Using Locally Weighted Regression to Estimate the Functional Size of Software: an Empirical Study

Luigi Lavazza Angela Locoro
Dipartimento di Scienze Teoriche e Applicate
Università degli Studi dell'Insubria
Varese, Italy
email:luigi.lavazza, angela.locoro@uninsubria.it

Geng Liu
Hangzhou Dianzi University
Hangzhou, China
email:liugeng@hdu.edu.cn

Roberto Meli
DPO
Rome, Italy
email:roberto.meli@dpo.it

***Abstract*—In software engineering, measuring software functional size via the IFPUG (International Function Point Users Group) Function Point Analysis using the standard manual process can be a long and expensive activity, which is possible only when functional user requirements are known completely and in detail. To solve this problem, several early estimation methods have been proposed and have become *de facto* standard processes. Among these, a prominent one is High-level Function Point Analysis. Recently, the Simple Function Point method has been released by IFPUG; although it is a proper measurement method, it has a great level of convertibility to traditional Function Points and may be used as an estimation method. Both High-level Function Point Analysis and Simple Function Point skip the activities needed to weight data and transaction functions, thus enabling lightweight measurement based on coarse-grained requirements specifications. This makes the process faster and cheaper, but yields approximate measures. The accuracy of the mentioned method has been evaluated, also via large-scale empirical studies, showing that the yielded approximate measures are sufficiently accurate for practical usage. In this paper, locally weighted regression is applied to the problem outlined above. This empirical study shows that estimates obtained via locally weighted regression are more accurate than those obtained via High-level Function Point Analysis, but are not substantially better than those yielded by alternative estimation methods using linear regression. The Simple Function Point method appears to yield measures that are well correlated with those obtained via standard measurement. In conclusion, locally weighted regression appears to be effective and accurate enough for estimating software functional size.**

***Index Terms*—Function Point Analysis, Early Size Estimation, High-level FPA, Simple Function Points, LOcally Estimated Scatterplot Smoothing (LOESS)**

I. Introduction

This paper extends a previous study that examined a single functional measure dataset [1].

In the late seventies, Allan Albrecht introduced Function Points Analysis (FPA) at IBM [2], as a means to measure the functional size of software, with special reference to the "functional content" delivered by software providers. Albrecht aimed at defining a measure that might be correlated to the value of software from the perspective of a user, and could also be useful to assess the cost of developing software applications, based on functional user requirements.

FPA is a Functional Size Measurement Method (FSMM), compliant with the ISO/IEC 14143 standard, for measuring the size of a software application in the early stages of a project, generally before actual development starts. Accordingly, software size measures expressed in Function Points (FP) are often used for cost estimation.

The International Function Points User Group (IFPUG) is an association that keeps FPA up to date, publishes the official FP counting manual [3], and certifies professional FP counters. Unfortunately, in some conditions, performing the standard IFPUG measurement process may be too long and expensive, with respect to management needs, because standard FP measurement can be performed only when relatively complete and detailed requirements specifications are available, while functional measures could be needed much earlier for management purposes.

Many methods were invented and used to provide *estimates* of functional size measures, based on fewer or coarser-grained information than required by standard FPA. These methods are applied very early in software projects, even before deciding what process (e.g., agile or waterfall) will be used. One of these methods is the High-level FPA (HLFPA) method [4], which was developed by NESMA under the name of "NESMA estimated" method [5].

In 2010, a new FSMM called Simple Function Point (SiFP) was developed by Meli [6]. In 2019, IFPUG acquired the method and in 2021 the IFPUG branded Simple Function Point (SFP) method was delivered to the market [7].

HLFPA and SiFP have been evaluated by several studies, which found that the methods are usable in practice to approximate traditional FPA values, since they yield reasonably accurate estimates. However, the question if it is possible to get more accurate estimates from the basic information used by HLFPA remains open.

In this paper, we evaluate—via an empirical study—the usage of LOESS (LOcally Estimated Scatterplot Smoothing)—also known as LOWESS (LOcally WEighted Scatterplot Smoothing)—to build models that can be used for early estimation of functional size.

We also compare the standard IFPUG FPA measures, the estimates obtained via HLFPA and the estimates obtained via alternative methods (linear regression models and LOESS models) with the measures obtained via the Simple Function Point (SFP) method. SFP is a lightweight method that has also been adopted by IFPUG as an alternative to full-fledged FPA. SFP measurement requires even less time and effort than HLFPA, and it usually yields measures that are very well correlated with IFPUG standard measures.

The work presented here extends previous work [1] by using two datasets to evaluate functional size estimation methods. Specifically, the availability of two datasets allows for cross-dataset evaluations. That is, one dataset is used as the training set, and the other one is used as the test set. This is particularly interesting for practitioners that do not own historical data: our results show that by using a "foreign" dataset for training, it is possible to obtain estimates that appear accurate enough for being used in practice.

In general, the findings reported in this paper contribute to increase our knowledge of the techniques that are available for functional size estimation, their applicability conditions, and the accuracy of the results that can be expected.

The remainder of the paper is organized as follows. Section II provides an overview of functional size measurement methods, and other background information. Section III describes the empirical study and its results. In Section IV the results obtained in the empirical study are discussed, from the technical and managerial points of view. In Section V, we discuss the threats to the validity of the study. Section VI reports about related work. Finally, in Section VII, we draw some conclusions and outline future work.

Note that FPA 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 get an indicator that is expected to be better correlated to development effort. Noticeably, the ISO standardized only unadjusted FP, recognizing UFP as a proper measure of functional requirements [8]. Following the ISO, in this paper we deal only with UFP, even when we speak generically of Function Points or FP. As a matter of fact, also HLFPA aims at providing measures that are compatible with UFP, and not with adjusted FP.

II. Background

Function Point Analysis was originally introduced by Albrecht to measure the size of data-processing systems from the point of view of end-users, with the goal of estimating the value of an application and the development effort [2]. The fortunes of this measure led to the creation of the IFPUG (International Function Points User Group), which maintains the method and certifies professional measurers.

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 counted on the basis of Functional User Requirements (FURs) specifications, and constitute the IFPUG Function Points measure.

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. 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 the primary intent.

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

The complexity of a transaction depends on the number of FTRs—i.e., the number of File Types Referenced 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 [3].

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 measurers can determine the number of RET or FTR and DET involved in every function. Activity 3 is relatively time- and effort-consuming [9].

HLFPA 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.

The SFP method [7] does not require activities 2 and 3: it only requires that the complete sets of data and transaction functions are identified.

Both the HLFPA and SFP methods let measurers skip the most time- and effort-consuming activity, which also needs that requirements are fully specified; thus both methods are relatively fast and cheap. The SFP method does not even require classification, making size estimation even faster and less subjective (since different measurers can sometimes classify differently the same transaction, based on the subjective perception of the transaction's primary intent).

A. The High-level FPA method

NESMA defined two size estimation methods: the 'NESMA Indicative' and the 'NESMA Estimated' methods. IFPUG adopted these methods as early function point analysis methods, under the names of 'Indicative FPA' and 'High-level FPA,' respectively [4]. The Indicative FPA method proved definitely less accurate [10], [11]. 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, while EI, EQ and EO are assumed to be of average complexity. Hence, estimated size is computed as follows:

$$EstSize_{UFP} = 7\ \#ILF + 5\ \#EIF + 4\ \#EI + 5\ \#EO + 4\ \#EQ \quad (1)$$

where *#ILF* is the number of data functions of type ILF, *#EI* is the number of transaction functions of type EI, etc.

B. The Simple Function Point Method

The Simple Function Point measurement method [6] [7] has been specifically designed to be agile, fast, lightweight, easy to use, and with minimal impact on software development processes. It is easy to learn and provides reliable, repeatable, and objective results. Like IFPUG FPA, it is independent of the technologies used and technical design principles.

SFP requires only the identification of Elementary Processes (EP) and Logical Files (LF), based on the following assumptions: 1) a user gives value to a BFC as a whole independently of internal organization and details, and 2) a cost model based on SFP shows a precision that is comparable to that of a cost model based on a detailed FPA measure. The latter assumption has been verified by different studies [12] [13].

SFP assigns a numeric value directly to these BFCs:

$$SFP = 7\ \#LF + 4.6\ \#EP \quad (2)$$

thus significantly speeding up the functional sizing process, at the expense of ignoring the domain data model, and the primary intent of each Elementary Process.

The weights for each BFC were originally given to achieve the best possible approximation of FPA but as long as the method has become a measurement method, those weights became constants, which are not subject to update or change for approximation reasons and that are crystallized for stability, repeatability and comparability reasons. We can approximate the FPA by setting $EstSize_{UFP} = SFP$.

III. Empirical study

A. The Datasets

In the empirical study, we use two datasets. The first is an ISBSG dataset [14], which was also used previously to evaluate SFP [12]; this is the dataset we used in our original work [1].

The second dataset includes data from software projects developed and used by a Chinese financial enterprise (hence, sometimes we make reference to this dataset as the "Chinese" dataset). These data are subject to non-disclosure agreement, therefore we cannot publish them in a replication package. Also the Chinese dataset was used previously [15], [16] in studies concerning the estimation of functional size measures.

Both datasets contain several small project data. As a matter of fact, estimating the size of small projects is not very interesting. Therefore, we removed from the dataset the projects smaller than 200 UFP. The resulting ISBSG dataset includes data from 110 projects having size in the [207, 4202] range. Some descriptive statistics for this dataset are given in Table I (where all values are rounded to integer).

TABLE I
Descriptive statistics for the ISBSG dataset.

