Software Development Effort Estimation Using Function Points and Simpler Functional Measures: a Comparison

Luigi Lavazza[1,*], Angela Locoro[2] and Roberto Meli[3]

[1]Università degli Studi dell'Insubria, Varese, Italy
[2]Università degli Studi di Brescia, Brescia, Italy
[3]Data Processing Organization Srl, Roma, Italy

Abstract

Background — Functional Size Measures are widely used for estimating the development effort of software. After the introduction of Function Points, a few "simplified" measures have been proposed, aiming to make measurement simpler and quicker, but also to make measures applicable when fully detailed software specifications are not yet available. It has been shown that, in general, software size measures expressed in Function Points do not support more accurate effort estimation with respect to simplified measures.
Objective — Many practitioners believe that when considering "complex" projects, i.e., project that involve many complex transactions and data, traditional Function Points measures support more accurate estimates than simpler functional size measures that do not account for greater-then-average complexity. In this paper, we aim to produce evidence that confirms or disproves such belief.
Method — Based on a dataset that contains both effort and size data, an empirical study is performed, to provide some evidence concerning the relations that link functional size (measured in different ways) and development effort.
Results — Our analysis shows that there is no statistically significant evidence that Function Points are generally better at estimating more complex projects than simpler measures. Function Points appeared better in some specific conditions, but in those conditions they also performed worse than simpler measures when dealing with less complex projects.
Conclusions — Traditional Function Points do not seem to effectively account for software complexity. To improve effort estimation, researchers should probably dedicate their effort to devise a way of measuring software complexity that can be used in effort models together with (traditional or simplified) functional size measures.

Keywords

Unadjusted Function Points (UFP), Simple Function Points (SFP), effort estimation, simple functional size measures

IWSM Mensura'23: The Joint Conference of the 32nd International Workshop on Software Measurement (IWSM) and the 17th International Conference on Software Process and Product Measurement (MENSURA), September 14-15, 2023, Rome, Italy

*Corresponding author.

luigi.lavazza@uninsubria.it (L. Lavazza); angela.locoro@unibs.it (A. Locoro); roberto.meli@dpo.it (R. Meli)
0000-0002-5226-4337 (L. Lavazza); 0000-0002-6740-8620 (A. Locoro); 0000-0003-1069-8548 (R. Meli)

1. Introduction

Functional Size Measures (FSM) are widely used for estimating the development effort of software, mainly because they can be obtained in the early stages of development, when effort estimates are most needed. Function Point Analysis (FPA) was introduced to yield a measure of software size based exclusively on specifications [1].

After the introduction of Function Points (FP), a few "simplified" measures have been proposed, aiming to make measurement simpler and quicker, but also to make measures applicable when fully detailed software specifications are not yet available. Among the simplified measures are Simple Function Points (SFP) [2] (formerly known as SiFP [3]) and the sheer number of transaction functions.

It has been shown [4, 5] that, in general, software size measures expressed in Unadjusted Function Points[1] (UFP) do not support more accurate effort estimation with respect to "simplified" measures. However, many practitioners who use UFP for estimation believe that when considering "complex" projects, i.e., projects that involve many complex transactions and data, UFP measures support more accurate estimates than SFP or other measures that do not account for greater-then-average complexity. In this paper, we illustrate an empirical study that is meant to confirm or disprove the aforementioned belief. The study is based on the analysis of the ISBSG dataset [7], which has been widely used for studies concerning software functional size.

The results of the study will likely be helpful for the numerous software development organizations that use FPA: e.g., organizations that develop software for public administration and are thus required to provide software size measured via IFPUG FPA by local laws (as in in Brazil, Italy, Japan, South Korea, and Malaysia). Also, other organizations may need FP measures because they use effort estimation tools (like Galorath's Seer-SEM [8], for instance) that take the size expressed in FP as input (together with some parameters that account for the development process and technology, non-functional requirements, human factors, etc.).

The results of the study can be interesting also for organizations that use agile development processes. In fact, traditional functional size measurement is not very popular in agile contexts, because it is perceived as a "heavy" method, not suitable for agile development. However, simplified functional size measurement methods could fit easily in agile development practices, especially when the simplification is pushed to counting only transactions, i.e., functional elements that can be easily identified from user stories.

The paper is organized as follows. Section 2 recalls some basic notions concerning Functional Size Measurement (FSM) methods. Section 3 states the objectives of the work described here, also by formulating research questions. Section 4 describes the empirical study through which we addressed the research questions. The achieved results are also illustrated and discussed. In Section 5 research questions are answered. Section 6 discusses the threats to the validity of the study. Section 7 accounts for related work. Section 8 draws some conclusions and outlines future work.

[1]Following the ISO [6], we consider only unadjusted FP.

2. Background

In this section we provide a very brief introduction to Function Points, as well as to simplified measures, namely SFP and the number of transactions.

2.1. Function Point Analysis

Function Point Analysis was originally introduced by Albrecht to measure the size of software systems from end-users' point of view, with the goal of estimating the development effort [1]. Currently, FPA is officially documented by the IFPUG (International Function Points User Group) via the counting practices manual [9],

The basic idea of FPA 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 elementary processes or 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.

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. BFCs are data functions (also known as logical files), which are classified into internal logical files (ILF) and external interface files (EIF), and 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.

The size of every BFC is determined based on its type and its "complexity" (see the manual [9] for details). The functional size of a given application, expressed in unadjusted Function Points, is given by the sum of the sizes of all its BFCs.

The core of FPA involves the following main activities:

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

The first four of these activities can be carried out even if the FURs have not yet been fully detailed. On the contrary, the last two activities require that details are available.

