An Evaluation of Simple Function Point as a Replacement of IFPUG Function Point

Luigi Lavazza
Dipartimento di Scienze Teoriche e Applicate
Università degli Studi dell’Insubria
Varese, Italy
luigi.lavazza@uninsubria.it

Roberto Meli
Simple Function Point Association
Rome, Italy
roberto.meli@dpo.it

Abstract—Simple Function Point is a functional size measurement method that can be used in place of IFPUG Function Point, but requires a much simpler –hence less time and effort consuming– measurement process. Simple Function Point was designed to be equivalent to IFPUG Function Point in terms of numerical results. This paper reports an empirical study aiming at verifying the effectiveness of Simple Function Point as a functional size measurement method, especially suitable to support estimation of software development effort. The data from a large popular public dataset were analyzed to verify the correlation of Simple Function Point with IFPUG Function Point, and the correlation of both size measures to development effort. The results obtained confirm, at a reasonable level of confidence, the hypothesis that Simple Function Point can be effectively used in place of IFPUG Function Point.

Keywords—Functional Size Measurement Methods; IFPUG Function Point; Simple Function Point; Effort estimation

I. INTRODUCTION

Function Point Analysis (FPA) [1][2][3] is widely used. Among the reasons for the success of FPA is that it can provide measures of size from an user perspective and in earlier stages of software development if compared to other measures like Lines of Code.

Today’s software market requires fast, agile, effective functional size measurement methods with low impact on production processes, which require not too specialized skills, that are reliable in results, not dependent on expert’s opinions and technology. The resulting measures should be adequately correlated to effort, cost, duration of a new development or functional enhancement software project. Current Functional Size Measurement Methods (FSMM) are only partially compliant with these needs. In order to facilitate the adoption of a FSMM by the organizations that are not yet using any of them and to improve the usage by the organizations that have already adopted one or more of them, it is essential to adequately satisfy these needs.

The work reported here was supported by the FP7 Collaborative Project S-CASE (Grant Agreement No 610717), funded by the European Commission and by project “Metodi, tecniche e strumenti per l’analisi, l’implementazione e la valutazione di sistemi software,” funded by the Università degli Studi dell’Insubria.

IFPUG FP analysis, performed by a certified function point consultant, proceeds at a variable pace: between 400 and 600 function points (FP) per day, according to Capers Jones [4], between 200 and 300 function points per day according to experts from Total Metrics [5]. As a consequence, measuring the size of a moderately large application (in the range from 1000 FP to 1500 FP) may take an average effort of 5 days of a measurement expert. If cost estimation is urgently needed, this could be too much. Depending on the software domain it is often needed the cooperation of a domain expert for half the time, leading to a total average effort of 7 person-days. Since measurement should be repeated during life cycle from two to three times this means 21 person-day of measurement effort. Even if this is only a small fraction of the development cost many managers are reluctant to accept this apparently “non-productive” cost in the project budget. Finally, cost estimates may be needed when requirements have not yet been specified in detail and completely as needed by a standard IFPUG measurement.

To overcome the aforementioned problems, simplified IFPUG FP estimation processes have been proposed (see Section X).

Besides simplified FP estimation processes, a new functional measurement method –fully compatible with IFPUG FP and designed to be compliant with the ISO/IEC 14143 standards [6]– has been developed in 2010 by Roberto Meli. This measurement method, named Simple Function Point (SiFP) [7][8] features a measurement process that requires much less time and effort than IFPUG FPA. The advantage of SiFP sizing is clear: a SiFP size measure can be obtained earlier and at a smaller cost than an IFPUG FP measure, but the two measures are absolutely equivalent: so, for instance, if an effort estimation model requires that the size in FP is given, the size in SiFP can be used in its place.

This paper aims at showing that a) SiFP size measures are really fully compatible with IFPUG FP size measures, and b) that the performance of effort estimations model do not change if SiFP are used instead of FP. To this end, the data from the ISBSG database [9] have been analyzed using rigorous statistical techniques.