	UFP	HLFPA	SFP	*#EI*	*#EO*	*#EQ*	*#ILF*	*#EIF*	*#LF*	*#EP*
Mean	976	888	971	43	46	46	26	24	50	135
StDev	842	739	785	38	71	51	22	23	39	123
Median	639	607	674	29	17	32	20	18	37	82
Min	207	202	223	0	0	0	0	1	12	14
Max	4202	3755	4257	204	442	366	100	172	234	656

While the ISBSG dataset contains data form projects not greater than 4202 UFP, the Chinese dataset contains data also from much larger projects (up to a few thousands UFP). However, to make the results obtained with the two datasets comparable, we used a subset of the Chinese dataset, so that the size range covered by the two datasets is the same. Some descriptive statistics of the resulting dataset (which accounts for 276 projects) are given in Table II (where all values are rounded to integer).

TABLE II
Descriptive statistics for the Chinese dataset.

	UFP	HLFPA	SFP	*#EI*	*#EO*	*#EQ*	*#ILF*	*#EIF*	*#LF*	*#EP*
Mean	1357	1323	1452	34	14	111	44	87	48	242
Sd	1040	1038	1141	39	23	101	60	101	52	200
Median	1041	984	1074	21	4	80	24	52	29	171
Min	200	142	154	0	0	0	0	0	0	2
Max	4079	4689	5349	220	144	524	428	712	276	997

B. Method used

We built models of functional size using LOESS (locally estimated scatterplot smoothing) [17]. LOESS belongs to the family of computational methods, based on least squares regression, for the estimation of functions fitting subsets of points of a dataset, without the need to yield a global function as a model. The way it works is capturing the local variability of neighbour points of the current point analyzed, in order to build up a function that describes the deterministic part of the variation in the data, point by point. For this reason, it is said to combine k-nearest-neighbor-based models into a meta-model. The regression can be linear and non-linear, i.e., polynomial. The mechanism of neighbours selection depends on a smoothing parameter, α, which determines the inclusion span of point neighbours to be included in the fitting polynomial function. A polynomial function of zero degree turns the LOESS curve method into a mobile average smoothing curve. A weighted variant of LOESS is called LOWESS, which stands for "locally weighted scatterplot smoothing". In this variant, local points are weighted for relevance with respect to the analyzed point, which is proportional to the variance brought by each point, with the nearest point receiving more importance and the furthest ones having less importance during models fitting.

C. Procedure

The analysis was carried out using the R programming language and environment [18]. Specifically, we used the `loess` function from the `Stats` package, which is provided as part of the system libraries.

Through the `span` parameter, the `loess` function makes it possible to control the degree of smoothing. In the empirical study, we tried different values for the `span` parameter, namely 0.5, 0.75 and 0.95.

We aimed at building models using the same five variables (*#EI*, *#EO*, *#EQ*, *#ILF*, *#EIF*) used by HLFPA. However, the `loess` function from the `Stats` package does not allow more than 4 independent variables. To overcome this problem, we observe that in the HLFPA method, *#EI* and *#EQ* get the same weight; therefore, it is conceivable to consider EIs and EQs as a single class of transactions (only as far as size estimation is concerned). Accordingly, for each project we compute *#EIQ* $=$ *#EI* $+$ *#EQ*. Then we use four independent variables (*#EO*, *#EIQ*, *#ILF*, *#EIF*) to build size models via LOESS. In addition, we built models that use the same two variables (*#LF* and *#EP*) used by SFP. We also built Ordinary Least Square (OLS) linear regression models.

The evaluation was carried out via 10-time 10-fold cross validation. For all the estimates obtained from 10-time 10-fold cross validation, we compute estimation errors and a few indicators, as follows. The error (alias residual) for the i^{th} estimation is defined as $ee_i = S_i - E_i$, where S_i is the actual size of the element involved in the i^{th} estimation (i.e., the size measured according to the IFPUG standard process) and E_i is the estimated size. The computed indicators are:

- MAR is the Mean of Absolute Residuals, i.e., *MAR* $= \frac{1}{n}\sum_{i=0}^{n}|ee_i|$, where n is the number of estimates.
- MR is the MAR divided by the mean size $\frac{1}{n}\sum_{i=0}^{n}S_i$. It gives an idea of the relative importance or the estimation errors.
- MdAR is the median of absolute residuals.
- MdR is MdAR divided by the median size. It gives an idea of the relative importance or the estimation errors, while taking into account that the distribution of sizes is skewed.
- MMRE is the mean magnitude of relative errors. *MMRE* $= \frac{1}{n}\sum_{i=0}^{n}|re_i|$, where $re_i = \frac{ee_i}{S_i}$ is the relative error. MMRE has been widely criticized as a biased metric [19]: we report it for completeness. At any rate, we also report MR, which is not a biased metric, since the mean size is a characteristic of the given dataset: MR is a sort of normalization of the MAR.
- MdMRE is the median magnitude of relative errors.
- Finally, R^2 (the coefficient of determination) is given, since it is a quite reliable indicator of the models' accuracy [20].

To assess the effect size, we use the non-parametric statistic A by Vargha and Delaney [21], as provided by the R package `effsize` [22].

To evaluate if the estimates provided by a method are significantly better than those provided by another method, we tested the statistical significance of the differences among absolute errors yielded by the considered methods [19]. Namely, we compared the absolute residuals via Wilcoxon sign rank test [23] (using the `wilcox.test` function from the R `Stats` package).

D. Evaluation procedure

Our study was carried out in two steps, the first one dealing with within-dataset and the second one with cross-dataset evaluation.

The within-dataset evaluation was carried out using the ISBSG dataset (as reported [1]) and the Chinese dataset. In both cases, we carried out 10-times 10-fold cross validation. In the process, we did not always get usable results. Specifically, via OLS regression we sometimes obtained invalid models (e.g., models with not normally distributed residuals); via LOESS we obtained models that did not support estimation in extreme cases, i.e., for too large or too small independent variables. All these cases were not evaluated. They are a strict minority, hence the reported results represent the most likely outcome of estimation in practice.

Cross-dataset evaluation was straightforward: we built a model (for each of the considered types) using the ISBSG dataset as the training set, and evaluated it using the Chinese dataset as the test set. This operation was then repeated using the Chinese dataset for training and the ISBSG dataset for testing.

E. Results of within-dataset evaluations

This section reports the results obtained for the within-dataset evaluations obtained using first the ISBSG dataset, and then the Chinese dataset.

Results obtained with the ISBSG dataset

The accuracy indicators computed over the estimates that were obtained for the ISBSG dataset are given in Table III. Models LMv are built using OLS regression using v independent variables; models LWMv (where LWM stands for Locally Weighted Model) are built using LOESS, based on v independent variables. For LWMv we give in parentheses the value of the span value.

Table III suggests that OLS linear models provide quite good estimates. Surprisingly, LM4, i.e., the model based on *#EO*, *#EIQ*, *#ILF*, *#EIF* achieves better results than the LM5, i.e., the model based on *#EO*, *#EI*, *#EQ*, *#ILF*, *#EIF*.

We can also observe that estimation accuracy of LWM models varies with the `span`; specifically, accuracy improves with `span`. However, the improvement is modest for LWM2 (MAR decreases from 91.4 to 86.6), while it is quite large for LWM4 (MAR decreases from 93.7 to 55.6). Overall, it seems that when LOESS is used with two variables it is not

TABLE III
WITHIN-DATASET EVALUATION USING THE ISBSG DATASET: ACCURACY INDICATORS.

	MAR	MR	MdAR	MdR	MMRE	MdMRE	R^2
HLFPA	103.8	0.106	58.0	0.091	0.097	0.084	0.966
SFP	87.1	0.089	60.5	0.095	0.105	0.078	0.978
LM5	62.0	0.064	40.6	0.064	0.074	0.057	0.985
LM4	58.2	0.060	39.0	0.061	0.071	0.055	0.987
LM2	91.6	0.096	52.2	0.089	0.096	0.084	0.971
LWM4(0.5)	93.7	0.107	53.5	0.089	0.109	0.089	0.943
LWM2(0.5)	91.4	0.099	56.5	0.089	0.103	0.082	0.940
LWM4(0.75)	66.5	0.076	39.5	0.066	0.082	0.068	0.972
LWM2(0.75)	88.7	0.096	58.2	0.091	0.101	0.075	0.950
LWM4(0.95)	55.6	0.064	37.4	0.062	0.073	0.064	0.984
LWM2(0.95)	86.6	0.094	53.9	0.085	0.096	0.072	0.958

able to substantially improve the estimates provided by LM2; instead, LOESS used with four variables achieves good results, provided that `span` is sufficiently large. In fact, the minimum MAR is achieved by LWM4 with `span`=0.95.

It can also be observed that SFP measures provide an approximation that is better than HLFPA's, and not much worse than the best estimators'. Considering that SFP uses fixed weights and does not even require classifying data and transactions, and that the method is not specifically intended to approximate IFPUG measures, this is a quite remarkable result.

The results of the Wilcoxon sign rank test (which are all statistically significant at the usual $\alpha = 0.05$ level) are given in Table IV, where symbol "$>$" (respectively, "$<$" and "=") in the cell at row i and column j indicates that the model in row i has greater (respectively, smaller and equal) absolute residuals than the model in column j.

TABLE IV
WITHIN-DATASET EVALUATION USING THE ISBSG DATASET: COMPARISON OF MODELS' ABSOLUTE RESIDUALS VIA WILCOXON SIGN RANK TEST.