Simplified Functional Size Measurement methods aim at providing estimates of functional size measures by skipping one or more of the activities listed above. Specifically, simplified measurement methods tend to skip at least the determination of complexity, since this activity is time- and effort-consuming [10].

2.2. Simple Function Points

The Simple Function Point measurement method [3, 11] has been designed by Meli to be lightweight and easy to use. Like IFPUG FPA, it is independent of the technologies and of the technical design principles.

SFP requires only the identification of Elementary Processes (EP) and Logical Files (LF), based on the assumption that value to a BFC is given as a whole, independently of internal organization and details.

SFP assigns a numeric value directly to BFCs, as follows:

$$Size_{SFP} = 7\ \#LF + 4.6\ \#EP$$

thus 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 defined to achieve the best possible approximation of FPA. However, since SFP is a measurement method, those weights are constants, i.e., they are not subject to update or change for approximation reasons, and are now crystallized for stability, repeatability, and comparability reasons.

2.3. Extremely Simplified Functional Size Measures

As described in Section 2.2 above, SFP adopts fixed weights for elementary processes and logical data files. A further simplification consists in not considering at all data in the measurement of functional size. In this sense, the most straightforward measure is given by the sheer number of transactions (#TF). Note that applying a weight to the number of transactions would result in a simple scale transformation, with no improvement in the correlation with development effort.

3. Research Questions

Some research has already been dedicated to evaluating the possibility of using functional size measures that are definitely simpler than standard IFPUG UFP for effort estimation [4, 5]. Simpler metrics are of great interest for practitioners because they are quicker and less expensive to collect than traditional FP, and, even more important, simple measures can sometimes be applied before detailed and complete software requirements are available.

However, previous research proposed empirical studies whose conclusions were based on the evaluation of estimation accuracy over the entire test set. Such practice, although sound and informative, does not solve possible doubts about the performance of difference metrics when dealing with projects having different complexity.

In fact, in some environments, it is believed that traditional UFP are better at accounting for the complexity of projects, hence, when dealing with relatively complex projects, UFP are expected to support more accurate effort estimation with respect to simpler FSM methods. However, as far as we know, hardly any evidence has been produced to support this belief.

In this paper, we provide some evidence that can be used to either support or disprove the aforementioned belief. To this end, we formulate the following research questions:

RQ1 If project complexity is not taken into account, is it true that simple functional measures (namely, SFP and #TF) provide effort estimates that are as accurate as those provided by standard IFPUG UFP?

RQ2 For projects that have relatively high (respectively, low) complexity, do UFP and simple functional metrics (namely, SFP and #TF) support effort estimation at significantly different levels of accuracy?

It is well known that there are multiple ways for i) modeling the dependence of development effort on software functional size; ii) evaluating (in a statistically sound manner) the accuracy of the obtained estimates; iii) classifying projects as relatively complex or relatively simple, etc. Answering the research questions for all the possible ways of addressing the issues mentioned above is hardly possible. Therefore, in this paper we adopt reasonable models and classification techniques, preferring simper ones, to avoid the risk of getting results that depend on the intricacies of the technical instruments being used.

4. The Study

In this section we describe the empirical study that supports our answers to the research questions.

4.1. The Dataset

In our empirical study, we analyzed data from the ISBSG dataset [7], which includes data from real-life software development projects and has been widely used in studies involving Functional Size Measures.

The ISBSG dataset contains data from both projects addressing the development of new software products and projects addressing the enhancement of existing projects. In this paper, we limit our investigation to new developments. Dealing with enhancement projects will be the objective of future work.

For each project, many data are provided. Of these, we used the following:

- The type of project, i.e., new development or enhancement.
- The effort spent, expressed in PersonHours (PH).
- The size, expressed in IFPUG Function Points.
- #ILF, #EIF, #EI, #EO, and #EQ, each split per complexity (high, medium, low).

It is worth noting that the considered version of the ISBSG dataset contains some measures (namely, Effort, the size in UFP and the number of transactions) that we use as-is, as well as raw data (#ILF, #EIF, #EI, #EO, and #EQ) that we used to compute the size in #TF (#EI+#EO+#EQ) and SFP (7(#ILF+#EIF)+4.6#TF). The type of project was used only to select new development projects.

The dataset includes data from 533 new development projects. Descriptive statistics of the ISBSG dataset are given in Section 4.3.

4.2. The Method

4.2.1. The effort model

In this paper, we use a very simple method for building effort models. In fact, we assume that effort can be computed by dividing the size of the software product to be built by the observed productivity:

$$Effort = \frac{Size}{Productivity} \tag{1}$$

It is clear that formula (1) describes a very simple model of effort, since i) it assumes that effort depends only on functional size, and ii) it is structurally simple, especially when compared with models that can be obtained via sophisticated techniques like machine learning, neural networks, etc. We preferred this extremely simple model to avoid possible confounding effects.

Productivity is defined as [12]

$$Productivity = \frac{Size}{Effort} \tag{2}$$

However, the value of *Productivity* to be used in (1) can be obtained in different ways. In this paper, we consider three possible derivations of the *Productivity* value:

1. For each project in the dataset, we considered its *Productivity*, as defined in (2). Then we computed the mean value of the projects' productivity. In doing this, we used as a size measure UFP, SFP and #TF, thus obtaining $Productivity_{UFP}$, $Productivity_{SFP}$ and $Productivity_{\#TF}$.
2. We proceeded as described above, but the productivity was obtained as the median value of the projects' productivity.
3. Finally, we considered the case when the productivity is given. This is the case in some software acquisition markets, where the productivity is assumed to have a conventional value, so that every UFP (or SFP) has the corresponding value and price.

