. Determine EQ complexity and their contribution to the unadjusted function point count.

So, while measuring UFP requires performing *all* of the activities listed above, measuring just M(EI) requires performing a subset of such activities, namely activities 1, 4, 5, 6 (limitedly to EI) and 7. Each activity performed to measure M(EI) involves the same work needed for IFPUG UFP measurment. Therefore, we can safely conclude that measuring M(EI) generally involves a reduction in the duration, effort and cost of measurement with respect to UFP measurement.

This is very good news for organizations that need to collect historical data to build their own effort prediction models. Organizations that already own historical data concerning the development effort and the size of completed projects (i.e., UFP, M(EI), M(EO), etc.) can switch to the simplified versions of the size measures while preserving the validity of the whole set of historical data. In fact, they can simply ignore the data concerning BFC that do not appear relevant. This is actually a big advantage: with our approach, new effort models can be derived almost immediately using the available data.

9.2.2. Less subjective measures

Concerning subjectivity, let us consider the following – quite typical – situation. Requirements specifications mention some data

(let's call them 'A') in a way that leaves space for interpreting the requirements by an IFPUG FP measurer. Thus, it can easily happen that a measurer decides that these data are a single ILF (A) with two RET, while another measurer decides that there are two ILF (A1 and A2) with one RET each. If we suppose that the total number of DET of A (or the sum of DET of A1 plus the DET of A2) is 25, then we have that measurer 1 counts A as an average complexity ILF (10 UFP), while measurer 2 counts two low complexity ILF (14 UFP). The subjective interpretation of the requirements leads to differences in the measures.

The subjectivity of the measure of ILF (and of EIF too) also affects the measure of a transaction. For instance, if an EI updating the data mentioned above involves X FTR according to measurer 1, it will involve $X + 1$ FTR according to measurer 2. It is thus possible that the two measurers provide different measures for the same EI, again due to the subjective interpretations of the ILF involved in the transaction: if measurer 1 identifies two FTR (including A) and 10 DET exchanged through the application boundaries (an average complexity EI: 4 UFP), measurer 2 will identify three FTR (including A1 and A2) and 10 DET (a high complexity EI: 6 UFP). The same type of effect applies to EO and EQ transactions that involve data A.

The consequence of the observations reported above is that:

- When measurers 1 and 2 count the UFP of the application, they will get differences that are due to subjective evaluations in M(ILF), M(EI), M(EO), etc. All these differences sum up: the measure provided by measurer 2 will be greater than the measure provided by measurer 1 because of the subjectively increased size of ILF, EI, EO, etc.
- When measurers 1 and 2 measure the size of the application based only on EI, they get differences due to subjective interpretations only for M(EI). Thus, the difference between the two measures will be smaller, though not null.

We can conclude that, although the simplified measure proposed in the paper is not free from subjectivity, it is – by construction – less affected by subjectivity than measures involving multiple BFC.

9.2.3. Theoretical implications

A few authors (see for instance [18,36]) noted that the measures of some BFC are inter-correlated. Our analyses confirmed previous results concerning inter-correlations (see Table 3). By means of LMS analysis it is possible to quantify the inter-correlations: for the ISBSG dataset we found a statistically significant linear correlation between the measures of EO and EI: $M(\text{EO}) = 24.5 + 0.7\ M(\text{EI})$.

These findings cast serious doubts on the well-formedness of Function Points as a measure, as inter-correlation among BFC measures imply that some aspects are taken into account multiple times in the size measure. According to our analysis, including both M(EI) and M(EO) in the measure of size involves counting M(EI) 1.7 times. From the theoretical point of view, measuring some property or dimension that is (maybe partially) already measured – by taking into account other BFC – makes the resulting measure unstable.

Basing size measures on a single BFC completely removes this type of problems.

9.3. Future work

A possible evolution of the work reported above consists in repeating the analysis described in Section 6 with a larger dataset, to gain a better confidence in the possibility of building models that are based on a simplified definition of size and on a relatively sophisticated measure of complexity.

We shall extend our analysis to involve additional parameters that could explain *why* EI can be used as a replacement of UFP, or why for COBOL projects EQ appears to correlate to effort better – as far as R^2_{OLS} is concerned – than both EI and UFP (see Table 5).

Finally, we intend to extend the analysis reported above to COSMIC Function Points.

Acknowledgments

The work has been partially supported by Project "Metodi, tecniche e strumenti per l'analisi, l'implementazione e la valutazione di sistemi software" funded by the Università degli Studi dell'Insubria and Research Fund of School of Engineering of Austral University.