	HLFPA	SFP	LM5	LM4	LM2	LWM4 (0.5)	LWM2 (0.5)	LWM4 (0.75)	LWM2 (0.75)	LWM4 (0.95)	LWM2 (0.95)
HLFPA	–	>	>	>	>	>	>	>	>	>	>
SFP	<	–	>	>	>	=	>	>	>	>	>
LM5	<	<	–	>	<	<	<	<	<	>	<
LM4	<	<	<	–	<	<	<	<	<	<	<
LM2	<	<	>	>	–	<	<	>	=	>	>
LWM4(.5)	<	=	>	>	>	–	=	>	>	>	>
LWM2(.5)	<	<	>	>	>	=	–	>	>	>	>
LWM4(.75)	<	<	>	>	<	<	<	–	<	>	<
LWM2(.75)	<	<	>	>	=	<	<	>	–	>	>
LWM4(.95)	<	<	<	>	<	<	<	<	<	–	<
LWM2(.95)	<	<	>	>	<	<	<	>	<	>	–

To assess the effect size, we use the non-parametric statistic A by Vargha and Delaney [21], as provided by the R package `effsize` [22]. We obtained the results given in Table V, where each numeric result is accompanied by its interpretation [22]: 'n' and 's' indicate negligible and small effect size, respectively.

LWM4(0.95) appears to be the best model according to MAR (Table III). However, According to the Wilcoxon sign rank test, LM4 is the most accurate model. The disagreement between this two indications is explained by Vargha and Delaney's A, which is 0.51 for LM4 vs. LWM4(0.95), showing that the size effect is practically nil, i.e., LM4 is better, but by a practically irrelevant extent.

Finally, we look into the error distributions yielded by the estimation methods that we used in the study.

Figure 1 shows the boxplots of estimation errors for each of the used methods. It can be noticed that LWM2 models provide exceedingly large errors in a few cases.

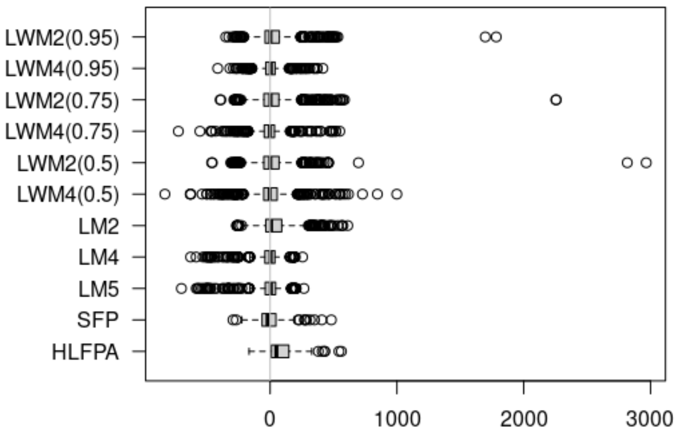

Fig. 1. Within-dataset evaluation using the ISBSG dataset: error boxplots.

Figure 2 provides the same information as Figure 1, but omitting outliers. It can be seen that the various models do not yield dramatically different accuracy levels, when the outliers are excluded. However, it is noteworthy that HLFPA tends to underestimate (as already noted in [16]). The other models provide more balanced errors, with medians very close to zero.

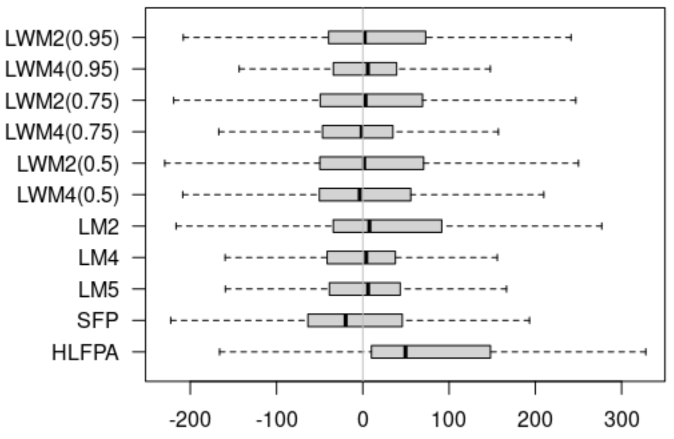

Fig. 2. Within-dataset evaluation using the ISBSG dataset: error boxplots (no outliers).

Figure 3 shows the boxplots of absolute estimation errors for each of the used methods, excluding outliers. The mean absolute error (i.e., the MAR) is shown as an orange diamond. Also according to Figure 3, LM4, LM5 and LWM4(0.95) are the most accurate models.

TABLE V
WITHIN-DATASET EVALUATION USING THE ISBSG DATASET: EFFECT SIZE ACCORDING TO VARGHA AND DELANEY'S A.

	HLFPA	SFP	LM5	LM4	LM2	LWM4 (0.5)	LWM2 (0.5)	LWM4 (0.75)	LWM2 (0.75)	LWM4 (0.95)	LWM2 (0.95)
HLFPA	NA	0.52(n)	0.61(s)	0.62(s)	0.54(n)	0.53(n)	0.53(n)	0.59(s)	0.55(n)	0.61(s)	0.56(n)
SFP	0.48(n)	NA	0.60(s)	0.62(s)	0.52(n)	0.51(n)	0.51(n)	0.58(s)	0.53(n)	0.60(s)	0.54(n)
LM5	0.39(s)	0.40(s)	NA	0.52(n)	0.44(n)	0.42(s)	0.42(s)	0.49(n)	0.43(n)	0.51(n)	0.45(n)
LM4	0.38(s)	0.38(s)	0.48(n)	NA	0.42(s)	0.40(s)	0.40(s)	0.47(n)	0.42(s)	0.49(n)	0.43(n)
LM2	0.46(n)	0.48(n)	0.56(n)	0.58(s)	NA	0.48(n)	0.49(n)	0.55(n)	0.50(n)	0.57(n)	0.51(n)
LWM4(0.5)	0.47(n)	0.49(n)	0.58(s)	0.60(s)	0.52(n)	NA	0.50(n)	0.57(n)	0.52(n)	0.59(s)	0.53(n)
LWM2(0.5)	0.47(n)	0.49(n)	0.58(s)	0.60(s)	0.51(n)	0.50(n)	NA	0.56(n)	0.51(n)	0.58(s)	0.52(n)
LWM4(0.75)	0.41(s)	0.42(s)	0.51(n)	0.53(n)	0.45(n)	0.43(n)	0.44(n)	NA	0.45(n)	0.52(n)	0.47(n)
LWM2(0.75)	0.45(n)	0.47(n)	0.57(n)	0.58(s)	0.50(n)	0.48(n)	0.49(n)	0.55(n)	NA	0.57(n)	0.51(n)
LWM4(0.95)	0.39(s)	0.40(s)	0.49(n)	0.51(n)	0.43(n)	0.41(s)	0.42(s)	0.48(n)	0.43(n)	NA	0.45(n)
LWM2(0.95)	0.44(n)	0.46(n)	0.55(n)	0.57(n)	0.49(n)	0.47(n)	0.48(n)	0.53(n)	0.49(n)	0.55(n)	NA

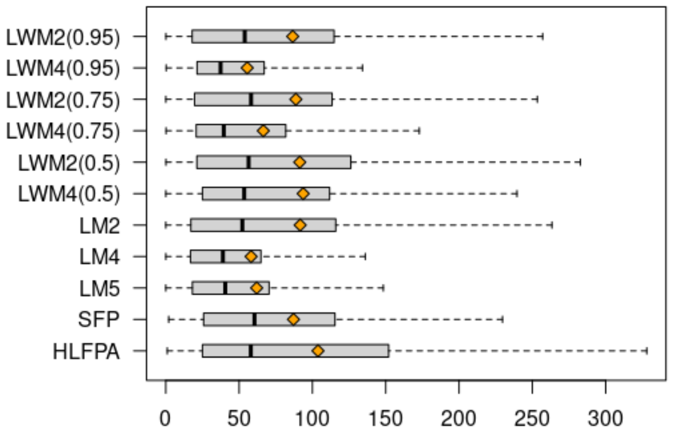

Fig. 3. Within-dataset evaluation using the ISBSG dataset: absolute error boxplots (no outliers).

Results obtained with the Chinese dataset

The accuracy indicators computed over the estimates that were obtained for the Chinese dataset are given in Table VI.

TABLE VI
WITHIN-DATASET EVALUATION USING THE CHINESE DATASET: ACCURACY INDICATORS.

	MAR	MR	MdAR	MdR	MMRE	MdMRE	R^2
HLFPA	119.0	0.088	69.0	0.066	0.095	0.077	0.970
SFP	154.3	0.114	91.9	0.088	0.124	0.108	0.945
LM5	121.0	0.089	78.1	0.076	0.104	0.087	0.972
LM4	128.3	0.095	75.7	0.074	0.105	0.087	0.964
LM2	131.8	0.097	75.8	0.073	0.108	0.088	0.960
LWM4(0.5)	151.9	0.115	82.3	0.081	0.119	0.098	0.942
LWM2(0.5)	116.6	0.087	69.3	0.068	0.104	0.083	0.970
LWM4(0.75)	154.7	0.117	79.9	0.079	0.120	0.104	0.939
LWM2(0.75)	118.7	0.089	74.9	0.073	0.104	0.083	0.970
LWM4(0.95)	123.6	0.094	74.9	0.074	0.106	0.090	0.966
LWM2(0.95)	118.8	0.089	77.8	0.076	0.104	0.082	0.970

Table VI shows that HLFPA provides quite good estimates: better than those achieved for the ISBSG dataset, according to MR. OLS linear models provide estimates that are slightly less accurate than HLFPA's; as expected, the fewer independent variables are used, the less accurate the estimates. Surprisingly, models LM4 (regardless `span`) perform worse than LMW2, which achieve the smallest MAR.