Productivity was then used to compute, via formula (1), the estimated effort for each project in the dataset.

Estimation errors *EstErr* were then computed:

$$EstErr = ActualEffort - EstimatedEffort = ActualEffort - \frac{Size}{Productivity} \tag{3}$$

Specifically, the computation described by formula (3) was computed for the three considered Functional size measures, i.e., UFP, SFP and #TF. For instance,

$$EstErr_{UFP} = ActualEffort - \frac{Size_{UFP}}{Productivity_{UFP}}$$

where $Size_{UFP}$ is the functional size of the considered project, expressed in UFP. Similarly, we obtained $EstErr_{SFP}$ and $EstErr_{\#TF}$ for each project in the ISBSG dataset.

4.2.2. Evaluation of estimation accuracy

We performed a sign test to evaluate whether any of the considered measures supports more accurate effort estimates than the other considered FSM methods. For instance, we counted for how many projects it is $|EstErr_{UFP}| < |EstErr_{SFP}|$: let n_{UFP} be that number; using the binomial test (with $\alpha = 0.05$), we evaluated whether we can safely conclude that estimates based on UFP are more accurate than estimates based on SFP. In practice, we tested if the probability of achieving $\frac{n_{UFP}}{n}$ (where n is the total number of projects) is greater than $\frac{1}{2}$.

Similarly, we evaluated UFP vs #TF and SFP vs #TF.

4.2.3. Classification of projects according to complexity

Research question **RQ2** requires identifying projects that are "complex." To this end, we need to properly define the notion of complexity. In the context of Function Point Analysis, complexity is evaluated by weighting base functional components. Therefore, we exploit this practice to evaluate projects' complexity. Noticeably, the ISBSG dataset does not provide other thorough and consistent information about projects' complexity, hence our choice was actually forced by the available data.

Accordingly, we proceeded as follows:

1. For each project, we computed the proportion tf_{cplx} of high complexity transactions over the total number of transactions.
2. We computed the 20^{th} and 80^{th} percentiles from the distribution of tf_{cplx}, thus obtaining $tf_{20} = 0.037$ and $tf_{80} = 0.493$.
3. We selected the projects having $tf_{cplx} < tf_{20}$ as simple, those having $tf_{cplx} > tf_{80}$ as complex, and those with $tf_{20} \leq tf_{cplx} \leq tf_{80}$ as medium complexity ones.

4.2.4. Dealing with fixed productivity

As mentioned in Section 4.2, we used both productivity values derived from the ISBSG data and a fixed market value. The latter is the value imposed by CONSIP (Concessionaria Servizi Informativi Pubblici) for contracts involving the public administration in Italy. Currently, such productivity value is 1.7 UFP/PD, where a PersonDay (PD) includes 8 PH, hence 1.7 UFP/PD is approximately 0.283 UFP/PH. Note that 0.283 UFP/PH is much greater than both the mean and median values from the ISBSG dataset (see Table 1).

Once the productivity in UFP/PH had been established, we had to devise proper corresponding values for the productivity in SFP/PH and the productivity in #TF/PH. This task is not easy, because it involves assuming ratios among the considered size measures; e.g., we have to assume that X UFP corresponds to Y SFP.

In the empirical study described here, we assume that 1 UFP corresponds to 1 SFP, based on previous research [3, 4].

To the best of our knowledge, there are no results available concerning the ratio among the #TF and UFP or SFP; therefore, we computed such ratio using the available ISBSG data. It turned out that the mean value of #TF/UFP is approximately 8.307; hence, we assumed that the productivity in #TF corresponding to 0.283 UFP/PH is $0.283 \times 8.307 = 1.765$ #TF/PH.

4.3. Results

In this section, the results of our analyses are described. Section 4.3.1 illustrates the results obtained when considering all the new development projects from the ISBSG dataset, while Section 4.3.2 concerns only those projects that are more effort-consuming.

In both sections, all combinations of software complexity and productivity are considered: complexity is either ignored (i.e., all projects are considered together) or it is used to split the dataset into low, mid and high complexity ones; productivity is obtained as the mean or the median of ISBSG projects', or it is a given value.

Table 1
Descriptive statistics of new development projects

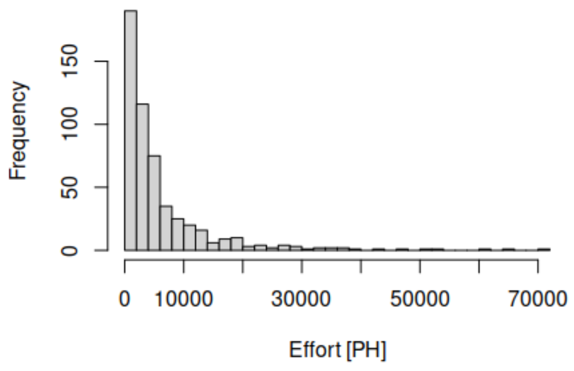
FSM	Size					Productivity	
	mean	st.dev.	median	min	max	mean	median
UFP	541.8	618.6	312	6	3968	0.1838	0.1109
SFP	546.4	613.2	320	9.2	4250.4	0.1932	0.1129
#TF	80.5	99.5	44	1	679	0.0259	0.0146

4.3.1. Results obtained from all new development projects

Table 1 provides descriptive statistics for the new development projects contained in the dataset.

Figure 1 shows the distribution of all new development projects' effort in PH.

