The Journal of Systems & Software 174 (2021) 110888

Contents lists available at ScienceDirect

The Journal of Systems & Software

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

Early and quick function points analysis: Evaluations and proposals ☆

Geng Liu [a], Luigi Lavazza [b,*]

[a] Hangzhou Dianzi University, Hangzhou, China
[b] Università degli Studi dell'Insubria, Varese, Italy

ARTICLE INFO

Article history:
Received 3 February 2020
Received in revised form 5 November 2020
Accepted 18 December 2020
Available online 23 December 2020

Keywords:
Function Points
Functional Size Measurement
NESMA Estimated
Early size estimation
Function point analysis
High-level FPA

ABSTRACT

Measuring Function Points following the standard process is sometimes long and expensive. To solve this problem, several early estimation methods have been proposed. Among these, the "NESMA Estimated" method is one of the most widely used; it has also been selected by the International Function Point User Group as the official early function point analysis method, under the name of 'High-level FPA' method. A large-scale empirical study has shown that the High-level FPA method – although sufficiently accurate – tends to underestimate the size of software. Underestimating the size of the software to be developed can easily lead to wrong decisions, which can even result in project failure. In this paper we investigate the reasons why the High-level FPA method tends to underestimate. We also explore how to improve the method to make it more accurate. Finally, we propose size estimation models built using different criteria and we evaluate the estimation accuracy of these new models. Our results show that it is possible to derive size estimation models from historical data using simple regression techniques: these models are slightly less accurate than those delivered by the High-level FPA method in terms of absolute estimation errors, but can be used earlier than the High-level FPA method, are cheaper, and do not underestimate software size.

1. Introduction

Function Points (FP) were introduced by Albrecht at IBM in the late seventies (Albrecht, 1979), to measure functional requirements of software. Albrecht aimed at defining a measure that was correlated to the value of software, and could be useful to assess the cost of developing software applications.

Today, Function Point Analysis (FPA) is still widely used (Jones, 2017; Georgi and Vogt, 2008), since it is a practical method for measuring the size of a software application in the very early stages of a project, often before actual development starts. Accordingly, software size measures expressed in FP are often used for cost estimation.

The success of Function Point Analysis led to the creation of the International Function Points User Group (IFPUG), an association that keeps FPA up to date with respect to the evolving software technologies, publishes the official FP counting manual (International Function Point Users Group (IFPUG), 2010), and certifies professional FP counters. FPA was also recognized as an international standard by the ISO (International Standardization Organization (ISO), 2003). Function Points are also the subject of many research activities (de Freitas Junior et al., 2015).

In some conditions, performing the standard FP measurement process may be too long and expensive, with respect to developers' needs. In fact, standard FP measurement can be performed only after completing the software requirements elicitation stage, while functional measures could be needed earlier, before functional requirements have been elicited completely and at the required detail level.

Therefore, many methods were invented and used to provide *estimates* of functional size measures based on less or coarser grained information than required by standard FPA. Among these early estimation methods, the most widely known and used method is the "NESMA estimated" method (van Heeringen et al., 2009), which was adopted by IFPUG as the official early function point analysis methods (Timp, 2015), under the name of 'High-level FPA' (HLFPA) method.

A large-scale empirical study (Lavazza and Liu, 2019) has shown that the HLFPA method – although sufficiently accurate for early and quick estimation of functional size – tends to underestimate the size of software. Underestimating the size of the software to be developed can easily lead to wrong decisions, which can increase the risk of project failure. This risk is particularly relevant because early estimation methods are used also as a replacement of the standard FPA process, not only when the requirements are not fully specified, but also when the analysis and specification phase is complete, and actual development have to be planned and organized. In this case, the motivation for

☆ Editor: [DANIELA DAMIAN].

* Corresponding author.
E-mail addresses: liugeng@hdu.edu.cn (G. Liu), luigi.lavazza@uninsubria.it (L. Lavazza).

https://doi.org/10.1016/j.jss.2020.110888
0164-1212/

using an early estimation method is just to save time and money: this is actually the usage envisioned by industry people working with NESMA (van Heeringen et al., 2009), and one of the uses anticipated by IFPUG (Timp, 2015).

In this paper we investigate the reasons why the HLFPA method tends to under-estimate. We also explore how to improve the method to make it less prone to under- (or over-) estimating.

The goals of the paper can be stated via the following research questions:

RQ1 Why does the HLFPA method underestimate?

RQ2 Is it possible to modify the HLFPA method so that its estimation accuracy improves, especially with respect to underestimation?

RQ3 Is it possible to build alternative estimation methods that are as easy and quick to apply as HLFPA, but are more accurate?

RQ4 Is it possible to build estimation methods that are even easier and quicker to apply than HLFPA, but are no less accurate?

The main contributions of the paper are the following:

1. We illustrate the reasons why the HLFPA method underestimates. These reasons can be generalized to similar estimation methods and situations.
2. We show that it is possible to build regression models that are structurally equivalent to the HLFPA method, i.e., they require exactly the same data, hence involve the same estimation cost as HLFPA. These models allow estimating size without underestimating (on average), but are slightly less accurate than HLFPA.
3. A method requiring less information than the HLFPA method – hence quicker and cheaper – is also illustrated. This method is slightly less accurate than HLFPA, though. We give the data that illustrate the expected accuracy level that can be achieved, thus enabling practitioners to evaluate the best trade-off between earliness and speed on one side and accuracy on the other side.

The mentioned contributions will likely be helpful for the numerous software development organizations that use Function Points. For instance, organizations that develop software for public administration in Brazil, Italy, Japan, South Korea, and Malaysia must provide software size measured in IFPUG FP, because local laws make this compulsory. For these organizations, obtaining early size estimations is extremely important, particularly for bidding purposes. Other organizations need FP measures because they use effort estimation tools (like Galorath's SEER-SEM Fischman et al., 2005, for instance) that require the size expressed in FP as input (together with a number of parameters that account for the development process and technology, non functional requirements, human factors, etc.).

Structure of the article

The remainder of the paper is organized as follows. Section 2 briefly presents Function Points Analysis and the High-level FPA method. Section 3 reports the results from the analysis of HLFPA method's estimation accuracy, published in a previous paper (Lavazza and Liu, 2019). Section 4 discusses the causes of HLFPA method's inaccuracy, especially as far as underestimation is concerned. Section 5 shows that the HLFPA method cannot be improved just by changing the values of parameters, based on the observation of the frequency of functional components' complexity levels. Section 6 illustrates the derivation of statistical models of functional size expressed in Function Points. The estimation accuracy of these models is evaluated and compared to the HLFPA method's. Section 7 provides guidelines for using our results in practical size estimation. In Section 8 we discuss the threats to the validity of this study. Section 9 reports about related work. Finally, in Section 10 we draw some conclusions and outline future work.

Scope of the paper

IFPUG defines both unadjusted FP (UFP) and adjusted FP. The former are a measure of functional requirements. The latter are obtained by correcting unadjusted FP in order to obtain an indicator that is better correlated to development effort. Noticeably, the ISO standardized only unadjusted FP, recognizing UFP as a proper measure of functional requirements (International Standardization Organization (ISO), 2003). Following the ISO, in this paper we deal only with UFP, even when we speak generically of Function Points or FP.

In this paper we deal only with the estimation of functional size. Such measure is considered important because it is the base of many effort estimation methods. However, effort estimation is out of this paper's scope.

2. Functional size measurement methods

To make the paper as self-contained as possible, Section 2.1 illustrates FP measurement, and Section 2.2 illustrates the High-level FPA method. Readers that are familiar with these methods can safely skip this section.

2.1. Function point analysis

This section provides a concise introduction to FPA. Readers are referred to the official documentation (International Function Point Users Group (IFPUG), 2010) for further details.

Function Point Analysis was originally introduced by Albrecht to measure the size of data-processing systems from end-users' point of view, with the goal of estimating the development effort (Albrecht, 1979).

The initial interest sparked by FPA, along with the recognition of the need for maintaining FPA counting practices updated, led to founding the IFPUG (International Function Points User Group). The IFPUG (http://www.ifpug.org/) maintains the counting practices manual (International Function Point Users Group (IFPUG), 2010), provides guidelines and examples, certifies professional FP counters, and oversees the standardization of the measurement method.