The paper is organized as follows: Section II provides a brief introduction to IFPUG Function Points, while Section III
introduces Simple Function Points. Section IV describes the empirical study; Section V reports about the analysis of the correlation of SiFP and IFPUG FP, based on the ISBSG dataset; Section VI validates the results given in Section V by applying SiFP measurement to a different dataset. Section VII shows that the correlation of SiFP with development effort is as good as the correlation of IFPUG FP to effort. Section VIII validates the idea of using SiFP for effort estimation by evaluating their accuracy in estimating applications that are more or less “complex” than average in IFPUG terms. Section IX discusses the possible reasons for choosing SiFP, while Section X accounts for related work and Section XI discusses the main threats to the validity of the empirical study. Finally, Section XII draws some conclusions and outlines future work.

Throughout the paper, we assume that the reader is familiar with the concepts of FPA and the IFPUG rules. Readers that need explanations and details about FP counting can refer to official documentation and manuals [2][3].

Throughout the paper, we refer exclusively to unadjusted function points (UFP), even when we talk generically of “Function Points” or “FP”.

II. FUNCTION POINTS

Function Point Analysis was originally proposed by Allan Albrecht to measure the size of data-processing systems from the end-user’s point of view, with the goal of estimating the development effort [1].

The initial interest sparked by FPA along with the recognition of the need for improvement in its counting practices led to founding the International Function Points User Group (http://www.ifpug.org/), which provides guidelines for carrying out measurement, makes FPA counting rules evolve along with the evolution in software modeling. In fact, the International Function Points User Group (IFPUG) maintains the definition of the method and publishes and regularly updates the official Function Point (FP) counting manual [2].

The IFPUG also oversees method standardization: IFPUG FPA is now an ISO standard in its “unadjusted” version [3].

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

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

IFPUG BFC are data functions (DF), which are classified into internal logical files (ILF) and external interface files (EIF), and transactional functions, which are classified into external inputs (EI), external outputs (EO), and external inquiries (EQ) according to the primary intent of the process. Each function, whether a data or transactional one, contributes to the final measure with a number of FP that depends on its “complexity.” Each function is weighted on the basis of its complexity according to given tables. Finally, the number of so-called Unadjusted Function Points (UFP) is obtained by summing up the contribution of the function types. Details about FP measurement can be found in the manual [2].

We use notation M(x) to indicate the measure of the size of BFC x, obtained according to IFPUG measurement rules. So, for instance, M(EI) is the measure of the size of external inputs. The size of a software application —expressed as the number of Unadjusted Function Points (UFP)— is

\[ UFP = M(EI) + M(EO) + M(EQ) + M(ILF) + M(EIF) \]  

The FP measurement process involves (among others) the following activities:

- Identifying logic data;
- Identifying elementary processes;
- Classifying logic data as internal logical files (ILF) or external interface files (EIF);
- Classifying elementary processes as external inputs (EI), outputs (EO), or queries (EQ);
- Weighting data functions;
- Weighting transaction functions.

Simplified measurement processes allow measurers to skip—possibly in part— one or more of the aforementioned activities, thus making the measurement process faster and cheaper.

III. SIMPLE FUNCTION POINTS

SiFP is a new measurement method, created by Roberto Meli and subsequently published by the Simple Function Point Association in an official Reference Manual [9] available in the public domain. SiFP is not just a different, simplified process to estimate IFPUG FP measures, although it was originated by the evolution of functional size estimating techniques like the estimated NESMA FP [15] and the Early & Quick FP [14]. The basic idea is that, in order to have a good evaluation of the functional size of an application, it is not necessary to identify several types of transactions (classified according to the primary intent) and files (classified as internal or external) and that a notion of complexity based on the number of logical data fields (DET - Data Element Type) or of cross references among transactions and files (FTR - File Type Referenced) or subgroups of data in a file (RET - Record Element Type) is not significant to the goal of representing functional size and of estimating effort or costs. Accordingly, the SiFP method defines only two BFCs (Fig. 1), known as:

- Unspecified Generic Elementary Process (UGEPI)
- Unspecified Generic Data Group (UGDG)

The first one is a transactional object type while the second is a data object type.
Using the same notation as before, the size of a software application – expressed as the number of Simple Function Points – is

$$\text{SiFP} = M (\text{UGEP}) + M (\text{UGDG})$$  \hspace{1cm} (2)

Actual formulas take into account the different types of measurement (new development, functional enhancement, software asset).

Even though SiFP measurement considers fewer details than IFPUG measurement, the convertibility of SiFP to IFPUG FP has been shown to be so strong that SiFP measures can be considered statistically equivalent to IFPUG FP measures. Section V is dedicated to the verification of the equivalence of SiFP and IFPUG FP measures, based on rigorous statistical analysis applied to a set of data from over 700 projects.

An empirical study may show us the quality of “convertibility” between the two measurement methods based on actual, real data. In the conversion, the average difference is influenced by the typical distribution of IFPUG BFC with respect to complexity. But there is also a different viewpoint to be considered. Due to the limited scales assigned by the IFPUG method to the BFC complexities we may compute the maximum theoretical error that could arise in any specific measurement.

Every UGEP has size 4.6 SiFP, while every UGDG has size 7.0 SiFP. This implies that –given a software application that includes X UGEP and Y UGDG– the difference between the SiFP size and the IFPUG FP size of the application is not greater than $2.4 \times X + 8 \times Y$. 2.4 is the maximum size difference for an elementary process: a high complexity EO is 7 FP and 4.6 SiFP; 8 is the maximum size difference for a logical file: a high complexity ILF is 15 FP and 7 SiFP. For instance, given an application involving 10 logical data files and 40 elementary processes (hence, sized $70 + 184 = 254$ SiFP) the maximum difference with respect to the IFPUG size is 176 FP (69%). This theoretical maximum is never approached in practice, since all logical files should be high complexity ILF, which is extremely unlikely.

IV. OVERVIEW OF THE EMPIRICAL STUDY

The statistical analyses reported in Sections V and VII were carried out using the ISBSG dataset [9]. A filter was used to include only data points with a quality score of A and B (the more reliable measures) and related to IFPUG method versions higher than 4.0. No other filter was used in order to have a variety of application domains, technologies, development methods etc.

The considered dataset includes 766 data points. The applications represented in the dataset have size ranging from a minimum of 10 UFP to a maximum of 3886 UFP. The average size is 371 UFP; the median size is 215.5 UFP.

We set $\alpha = 0.05$, as is customary in Empirical Software Engineering. When not explicitly mentioned, the models described in the remainder of the paper conform to the usual statistical validity criteria (p-value < $\alpha$, normally distributed residuals, etc.).

Throughout the paper, unless otherwise explicitly stated, we refer exclusively to Unadjusted Function Points –which are generally referred to as “UFP”– even when we talk generically of “Function Points”.

V. CORRELATION OF SiFP AND IFPUG FP

The correlation of SiFP and IFPUG FP was studied first via non-parametric correlation tests; then via Ordinary Least Square (OLS) regression. OLS regression was used with plain variables and after log-log transformation of variables.

A. Correlation tests

Both UFP and SiFP data are not normally distributed in the ISBSG dataset. Accordingly, we used the Spearman’s and Kendall’s tests to verify the correlation between UFP and SiFP.

Fig. 2. Plot of ISBSG applications: SiFP vs. UFP.
Spearman's rank correlation test provided a value of \( \rho = 0.988 \) (with p-value \(< 10^{-15}\)).

Kendall's rank correlation test provided a value of \( \tau = 0.907 \) (with p-value \(< 10^{-15}\)).

We can thus conclude that UFP and SiFP are very strongly correlated. This conclusion is supported also by the visual inspection of the data plot (Fig. 2).