Figure 1: The distribution of all new development projects' effort in PH



When using mean productivity in model (1), we obtained estimation errors whose distribution is described in Figure 2.

The boxplots in Figure 2 indicate that the three considered measures yield quite similar error distributions. This was confirmed by the sign test, whose results are summarized in Table 2.

Each cell of the table provides a symbol followed by two numbers in parentheses: the symbol indicates if the measure in row was better (">"), equivalent ("=") or worse ("<") than the measure in column; the first number indicates how many times the measure in row was better

Figure 2: Boxplots of estimation errors with (left) and without outliers (right) for all new development projects, when estimates are based on mean productivity

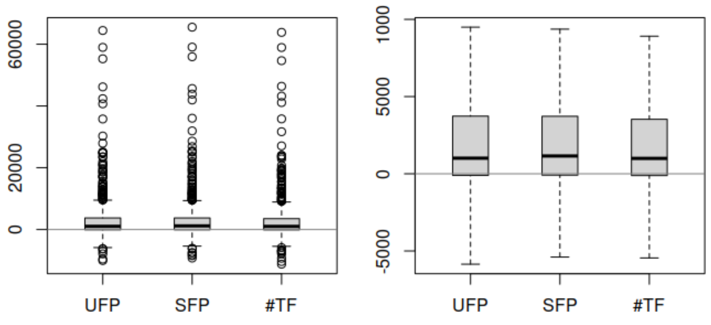


Table 2
Sign test results for all new development projects, mean productivity

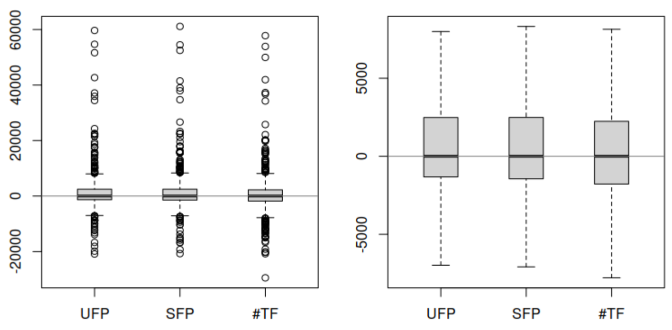
	UFP	SFP	#TF
UFP	—	= (285 vs 248)	= (261 vs 272)
SFP	= (248 vs 285)	—	= (256 vs 277)
#TF	= (272 vs 261)	= (277 vs 256)	—

than the measure in column; the second number indicates how many times the measure in column was better than the measure in row. For instance, the cell in row 1 and column 3 indicates that UFP supported more accurate estimates for 261 projects, while #TF supported more accurate estimates for 272 projects; the "=" symbol indicates that according to the binomial test the observed difference (261 vs 272) is not large enough to conclude that the probability of UFP being better or worse than #TF is not 0.5.

In practice, when considering all new development projects and using mean productivity in the effort estimation formula (1), there is no statistically significant evidence that the considered functional measures perform differently.

When using median productivity in model (1), we obtained the estimation errors described in Figure 3.

Figure 3: Boxplots of estimation errors with (left) and without outliers (right) for all new development projects, when estimates are based on median productivity



The results of the sign tests are summarized in Table 3.

Table 3
Sign test results for all new development projects, median productivity

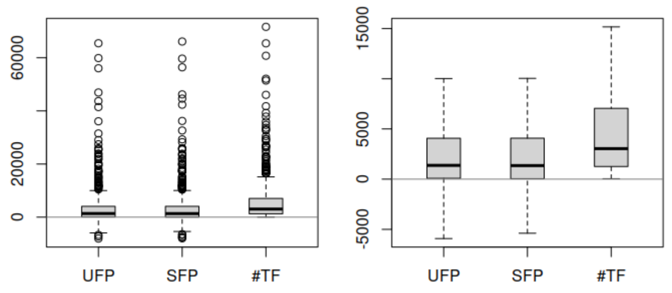
	UFP	SFP	#TF
UFP	—	= (275 vs 258)	> (287 vs 246)
SFP	= (258 vs 275)	—	> (291 vs 242)
#TF	< (246 vs 287)	< (242 vs 291)	—

The sign tests indicate:

- There is no evidence that the number of more accurate estimates obtained via UFP is different than the number of more accurate estimates obtained via SFP. That is, UFP and SFP appear equivalent in supporting effort estimation.
- Both UFP and SFP appear to support more accurate estimates than #TF.

We then used the fixed productivity values discussed in Section 4.2.4 in model (1) to get further estimates. When using such fixed productivity values, we obtained the estimation errors described in Figure 4.

Figure 4: Boxplots of estimation errors with (left) and without outliers (right) for all new development projects, when estimates are based on given productivity



The results of the sign test are summarized in Table 4.

Table 4
Sign test results for all new development projects, given productivity

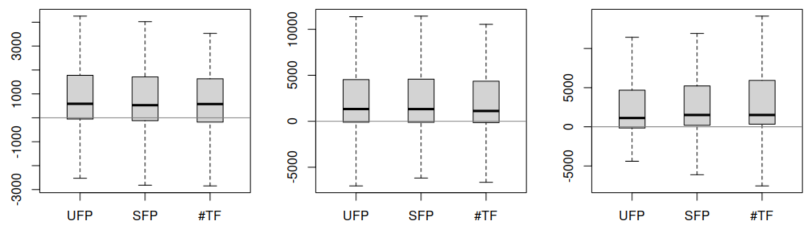
	UFP	SFP	#TF
UFP	—	< (242 vs 291)	> (488 vs 45)
SFP	> (289 vs 244)	—	> (482 vs 51)
#TF	< (45 vs 488)	< (51 vs 482)	—

We then proceeded to evaluate separately the high-, mid- and low-complexity projects. As mentioned in Section 4.2, we computed the 20^{th} and 80^{th} percentiles from the distribution of the proportion tf_{cplx} of high complexity transactions over the total number of transactions, obtaining $tf_{20} = 0.037$ and $tf_{80} = 0.493$. The new development projects of the ISBSG dataset are split by complexity as follows:

- 108 low-complexity ($tf_{cplx} < tf_{20}$) projects.
- 163 mid-complexity ($tf_{20} \leq tf_{cplx} \leq tf_{80}$) projects.
- 107 high-complexity ($tf_{cplx} > tf_{80}$) projects.