Albrecht's basic idea – which is still at the basis of the IFPUG method – is that the "amount of functionality" released to the user can be evaluated by taking into account (1) the data used by the application to provide the required functions, and (2) 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, IFPUG Function Points are counted on the basis of functional user requirements (FURs) specifications. The boundary indicates the border between the application being measured and the external applications and user domains.

FURs are modeled as a set of base functional components (BFCs), which are the measurable elements of FURs: each of the identified BFCs is measured, and the size of the application is obtained as the sum of the sizes of BFCs.

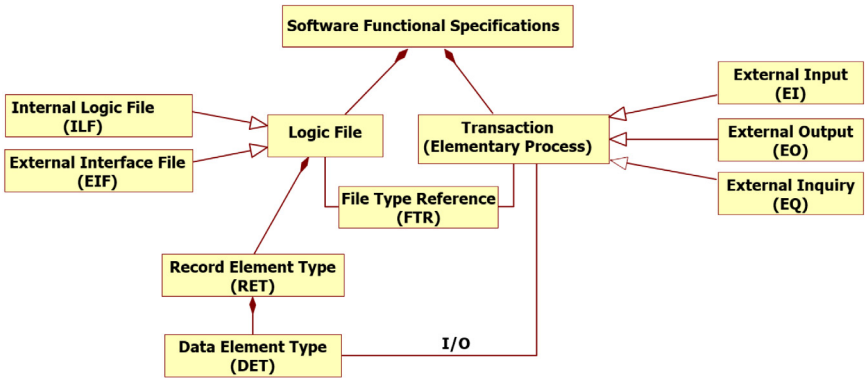

Fig. 1. The IFPUG model of software.

Table 1
FPA weight table.

Function type	Complexity		
	Low	Average	High
ILF	7	10	15
EIF	5	7	10
EI	3	4	6
EO	4	5	7
EQ	3	4	6

The IFPUG model of a software application to be measured is shown in Fig. 1 as a UML class diagram. IFPUG BFCs are data functions (also known as logical files), which are classified into internal logical files (ILF) and external interface files (EIF), and elementary processes (EP) – also known as transaction functions – which are classified into external inputs (EI), external outputs (EO), and external inquiries (EQ), according to the activities carried out within the considered process and its main intent.

So, the functional size of a given application, expressed in unadjusted Function Points, $Size_{UFP}$, is given by the sum of the sizes of the different types of functions, as shown in Eq. (1).

$$Size_{UFP} = \sum_{f \in ILFs} Size_{ILF}(f) + \sum_{f \in EIFs} Size_{EIF}(f) + \sum_{f \in EIs} Size_{EI}(f) + \sum_{f \in EOs} Size_{EO}(f) + \sum_{f \in EQs} Size_{EQ}(f) \quad (1)$$

In Eq. (1), *ILFs* is the set of all data functions of type *ILF*, *EIs* is the set of all transaction functions of type *EI*, etc. Also, $Size_X(f)$ is the weight of function f, which depends on its complexity, and its type $X \in \{ILF, EIF, EI, EO, EQ\}$, as described in Table 1.

The complexity of a data function (ILF or EIF) depends on the RETs (Record Element Types), which indicate how many types of information (e.g., sub-classes, in object-oriented terms) can be contained in the given logical data file, and DETs (Data Element Types), which indicate how many types of elementary information (e.g., attributes, in object-oriented terms) can be contained in the given logical data file.

The complexity of a transaction depends on the number of FTRs – i.e., the number of types of logical data files used while performing the required operation – and the number of DETs – i.e., the number of types of elementary data – that the considered transaction sends and receives across the boundaries of the application.

Details concerning the determination of complexity can be found in the official documentation (International Function Point Users Group (IFPUG), 2010).

The core of FPA involves three main activities:

1. Identifying data and transaction functions.
2. Classifying data functions as ILF or EIF and transactions as EI, EO or EQ.
3. Determining the complexity of each data or transaction function.

The first two of these activities can be carried out even if the FURs have not yet been fully detailed. On the contrary, activity 3 requires that all details are available, so that FP counters can determine the number of RET or FTR and DET involved in every function. Activity 3 is also relatively time- and effort-consuming (Lavazza, 2017). The HLFPA method does not require activity 3, thus allowing for size estimation when FURs are not fully detailed: it only requires that the complete sets of data and transaction functions are identified and classified. Since the method lets measurers skip the most time- and effort-consuming activity, it is relatively fast and cheap.

2.2. The high-level FPA method

NESMA defined two size estimation methods: the 'NESMA Indicative' and the 'NESMA Estimated' methods. IFPUG adopted these methods as the official early function point analysis methods, under the names of 'Indicative FPA' and 'High-level FPA,' respectively (Timp, 2015). The Indicative FPA method proved definitely less accurate (nesma; Lavazza and Liu, 2013). Hence, in this paper we consider only the High-level FPA method.

The High-level FPA method requires the identification and classification of all data and transaction functions, but does not require the assessment of the complexity of functions: ILF and EIF are assumed to be of low complexity, EI, EQ and EO are assumed to be of average complexity. Hence, estimated size is computed as follows:

$$EstSize_{UFP} = 7\ \#ILF + 5\ \#EIF + 4\ \#EI + 5\ \#EO + 4\ \#EQ \quad (2)$$

In formula (2), *#ILF* is the number of data functions of type ILF, *#EI* is the number of transaction functions of type EI, etc.

3. Evaluation of HLFPA estimation accuracy

In this section, we report an evaluation of the HLFPA method from previous work (Lavazza and Liu, 2019). The results reported here[1] motivate the research illustrated in the remainder of the paper.

[1] The results reported here are slightly different from those given in Lavazza and Liu (2019), because of an improved analysis procedure, which corrected a minor error in the previous analysis.

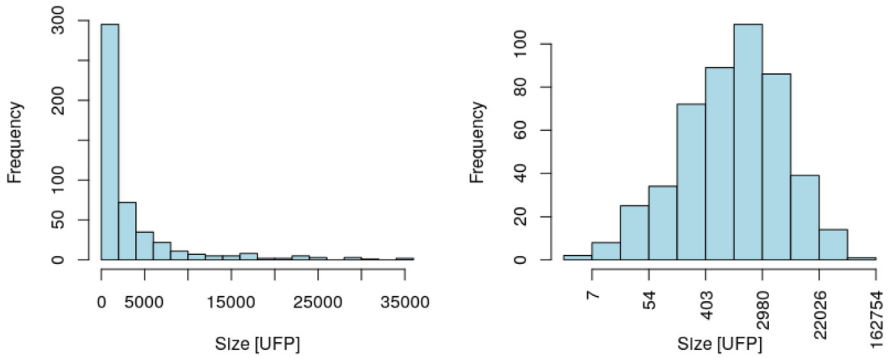

Fig. 2. Distributions of project sizes in the dataset (a logarithmic scale is used in the histogram on the right hand side).

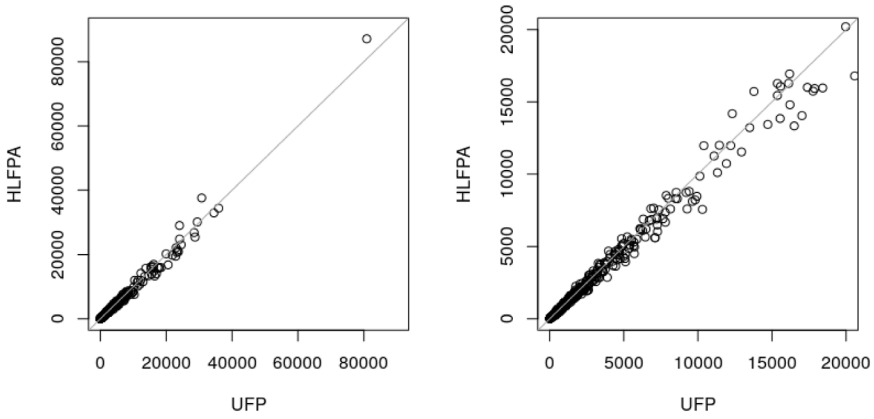

Fig. 3. Standard IFPUG UFP measures vs. HLFPA estimates: complete dataset (left) and zoom on projects not greater than 20,000 UFP (right).

Table 2
Descriptive statistics of the analyzed dataset.