**B. Linear regression**

We then proceeded to compute the OLS linear regressions. The following model was obtained (after eliminating 268 outliers – identified according to Cook’s distance [18]– out of 766 data points):

\[
\text{UFP} = 3 + 0.9924 \text{SiFP}
\]

The model is characterized by adjusted \( R^2 = 0.974 \). The accuracy of the resulting model is characterized by:

- MMRE (Mean Magnitude of Relative Errors) = 11.7%
- MdMRE (Median Magnitude of Relative Errors) = 9.7%
- Pred(25) = 92.95%
- Error range = [-47% .. 51%]

The model above has a problem: the normality of the distribution of residuals could not be verified with certainty (the p-value of the Shapiro-Wilk test for normality is 0.083, i.e., a borderline value).

Therefore, we computed OLS linear regression while forcing the intercept to be null. The resulting model (obtained after eliminating 321 outliers – identified according to Cook’s distance– out of 766 data points) is:

\[
\text{SiFP} = 0.998 \text{UFP}
\]

The model has adjusted \( R^2 = 0.994 \).
The distribution of relative residuals is shown in Fig. 5. The blue diamond is the average residual error.

C. Log-log regression

The distributions of both UFP and SiFP sizes of the applications in the ISBSG dataset are not normal. In these cases it is usual to perform a log-log transformation before computing the regression. By performing log-log transformation, we still got non normal variables. Moreover, the obtained model does not have normally distributed residuals.

The obtained model –having adjusted $R^2 = 0.9994$– is

$$\text{SiFP} = \text{UFP}^{1.00026}$$

Since the obtained model is very close to the one obtained via linear OLS and to $\text{SiFP} = \text{UFP}$, its accuracy is very close to the former models’ (see the boxplot of relative residuals in Fig. 6).

In conclusion, we have a model with statistical significance problems, which is essentially equivalent to other models that are statistically significant. Therefore, we safely adopt the linear model.

D. Equivalence of SiFP and UFP

The model obtained via OLS linear regression while forcing the intercept to be null is very close to 1 (namely, 0.998). That is, according to the model, the size in UFP is approximately also the size in SiFP (or vice versa).

To verify the statistical significance of the hypothesis $\text{SiFP} = \text{UFP}$, we computed the 95% confidence interval for the coefficient. We found that in the $\text{SiFP} = K \times \text{UFP}$ model, $K$ belongs to the interval [0.9907, 1.0052] with 95% confidence.

So, we can safely assume that $\text{SiFP} = \text{UFP}$, with a 95% confidence that the error is less than 1%. To give an idea of the entity of this error, by adopting the model $\text{SiFP} = \text{UFP}$, the difference with respect to the model found via OLS regression is at most one SiFP –that is, practically negligible– for sizes up to 732 UFP.

E. Comments on the correlations found

A good correlation of SiFP and IFPUG UFP was expected, since the very definition of SiFP was based on the analysis of the ISBSG dataset.

However, it should be remembered that SiFP are computed based on information that is coarser-grained than the information required for computing IFPUG UFP, which includes additional details on transactions and data complexity. So, in some sense SiFP are an “approximation” of IFPUG UFP. Accordingly, it could be expected that not considering complexity of data and transactions in the computation of SiFP could cause a relatively high level of approximation. The analyses reported above demonstrate that it is not so: there appears to be a very good correlation of SiFP and IFPUG UFP.