The results of the Wilcoxon sign rank test (which are all statistically significant at the usual $\alpha = 0.05$ level) are given in Table VII.

TABLE VII
WITHIN-DATASET EVALUATION OF THE CHINESE DATASET: COMPARISON OF MODELS' ABSOLUTE RESIDUALS VIA WILCOXON SIGN RANK TEST.

	HLFPA	SFP	LM5	LM4	LM2	LWM4 (0.5)	LWM2 (0.5)	LWM4 (0.75)	LWM2 (0.75)	LWM4 (0.95)	LWM2 (0.95)
HLFPA	–	<	<	<	<	<	=	<	<	<	<
SFP	>	–	>	>	>	=	>	>	>	>	>
LM5	>	<	–	>	>	<	>	<	>	>	>
LM4	>	<	<	–	=	<	>	<	>	>	>
LM2	>	<	<	=	–	<	>	<	=	=	>
LWM4(0.5)	>	=	>	>	>	–	>	=	>	>	>
LWM2(0.5)	=	<	<	<	<	<	–	<	<	<	<
LWM4(0.75)	>	<	>	>	>	=	>	–	>	>	>
LWM2(0.75)	>	<	<	<	=	<	>	<	–	=	>
LWM4(0.95)	>	<	<	<	=	<	>	<	=	–	>
LWM2(0.95)	>	<	<	<	<	<	>	<	<	<	–

According to the Wilcoxon sign rank test, HLFPA provides smaller absolute errors than all other models, except for LWM2(0.5). At any rate, Vargha and Delaney's A, indicates that all model pairs are likely to provide very similar absolute residuals. Finally, we look into the error distributions yielded

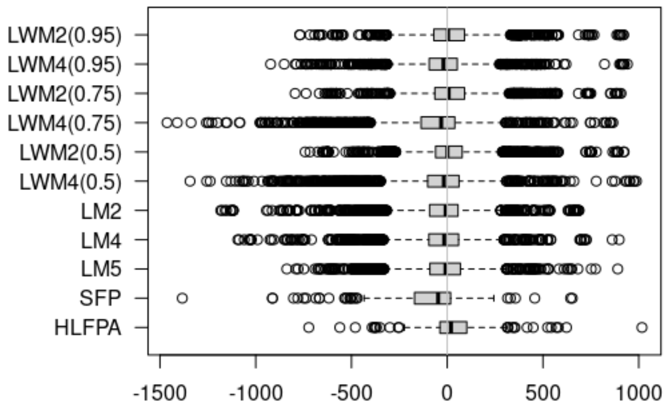

Fig. 4. Within-dataset evaluation using the Chinese dataset: error boxplots.

by the estimation methods that we used in the study. Figure 4 shows the boxplots of estimation errors for each of the used methods. The same boxplots are shown in Figure 5 without outliers, to improve readability. It can be noticed

TABLE VIII
WITHIN-DATASET EVALUATION USING THE CHINESE DATASET: EFFECT SIZE ACCORDING TO VARGHA AND DELANEY'S A.

	HLFPA	SFP	LM5	LM4	LM2	LWM4 (0.5)	LWM2 (0.5)	LWM4 (0.75)	LWM2 (0.75)	LWM4 (0.95)	LWM2 (0.95)
HLFPA	NA	0.45(n)	0.48(n)	0.48(n)	0.49(n)	0.46(n)	0.50(n)	0.46(n)	0.49(n)	0.49(n)	0.49(n)
SFP	0.55(n)	NA	0.53(n)	0.53(n)	0.53(n)	0.51(n)	0.55(n)	0.51(n)	0.54(n)	0.53(n)	0.54(n)
LM5	0.52(n)	0.47(n)	NA	0.51(n)	0.51(n)	0.48(n)	0.52(n)	0.48(n)	0.51(n)	0.51(n)	0.51(n)
LM4	0.52(n)	0.47(n)	0.49(n)	NA	0.50(n)	0.47(n)	0.52(n)	0.48(n)	0.50(n)	0.50(n)	0.51(n)
LM2	0.51(n)	0.47(n)	0.49(n)	0.50(n)	NA	0.47(n)	0.51(n)	0.48(n)	0.50(n)	0.50(n)	0.50(n)
LWM4(0.5)	0.54(n)	0.49(n)	0.52(n)	0.53(n)	0.53(n)	NA	0.54(n)	0.50(n)	0.53(n)	0.53(n)	0.53(n)
LWM2(0.5)	0.50(n)	0.45(n)	0.48(n)	0.48(n)	0.49(n)	0.46(n)	NA	0.46(n)	0.49(n)	0.49(n)	0.49(n)
LWM4(0.75)	0.54(n)	0.49(n)	0.52(n)	0.52(n)	0.52(n)	0.50(n)	0.54(n)	NA	0.53(n)	0.53(n)	0.53(n)
LWM2(0.75)	0.51(n)	0.46(n)	0.49(n)	0.50(n)	0.50(n)	0.47(n)	0.51(n)	0.47(n)	NA	0.50(n)	0.50(n)
LWM4(0.95)	0.51(n)	0.47(n)	0.49(n)	0.50(n)	0.50(n)	0.47(n)	0.51(n)	0.47(n)	0.50(n)	NA	0.50(n)
LWM2(0.95)	0.51(n)	0.46(n)	0.49(n)	0.49(n)	0.50(n)	0.47(n)	0.51(n)	0.47(n)	0.50(n)	0.50(n)	NA

that, as already observed for the ISBSG dataset, HLFPA tends to underestimate. All the other models either provide estimation errors that are equally distributed between negative and positive, or (like SiFP, LWM4(0.75) and LWM4(0.95)) overestimate.

Figure 6 shows the boxplots of absolute estimation errors for each of the used methods, excluding outliers. The mean absolute error (i.e., the MAR) is shown as an orange diamond. Figure 6 shows that most models provide similar accuracy. The only models that yield evidently less accurate estimates are SiFP, LWM4(0.5) and LWM4(0.75). Concerning SiFP, it is useful reminding that it is not an estimation method, hence it is not correct to talk about estimation errors, in this case; rather, we should talk about the distance between SiFP measures and standard FPA size.

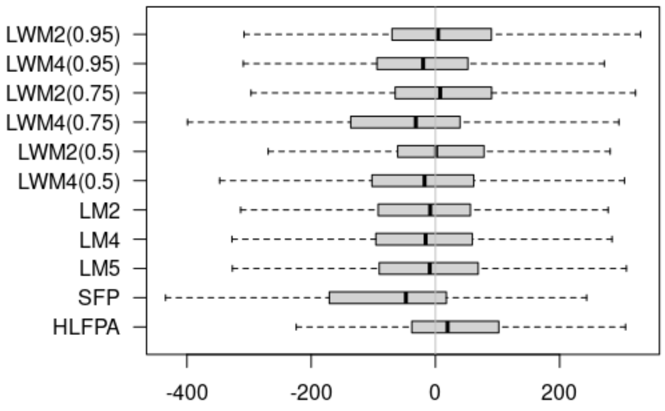

Fig. 5. Within-dataset evaluation using the Chinese dataset: error boxplots (no outliers).

With both datasets, the lowest MAR is obtained by using a LOESS approach, although with different spans. This confirms the flexibility of the method and its adaptability to different datasets after a tuning phase regarding the configuration of the span based on the peculiarities of each dataset.

F. Results of cross-dataset evaluations

This activity consisted of two steps:

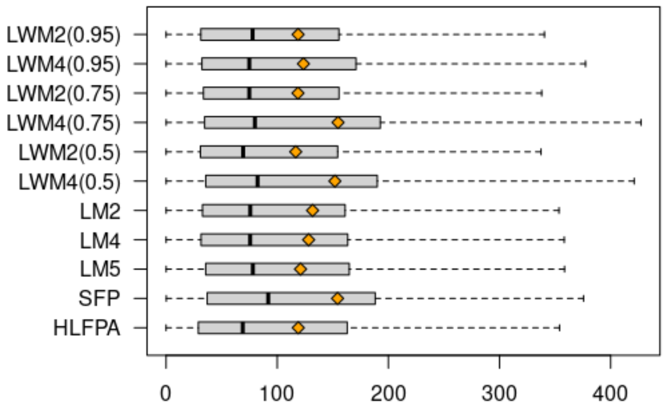

Fig. 6. Within-dataset evaluation using the Chinese dataset: absolute error boxplots (no outliers).

1) We built models using the ISBSG dataset as the training set and used the obtained model to estimate the size of projects in the Chinese dataset.
2) We built models using the Chinese dataset as the training set and used the obtained model to estimate the size of projects in the ISBSG dataset.

TABLE IX
CROSS-DATASET EVALUATION (TRAINING SET ISBSG, TEST SET CHINESE): ACCURACY INDICATORS.