	IFPUG FP	HLFPA	#ILF	#EIF	#EI	#EO	#EQ
Mean	3 554	3 435	80.6	42.3	269.3	130.0	233.0
Stdev	6 673	6 694	172.2	123.9	495.0	561.3	548.0
Median	1 155	1 122	22	5	90	23	57
Min	4	4	0	0	0	0	0
Max	80 880	87 100	2169	1198	3551	10 452	7099

3.1. The dataset

The dataset that was used in Lavazza and Liu (2019), and is used in this paper as well, includes data from 479 software projects developed and used by a Chinese financial enterprise. The data are subject to non-disclosure agreement, therefore we cannot publish them is a replication package. Some descriptive statistics of the dataset are given in Table 2.

Fig. 2 shows the distribution of the projects' sizes. The distribution of sizes in the dataset is skewed, as is often the case in empirical software engineering data. Specifically, most projects are small. However, 257 projects (54%) are over 1000 FP, and large projects are well represented (91 projects have size exceeding 5000 FP).

3.2. The analysis

For each of the 479 project in the dataset, we computed the estimated size according to the HLFPA method described in Section 2.2.

To assess the estimates, in Fig. 3(left) we plot the values of the estimates with respect to the actual size measured according to the standard IFPUG counting manual (International Function Point Users Group (IFPUG), 2010). In the figure, we also draw the HLFPA estimates = UFP line: if the estimates were perfect, all the points would lie on this line. As a matter of fact, most points are quite close to the line, thus indicating that in general the estimates are close to the actual measures.

To better appreciate the accuracy of estimates, in Fig. 3(right) the situation for the 462 out of 479 projects having size not greater than 20,000 UFP is shown. It can be observed that most points are below the y = x line, thus indicating that the HLFPA method tends to underestimate.

To verify if and to what extent the HLFPA method underestimates the IFPUG size, in Fig. 4 a boxplot of HLFPA estimation errors is given (errors are defined as the standard IFPUG size measure minus the estimate). It can be seen that both the median (shown as a thick horizontal segment) and the mean (shown as a blue diamond) are above the zero error line; namely the median error is 10 UFP, while the mean error is 119 UFP. Actually, 283 out of 479 projects (over 59%) are underestimated. We can thus conclude that – in the considered dataset – the HLFPA method tends to underestimate functional size.

3.3. Accuracy evaluation

It is now necessary to evaluate quantitatively the accuracy of HLFPA estimates. As suggested by Shepperd and McDonell (Shepperd and MacDonell, 2012), HLFPA estimates were compared with the estimates yielded by "baseline" models. Shepperd and McDonell also proposed that the accuracy of a given estimation method be measured via the Mean Absolute Residual (MAR):

$$MAR = \frac{\sum_{i=1..n} |y_i - \hat{y}_i|}{n} \quad (3)$$

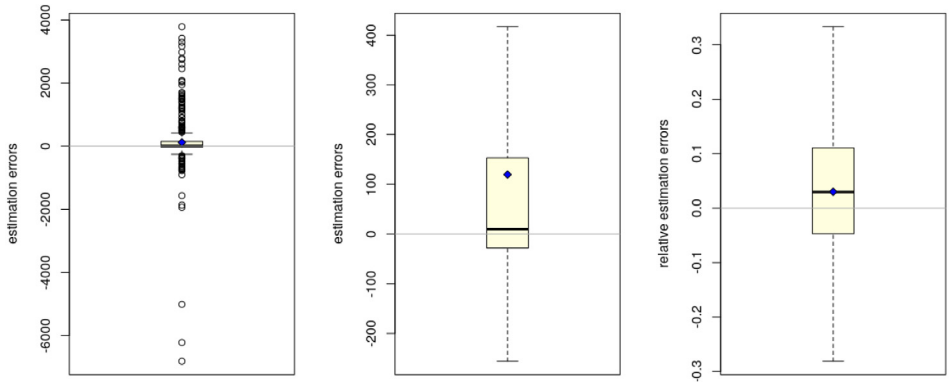

Fig. 4. Boxplot of HLFPA estimation errors: all errors (left), errors excluding outliers (center), all relative errors (right).

where y_i indicates the actual size of the ith project and $\hat{y}_i$ indicates the estimated size of the ith project (Shepperd and MacDonell, 2012).

As a referenced model, Shepperd and MacDonell suggest to use random (*rnd*) estimation, based on the known (actual) values of previously measured projects. A random estimation $\hat{y}_i$ is obtained by picking at random y_j, with $j \neq i$. In our case, the size of a software application would be estimated by picking at random the size of one of the other applications from the historical dataset. Of course, in this way there are $n-1$ possible estimates for y_i, so to compute the MAR of *rnd* we need to average all these possible values. Shepperd and MacDonell suggest to make a large number of random estimates (typically, 1000), and then take the mean $\overline{MAR_{rnd}}$. Achieving a MAR value substantially smaller than $\overline{MAR_{rnd}}$ is a necessary condition that estimation methods must satisfy, otherwise we could simply guess and get equivalent or even better estimates.

Shepperd and MacDonell observed also that the value of the 5% quantile of the random estimate MARs can be interpreted like α for conventional statistical inference, that is, any accuracy value that is better than this threshold has a less than one in twenty chance of being a random occurrence. Accordingly, the MAR of a proposed model should be compared with the 5% quantile of the random estimate MARs, to make us reasonably sure that the model is actually more accurate than the random estimation.

We also used "constant" models as baseline, proposed, among others, by Lavazza and Morasca (2017) and Di Martino et al. (2020). With the mean – respectively, median – constant model, the estimated effort for a project is given by the mean – respectively, median – of all other projects' actual efforts.

In Fig. 5, the distribution of absolute errors is given. The blue diamond is the mean, i.e, the MAR of the estimates. The median of absolute residuals is 75 UFP, however the MAR is definitely greater (315 UFP), because of several large errors. In Fig. 5, the 5% quantile of absolute residuals for random estimates, the MAR of the mean model and the MAR of the median model are shown as, respectively, solid, dashed and dotted lines. The MAR of HLFPA estimates is much smaller than any baseline. Consequently, the HLFPA method easily satisfies the necessary conditions for being considered an acceptable estimation method.

In general, relatively large estimation errors are deemed acceptable in very large projects. To help practitioners appreciate the "importance" of errors with respect to the size of the project, in Fig. 6 we give the boxplot representing the distribution of absolute relative errors (the relative error of an estimate is the estimation error divided by the actual size).

Fig. 6 shows that the great majority of estimate errors are less than 10% (the median absolute relative error is 0.083); moreover, in only 34 out of 479 cases, errors are greater than 25%. Considering that HLFPA estimates are produced without considering the details of functional requirements, this level of accuracy is likely acceptable by most practitioners.

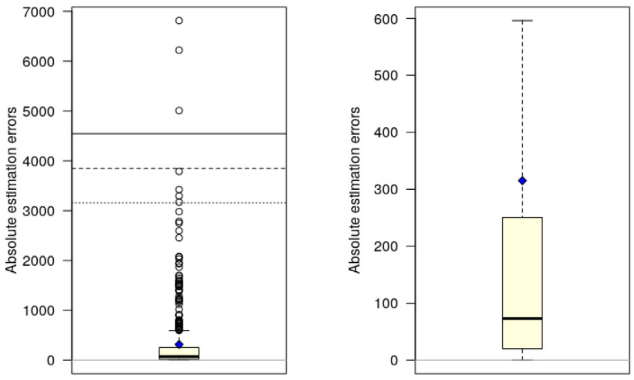

Fig. 5. Boxplot of HLFPA absolute estimation errors: all errors (left) and excluding outliers (right).

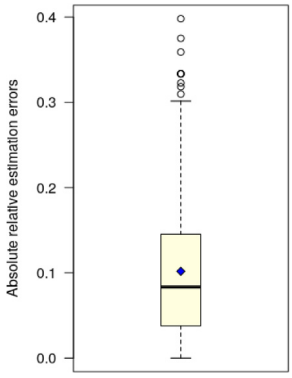

Fig. 6. Boxplot of absolute relative errors.

In Fig. 6 the blue diamond indicates the MMRE (Mean Magnitude of Relative Errors): its value is 0.1. MMRE has been criticized for being a biased indicator (see for instance Kitchenham et al., 2001); nevertheless we report its value because – when used in conjunction with sound indicators – it can provide an intuitive and representative indication of the "importance" of estimation errors.