References

[1] B.W. Boehm, Software Engineering Economics, Prentice-Hall, 1981.
[2] G. Finnie, G. Wittig, J. Desharnais, A comparison of software effort estimation techniques: using function points with neural networks, case-based reasoning and regression models, Journal of Systems and Software 39 (3) (1997) 281–289.
[3] R. Jeffery, M. Ruhe, I. Wieczorek, Using public domain metrics to estimate software development effort, in: Software Metrics Symposium, 2001. METRICS 2001. Proceedings. Seventh International, IEEE, 2001, pp. 16–27.
[4] L. Buglione, C. Ebert, Estimation tools and techniques, Software, IEEE 28 (3) (2011) 91–94.
[5] A.J. Albrecht, J.E. Gaffney, Software function, lines of code and development effort prediction: a software science validation, IEEE Transactions on Software Engineering 9 (6) (1983) 639–647.
[6] C. Symons, Function point analysis: difficulties and improvements, IEEE Transactions on Software Engineering 14 (1988) 2–11.
[7] I. ISO, IEC 24570: 2004, Software Engineering-NESMA Functional Size Measurement Method Version 2.1 – Definitions and Counting Guidelines for the Application of Function Point Analysis, International Organization for Standardization, Geneva.
[8] Finnish Software Measurement Association, FiSMA FSM Method 1.1 (2004).
[9] COSMIC – Common Software Measurement International Consortium, The COSMIC Functional Size Measurement Method – Version 3.0 Measurement Manual (The COSMIC Implementation Guide for ISO/IEC 19761: 2003) (2007).
[10] Total Metrics, Methods for Software Sizing How to Decide which Method to Use (2007).
[11] C.F. Kemerer, Reliability of function points measurement: a field experiment, Communications of the ACM 36 (2) (1993) 85–97.
[12] J. Jeffery, G. Low, M. Barnes, Comparison of function point counting techniques, IEEE Transactions on Software Engineering 19 (5) (1993) 529–532.
[13] P. Rule, The importance of the size of software requirements, in: Software Measurement Services, NASSCOM Conference, India, 2001.
[14] J. Cuadrado-Gallego, L. Buglione, M. de Sevilla, P. Rodríguez-Soria, M. Dominguez, Horizontal dispersion of software functional size with IFPUG and cosmic units, Advances in Engineering Software 41 (2) (2010) 262–269.
[15] O. Mendes, A. Abran, P. Bourque, Function Point Tool Market Survey, Tech. Rep., Université du Québec a Montréal, Montreal, 1996.
[16] A. Stambollian, A. Abran, Survey of automation tools supporting COSMIC-FFP ISO 19761, in: Proceedings of the 16th IWSM-Metrikom 2006, Postdam, Germany, 2006.
[17] D. Kempisty, M. Harris, Is automated function point counting useful yet? in: United Kingdom Software Metrics Association Conference – UKSMA, 2009.
[18] B. Kitchenham, K. Känsälä, Inter-item correlations among function points, in: Proceedings of the 15th International Conference on Software Engineering, ICSE '93, IEEE Computer Society Press, Los Alamitos, CA, USA, 1993, pp. 477–480.
[19] ISO/IEC, ISO/IEC 24750:2005, Software Engineering NESMA Functional Size Measurement Method, Version 2.1, Definitions and Counting Guidelines for the Application of Function Point Analysis (2005).
[20] L. Santillo, M. Conte, R. Meli, Early and quick function point: sizing more with less, in: Software Metrics, 2005. 11th IEEE International Symposium, Como, 19–22 September 2005.
[21] ISO, ISO/IEC 20926: 2003, Software Engineering – IFPUG 4.1 Unadjusted Functional Size Measurement Method – Counting Practices Manual (2003).
[22] L. Buglione, C. Gencel, Impact of base functional component types on software functional size based effort estimation, Product-Focused Software Process Improvement (2008) 75–89.
[23] A. Abran, P.N. Robillard, Function points analysis: an empirical study of its measurement processes, IEEE Transactions on Software Engineering 22 (1996) 895–910.
[24] S. Morasca, On the use of weighted sums in the definition of measures international conference on software engineering, in: 2010 ICSE: Workshop on Emerging Trends in Software Metrics, 2010, pp. 8–15.
[25] R.D. Cook, S. Weisberg, Residuals and Influence in Regression, Chapman and Hall, London, 1982.