The estimation errors obtained when using the mean productivity are shown in Figure 5.

Figure 5: Boxplots of estimation errors for low (left), mid (center) and high (right) new development projects, when estimates are based on mean productivity. Outliers omitted.



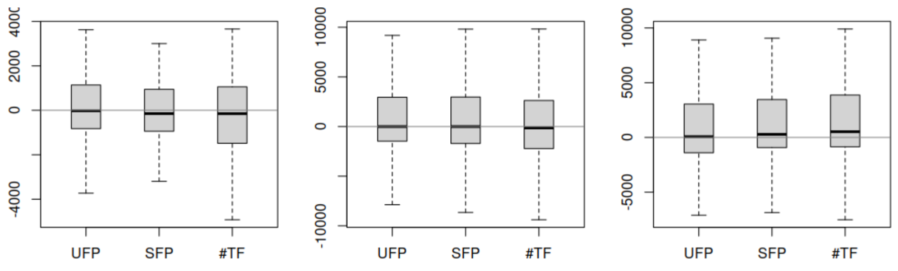
The results of the sign tests are summarized in Table 5.

Table 5
Sign test results for new development projects split per complexity, mean productivity

	low complexity			mid complexity			high complexity		
	UFP	SFP	#TF	UFP	SFP	#TF	UFP	SFP	#TF
UFP	—	<(42 vs 66)	= (52 vs 56)	—	= (163 vs 155)	<(143 vs 175)	—	>(80 vs 27)	>(66 vs 41)
SFP	>(66 vs 42)	—	= (54 vs 54)	= (155 vs 163)	—	<(141 vs 177)	<(27 vs 80)	—	= (61 vs 46)
#TF	= (56 vs 52)	= (54 vs 54)	—	>(175 vs 143)	>(177 vs 141)	—	<(41 vs 66)	= (46 vs 61)	—

The estimation errors obtained when using the median productivity are shown in Figure 6.

Figure 6: Boxplots of estimation errors for low (left), mid (center) and high (right) new development projects, when estimates are based on median productivity. Outliers omitted.



The results of the sign tests are summarized in Table 6.
The estimation errors obtained when using a given fixed productivity are shown in Figure 7.
The results of the sign tests are summarized in Table 7.

4.3.2. Results obtained from selections of new development projects

As shown in Figure 1, the great majority of ISBSG new development projects required a relatively small effort. Specifically, 30% of the projects required no more than a PersonYear, while more than

Table 6
Sign test results for new development projects split per complexity, median productivity

	low complexity			mid complexity			high complexity		
	UFP	SFP	#TF	UFP	SFP	#TF	UFP	SFP	#TF
UFP	—	= (51 vs 57)	>(68 vs 40)	—	= (168 vs 150)	= (159 vs 159)	—	= (56 vs 51)	= (60 vs 47)
SFP	= (57 vs 51)	—	>(66 vs 42)	= (150 vs 168)	—	= (163 vs 155)	= (51 vs 56)	—	= (62 vs 45)
#TF	<(40 vs 68)	<(42 vs 66)	—	= (159 vs 159)	= (155 vs 163)	—	= (47 vs 60)	= (45 vs 62)	—

Figure 7: Boxplots of estimation errors for low (left), mid (center) and high (right) new development projects, when estimates are based on fixed productivity. Outliers omitted.

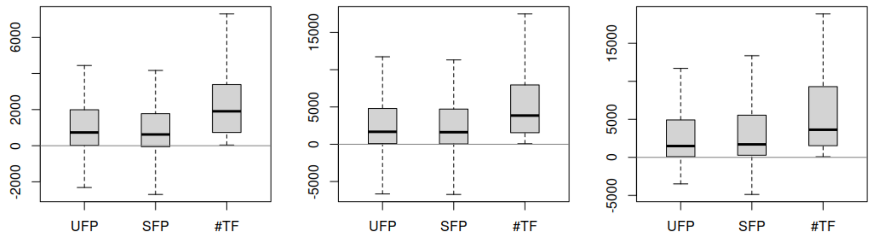


Table 7
Sign test results for new development projects split per complexity, fixed productivity

	low complexity			mid complexity			high complexity		
	UFP	SFP	#TF	UFP	SFP	#TF	UFP	SFP	#TF
UFP	—	< (33 vs 75)	> (95 vs 13)	—	< (134 vs 184)	> (294 vs 24)	—	> (75 vs 32)	> (99 vs 8)
SFP	> (75 vs 33)	—	> (93 vs 15)	> (184 vs 134)	—	> (290 vs 28)	< (30 vs 77)	—	> (99 vs 8)
#TF	< (13 vs 95)	< (15 vs 93)	—	< (24 vs 294)	< (28 vs 290)	—	< (8 vs 99)	< (8 vs 99)	—

50% require less than 2 PersonYear. We can thus conclude that the results reported in Section 4.3.1 are determined mainly by small (in terms of effort) projects. It is thus necessary to reconsider the research questions in the context of projects that require considerable development effort. To this end, we repeated the analysis described in Section 4.3.1, considering only projects that require considerable development effort. For the sake of space, in this section we report only the results of the sign tests, while estimation error boxplots are omitted.