4. Observations on HLFPA estimation accuracy

The data available from the dataset support some observations concerning the foundations of the HLFPA estimation method.

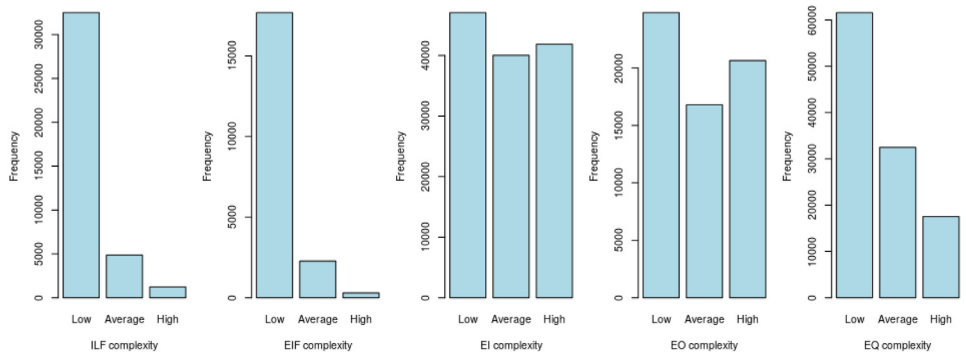

Fig. 7. Distributions of BFCs.

Table 3
Total HLFPA estimation errors, by function type.

Function type	Estimation error	
	UFP	%
ILF	24421	1.43
EIF	6053	0.36
EI	36666	2.15
EO	16475	0.97
EQ	−26446	−1.55
All	57569	3.36

As described in Section 2.2, the HLFPA estimation method weights all data functions as being of low complexity and all transaction functions as being of average complexity. We checked whether these conditions hold in our dataset. Fig. 7 shows the distributions of base functional components into the low, average and high complexity classes, evaluated according to the IFPUG counting manual. It is easy to see that HLFPA assumptions hold only for data functions. This mismatch between HLFPA assumptions and the actual distributions of functions' complexity may explain the tendency of HLFPA method to underestimate.

To better understand the origin of estimation errors, we compute the total approximation error due to each function type. Table 3 shows the estimations error for the entire dataset, by function type. Remember that estimation errors are defined as the actual size minus the estimated size, hence positive errors indicate underestimation. Table 3 shows that the HLFPA method underestimates all types of functions, except EQ; however, the overestimation of EQ does not compensate for the underestimation of the other function types. As a result, HLFPA underestimates the size of most software applications (as shown in Fig. 4) and the size of the entire set of applications (as shown in the bottom row of Table 3).

The observations reported above let us answer RQ1 (Why does HLFPA method underestimate?). In fact, the hypothesis that most data functions are of low complexity appears correct, but the average and high complexity data files lead the HLFPA method to underestimate the size of data functions. Concerning transactional functions, the hypothesis that most transactional functions are of average complexity appears incorrect for all types of transactions. The relatively high number of high complexity EI and EO leads the HLFPA method to underestimate both EI and EO. The relatively high number of low complexity EQ leads the HLFPA method to overestimate the size of EQ, but not sufficiently to compensate the underestimation of all other types of functions.

5. Alternative weighting

The analysis of the distributions of function complexity illustrated in Section 4 suggests that considering all functions of a given type as low complexity leads inevitably to underestimation. Similarly, considering all functions of a given type as high complexity would lead to overestimating. Hence, to avoid both underestimation and overestimation, it looks reasonable to use weights corresponding to average complexity (this is actually the simplified FP (sFP) approach Bernstein and Yuhas, 2005). In conclusion, we can redefine the estimation method as in the following formula (4), which is structurally identical to the original HLFPA formula (2), but with weights corresponding to average complexity for all function types.

$$EstSize_{UFP} = 10\ \#ILF + 7\ \#EIF + 4\ \#EI + 5\ \#EO + 4\ \#EQ \qquad (4)$$

We computed the estimated size of every application in the dataset using this 'modified HLFPA' method. Then, we computed the errors of the modified HLFPA method's estimates and compared them with the HLFPA method's errors.

Fig. 8(left) shows the boxplots representing the distributions of errors of the estimated and modified HLFPA methods (outlier errors are not reported to keep the figure readable). The modified HLFPA method tends to overestimate; it is also characterized by higher mean and median absolute errors, as shown in Fig. 8(right).

The results reported above let us answer RQ2 (Is it possible to modify the HLFPA method so that its estimation accuracy improves, especially with respect to underestimation?) even without applying formal statistical tests. It appears that adopting a fixed weight for all the functions of a given type does not let us improve the accuracy of the HLFPA method.

Other choices of weights could not correct the problem either. Specifically, we tested several combinations of weights without obtaining any improvement over HLFPA. For instance, assuming that ILF, EIF, EI, EO are of average complexity, and EQ are of low complexity, yields MAR = 411 UFP, while HLFPA's MAR is 311 UFP.

6. Basing estimates on statistical models

Organizations that need a quick estimation method for functional size and have historical data can consider building their own models, instead of using predefined estimation models. In fact, based on quantitative data from previous software projects, an organization can derive statistical models that correlate the size of software applications to the number of involved ILF, EIF, EI, EO and EQ. In this section we look for such models and evaluate the accuracy of the estimates they yield.

All the statistical models presented here satisfy the conditions usually required from statistically valid ordinary least squares (OLS) regression models (e.g., p-value $<$ 0.05, normally distributed residuals, etc.), also when not explicitly stated.

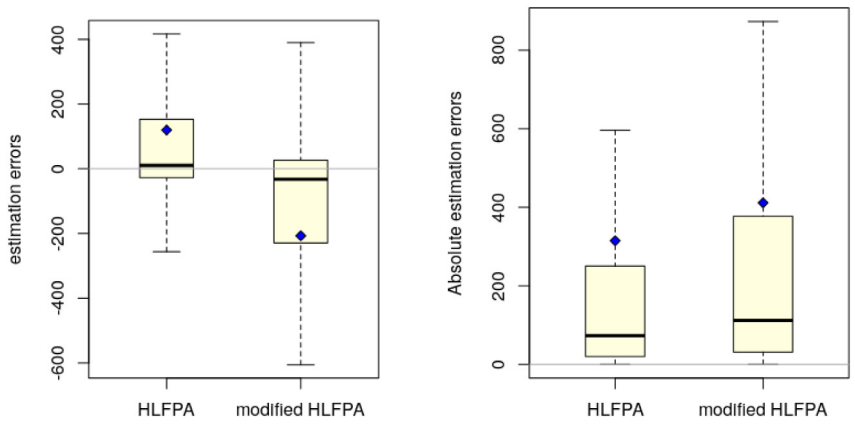

Fig. 8. Distributions of HLFPA and modified HLFPA methods' estimation errors (left) and absolute errors (right).

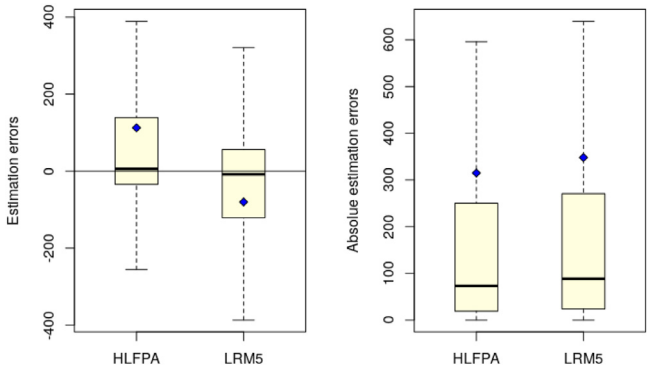

Fig. 9. The distributions of HLFPA method's and LRM5's estimation errors (left) and absolute errors (right).

6.1. A linear regression model based on FPA functions

We considered building a linear regression model (LRM) having the number of ILF, EIF, EI, EO and EQ (i.e., #ILF, #EIF, #EI, #EO, #EQ) as independent variables and the actual size measured in FP as the dependent variable. We shall refer to this model as LRM5, to stress that it uses 5 independent variables.

Before proceeding to build models, we checked the multicollinearity among the independent variables in the dataset by using the Farrar–Glauber multicollinearity test (Farrar and Glauber, 1967). The test showed that the candidate independent variables are not multicollinear, hence we proceeded to build the model using all of these variables.