	MAR	MR	MdAR	MdR	MMRE	MdMRE	R^2
HLFPA	119.0	0.088	69.0	0.066	0.095	0.077	0.970
SFP	154.3	0.114	91.9	0.088	0.124	0.108	0.945
LM5	140.7	0.104	82.8	0.080	0.112	0.088	0.955
LM4	143.1	0.106	85.3	0.082	0.113	0.090	0.953
LWM4(0.5)	403.3	0.322	147.8	0.135	0.273	0.188	0.144
LWM2(0.5)	218.1	0.148	144.5	0.114	0.146	0.123	0.859
LWM4(0.75)	315.7	0.252	129.9	0.119	0.208	0.152	0.534
LWM2(0.75)	180.2	0.122	114.2	0.090	0.119	0.107	0.920
LWM4(0.95)	241.4	0.192	106.4	0.097	0.163	0.115	0.724
LWM2(0.95)	168.5	0.114	113.5	0.090	0.113	0.100	0.929

When considering point 1) the comparison of Table VI with Table IX shows that prediction accuracy decreases for all models when "foreign" data are used for training. Of course, the accuracy obtained by HLFPA and SFP do not change, since these predictions are not obtained from any dataset.

Noticeably, models obtained via linear regression achieve a level of accuracy that is quite close to HLFPA's and slightly

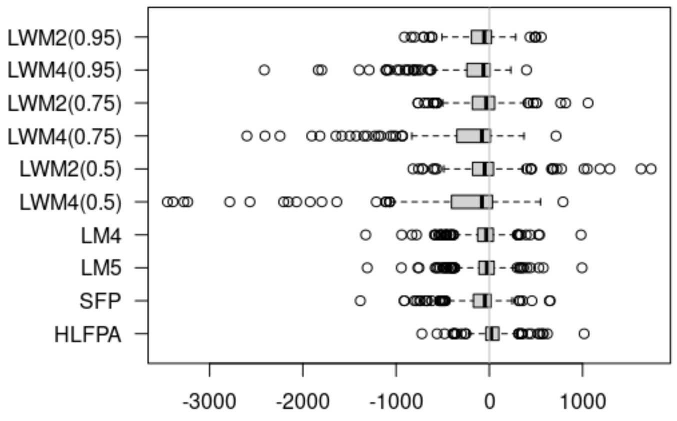

Fig. 7. Cross-dataset evaluation (training set ISBSG, test set Chinese): error boxplots.

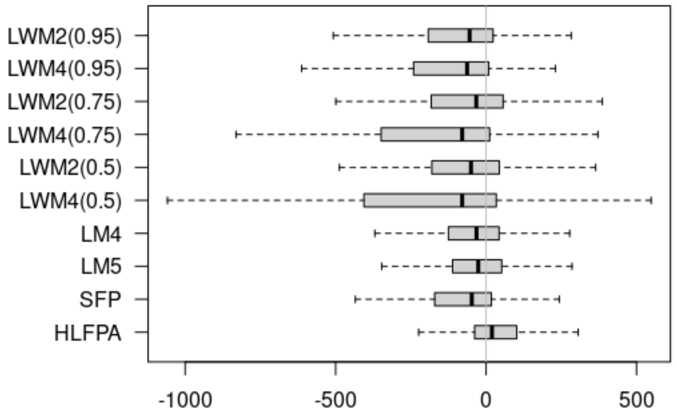

Fig. 8. Cross-dataset evaluation (training set ISBSG, test set Chinese): error boxplots (no outliers).

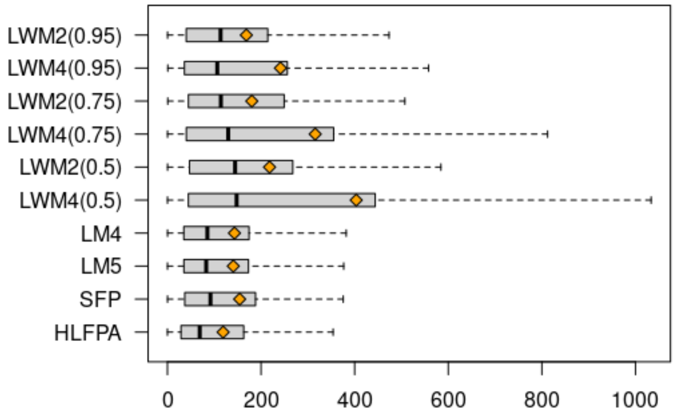

Fig. 9. Cross-dataset evaluation (training set ISBSG, test set Chinese): absolute error boxplots (no outliers).

TABLE X
CROSS-DATASET EVALUATION (TRAINING SET ISBSG, TEST SET CHINESE): COMPARISON OF MODELS' ABSOLUTE RESIDUALS VIA WILCOXON SIGN RANK TEST.

	HLFPA	SFP	LM5	LM4	LWM4 (0.5)	LWM2 (0.5)	LWM4 (0.75)	LWM2 (0.75)	LWM4 (0.95)	LWM2 (0.95)
HLFPA	–	<	<	<	<	<	<	<	<	<
SFP	>	–	>	>	<	<	<	<	<	<
LM5	>	<	–	=	<	<	<	<	<	<
LM4	>	<	=	–	<	<	<	<	<	<
LWM4(0.5)	>	>	>	>	–	>	>	>	>	>
LWM2(0.5)	>	>	>	>	<	–	=	>	>	>
LWM4(0.75)	>	>	>	>	<	=	–	>	>	>
LWM2(0.75)	>	>	>	>	<	<	<	–	>	>
LWM4(0.95)	>	>	>	>	<	<	<	<	–	=
LWM2(0.95)	>	>	>	>	<	<	<	<	=	–

better than SFP's. However, this applies for models using 4 or 5 variables; no valid model using 2 variables could be found via linear regression. LOESS models appear definitely less accurate, although LWM2(0.95) appear only slightly less accurate than SFP.

The results of the Wilcoxon sign rank test are given in Table X. The results of the Vargha and Delaney's A test are given in Table XI.

According to the Wilcoxon sign rank test, HLFPA is the most accurate method, although according to A, the difference in accuracy is negligible when compared to SFP and linear regression models, and small when compared to LOESS models.

Figure 7 and Figure 8 show the boxplots of estimation errors for each of the used methods with and without outliers, respectively.

From both figures it can be noticed that, as already observed for the ISBSG and the Chinese dataset, HLFPA tends to underestimate. All the other models tend to overestimate, in some cases by fairly large amounts.

Figure 9 shows the boxplots of absolute estimation errors for each of the used methods, excluding outliers. It can be noticed that HLFPA, SFP and LM models provide similar and the better accuracy. All LWM4 models yield evidently less accurate estimates than LWM2.

When considering point 2) i.e., the estimation of the ISBSG dataset via models obtained from the Chinese dataset, the comparison of Table III with Table XII confirms that prediction accuracy decreases for all models when "foreign" data are used for training.

However, both linear regression and LOESS models achieve better results than HLFPA when using 4 or 5 variables. Among 2-variable models, SFP and linear regression appear more accurate than HLFPA, while LOESS models achieve slightly worse accuracy.

The results of the Wilcoxon sign rank test are given in Table XIII. The results of the Vargha and Delaney's A test are given in Table XIV.

According to the Wilcoxon sign rank test, LOESS models using 4 variables are the most accurate. According to A, LOESS models using 4 variables provide a small advantage

TABLE XI
CROSS-DATASET EVALUATION (TRAINING SET ISBSG, TEST SET CHINESE): EFFECT SIZE ACCORDING TO VARGHA AND DELANEY'S A.

	HLFPA	SFP	LM5	LM4	LWM4 (0.5)	LWM2 (0.5)	LWM4 (0.75)	LWM2 (0.75)	LWM4 (0.95)	LWM2 (0.95)
HLFPA	NA	0.45(n)	0.47(n)	0.47(n)	0.34(s)	0.36(s)	0.39(s)	0.40(s)	0.42(s)	0.42(s)
SFP	0.55(n)	NA	0.52(n)	0.51(n)	0.38(s)	0.41(s)	0.43(s)	0.44(n)	0.46(n)	0.46(n)
LM5	0.53(n)	0.48(n)	NA	0.50(n)	0.37(s)	0.39(s)	0.41(s)	0.43(n)	0.45(n)	0.45(n)
LM4	0.53(n)	0.49(n)	0.50(n)	NA	0.37(s)	0.40(s)	0.41(s)	0.43(n)	0.45(n)	0.45(n)
LWM4(0.5)	0.66(s)	0.62(s)	0.63(s)	0.63(s)	NA	0.54(n)	0.54(n)	0.57(n)	0.58(s)	0.59(s)
LWM2(0.5)	0.64(s)	0.59(s)	0.61(s)	0.60(s)	0.46(n)	NA	0.50(n)	0.53(n)	0.54(n)	0.56(n)
LWM4(0.75)	0.61(s)	0.57(s)	0.59(s)	0.59(s)	0.46(n)	0.50(n)	NA	0.53(n)	0.54(n)	0.54(n)
LWM2(0.75)	0.60(s)	0.56(n)	0.57(n)	0.57(n)	0.43(n)	0.47(n)	0.47(n)	NA	0.51(n)	0.52(n)
LWM4(0.95)	0.58(s)	0.54(n)	0.55(n)	0.55(n)	0.42(s)	0.46(n)	0.46(n)	0.49(n)	NA	0.50(n)
LWM2(0.95)	0.58(s)	0.54(n)	0.55(n)	0.55(n)	0.41(s)	0.44(n)	0.46(n)	0.48(n)	0.50(n)	NA

over HLFPA and SFP, while the advantage is negligible with respect to linear regression models.

Figure 10 and Figure 11 show the boxplots of estimation errors for each of the used methods, with and without outliers, respectively. The boxplots show that most methods tend to underestimate. LWM4 models are either well balanced or tend to overestimate. Similarly, SFP tends to overestimate.