As a first step, we had to decide which projects should be involved in the analysis. We decided to retain the projects that required no less than two PersonYears, i.e., $2 \times 210 \times 8 = 3360$ PH (assuming 210 working day per year and 8 working hours per day). In this way, we selected 247 projects. The descriptive statistics of this dataset are given in Table 8. In the rest of the paper, these projects are named conventionally "not too small."

Table 8
Descriptive statistics of not too small new development projects

FSM	Size					Productivity	
	mean	st.dev.	median	min	max	mean	median
UFP	842.6	749.5	578	117	3968	0.0908	0.0698
SFP	838.5	739.8	566.4	106.4	4250.4	0.0911	0.0699
#TF	126.6	121.1	82	5	679	0.0134	0.0097

When using mean productivity in model (1), the sign tests applied to absolute residuals yielded the results summarized in Tables 9 (all projects) and 12 (only not too small projects).

Table 9
Sign test results for not too small new development projects, mean productivity

	UFP	SFP	#TF
UFP	—	= (126 vs 121)	= (134 vs 113)
SFP	= (121 vs 126)	—	= (133 vs 114)
#TF	= (113 vs 134)	= (114 vs 133)	—

When using median productivity in model (1), the sign tests applied to absolute residuals yielded the results summarized in Tables 10 (all projects) and 13 (only not too small projects).

Table 10
Sign test results for not too small new development projects, median productivity

	UFP	SFP	#TF
UFP	—	= (122 vs 125)	= (127 vs 120)
SFP	= (125 vs 122)	—	> (137 vs 110)
#TF	= (120 vs 127)	< (110 vs 137)	—

When using the given fixed productivity in model (1), the sign tests applied to absolute residuals yielded the results summarized in Tables 11 (all projects) and 14 (only not too small projects).

Table 11
Sign test results for not too small new development projects, fixed given productivity

	UFP	SFP	#TF
UFP	—	= (116 vs 131)	> (246 vs 1)
SFP	= (131 vs 116)	—	> (245 vs 2)
#TF	< (1 vs 246)	< (2 vs 245)	—

Table 12
Sign test results for not too small new development projects split per complexity, mean productivity

	low complexity			mid complexity			high complexity		
	UFP	SFP	#TF	UFP	SFP	#TF	UFP	SFP	#TF
UFP	—	= (21 vs 29)	= (20 vs 30)	—	= (75 vs 72)	> (84 vs 63)	—	= (30 vs 20)	= (30 vs 20)
SFP	= (29 vs 21)	—	= (21 vs 29)	= (72 vs 75)	—	= (83 vs 64)	= (20 vs 30)	—	= (29 vs 21)
#TF	= (30 vs 20)	= (29 vs 21)	—	< (63 vs 84)	= (64 vs 83)	—	= (20 vs 30)	= (21 vs 29)	—

4.3.3. Results for enhancement projects

In this paper, we focused on projects involving the development of new software (i.e., development from scratch). The ISBSG dataset provides also data concerning projects that involved

Table 13
Sign test results for not too small new development projects split per complexity, median productivity

	low complexity			mid complexity			high complexity		
	UFP	SFP	#TF	UFP	SFP	#TF	UFP	SFP	#TF
UFP	—	= (24 vs 26)	= (23 vs 27)	—	= (71 vs 76)	= (78 vs 69)	—	= (27 vs 23)	= (26 vs 24)
SFP	= (26 vs 24)	—	= (26 vs 24)	= (76 vs 71)	—	> (84 vs 63)	= (23 vs 27)	—	= (27 vs 23)
#TF	= (27 vs 23)	= (24 vs 26)	—	= (69 vs 78)	< (63 vs 84)	—	= (24 vs 26)	= (23 vs 27)	—

Table 14
Sign test results for not too small new development projects split per complexity, fixed productivity

	low complexity			mid complexity			high complexity		
	UFP	SFP	#TF	UFP	SFP	#TF	UFP	SFP	#TF
UFP	—	< (9 vs 41)	> (50 vs 0)	—	< (62 vs 85)	> (147 vs 0)	—	> (45 vs 5)	> (49 vs 1)
SFP	> (41 vs 9)	—	> (49 vs 1)	> (85 vs 62)	—	> (147 vs 0)	< (5 vs 45)	—	> (49 vs 1)
#TF	< (0 vs 50)	< (1 vs 49)	—	< (0 vs 147)	< (0 vs 147)	—			

enhancing existing software. For time and space reasons, we could not extend our analysis to this type of projects in this paper. However, preliminary analysis of enhancement projects seem to confirm the results concerning new developments.

5. Discussion

In this section, we answer the research questions enunciated in Section 3. Having considered two simple functional size measures (SFP and #TF), we answer each question separately for UFP vs SFP and UFP vs #TF.

In interpreting the results given below, we recommend some caution when a given productivity is used: in those cases, the given productivity concerns UFP, while the equivalent productivities for SFP and #FP are not given; hence we had to devise them (see Section 4.2.4) in a somewhat subjective manner.

5.1. Answer to RQ1

Research question **RQ1** asks if it is true that SFP and #TF provide effort estimates that are as accurate as those provided by standard IFPUG UFP, when project complexity is not taken into account.