We built an ordinary least squares linear regression model with null intercept. In this way, the obtained model is structurally identical to formulae (2) and (4). In fact, the obtained model is

$$Size_{UFP} = 6.82\ \#ILF + 4.03\ \#EIF + 5.64\ \#EI + 5.08\ \#EO + 3.34\ \#EQ \quad (5)$$

The adjusted R^2 of the model is 0.996; such a high value was expected, since the correlation between the independent and dependent variable is a consequence of the definition of Function Points.

To evaluate the accuracy of estimates based on linear regression models, we performed a 10-times 10-fold cross validation.

Fig. 9(left) shows the boxplots of the estimation errors of the HLFPA method and LRM5. It is easy to observe that LRM5 does not underestimate size, on the contrary, it tends to overestimate, much like HLFPA tends to underestimate.

Table 4
Comparison of estimation models' accuracy.

	Error		Relative error		Absolute error		Abs. rel. error	
	Mean	Median	Mean	Median	Mean	Median	Mean	Median
HLFPA	112	6	0.025	0.026	315	73	0.100	0.081
LRM5	−80	−9	−0.032	−0.023	348	88	0.121	0.094
LRM2	7	2	−0.004	0.007	342	89	0.119	0.100

Concerning the accuracy of estimates, Fig. 9(right) shows the boxplots illustrating the distributions of absolute estimation errors of the HLFPA method and of LRM5. It is easy to see that the performances of the two methods are extremely similar, HLFPA being slightly more accurate. The MAR of HLFPA estimates is 315 UFP, while the MAR of LRM5 is 348 UFP (see also Table 4): considering that the mean size of the projects in our dataset is over 3500 UFP (see Table 2), a 33 UFP MAR difference is hardly noticeable.

To evaluate if the estimates provided by a method are significantly better than those provided by the other method, we tested the statistical significance of the differences among absolute errors yielded by the used methods (Kitchenham et al., 2001). Namely, we compared the absolute residuals provided by the two methods via Wilcoxon rank sum test (Cohen, 1988): the test indicated that LRM5's absolute residuals are greater than HLFPA's. However the effect size – computed via Hedges's g (Rosenthal et al., 1994) – is definitely negligible: g = 0.04 indicates a close to nil difference.

6.2. A linear regression model based on unclassified data and transaction functions

Some Function Points estimation methods do not require that data functions are classified into ILF and EIF and transaction functions are classified into EI, EO, and EQ. So, the estimate is based on the number of data functions (#DF) and transaction function (#TF) only. Clearly, skipping the classification makes the estimation faster and cheaper, though possibly at the expenses of accuracy.

We looked for OLS linear regression models that use as independent variable the number of data functions and the number of transaction functions. We refer to this model as LRM2, to stress that the model uses 2 independent variables.

We obtained the following model:

$$Size_{UFP} = 5.76\ \#DF + 4.42\ \#TF \quad (6)$$

The adjusted R^2 of the model is 0.995.

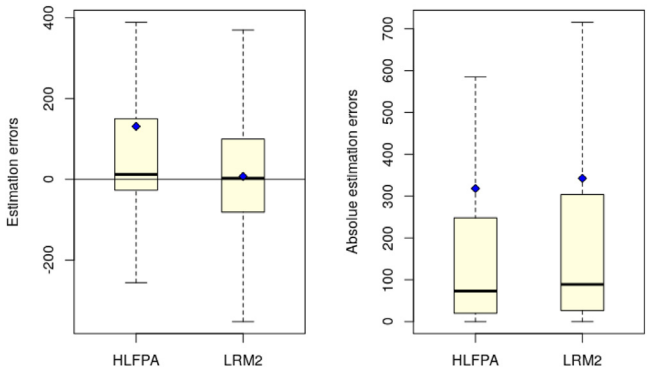

Fig. 10. The distributions of HLFPA method's and LRM2's estimation errors (left) and absolute errors (right).

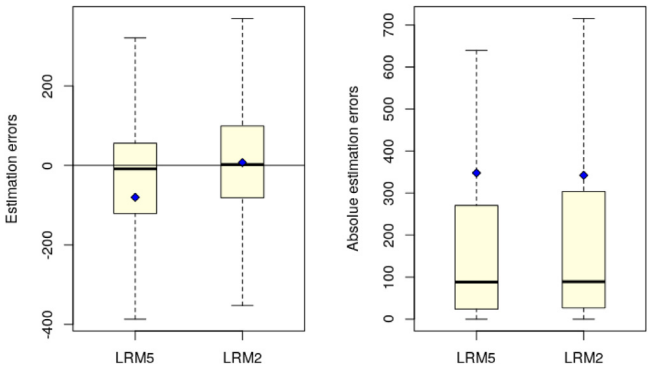

Fig. 11. The distributions of LRM5's and LRM2's estimation errors (left) and absolute errors (right).

To evaluate the accuracy of estimates yielded by LRM2, we performed a 10-times 10-fold cross validation.

Fig. 10 shows the boxplots of the estimation errors of the HLFPA method and LRM2. It is easy to see that the LRM2 model does neither underestimate nor overestimate size, on average.

Concerning accuracy, Fig. 10(right) shows the boxplots illustrating the distributions of absolute estimation errors of the HLFPA method and of LRM2. It appears that the performances of the two methods are similar, HLFPA being slightly more accurate. The MAR of HLFPA estimates is 315 UFP, while the MAR of LRM2 is 342 UFP (see also Table 4).

To evaluate if the estimates provided by LRM2 are significantly better than those provided by the HLFPA method, we compared the absolute residuals provided by the two methods via Wilcoxon rank sum test, which indicated that LRM2's absolute residuals are greater than HLFPA's. However the effect size – computed via Hedges's g (Rosenthal et al., 1994) – is definitely negligible: g = 0.03 indicates a close to nil difference.

In conclusion, even without classifying function into ILF, EIF, EI, EO and EQ, it is possible to obtain a statistically significant model that does not underestimate and provides marginally less accurate estimates than the HLFPA method.

6.3. Final observations on statistical models

The observations reported in Section 6.1 let us answer RQ3 (Is it possible to build alternative estimation methods that are as easy and quick to apply as HLFPA, but are more accurate?). Our results support only a negative answer: LRM5, which is based on the number of ILF, EIF, EI, EO and EQ, is marginally less accurate than HLFPA, as summarized in Table 4. However, some project managers could prefer LRM5 because it tends to overestimate, while HLFPA tends to underestimate.

Concerning RQ4 (Is it possible to build estimation methods that are even easier and quicker to apply than HLFPA, but are no less accurate?), the results described in Section 6.2 let us give a partially positive answer. On the one hand, HLFPA is slightly more accurate: it features MAR = 315 UFP vs. LRM2' MAR = 342 UFP, and HLFPA's MMRE is 10%, while LRM2's MMRE is 11.9%. Wilcoxon rank sum test confirms that LRM2's absolute residuals are greater than HLFPA's. On the other hand, the difference between LRM2's errors and HLFPA's errors is negligible according to Hedges's g (and probably also to many developers). In conclusion, the best answer to RQ4 is probably that even without classifying functions – hence, more quickly and cheaply than via HLFPA – it is possible to get a model (LRM2) that is only very slightly less accurate than the HLFPA method and makes the probability of overestimation and underestimation approximately equal, as shown in Fig. 10.

The availability of models based on the number of unclassified data and transaction functions is particularly interesting, since it allows software project managers to get size estimates even earlier than with the HLFPA method or the LRM based on classified functions. To this end, practitioners are likely interested in knowing how much LRM2 and LRM5 differ in terms of accuracy. Fig. 11 and Table 4 compare LRM5's and LRM2's distributions of errors and absolute errors. LRM5's accuracy is so close to LRM2' that most practitioners will likely prefer LRM2, which allows for faster and cheaper size estimates.

Finally, it can be observed that the HLFPA method is ready to use: after the ILF, EIF, EI, EO and EQ have been counted, the arguments required by formula (2) are available, and a size estimate can be produced. On the contrary, LRM models have to be derived from the available data via statistical analysis. However, carrying out the statistical analysis is fairly easy and takes a very short time. In fact, the core of the program for performs the analysis is just 50 lines of R code, and running the program takes just a few seconds.