VI. VALIDATION OF THE STUDY ABOUT CORRELATION OF SI FP AND IFPUG FP

The results reported in Section V were partially expected, because the analysis was carried out using the ISBSG dataset, i.e., the same dataset used for defining SiFP. Even though the residuals are fairly small and normally distributed, and $R^2$ is quite high, further validation is desirable. So, to validate the results about the correlation of the SiFP and IFPUG FP, we analyzed a second dataset.
A. The validation dataset

This new dataset includes data from 140 software applications, ranging from a minimum size of 103 FP to a maximum size of 4202 FP. The average size is 801 FP (standard deviation 475) and the median size is 818 FP. That is, the measurements in the new dataset are fairly larger than those in the ISBSG dataset. This was due to the fact that the second data set is mainly made of “asset” measurements (software applications baselines), which are, of course, greater than the corresponding development or functional enhancement projects. Measurements belong to the same organization and to the same application domain, which is classifiable as MIS (Management Information System) with a high presence of enquiry functions (DSS and Business intelligence).

The data from this dataset are plotted in Fig. 8.

B. Correlation tests

The visual inspection of the data plot (Fig. 8) clearly shows that SiFP and IFPUG FP are correlated. The existence of a very strong correlation is supported also by tests:

– Spearman's rank correlation rho is 0.9926, with p-value < $10^{-15}$.
– Kendall's rank correlation tau is 0.9336, with p-value < $10^{-15}$.

C. Analysis of differences between SiFP and IFPUG FP

The distribution of the relative differences of SiFP and IFPUG FP sizes (i.e., (SiFP-IFPUG)/IFPUG) is represented via the box-plot shown in Fig. 9. The mean and median values are practically equal (5.2%). According to the Shapiro-Wilk test, we cannot reject the hypothesis that the distribution of the relative differences is normal.

Fig. 9. Boxplot of the relative distance of SiFP and IFPUG FP.

When considering the absolute relative differences between the two size measures, we got:

– mean absolute relative difference = 10.6%
– median absolute relative difference = 8.2%

These results are perfectly in line with those given in Section V.B. Actually, the mean absolute relative difference is even better than the MMRE of the model reported in Section V.B. Similarly, the median absolute relative difference is slightly better than the MdMRE of the same model.

In conclusion, we observed that SiFP measures are –on average– very close to IFPUG measures also in a sufficiently large and representative dataset that is totally independent on ISBSG dataset.

VII. EFFORT ESTIMATION

Since functional size measures are used mainly for development effort estimation, it is necessary to verify that estimates based on SiFP are at least as good as the estimates based on UFP. To this end, we:

1) analyzed the correlation of SiFP with the development effort;
2) analyzed the correlation of IFPUG UFP with the development effort;
3) we compared the residuals obtained in the two cases.

The following statistically significant model describes the relationship between Effort and size in UFP:

\[ \text{Effort} = 2.9204 \times \text{UFP}^{0.9074} \]

The model features adjusted $R^2 = 0.58$. In conclusion, we have that in the validation dataset SiFP measures are extremely close to IFPUG measures, with an average overestimation of size (+5.2%).
The model is not very accurate. In fact, it features MMRE = 116%, MdMRE = 52%, Pred(25) = 21.6%, Error range = [-89%, 1324%]). However, this is not relevant for our purposes: we just want to check if a model based on SiFP is as accurate as a model based on UFP or not. The fact that the two models are good or not depends on several factors (the homogeneity of the dataset, the number of parameters taken into account, etc.).

The following statistically significant model describes the relationship between Effort and size in SiFP:

\[
\text{Effort} = 2.773 \times \text{SiFP}^{0.92}
\]

The model features adjusted \( R^2 = 0.538 \).

Like the model based on UFP, this is not a very accurate model: it features MMRE = 117%, MdMRE = 51%, Pred(25) = 20.6%, Error range = [-91% .. 1362%].

The two models appear essentially equivalent with respect to effort estimation errors. This is quite clear from the boxplots given in Fig. 10 and Fig. 11, which compare the estimation errors of the UFP-based and SiFP-based effort models.

To prove that the two models provide equivalent estimation accuracy, we used the Wilcoxon sign rank test and the Mann-Whitney (Wilcoxon rank sum) test. Neither test could reject the hypothesis that the relative absolute residuals of the UFP-based model are equal to those of the SiFP-based model.