Figure 12 shows the boxplots of absolute estimation errors for each of the used methods, excluding outliers. It can be noticed that the better accuracy is provided by LMW4 methods, while HLFPA, LM2 and all the LMW2 provide similar and worse accuracy with respect to the other methods. LM methods are between those extremes.

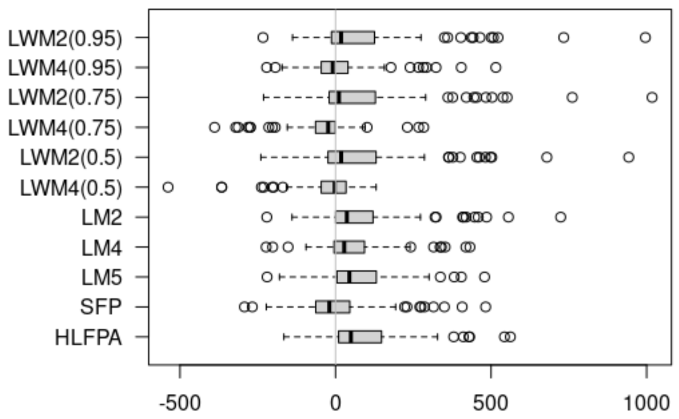

Fig. 10. Cross-dataset evaluation (training set Chinese, test set ISBSG): error boxplots.

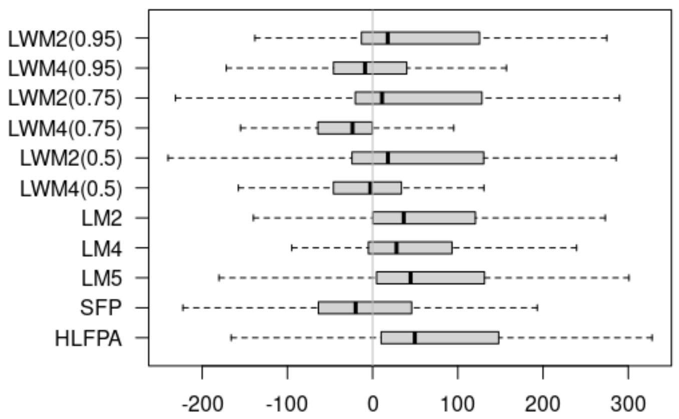

Fig. 11. Cross-dataset evaluation (training set Chinese, test set ISBSG): error boxplots (no outliers).

TABLE XII
CROSS-DATASET EVALUATION (TRAINING SET CHINESE, TEST SET ISBSG): ACCURACY INDICATORS.

	MAR	MR	MdAR	MdR	MMRE	MdMRE	R2
HLFPA	103.8	0.106	58.0	0.091	0.097	0.084	0.966
SFP	87.1	0.089	60.5	0.095	0.105	0.078	0.978
LM5	90.3	0.093	51.8	0.081	0.090	0.083	0.976
LM4	81.9	0.084	48.5	0.076	0.086	0.080	0.978
LM2	108.0	0.111	57.8	0.091	0.106	0.100	0.959
LWM4(0.5)	63.6	0.069	35.5	0.057	0.075	0.058	0.980
LWM2(0.5)	115.1	0.118	62.7	0.098	0.107	0.094	0.947
LWM4(0.75)	69.8	0.076	50.8	0.082	0.082	0.063	0.979
LWM2(0.75)	118.5	0.121	65.2	0.102	0.108	0.088	0.941
LWM4(0.95)	66.9	0.073	42.9	0.069	0.077	0.059	0.978
LWM2(0.95)	113.7	0.117	58.4	0.091	0.102	0.087	0.945

TABLE XIII
CROSS-DATASET EVALUATION (TRAINING SET CHINESE, TEST SET ISBSG): COMPARISON OF MODELS' ABSOLUTE RESIDUALS VIA WILCOXON SIGN RANK TEST.

	HLFPA	SFP	LM5	LM4	LWM4 (0.5)	LWM2 (0.5)	LWM4 (0.75)	LWM2 (0.75)	LWM4 (0.95)	LWM2 (0.95)	
HLFPA	–	>	>	>	=	>	=	>	=	>	>
SFP	<	–	=	>	<	>	=	>	=	>	=
LM5	<	=	–	>	<	>	<	>	=	>	=
LM4	<	<	<	–	<	>	<	>	<	>	<
LM2	=	>	>	>	–	>	=	>	=	>	>
LWM4(0.5)	<	<	<	<	<	–	<	<	<	=	<
LWM2(0.5)	=	=	>	>	=	>	–	>	=	>	=
LWM4(0.75)	<	<	<	<	<	>	<	–	<	>	<
LWM2(0.75)	=	=	=	>	=	>	=	>	–	>	=
LWM4(0.95)	<	<	<	<	<	=	<	<	<	–	<
LWM2(0.95)	<	=	=	>	<	>	=	>	=	>	–

IV. DISCUSSION

In this section we discuss the obtained results from two points of view: a technical one (in Section IV-A) and a managerial one (in Section IV-B).

TABLE XIV
CROSS-DATASET EVALUATION (TRAINING SET CHINESE, TEST SET ISBSG): EFFECT SIZE ACCORDING TO VARGHA AND DELANEY'S A.

	HLFPA	SFP	LM5	LM4	LM2	LWM4 (0.5)	LWM2 (0.5)	LWM4 (0.75)	LWM2 (0.75)	LWM4 (0.95)	LWM2 (0.95)
HLFPA	NA	0.52(n)	0.52(n)	0.55(n)	0.50(n)	0.61(s)	0.51(n)	0.59(s)	0.51(n)	0.60(s)	0.53(n)
SFP	0.48(n)	NA	0.51(n)	0.54(n)	0.48(n)	0.60(s)	0.49(n)	0.58(s)	0.49(n)	0.59(s)	0.51(n)
LM5	0.48(n)	0.49(n)	NA	0.53(n)	0.47(n)	0.58(s)	0.48(n)	0.56(n)	0.49(n)	0.58(s)	0.50(n)
LM4	0.45(n)	0.46(n)	0.47(n)	NA	0.45(n)	0.56(n)	0.45(n)	0.53(n)	0.45(n)	0.55(n)	0.47(n)
LM2	0.50(n)	0.52(n)	0.53(n)	0.55(n)	NA	0.61(s)	0.50(n)	0.58(s)	0.51(n)	0.60(s)	0.52(n)
LWM4(0.5)	0.39(s)	0.40(s)	0.42(s)	0.44(n)	0.39(s)	NA	0.40(s)	0.47(n)	0.41(s)	0.49(n)	0.42(s)
LWM2(0.5)	0.49(n)	0.51(n)	0.52(n)	0.55(n)	0.50(n)	0.60(s)	NA	0.58(s)	0.50(n)	0.59(s)	0.51(n)
LWM4(0.75)	0.41(s)	0.42(s)	0.44(n)	0.47(n)	0.42(s)	0.53(n)	0.42(s)	NA	0.43(s)	0.52(n)	0.44(n)
LWM2(0.75)	0.49(n)	0.51(n)	0.51(n)	0.55(n)	0.49(n)	0.59(s)	0.50(n)	0.57(s)	NA	0.59(s)	0.52(n)
LWM4(0.95)	0.40(s)	0.41(s)	0.42(s)	0.45(n)	0.40(s)	0.51(n)	0.41(s)	0.48(n)	0.41(s)	NA	0.43(n)
LWM2(0.95)	0.47(n)	0.49(n)	0.50(n)	0.53(n)	0.48(n)	0.58(s)	0.49(n)	0.56(n)	0.48(n)	0.57(n)	NA

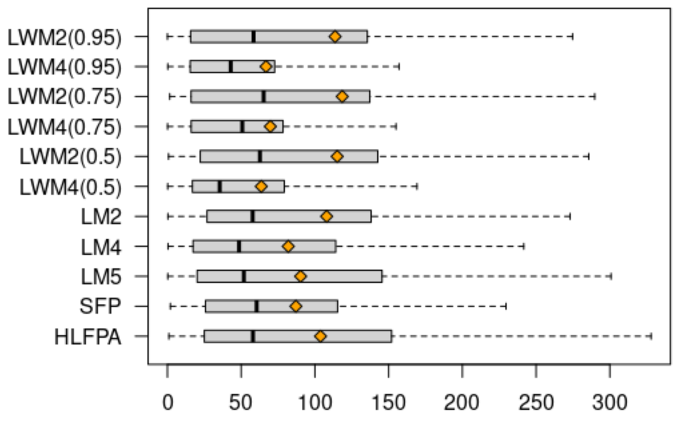

Fig. 12. Cross-dataset evaluation (training set Chinese, test set ISBSG): absolute error boxplots (no outliers).

A. Technical discussion

The approaches to size estimation presented in the previous sections correspond to different model building strategies, which are based on different assumptions and require different types of knowledge. In fact,

- HLFPA exploits the knowledge of how FPA works. According to FPA, the measure of size is obtained as a weighted sum of the numbers of EI, EO, EQ, ILF and EIF. HLFPA adopts exactly the same schema. HLFPA does not rely on any data, i.e., the model is fixed and does not depend on the characteristics of the known projects. In other words, HLFPA does not try to learn from data; instead, it simply adopts fixed weights, namely low complexity weights for data and medium complexity weights for transactions.
- SFP works along similar lines. Structurally, it is a simplified version of FPA. Like HLFPA, it does not learn from data, i.e., it does not try to adapt to the characteristics of the known projects. Even though the weights to be used were originally derived by the observation of data from real projects, these weights are now fixed and apply to whatever project has to be measured.
- OLS exploit, like HLFPA, the knowledge of the structure of FPA sizing, in that they model size as a linear function of the numbers of EI, EO, EQ, ILF and EIF. In addition, OLS linear regression models also exploit data from known projects, since they derive the weights to be used in the computation of size from historical data. Accordingly, any organization owing suitable historical data can build its own OLS model.
- LOESS is a more flexible method with respect to OLS in that it builds models based on ML approaches (like nearest neighbours), also keeping the simplicity of regression models. Using locality principles, it may possibly yield more accurate estimates than OLS methods.