Table 15 summarizes the results illustrated in Section 4.3.1 that are relevant for **RQ1**.

Based on the collected results, we can state that there is hardly any difference in estimation accuracy when using SFP instead of UFP.

When #TF are used, the answer clearly depends on the method used to compute the productivity.

5.2. Answer to RQ2

Research question **RQ2** asks if UFP and simple functional metrics support effort estimation at significantly different levels of accuracy for projects that have different complexity.

Table 15
Summary of results, when complexity is not considered.

Projects considered	Productivity	Result: UFP vs SFP	Results: UFP vs #TF
All	mean	no difference	no difference
All	median	no difference	UFP more accurate
All	given	SFP more accurate	UFP more accurate
“Not too small”	mean	no difference	no difference
“Not too small”	median	no difference	no difference
“Not too small”	given	no difference	UFP more accurate

Table 16 summarizes the results illustrated in Section 4.3.2 that are relevant for **RQ2**.

Table 16
Summary of results, when complexity is considered.

Projects considered	Complexity	Productivity	Result: UFP vs SFP	Results: UFP vs #TF
All	Low	mean	SFP more accurate	no difference
All	Mid	mean	no difference	#TF more accurate
All	High	mean	UFP more accurate	UFP more accurate
All	Low	median	no difference	UFP more accurate
All	Mid	median	no difference	no difference
All	High	median	no difference	no difference
All	Low	given	SFP more accurate	UFP more accurate
All	Mid	given	SFP more accurate	UFP more accurate
All	High	given	UFP more accurate	UFP more accurate
“Not too small”	Low	mean	no difference	no difference
“Not too small”	Mid	mean	no difference	UFP more accurate
“Not too small”	High	mean	no difference	no difference
“Not too small”	Low	median	no difference	no difference
“Not too small”	Mid	median	no difference	no difference
“Not too small”	High	median	no difference	no difference
“Not too small”	Low	given	SFP more accurate	UFP more accurate
“Not too small”	Mid	given	SFP more accurate	UFP more accurate
“Not too small”	High	given	UFP more accurate	UFP more accurate

Based on the collected results, it can be observed that in a few cases (depending on how productivity is computed) UFP appear more accurate when dealing with high-complexity software, while SFP appear more accurate when dealing with low-complexity software. This suggests that UFP may be slightly biased in considering complexity.

When considering #TF, answering **RQ2** is very difficult. There is no clear pattern in the performances of UFP vs #TF: for instance, when using mean productivity for mid-complexity software, UFP perform better with more effort-consuming (or “not too small”) projects, while #TF perform better with less effort-consuming projects.

6. Threats to validity

A typical concern for considering only empirical data is the lack of theoretical point of view, for example in defining complexity and complex software projects. However, we started from some consolidated empirical evidence and practices about the criteria of software functional size, and we followed the common praxis of the community. One of the reasons why our results challenge a strong believe in the community (the more the UFP measure is "complex", the better it is correlated to effort), probably comes from the lack of too theoretical reflections. However, this is not a limitation of our paper only, but a more generalized problem.

Some decisions made while carrying out the study might have influenced the results. However, such decisions were necessary to perform the analysis. When dealing with the choices that most obviously could affect our results, we carried out some sensitivity analysis. For instance, concerning the criteria used to identify "not too small" projects when the median productivity model is used, we tried increasing (up to doubling) the minimum effort threshold that qualifies a project as "not to small" and we noticed no differences.

Another major concern in these kinds of studies is the generalizability of results outside the scope and context of the analyzed dataset. The ISBSG dataset is deemed the standard benchmark among the community, and it includes data from several application domains. Therefore, our results should be representative of a fairly comprehensive situation. However, additional studies are needed for confirming the generalizability of the results presented above. This is particularly true of studies concerning enhancement projects, that we could not treat adequately in this paper.

7. Related Work

Since the introduction of Function Point Analysis, many researchers and practitioners strived to develop simplified versions of the FP measurement process, both to reduce the cost and duration of the measurement process, and to make it applicable when full-fledged requirements specifications are not yet available [13, 14, 15, 16, 17, 18, 19, 3, 20].

These simplified measurement methods were then evaluated with respect to their ability to support accurate effort estimation [21, 22, 23, 24, 25, 26, 27, 28, 4, 29].

More recently, Lavazza et al. considered using only the number of transaction to estimate effort [5]: it was found that effort models based on the number of transactions appear marginally less accurate than models based on standard IFPUG Function Points for new development projects, and marginally more accurate for projects extending previously developed software.

To the best of our knowledge, no studies considered classifying projects according to degrees of complexity.

Since the '90s, the early estimation of software was achieved with different methods. Among them, using regression-like methods [30] or by the "Early & Quick Function Point" (EQFP) method [31], which uses analogy to discover similarities between new and previously measured pieces of software, and analysis to provide weights for software objects. Statistical estimation methods were first introduced by Lavazza et al., who studied the relationships between BFCs and size measures expressed in FP [32].

More recently, software effort estimation has been done with machine learning methods. Case-Based Reasoning and Genetic Algorithm were exploited with benchmark datasets, and improved accuracy of effort estimation [33]. In another study [34], agile development is the target of the effort estimation research: a mixed method using also qualitative interviews of software projects teams revealed the necessity to assess 12 hypothesis tests for effort estimation.

Being those methods far away from the approach presented in this paper, we will not go further in detail into these researches. It is worth notice that these approaches are complementary to the one adopted in this paper, which may hence be applied in synergy with the one currently presented.