The analysis described above was repeated for Enhancement projects, with similar results. It appears that models of enhancement effort based on size expressed in UFP are equivalent to models based on size in SiFP.

On the one hand, we can conclude that SiFP can be used in place of UFP without any disadvantage. On the other hand, using SiFP instead of UFP implies that the measurement can be carried out earlier, faster and at a much lower cost.

VIII. VALIDATION OF SIFP FOR EFFORT ESTIMATION

The definition of SiFP does not take into consideration the details of functions that the IFPUG method uses to assign a weight to each function; rather, fixed average weights are assigned to data functions (UGDG) and transaction functions (UGE). Accordingly, many projects have similar IFPUG FP and SiFP size measures, as shown in Section V. However, several applications have different SiFP and IFPUG sizes: for instance, the project having ID 10337 in the ISBSG dataset has size 87 SiFP and 120 UFP, i.e., the IFPUG size is 38% bigger than the SiFP size. So, when considering individual software applications it is possible that the equivalence of IFPUG FP and SiFP size measures be less good than observed in Section V.

However, when considering effort estimation, the fact that for some projects SiFP size measures are considerably different from IFPUG FP measures does not imply that SiFP-based effort estimation is less accurate than IFPUG FP-based estimation. Actually, the utility of SiFP lies in the possibility of performing simpler measures that are as good as IFPUG measures for effort estimation.

Let us consider the ISBSG dataset: 214 out of 292 new development projects concern software applications having similar SiFP and IFPUG size measures, or, more precisely, UFP \( \geq 0.85 \) SiFP \( \land \) UFP \( \leq 1.15 \) SiFP. It is quite reasonable that
when estimating development Effort based on functional size for these projects, using SiFP or IFPUG FP measures does not make a great difference, the two measures being quite similar.

On the contrary, when we consider projects such that UFP $\geq 1.15$ SiFP (i.e., projects considered “complex” by the IFPUG method) or projects such that UFP $\leq 0.85$ SiFP (i.e., projects having “low complexity”, according to the IFPUG method) there are two possibilities:

1) The development effort is affected by the details (FTR, RET, DET, etc.) accounted for by IFPUG and ignored by SiFP. If this is the case, we should find a correlation of IFPUG FP with Effort that is significantly better than the correlation of SiFP with Effort.

2) The development effort is not affected by the details (FTR, RET, DET, etc.) accounted for by IFPUG and ignored by SiFP. If this is the case, we should find a correlation of IFPUG FP with Effort that is similar to the correlation of SiFP with Effort.

To test assess the two hypotheses, we performed the following empirical study:

a) We selected from the ISBSG dataset the new development projects for which the UFP/SiFP ratio is greater than 1+K or less than 1-K, with K $\geq 0.15$.

b) We computed the Effort vs. UFP and Effort vs. SiFP regression models (as done in Section VII for the entire dataset).

c) Finally, we compared the residuals of the effort models obtained at point b.

We performed the analysis for K=0.15 and 0.2. For values of K greater than 0.2, the number of selected projects is too small to support significant statistical analysis.

With K = 0.15 we obtained a dataset of 78 data points supporting the following models:

- Effort =13.22 $\times$ UFP$^{0.971}$ (adjusted $R^2$ = 0.566)
- Effort = 12.38 $\times$ SiFP$^{0.95}$ (adjusted $R^2$ = 0.536)

The accuracy indicators of these models are given in Table 1. It is quite clear that the two models feature quite similar accuracy levels. This is also confirmed by the Wilcoxon sign rank test and the Mann-Whitney (Wilcoxon rank sum) test: neither test can reject the hypothesis that the relative absolute residuals of the two models are equal.