The size of the dataset may hinder the performance of the LOESS method. As a counterpart, in cross-dataset validation, LOESS models showed the best performances of the whole set of experiments. This may suggest that the generalizability of this approach should be further analyzed in search of specific conditions for a better performance of the algorithm.

B. Managerial Discussion

From the managerial standpoint, LOESS has some limitations and potential, depending on its use and application context.

With respect to HLFPA and SFP, the LOESS-based methods have the disadvantange that they need to be trained on a dataset, while the former models are fixed formulae (see (1) and (2)) that just need measures from the project being estimated. Therefore, an historical dataset is needed, and using "foreign" data may not work well, as in the case of the models trained on the ISBSG dataset and use to estimate projects from the Chinese dataset. However, using LOESS models yielded quite accurate estimates in several cases, therefore it is seems that build LOESS models is worth trying, when data are available for training. In this respect, the work needed to build LOESS models is similar to the work needed to build linear regression models, which is a fairly common activity.

It must also be considered that in some contexts, like public sectors, for instance, estimates base on LOESS models may be difficult to accept, depending on the kind of contractors. An estimation tool like LOESS could seem not transparent enough to yield reliable estimated to be agreed upon.

However, in general, from the organizational and managerial perspectives, using the LOESS method could be useful for the early assessment of the feasibility of a project before any elicitation phase, as in the case of agile methodologies. Pursuing the study of functional size estimation via LOESS may act as a proof of concept mechanism to help identify project features; to simulate and quantify the average error and intrinsic residuality of early estimation methods vs post-hoc measurement; to help compare functional size models and estimation procedures and their measurement validity and reliability.

A further opportunity represented by this approach is that of introducing evolutionary-wise estimation methods, whereby different outcomes may come from the identification of the same BFC (and whereby, in this respect, fixed weights methods would always return the same outcome). In this light, LOESS may represent a more situated approach, evolving through time and in line with factors characterizing and influencing from time to time the productive system.

C. Applicability

In this section, the practical applicability of LOESS for functional size estimation is briefly discussed.

First of all, to use LOESS, we need historical data. Besides the usual requirements for data, we need data that represent the entire size range in which we are interested. Specifically, LOESS requires fairly large, densely sampled data sets in order to produce good models. Remember that LOESS performs local fitting, therefore fairly complete information concerning BFC configurations have to be available.

Besides data, we just need a reasonable computer environment. A modern PC running the R environment (the `loess` function is available by default).

As we reported above, LOESS works well, but does not always provide the best estimates. Therefore, we do not recommend replacing estimation practices based on HLFPA or linear regression models, for instance, with LOESS right away. Instead, it can be useful to use LOESS alongside other estimation methods. In this way, if LOESS results agree with other methods' estimates you increase your confidence in the correctness of estimates. Otherwise, i.e., if LOESS disagrees with other methods' estimate, you should regard all the obtained estimates as subject to some uncertainty.

V. Threats to validity

A typical concern in this kind of studies is the generalizability of results outside the scope and context of the analyzed dataset. In our case, the ISBSG dataset is deemed the standard benchmark among the community, and it includes data from several application domains. Therefore our results may be valid in general. However, this dataset resulted too small for local approaches like LOESS, which showed its effectiveness and efficiency when applied to a larger dataset as the Chinese one. This may also suggest a limitation of the approach related to the specific dataset that each time is used. For this reason, the problem of generalizability remains crucial.

The usage of MMRE is questionable, since it is has been shown to be a biased indicator (see for instance [19]). Nonetheless, we used MMRE together with other indicators—like MAR, the boxplots of residuals and R^2—to provide a more complete and balanced picture of the accuracy of our results, and compared the precision of different models via sound statistical tests, namely Wilcoxon sign rank test and Vargha and Delaney's A measure of effect size. Therefore, the role of MMRE in the presented evaluations is marginal. Although the comparison of precision did not always yield significant differences, it is nonetheless a formal and robust method for comparing the used techniques.

VI. Related work

The quest for measures that are available in the early stages of the software lifecycle dates back to decades ago [37] [38] [30].

The "Early & Quick Function Point" (EQFP) method [32] uses analogy (similarities between a new and a classified piece of software) and analysis (statistical analysis of the estimated similarity) to get size estimates. It was reported that estimates are within $\pm 10\%$ of the real size in most real cases, while the savings in time and costs are between 50% and 90%.

"Easy Function Points," [39], adopt probabilistic approaches to estimate not only the size, but also the probability that the actual size is equal to the estimate.

Lavazza et al. built estimation models for UFP based on BFCs [40] using Least Median Squares robust regression models. They observed that FP measures could be altogether replaced by measured based on a smaller set of BFCs.

Several other early estimation methods were proposed: Table XV list the most popular ones.

Lavazza and Liu [11] used 7 real-time applications and 6 non real-time applications to evaluate the accuracy of the E&QFP [30] 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.

Lavazza and Liu [16] used a dataset containing data from 479 projects to compare the accuracy of HLFPA method with Ordinary Least Squares method, with both 5 predictors (LM5) and only 2 predictors (LM2). Their conclusions were that, although HLFPA method is sufficiently accurate for practical usage, it tends to underestimate effort. Since underestimation may lead to unrealistic development plans and possibly to project failure, the authors looked for motivations of HLFPA method underestimation behaviour, finding that it assumes that data functions are mainly of low complexity and transaction functions are mainly of medium complexity, while in the considered dataset it was not so. An alternative strategy they derived from it is to compute linear regression in order to derive the most likely weight by analyzing the data from projects. They found that (1) unlike HLFPA, linear regression models do not underestimate, (2) linear regression models yield slightly less accurate estimates, and (3) models based on only two variables yield marginally less accurate estimates.

TABLE XV
EARLY ESTIMATION METHODS: DEFINITIONS AND EVALUATIONS

Method name	Definition	Used functions	Weight	Evaluation
NESMA indicative	[24] [25]	data	fixed	[5] [15], [26]–[29] [11]
NESMA estimated	[24] [25]	all functions	fixed	[5] [15], [26]–[29] [11]
Early & Quick FP	[30] [31] [32]	all functions	statistics	[11] [33]
Tichenor ILF model	[34]	ILF	fixed	[11]
simplified FP (sFP)	[35]	all functions	fixed	[11]
ISBSG average weights	[36]	all functions	statistics	[11]
SiFP	[6]	data and trans.	statistics	[12] [13]

Also Machine learning (ML) techniques have proved to provide quite good estimation models, in several different domains and situations, and are increasingly being used in software project management activities [41], [42]. A review of the usage of ML for software project management [42] reported that ML is used for software effort and cost estimation: the reported accuracy spans from 91% for cost estimation with K-NN (K-Nearest Neighbours), to 92% for effort prediction with Decision Trees, and 99% for effort estimation with Random Forests.

Local regression methods are extensively used for DNA microarray normalization studies [43], as well as for studying spatiotemporal trends, and improving image resolution and forecasts predictions. They have also been used for hand tracking rapid movements in Human-Computer Interaction studies [44]. However, regarding software size estimation, only a few study have focused on the use of LOESS (see for example [45]), by comparing this method with other ML approaches. In this paper, we are interested in the estimation of functional size, which is generally the main input for effort estimation. Approaches based on local regression have been rarely adopted in this field. We hope to have contributed in a constructive way to better introduce this technique for the analysis and the modelling of software functional size.

VII. Conclusion

Measuring software functional size via IFPUG FPA with the standard manual process is sometimes a long and expensive activity, and it is simply impossible when the details of a functional specification are not available for any reason. To solve this problem, several early estimation methods have been proposed. In this paper, we compare the estimates obtained via a standard estimation methods, namely HLFPA, and a new functional size measurement method, namely IFPUG SFP, with the estimates obtained with traditional (namely, linear regression) models and LOESS models.

To evaluate the accuracy of the functional size estimates provided by the considered methods, we performed both within-dataset and cross-dataset studies. Specifically, we performed two within-datasets analyses, one using an ISBSG dataset containing data from 110 projects and one using a dataset containing data from 276 software projects developed and used by a Chinese financial enterprise. We then performed two cross-datasets analysis: in the first one the ISBSG dataset was used for training and the Chinese dataset was used for testing; in the second one the Chinese dataset was used for training and the ISBSG dataset was used for testing.

When performing within-dataset evaluation using the ISBSG dataset, the LOESS and linear regression models provided the best MAR. Among models using only two variables (unclassified data and transaction functions) the LOESS and SFP models provided the best MAR.

When performing within-dataset evaluation using the Chinese dataset, HLFPA provided the best MAR, with the linear regression and LOESS models providing very similar performance. Among models using only two variables the LOESS model provided the best results, even better than HLFPA's.

When using the ISBSG dataset to train models and the Chinese dataset for testing, HLFPA was definitely most accurate than other models. However, when using the Chinese dataset to train models and the ISBSG dataset for testing, the LOESS model provided definitely the best results. SFP proved also quite good.