8. Conclusions

Simplified functional size measures ignore the "complexity" of transactions, which is instead accounted for by traditional Function Point Analysis. Some believe that this type of omission makes simplified measures less suitable for effort estimation, when relatively complex software products are involved. To assess the truth of this belief, an empirical study was conducted, based on the analysis of the data from the ISBSG dataset.

Our analysis shows that UFP do not appear to support more accurate effort estimation when performances over an entire dataset are considered. Instead, when splitting the given dataset according to transaction complexity, UFP-based estimates appear sometimes more accurate; however, it also appears that

1. UFP performance depends on how productivity is computed. For instance, when the median productivity is used, no significant difference between UFP and SFP is observed.
2. When UFP appear more accurate in estimating complex projects, they also appear less accurate in estimating less complex projects.

To sum up, the belief that when considering "complex" projects, i.e., projects that involve many complex transactions and data, traditional Function Points measures support more accurate estimates than simpler functional size measures that do not account for greater-then-average complexity, seems not to be confirmed. It seems justified only in specific circumstances, e.g., whenever the productivity is imposed and definitely higher than what it is detectable by looking at the same data. Furthermore, often UFP performs better than SFP to estimate complex projects, but worse in less complex projects. This may suggest that UFP are not better than SFP, but rather that they are biased towards more complex projects instead.

In conclusion, our results show that the complexity of software (even measured using the rather rough concepts of FPA) can affect effort estimation accuracy; at the same time, embedding the notion of such complexity in Unadjusted Function Points does not guarantee good results.

Accordingly, some interesting topics for future work include involving the notion of complexity in effort models. A first straightforward manner of doing this consists in building a model

$$EstimatedEffort = Productivity_{cplx} \times Size$$

where *Size* and $Productivity_{cplx}$ are the size and the expected productivity, computed according to the class of complexity (high, medium or low) of the software to be developed. More complex models could be achieved, e.g., via machine learning techniques.

Acknowledgments

This work has been partially supported by the “Fondo di ricerca d’Ateneo” of the Università degli Studi dell’Insubria.

References

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

[2] International Function Point Users Group (IFPUG), Simple Function Point (SFP) Counting Practices Manual Release v2.1, 2022.

[3] R. Meli, Simple function point: a new functional size measurement method fully compliant with IFPUG 4. x, in: Software Measurement European Forum, 145–152, 2011.

[4] L. Lavazza, R. Meli, 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, 196–206, 2014.

[5] L. Lavazza, G. Liu, R. Meli, Using Extremely Simplified Functional Size Measures for Effort Estimation: an Empirical Study, in: Proceedings of the 14th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM), 1–9, 2020.

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

[7] International Software Benchmarking Standards Group, “Worldwide Software Development: The Benchmark, release April 2019, 2019.

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

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

[10] L. Lavazza, On the Effort Required by Function Point Measurement Phases, International Journal on Advances in Software Volume 10, Number 1 & 2, 2017 .

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

[12] B. W. Boehm, Improving software productivity, Computer 20 (09) (1987) 43–57.

[13] G. Horgan, S. Khaddaj, P. Forte, Construction of an FPA-type metric for early lifecycle estimation, Information and Software Technology 40 (8) (1998) 409–415.

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

[15] 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.

[16] 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.

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

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

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

[20] L. Lavazza, A. Locoro, G. Liu, R. Meli, Estimating software functional size via machine learning, ACM Transactions on Software Engineering and Methodology .

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

[22] F. G. Wilkie, I. R. McChesney, P. Morrow, C. Tuxworth, N. Lester, The value of software sizing, Information and Software Technology 53 (11) (2011) 1236–1249.

[23] J. Popović, D. Bojić, A comparative evaluation of effort estimation methods in the software life cycle, Computer Science and Information Systems 9 (1) (2012) 455–484.

[24] P. Morrow, F. G. Wilkie, I. McChesney, Function point analysis using NESMA: simplifying the sizing without simplifying the size, Software Quality Journal 22 (4) (2014) 611–660.

[25] L. Lavazza, G. Liu, An Empirical Evaluation of the Accuracy of NESMA Function Points Estimates, in: The 14th International Conference on Software Engineering Advances (ICSEA 2019), 24–29, 2019.

[26] S. Di Martino, F. Ferrucci, C. Gravino, F. Sarro, Assessing the effectiveness of approximate functional sizing approaches for effort estimation, Information and Software Technology 123 (106308).

[27] L. Lavazza, G. Liu, An Empirical Evaluation of Simplified Function Point Measurement Processes, Journal on Advances in Software 6 (1& 2).

[28] R. Meli, Early & Quick Function Point Method-An empirical validation experiment, in: Int. Conf. on Advances and Trends in Software Engineering, Barcelona, Spain, 2015.

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

[30] D. B. Bock, R. Klepper, FP-S: a simplified function point counting method, Journal of Systems and Software 18 (3) (1992) 245–254.

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

[32] L. Lavazza, S. Morasca, G. Robiolo, Towards a simplified definition of Function Points, Information and Software Technology 55 (10) (2013) 1796–1809.

[33] S. Hameed, Y. Elsheikh, M. Azzeh, An optimized case-based software project effort estimation using genetic algorithm, Information and Software Technology 153 (2023) 107088.

[34] S. A. Butt, T. Ercan, M. Binsawad, P.-P. Ariza-Colpas, J. Diaz-Martinez, G. Pineres-Espitia, E. De-La-Hoz-Franco, M. A. P. Melo, R. M. Ortega, J.-D. De-La-Hoz-Hernandez, Prediction based cost estimation technique in agile development, Advances in Engineering Software 175 (2023) 103329.