7. Guidelines

7.1. Function size estimation with no historical data

Organizations that do not own historical data can use the High-level FPA method to estimate the size of software. However, HLFPA estimates are affected by some error, as all estimates are. When using the HLFPA method, software developers must consider that – based on our findings – the obtained size measures are probably underestimated, as Figs. 4 and 6 show. Therefore, one could try to compensate for the underestimation by scaling the estimates obtained via the HLFPA estimated method: Fig. 6 suggests that by increasing the estimates by 3.5% we should obtain more realistic and prudent estimates. In fact, such correction leads to size estimation errors that have the distribution illustrated by the boxplot in Fig. 12: the median and the mean errors (which are definitely above zero in Fig. 4) are close to zero. The MAR is 303 UFP, while the median absolute error is 79 UFP.

7.2. Function size estimation with historical data

Organizations that have historical data can build their own models – as we did in Section 6 – to estimate the size of software. In such case, it is recommended that the accuracy of the obtained models is evaluated, based on historical data, to get a rough idea of the likely reliability of new estimates. For instance, Fig. 6 shows

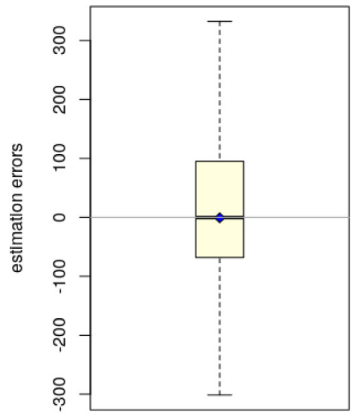

Fig. 12. Estimation errors of scaled HLFPA estimates.

that for the projects in our dataset, estimation errors in the [4%, 14%] range are quite probable, while errors above 30% are very rare.

The computation of estimation errors lets project managers answer important questions; for instance, they can evaluate the impact of size estimation errors on effort estimates. Consider for instance an organization that estimates effort using the model $\widehat{Eff} = 30\ Size^{1.2}$. If they feed the model with a size estimate that is underestimated by 10%, they get $30\ (0.9\ Size)^{1.2} = 0.88\ 30\ Size^{1.2} = 0.88\ \widehat{Eff}$. So, they know that underestimating the size by 10% results in underestimating effort by 12%. Based on this knowledge, they can decide whether the size estimation method being used is acceptably accurate or not.

Functional measures are used – to some extent – in agile processes as well. Usman et al. performed a survey concerning effort estimation in agile environments (Usman et al., 2015). They collected data from 60 agile practitioners from 16 different countries and found that 17% of the projects used function points (possibly in combination with other measures). To be effectively usable in an agile context, measures must be obtained easily and seamlessly. In this respect, it can be observed that the number of transactions and the number of logic data files can be easily derived from stories. Consider for instance the story "As a student, I want to purchase a parking pass, so that I can drive to school:" realizing that "parking pass" is a logical data file, and that "pass purchasing" is a transaction is immediate. Hence, the story accounts for a data file and a transaction functions, and – according to (6) – its size is 5.76 + 4.42 = 10.18 FP. In conclusion, in principle the LRM2 we propose can be applied in an agile context as well, since its independent variables are very easy to collect based on artifacts that are commonly used in agile processes.

8. Threats to validity

First, we should consider the correctness of the given data. In fact, the data in the analyzed dataset were derived from the analysis and measurement of functional requirements: both these activities could be affected by errors, which would end up in the dataset. Concerning this threat, we are reasonably sure that the used data are of good quality, since they were collected by professionals in industrial contexts where functional size measures are quite important, hence great attention is posed in the measurement activities. Even so, we cannot exclude that some errors are present; however, in such case most errors would affect IFPUG measures and estimates in similar ways. Hence, these errors should not be able to affect our results to an appreciable extent.

Second, we need to consider external validity, specifically, whether we can generalize the results of our study outside the scope and context that characterize the considered software projects. On the one hand, our dataset is much larger than the datasets usually involved in software engineering empirical studies; besides, our dataset includes data from fairly big projects (e.g., over 20,000 FP), which are rarely represented in historical datasets. In this sense, our dataset represents a quite large and varied sample. On the other hand, all the considered projects are from the economic, financial and banking domain, hence we cannot be sure that the results of our study apply equally well in other domains. In this respect, readers are reminded that previous studies show some difference in accuracy when estimates concern real-time software applications (Lavazza and Liu, 2013).

Another threat may come from using MMRE among the accuracy indicators, since MMRE is a biased indicator, as pointed out in the literature (see for instance (Kitchenham et al., 2001)). We used MMRE, together with other indicators – like MAR and the boxplots of residuals – to provide a more complete picture of the accuracy of our results, and compared the precision of different models via sound statistical tests, namely Wilcoxon rank sum test (also known as Mann–Whitney test). Therefore, the role of MMRE in the presented evaluations is marginal.

9. Related work

The quest for measures that are available in the early stages of the software lifecycle dates back to decades ago. In 1992, Bock and Klepper proposed the FP-S method (Bock and Klepper, 1992). Bock and Klepper were motivated by the need of reducing the time required to complete the function point counting task at McDonnell Douglas Corporation. To avoid the time-consuming activity required to determine the complexity of each BFC, they used a multiple regression procedure to determine the weight to be applied to each BFC. They used a fairly small historical dataset (39 projects), and obtained acceptably accurate estimates. Bock and Klepper reported as one of the main positive aspects of FP-S the elimination of measure variance due to the subjectivity that is inherently present in FP measures. The absence of variance due to subjective evaluations characterizes also HLFPA estimates, as well as those proposed in Sections Section 6.

In 1998, Horgan et al. proposed a size estimation method named "Early E model" (Horgan et al., 1998). With the Early E model, the estimated size is computed based on the Raw Function Points (RFP), which are defined as in formula (7) (in practice, RFP is the number of Base Functional Components).

$$RFP = \#ILF + \#EIF + \#EI + \#EO + \#EQ \tag{7}$$

The estimated size is obtained as $EstSize_{UFP} = W \cdot RFP$, where W is the Weighting Constant, which is computed based on historical project data, as follows:

$$W = \frac{\sum_{i=1}^{n} \frac{FP_i}{RFP_i}}{n} \tag{8}$$

In formula (8) n is the number of projects in the historical dataset, while FP_i is the size of the ith project computed according to Albrecht's FPA, and RFP_i is its RFP number.

The Early E model is applicable in the early stage of the software lifecycle. In fact, it only requires that the first of the activities listed in Section 2.1 is performed. As such, it is applicable even before the High-level FPA and LRM5 methods.

Horgan et al. reported that the proposed model provides estimation errors not greater than 25% for 94% of the estimates. With our dataset, we found – via a 10-times 10-fold cross-validation – that Early E model provides estimation errors not greater than

25% for almost 90% of the estimates. We also found that Early E's MAR is 360 UFP (and MMRE = 12.3%). We may conclude that Early E appears usable, but it is less accurate than the other methods considered in the paper, and it is not even much faster to apply: LRM2 only requires that BFCs are classified into data and transactions (which is practically immediate) and yields a slightly smaller MAR (342 UFP).

Santillo, Conte and Meli proposed the "Early & Quick Function Point" (EQFP) method. The EQFP method (Santillo et al., 2005) uses analogy and analysis. The former involves discovering similarities between a new piece of a software application and similar pieces that were classified and measured during previous measurement activities. The latter provides software object with weights derived from statistical analysis, thus guaranteeing stability for the estimates, at least in a given environment. Santillo et al. reported that estimates are within ±10% of the real size in most real cases, while the savings in time and costs are between 50% and 90%.

Santillo also proposed "Easy Function Points", an approximation technique for functional size measures (Santillo, 2012). Santillo suggested probabilistic approaches, where the measurer can indicate the minimum, medium and maximum weight of each BFC, together with the expected probability that the weight is actually minimum, medium or maximum. This leads to estimate not only the size, but also the probability that the actual size is equal to the estimate.

Lavazza et al. studied the relationships between BFCs and size measures expressed in FP and found that it is possible to build statistically significant estimation models for UFP based on BFCs (Lavazza et al., 2013). Specifically, they used Least Median Squares robust regression models. They conclude that not only BFCs can be used as predictors of the size expressed in UFP, but FP measures could be altogether replaced by measured based on a smaller set of BFCs.