We assessed the effect size via the non-parametric statistic A by Vargha and Delaney; we also compared the absolute residuals via Wilcoxon sign rank test to evaluate if the estimates provided by a method are significantly better than those provided by another method. In general, the obtained results show that no methods appears consistently better than others, and the differences are small or even negligible.

In conclusion, even though there is no clear winner, the LOESS method provided generally quite good results; therefore, practitioners needing to estimate software functional size in the early stages of projects are advised to try also LOESS models.

Among future work, we envision the following activities:

- Comparing LOESS estimates with those produced by machine learning techniques [46].
- Study LOESS estimates with confidence intervals.
- Evaluating size estimates obtained via LOESS models, when used for effort estimation.

Acknowledgment

The work reported here was partly supported by Fondo per la Ricerca di Ateneo, Università degli Studi dell'Insubria.

References

[1] L. Lavazza, A. Locoro, and R. Meli, "Using Locally Weighted Regression to Estimate the Functional Size of Software: a Preliminary Study," in Proceedings of IARIA Congress 2022: The 2022 IARIA Annual Congress on Frontiers in Science, Technology, Services, and Applications, 2022, pp. 20–24.

[2] A. J. Albrecht, “Measuring application development productivity,” in Proceedings of the joint SHARE/GUIDE/IBM application development symposium, vol. 10, 1979, pp. 83–92.

[3] International Function Point Users Group (IFPUG), “Function point counting practices manual, release 4.3.1,” 2010.

[4] A. Timp, “uTip – Early Function Point Analysis and Consistent Cost Estimating,” 2015, uTip # 03 – (version # 1.0 2015/07/01).

[5] H. van Heeringen, E. van Gorp, and T. Prins, “Functional size measurement-accuracy versus costs–is it really worth it?” in Software Measurement European Forum (SMEF), 2009.

[6] R. Meli, “Simple function point: a new functional size measurement method fully compliant with IFPUG 4.x,” in Software Measurement European Forum, 2011.

[7] IFPUG, “Simple Function Point (SFP) Counting Practices Manual Release 2.1,” 2021.

[8] International Standardization Organization (ISO), “ISO/IEC 20926: 2003, Software engineering – IFPUG 4.1 Unadjusted functional size measurement method – Counting Practices Manual,” 2003.

[9] L. Lavazza, “On the effort required by function point measurement phases,” International Journal on Advances in Software, vol. 10, no. 1 & 2, 2017.

[10] nesma, “Early Function Point Analysis,” https://nesma.org/themes/sizing/function-point-analysis/early-function-point-counting/ last access 6/6/22.

[11] L. Lavazza and G. Liu, “An empirical evaluation of simplified function point measurement processes,” Journal on Advances in Software, vol. 6, no. 1& 2, 2013.

[12] L. Lavazza and R. Meli, “An evaluation of simple function point as a replacement of IFPUG function point,” in IWSM–MENSURA 2014. IEEE, 2014, pp. 196–206.

[13] F. Ferrucci, C. Gravino, and L. Lavazza, “Simple function points for effort estimation: a further assessment,” in 31st Annual ACM Symposium on Applied Computing. ACM, 2016, pp. 1428–1433.

[14] International Software Benchmarking Standards Group, ““Worldwide Software Development: The Benchmark, release 11,” ISBSG, 2009.

[15] L. Lavazza and G. Liu, “An Empirical Evaluation of the Accuracy of NESMA Function Points Estimates,” in ICSEA, 2019, pp. 24–29.

[16] G. Liu and L. Lavazza, “Early and quick function points analysis: Evaluations and proposals,” Journal of Systems and Software, vol. 174, 2021, p. 110888.

[17] W. S. Cleveland, “Robust locally weighted regression and smoothing scatterplots,” Journal of the American statistical association, vol. 74, no. 368, 1979, pp. 829–836.

[18] R core team, “R: a language and environment for statistical computing,” 2015.

[19] B. Kitchenham, L. Pickard, S. MacDonell, and M. Shepperd, “What accuracy statistics really measure [software estimation],” in Software, IEE Proceedings-, vol. 148, no. 3. IET, 2001, pp. 81–85.

[20] D. Chicco, M. J. Warrens, and G. Jurman, “The coefficient of determination R-squared is more informative than SMAPE, MAE, MAPE, MSE and RMSE in regression analysis evaluation,” PeerJ Computer Science, vol. 7, 2021, p. e623.

[21] A. Vargha and H. D. Delaney, “A critique and improvement of the cl common language effect size statistics of mcgraw and wong,” Journal of Educational and Behavioral Statistics, vol. 25, no. 2, 2000, pp. 101–132.

[22] M. Torchiano et al., “effsize: Efficient effect size computation,” R package version 0.7, vol. 1, 2017.

[23] J. Cohen, “Statistical power analysis for the behavioral sciences Lawrence Earlbaum Associates,” Hillsdale, NJ, 1988, pp. 20–26.

[24] NESMA–the Netherlands Software Metrics Association, “Definitions and counting guidelines for the application of function point analysis. NESMA Functional Size Measurement method compliant to ISO/IEC 24570 version 2.1,” 2004.

[25] International Standards Organisation, “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,” 2005.

[26] F. G. Wilkie, I. R. McChesney, P. Morrow, C. Tuxworth, and N. Lester, “The value of software sizing,” Information and Software Technology, vol. 53, no. 11, 2011, pp. 1236–1249.

[27] J. Popović and D. Bojić, “A comparative evaluation of effort estimation methods in the software life cycle,” Computer Science and Information Systems, vol. 9, no. 1, 2012, pp. 455–484.

[28] P. Morrow, F. G. Wilkie, and I. McChesney, “Function point analysis using nesma: simplifying the sizing without simplifying the size,” Software Quality Journal, vol. 22, no. 4, 2014, pp. 611–660.

[29] S. Di Martino, F. Ferrucci, C. Gravino, and F. Sarro, “Assessing the effectiveness of approximate functional sizing approaches for effort estimation,” Information and Software Technology, vol. 123, July 2020.

[30] L. Santillo, M. Conte, and R. Meli, “Early & Quick Function Point: sizing more with less,” in 11th IEEE International Software Metrics Symposium (METRICS’05). IEEE, 2005, pp. 41–41.

[31] T. Iorio, R. Meli, and F. Perna, “Early&quick function points® v3. 0: enhancements for a publicly available method,” in SMEF, 2007, pp. 179–198.

[32] DPO, “Early & Quick Function Points Reference Manual - IFPUG version,” DPO, Roma, Italy, Tech. Rep. EQ&FP-IFPUG-31-RM-11-EN-P, April 2012.

[33] R. Meli, “Early & quick function point method-an empirical validation experiment,” in Int. Conf. on Advances and Trends in Software Engineering, Barcelona, Spain, 2015.

[34] C. Tichenor, “The IRS development and application of the internal logical file model to estimate function point counts,” in IFPUG Fall Conf., 1997.

[35] L. Bernstein and C. M. Yuhas, Trustworthy systems through quantitative software engineering. John Wiley & Sons, 2005, vol. 1.

[36] R. Meli and L. Santillo, “Function point estimation methods: A comparative overview,” in FESMA, vol. 99. Citeseer, 1999, pp. 6–8.

[37] D. B. Bock and R. Klepper, “FP-S: a simplified function point counting method,” Journal of Systems and Software, vol. 18, no. 3, 1992, pp. 245–254.

[38] G. Horgan, S. Khaddaj, and P. Forte, “Construction of an FPA-type metric for early lifecycle estimation,” Information and Software Technology, vol. 40, no. 8, 1998, pp. 409–415.

[39] L. Santillo, “Easy Function Points – ‘Smart’ Approximation Technique for the IFPUG and COSMIC Methods,” in IWSM–MENSURA, 2012.

[40] L. Lavazza, S. Morasca, and G. Robiolo, “Towards a simplified definition of function points,” Information and Software Technology, vol. 55, no. 10, 2013, pp. 1796–1809.

[41] P. Pospieszny, B. Czarnacka-Chrobot, and A. Kobylinski, “An effective approach for software project effort and duration estimation with machine learning algorithms,” Journal of Systems and Software, vol. 137, 2018, pp. 184–196.

[42] M. N. Mahdi, M. H. Mohamed Zabil, A. R. Ahmad, R. Ismail, Y. Yusoff, L. K. Cheng, M. S. B. M. Azmi, H. Natiq, and H. Happala Naidu, “Software project management using machine learning technique—a review,” Applied Sciences, vol. 11, no. 11, 2021, p. 5183.

[43] X. Liu, N. Li, S. Liu, J. Wang, N. Zhang, X. Zheng, K.-S. Leung, and L. Cheng, “Normalization methods for the analysis of unbalanced transcriptome data: a review,” Frontiers in bioengineering and biotechnology, vol. 7, 2019, p. 358.

[44] T. Kuronen, T. Eerola, L. Lensu, J. Takatalo, J. Häkkinen, and H. Kälviäinen, “High-speed hand tracking for studying human-computer interaction,” in Scandinavian Conference on Image Analysis. Springer, 2015, pp. 130–141.

[45] L. Q. Leal, R. A. Fagundes, R. M. de Souza, H. P. Moura, and C. M. Gusmão, “Nearest-neighborhood linear regression in an application with software effort estimation,” in 2009 IEEE International Conference on Systems, Man and Cybernetics. IEEE, 2009, pp. 5030–5034.

[46] L. Lavazza, A. Locoro, G. Liu, and R. Meli, “Estimating software functional size via machine learning,” ACM Transaction on Software Engineering and Methodology, vol. to appear, no. ?, 2023?, p. ?