[26] P.J. Rousseeuw, A.M. Leroy, Robust Regression and Outlier Detection, John Wiley & Sons, Inc., New York, NY, USA, 1987.
[27] S.G. MacDonell, Establishing relationships between specification size and software process effort in CASE environments, Information and Software Technology 39 (1) (1997) 35–45.
[28] B. Kitchenham, S. MacDonell, L. Pickard, M. Shepperd, Assessing Prediction Systems, Technical report, University of Otago, 1999.
[29] L. Baresi, S. Morasca, Three empirical studies on estimating the design effort of web applications, ACM Transactions on Software Engineering and Methodology 16 (4) (2007) 15.
[30] S. Morasca, Building statistically significant robust regression models in empirical software engineering, in: PROMISE '09: Proceedings of the 5th International Conference on Predictor Models in Software Engineering, ACM, New York, NY, USA, 2009, pp. 1–10.
[31] B. Kitchenham, L. Pickard, S. MacDonell, M. Shepperd, What accuracy statistics really measure [software estimation], in: Software, IEE Proceedings, vol. 148, IET, 2001, pp. 81–85.
[32] International Software Benchmarking Standards Group, Worldwide Software Development: The Benchmark, Release 11 (2009).
[33] E. Mendes, C. Lokan, R. Harrison, C. Triggs, A replicated comparison of cross-company and within-company effort estimation models using the ISBSG database, in: Software Metrics, 2005. 11th IEEE International Symposium, IEEE, 2005, pp. 10.
[34] C. Gencel, L. Buglione, Do base functional component types affect the relationship between software functional size and effort?, Software Process and Product Measurement (2008) 72–85
[35] P. Pendharkar, J. Rodger, The relationship between software development team size and software development cost, Communications of the ACM 52 (1) (2009) 141–144.
[36] C.J. Lokan, An empirical study of the correlations between function point elements, IEEE International Symposium on Software Metrics 0 (1999) 200.
[37] C. Symons, The Performance of Real-Time, Business Application and Component Software Projects: An Analysis of COSMIC-Measured Projects in the ISBSG Database, Tech. Rep., ISBSG, 2009.
[38] International Software Benchmarking Standards Group, Practical Software Project Estimation: A Toolkit for Estimating Software Development Effort & Duration, McGraw-Hill, 2011.
[39] B. Kitchenham, E. Mendes, Why comparative effort prediction studies may be invalid, in: Proceedings of the 5th International Conference on Predictor Models in Software Engineering, ACM, 2009, p. 4.
[40] B. Kitchenham, S. Pfleeger, B. McColl, S. Eagan, An empirical study of maintenance and development accuracy, Journal of Systems and Software 64 (1) (2002) 57–77.
[41] E. Stensrud, I. Myrtveit, Human performance estimating with analogy and regression models: an empirical validation, in: Proceedings of the 5th International Symposium on Software Metrics, 1998.
[42] D. Sheskin, Handbook of Parametric and Nonparametric Statistical Procedures, third ed., Chapman & Hall/CRC, 2004.
[43] L. Lavazza, G. Robiolo, The role of the measure of functional complexity in effort estimation, in: Proceedings of the 5th International Conference on Predictor Models in Software Engineering – PROMISE 2010, Timisoara, 2010.
[44] L. Lavazza, G. Robiolo, Introducing the evaluation of complexity in functional size measurement: a UML-based approach, in: Proceedings of the 4th International Symposium on Empirical Software Engineering and Measurement – ESEM 2010, Bozen, 2010.
[45] G. Robiolo, R. Orosco, Employing use cases to early estimate effort with simpler metrics, Innovations in Systems and Software Engineering 4 (1) (2008) 31–43.
[46] A. Albrecht, Measuring application development productivity, in: I.B.M. Press (Ed.), IBM Application Development Symp., 1979, pp. 83–92.
[47] R. Jeffery, J. Stathis, Function point sizing: structure, validity and applicability, Journal of Empirical Software Engineering 1 (1) (1996) 11–30.
[48] A. Abran, B. Gil, É. Lefebvre, Estimation models based on functional profiles, in: International Workshop on Software Measurement–IWSM/MetriKon, Kronisburg, Germany, 2004, pp. 195–211.
[49] C. Gencel, L. Buglione, Do different functionality types affect the relationship between software functional size and effort, in: Proceedings of the Int. Conf. on Software Process and Product Measurement (IWSM-MENSURA 2007), Palma de Mallorca, 2007, pp. 235–246.
[50] L. Buglione, F. Ferrucci, C. Gencel, C. Gravino, F. Sarro, Which COSMIC Base Functional Components are Significant in Estimating Web Application Development? A Case Study, in: 20th International Workshop on Software Measurement (IWSM)/Metrikon/MENSURA Joint Conference, Shaker Verlag, 2010.
[51] M. Conte, T. Iorio, R. Meli, L. Santillo, E&Q: An Early & Quick Approach to Functional Size Measurement Methods, in: Software Measurement European Forum, 2004.
[52] M. Heričko, A. Živkovič, The size and effort estimates in iterative development, Information and Software Technology 50 (7) (2008) 772–781.
[53] A. Živkovič, I. Rozman, M. Heričko, Automated software size estimation based on function points using UML models, Information and Software Technology 47 (13) (2005) 881–890.
[54] C. Gencel, O. Demirors, Functional size measurement revisited, ACM Transactions on Software Engineering and Methodology 17 (3).