Several other early estimation methods were proposed. We cannot give here a complete account of all Function Points early estimation methods; instead, in Table 5, we provide a list of the most popular ones. For each method, references to its definition and to empirical evaluations are given, together with the indication of which data are used and how they are weighted. For instance, the ISBSG average weight method computes estimates as the weighted sum of all the function types (EI, EO, EQ, ILF and EIF), each weighted with the corresponding mean weight observed in the ISBSG dataset. A review of older FP estimation methods was published by Meli and Santillo (1999b).

Lavazza and Liu (2013) used 7 real-time applications and 6 non real-time applications to evaluate the accuracy of the E&QFP (Santillo et al., 2005) and HLFPA methods with respect to full-fledged Function Point Analysis. The results showed that the Indicative FPA method yields the greatest errors. On the contrary, the HLFPA method yields size estimates that are close to the actual size. Specifically, the HLFPA method proved fairly good in estimating both Real-Time and non Real-Time applications.

Using a database concerning over 100 applications, NESMA empirically evaluated the accuracy of the NESMA Estimated and Indicative FPA approximation methods (nesma). The results showed that size measures of the high-level function point analysis and the detailed function point analysis are very close. Moreover, the Indicative method gives a good estimate of the size of several applications.

van Heeringen described the accuracy – as well as the difference in measurement effort – of the NESMA Estimated and NESMA Indicative methods, by measuring 42 projects (van Heeringen et al., 2009). The results show that the estimation error of NESMA Estimated was in the [-6%, +15%] range, with average 1.5%; the estimation error of NESMA Indicative was in the [-15%, +50%] range, with average 16.3%. In both cases the estimation error was evaluated with respect to detailed measurement.

Cândido and Sanches used the NESMA method to estimate the size of Web applications (Cândido and Sanches, 2004). They found that the choice of weights used by the NESMA method did not match the complexity of most BFCs in their Web applications. As a solution, they proposed to use the weights corresponding to low complexity for all BFCs. In that way, they improved the accuracy of estimates. This solution is clearly not generalizable: with our dataset, weighting all BFCs as they were of low complexity leads to less accurate estimates.

Popović and Bojić compared different functional size measures – including NESMA Indicative and Estimated – by evaluating their accuracy in effort estimation in various phases of the development lifecycle (Popović and Bojić, 2012). Not surprisingly, they found that the NESMA Indicative method provided the best accuracy at the beginning of the project. With respect to Popović and Bojić, we evaluated the NESMA method with respect to its accuracy in estimating the actual size of software, rather than development effort.

Wilkie et al. (2011) used five commercial projects to evaluate the cost-benefit trade-off of size measurement with respect to size estimation; they concluded that whilst the NESMA Indicative method was insufficiently accurate for the involved commercial organization, the NESMA Estimated approach was definitely viable.

Morrow et al. used a dataset of 11 projects to evaluate the quality of sizing estimates provided by NESMA methods (Morrow et al., 2014). They also adapted NESMA methods' general principles to enhance their accuracy and extent of relevance, and empirically validated such an adapted approach using commercial software projects.

10. Conclusions

After over 40 years since their introduction, Function Points are widely used (Jones, 2017; Georgi and Vogt, 2008) and studied (Liu et al., 2018; Silhavy et al., 2018; Dewi et al., 2018; Liu et al., 2020). However, the effort and time required to perform FP measurement, together with the need for relatively detailed specifications of functional user requirements, induced practitioners and researchers to look for fast and easy ways of getting estimates of size measures. To this end, several techniques were proposed (see Section 9).

Both practitioners and researchers are interested in obtaining the best possible trade-off between estimation accuracy on one side and measurement time and effort on the other side. Several FP estimation techniques were proposed in pursuing the optimization of such trade-off.

The "NESMA Estimated" method emerged as one of the most effective, and in 2015 it was finally adopted by the International Function Point User Group as the IFPUG preferred method for "early Function Point Analysis and consistent cost estimating", under the name of High-level FPA (Timp, 2015). The High-level FPA method avoids the most time-consuming phase of FP measurement, namely the evaluation of complexity and consequent weighting of BFCs. In addition, several empirical studies showed that the method is sufficiently accurate for practical usage.

However, a recent study, carried out using an unusually large dataset (479 project from the banking and financial domain) showed that the HLFPA method tends to underestimate. Since underestimation may lead to unrealistic development plans and possibly to project failure, it is quite important to evaluate why does the HLFPA method underestimate, and, if so, whether it is possible to overcome the problem.

In this paper we showed that the reason for size underestimation is that the HLFPA method assumes that data functions are

Table 5
Early estimation methods: definitions and evaluations.

Method name	Definition	Used functions	Weight	Evaluation
NESMA indicative	NESMA–the Netherlands Software Metrics Association (2004), International Standards Organisation (2005)	Data	Fixed	van Heeringen et al. (2009), Wilkie et al. (2011), Popović and Bojić (2012), Morrow et al. (2014), Lavazza and Liu (2019), Di Martino et al. (2020), Lavazza and Liu (2013)
NESMA estimated (HLFPA)	NESMA–the Netherlands Software Metrics Association (2004), International Standards Organisation (2005)	All functions	Fixed	van Heeringen et al. (2009), Wilkie et al. (2011), Popović and Bojić (2012), Morrow et al. (2014), Lavazza and Liu (2019), Di Martino et al. (2020), Lavazza and Liu (2013)
Early & Quick FP	Santillo et al. (2005), Iorio et al. (2007)	All functions	Statistics	Lavazza and Liu (2013), Meli (2015)
Tichenor ILF model	Tichenor (1997)	ILF	Fixed	Lavazza and Liu (2013)
simplified FP (sFP)	Bernstein and Yuhas (2005)	All functions	Fixed	Lavazza and Liu (2013)
ISBSG average weights	Meli and Santillo (1999a)	All functions	Statistics	Lavazza and Liu (2013)
SiFP	Meli (2011)	Data and trans.	Statistics	Lavazza and Meli (2014), Ferrucci et al. (2016)

mainly of low complexity and transaction functions are mainly of medium complexity, while in the considered dataset it is not so. Unfortunately, whatever complexity you expect for BFCs, you may find an application domain or specific software applications characterized by different complexities (for instance, this was observed in Cândido and Sanches, 2004, where BFCs had mostly low complexity).

An alternative strategy to size estimation involves deriving the most likely weight in a given environment by analyzing the data from projects previously carried out in the same environment. We did that, and derived ordinary least squared linear regression models linking the size of projects, expressed in unadjusted FP, to the number of classified BFCs (i.e., the numbers of ILF, EIF, EI, EO and EQ). Via 10-times 10-fold cross-validation, we evaluated the accuracy of these models, and obtained two opposing results: (1) unlike the HLFPA method, the linear regression models do not underestimate, and (2) linear regression models yield slightly less accurate estimates.

We also considered further simplifying the HLFPA method, by avoiding the classification of transactions into EI, EO and EQ, and the classification of data into ILF and EIF. We derived ordinary least squared linear regression models linking the size of projects expressed in FP to the number of data functions and the number of transaction functions. The obtained model appears as an interesting alternative to the HLFPA method, being marginally less accurate while avoiding underestimation and allowing for even faster estimations.

Future work includes experimenting with additional datasets, if available, and different techniques for deriving size estimation models.

CRediT authorship contribution statement

Geng Liu: Investigation, Methodology, Writing - review & editing, Funding acquisition. **Luigi Lavazza:** Conceptualization, Methodology, Software, Writing - original draft.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work have been supported by the "Fondo di Ricerca d'Ateneo" funded by the Università degli Studi dell'Insubria, Italy, by Zhejiang Provincial Science Foundation of China under grant no. LY19F020046, and by the Chinese Scholarship Council under grant no. 201708330399.

References

Albrecht, A.J., 1979. Measuring application development productivity. In: Proceedings of the Joint SHARE/GUIDE/IBM Application Development Symposium, Vol. 10. pp. 83–92.

Bernstein, L., Yuhas, C.M., 2005. Trustworthy Systems Through Quantitative Software Engineering, Vol. 1. John Wiley & Sons.

Bock, D.B., Klepper, R., 1992. FP-S: a simplified function point counting method. J. Syst. Softw. 18 (3), 245–254.

Cândido, E.J., Sanches, R., 2004. Estimating the size of web applications by using a simplified function point method. In: WebMedia and la-Web, 2004. Proceedings. IEEE, pp. 98–105.

Cohen, J., 1988. Statistical Power Analysis for the Behavioral Sciences. Lawrence Earlbaum Associates, Hillsdale, NJ, pp. 20–26.

Dewi, R.S., Subriadi, A.P., et al., 2018. Game complexity factor: A collaborative study of leblanc taxonomy and function points method. In: 2018 International Conference on Electrical Engineering and Computer Science. ICECOS, IEEE, pp. 153–158.

Di Martino, S., Ferrucci, F., Gravino, C., Sarro, F., 2020. Assessing the effectiveness of approximate functional sizing approaches for effort estimation. Inf. Softw. Technol. 123 (106308).

Farrar, D.E., Glauber, R.R., 1967. Multicollinearity in regression analysis: The problem revisited. Rev. Econ. Stat. 49 (1), 92–107, URL http://www.jstor.org/stable/1937887.

Ferrucci, F., Gravino, C., Lavazza, L., 2016. Simple function points for effort estimation: a further assessment. In: Proceedings of the 31st Annual ACM Symposium on Applied Computing. ACM, pp. 1428–1433.

Fischman, L., McRitchie, K., Galorath, D.D., 2005. Inside SEER-SEM. CrossTalk 18 (4).

de Freitas Junior, M., Fantinato, M., Sun, V., 2015. Improvements to the function point analysis method: A systematic literature review. IEEE Trans. Eng. Manage. 62 (4), 495–506.

Georgi, R., Vogt, T., 2008. Illustrative example of a function point analysis for the NASA crew exploration vehicle guidance, navigation and control flight software.

van Heeringen, H., van Gorp, E., Prins, T., 2009. Functional size measurement-Accuracy versus costs–Is it really worth it? In: Software Measurement European Forum. SMEF 2009.

Horgan, G., Khaddaj, S., Forte, P., 1998. Construction of an FPA-type metric for early lifecycle estimation. Inf. Softw. Technol. 40 (8), 409–415.

International Function Point Users Group (IFPUG), 2010. Function point counting practices manual, release 4.3.1.

International Standardization Organization (ISO), 2003. ISO/IEC 20926: 2003, Software engineering – IFPUG 4.1 Unadjusted functional size measurement method – counting practices manual.

International Standards Organisation, 2005. ISO/IEC 24570:2005 – software engineering – NESMA functional size measurement method version 2.1 – definitions and counting guidelines for the application of function point analysis.

Iorio, T., Meli, R., Perna, F., 2007. Early & Quick Function Points® v3. 0: enhancements for a Publicly Available Method. In: Proceedings Software Measurement European Forum. SMEF. pp. 179–198.

Jones, C., 2017. Quantifying Software: Global and Industry Perspectives. CRC Press.

Kitchenham, B., Pickard, L., MacDonell, S., Shepperd, M., 2001. What accuracy statistics really measure [software estimation]. In: Software, IEE Proceedings, Vol. 148. (3), IET, pp. 81–85.

Lavazza, L., 2017. On the effort required by function point measurement phases. Int. J. Adv. Softw. 10 (1), 2.

Lavazza, L., Liu, G., 2013. An empirical evaluation of simplified function point measurement processes. J. Adv. Softw. 6 (1& 2).

Lavazza, L., Liu, G., 2019. An empirical evaluation of the accuracy of NESMA function points estimates. In: The 14th International Conference on Software Engineering Advances. ICSEA 2019. pp. 24–29.
Lavazza, L., Meli, R., 2014. An evaluation of simple function point as a replacement of ifpug function point. In: 2014 Joint Conference of the International Workshop on Software Measurement and the International Conference on Software Process and Product Measurement. IWSM-MENSURA, IEEE, pp. 196–206.
Lavazza, L., Morasca, S., 2017. On the evaluation of effort estimation models. In: Proceedings of the 21st International Conference on Evaluation and Assessment in Software Engineering. ACM, pp. 41–50.
Lavazza, L., Morasca, S., Robiolo, G., 2013. Towards a simplified definition of function points. Inf. Softw. Technol. 55 (10), 1796–1809.
Liu, J., Du, Q., Xu, J., 2018. A learning-based adjustment model with genetic algorithm of function point estimation. In: 2018 IEEE 20th International Conference on High Performance Computing and Communications; IEEE 16th International Conference on Smart City; IEEE 4th International Conference on Data Science and Systems. HPCC/SmartCity/DSS. pp. 51–58.
Liu, G., Lavazza, L., Tosi, D., 2020. Evolution of functional size measures through ICONIX process phases. J. Softw.: Evol. Process 32 (5), e2240.
Meli, R., 2011. Simple function point: a new functional size measurement method fully compliant with IFPUG 4. x. In: Software Measurement European Forum.
Meli, R., 2015. Early & Quick Function Point Method-An empirical validation experiment. In: Int. Conf. on Advances and Trends in Software Engineering. Barcelona, Spain.
Meli, R., Santillo, L., 1999a. Function point estimation methods: A comparative overview. In: FESMA, Vol. 99. Citeseer, pp. 6–8.
Meli, R., Santillo, L., 1999. Function point estimation methods: a comparative overview. In: Software Measurement European Forum. FESMA 1999.
Morrow, P., Wilkie, F.G., McChesney, I., 2014. Function point analysis using NESMA: simplifying the sizing without simplifying the size. Softw. Qual. J. 22 (4), 611–660.
nesma, 0000. Early function point analysis. https://nesma.org/themes/sizing/function-point-analysis/early-function-point-counting/.
NESMA–the Netherlands Software Metrics Association, 2004. Definitions and counting guidelines for the application of function point analysis. nesma functional size measurement method compliant to iso/iec 24570 version 2.1.
Popović, J., Bojić, D., 2012. A comparative evaluation of effort estimation methods in the software life cycle. Comput. Sci. Inf. Syst. 9 (1), 455–484.
Rosenthal, R., Cooper, H., Hedges, L., 1994. Parametric measures of effect size. In: The Handbook of Research Synthesis, Vol. 621. pp. 231–244.
Santillo, L., 2012. Easy function points – 'Smart' approximation technique for the IFPUG and COSMIC methods. In: Joint Conf. of the 22nd Int. Workshop on Software Measurement and the 7th Int. Conf. on Software Process and Product Measurement.
Santillo, L., Conte, M., Meli, R., 2005. Early & quick function point: sizing more with less. In: 11th IEEE International Software Metrics Symposium. METRICS'05, IEEE, p. 41.
Shepperd, M., MacDonell, S., 2012. Evaluating prediction systems in software project estimation. Inf. Softw. Technol. 54 (8), 820–827.
Silhavy, P., Silhavy, R., Prokopova, Z., 2018. Stepwise regression clustering method in function points estimation. In: Proceedings of the Computational Methods in Systems and Software. Springer, pp. 333–340.
Tichenor, C., 1997. The IRS development and application of the internal logical file model to estimate function point Counts. In: IFPUG Fall Conference.
Timp, A., 2015. uTip – Early Function Point Analysis and Consistent Cost Estimating. IFPUG, uTip # 03 – (version # 1.0 2015/07/01).
Usman, M., Mendes, E., Börstler, J., 2015. Effort estimation in agile software development: a survey on the state of the practice. In: Proceedings of the 19th International Conference on Evaluation and Assessment in Software Engineering. pp. 1–10.
Wilkie, F.G., McChesney, I.R., Morrow, P., Tuxworth, C., Lester, N., 2011. The value of software sizing. Inf. Softw. Technol. 53 (11), 1236–1249.

Geng Liu is a lecturer at Hangzhou Dianzi University, China. He received the Ph.D. degree in Computer Science from University of Insubria (Italy) in 2014. His research interests include, among others, Software Metrics, Software Quality, Requirements Modeling and Requirement-driven Design.

Luigi Lavazza is an Associate Professor at the University of Insubria at Varese, Italy. His research interests include: Empirical software engineering, software metrics and software quality evaluation; Software project management and effort estimation; Software process modeling, measurement and improvement; Open Source Software. He is co-author of over 170 scientific articles, published in international journals, in the proceedings of international conferences, or in books. He has served on the PC of several international Software Engineering conferences, and in the editorial board of international journals.