<table>
<thead>
<tr>
<th></th>
<th>Effort =13.22 $\times$ UFP$^{0.971}$</th>
<th>Effort = 12.38 $\times$ SiFP$^{0.95}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>MMRE</td>
<td>97.4%</td>
<td>91.8%</td>
</tr>
<tr>
<td>MdMRE</td>
<td>49%</td>
<td>47.5%</td>
</tr>
<tr>
<td>Pred(25)</td>
<td>20.5%</td>
<td>21.8%</td>
</tr>
<tr>
<td>Error range</td>
<td>[-79%, 987%]</td>
<td>[-84%, 933%]</td>
</tr>
</tbody>
</table>

Table 1. Accuracy of Effort models based on UFP and SiFP.

With K = 0.20 we obtained a dataset of 30 data points supporting the following models:

- Effort =29.23 $\times$ UFP$^{0.789}$ (adjusted $R^2$ = 0.551)
- Effort = 17.22 $\times$ SiFP$^{0.86}$ (adjusted $R^2$ = 0.539)

The accuracy indicators of these models are given in Table 2. It is quite clear that the two models feature quite similar accuracy levels. This is also confirmed by the Wilcoxon sign rank test and the Mann-Whitney (Wilcoxon rank sum) test: neither test can reject the hypothesis that the relative absolute residuals of the two models are equal.

<table>
<thead>
<tr>
<th></th>
<th>Effort =29.23 $\times$ UFP$^{0.789}$</th>
<th>Effort = 17.22 $\times$ SiFP$^{0.86}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>MMRE</td>
<td>56.8%</td>
<td>58.1%</td>
</tr>
<tr>
<td>MdMRE</td>
<td>41%</td>
<td>39.3%</td>
</tr>
<tr>
<td>Pred(25)</td>
<td>33.3%</td>
<td>40%</td>
</tr>
<tr>
<td>Error range</td>
<td>[-85%, 247%]</td>
<td>[-88%, 269%]</td>
</tr>
</tbody>
</table>

Table 2. Accuracy of Effort models based on UFP and SiFP.

We also tested the effectiveness of SiFP to estimate the development effort of projects for which UFP $\geq (1+K)$ SiFP and projects for which UFP $\leq (1-K)$ SiFP separately. Also in these cases, the results of our analyses indicated that there is no difference in accuracy when using SiFP for estimation instead of UFP.

In conclusion, we must conclude that hypothesis 2) is true: considering details of logical data and elementary processes in functional sizing does not appear to improve the accuracy of effort estimation.

A possible interpretation of the results described above is based on the observation that the adjusted $R^2$ of Effort vs. Functional Size models is always around 0.5. This means that only about half of the variation of development effort of the considered projects is due to variations of the functional size; the remaining half is due to other factors, which do not include the details measured by IFPUG UFP and ignored by SiFP. In other words, if one wants to achieve accurate effort estimates, he/she should properly characterize the development effort, like the software complexity (in the sense of McCabe complexity, for instance), the experience and ability of developers, etc.

IX. WHY USE SI FP

In this section, the main reasons for using SiFP instead of IFPUG FP are summarized.

A. SiFP is easy to learn.

Mastering IFPUG FP Analysis is a quite long and expensive process. A basic 2 or 3 day long training is needed as a minimum to catch the fundamentals elements of the method needed to start measuring, but many more days of practice are essential to consolidate measurement capabilities. Much more training and field experiences are needed to become a Certified FP Specialist.

An empirical study on the SiFP learnability has been conducted in 2013 at DPO, involving more than 180 analysts with no previous knowledge of Function Points. They were exposed to 3 hours of training, assigned randomly to 75 small groups and asked to measure within 40 minutes a simple software application sizing 267,8 SiFP. The functional specifications were not sufficient for a detailed IFPUG measurement but contained all the needed BFCs. The results
were encouraging: 75% of participants measured a value within -10% and +7% of the proposed value and 89% of participants measured a value within -18% and +16% of the proposed value. The results of the experiment are shown in Fig. 12.

<table>
<thead>
<tr>
<th>Class</th>
<th>Frequency</th>
<th>% cumulative</th>
</tr>
</thead>
<tbody>
<tr>
<td>-35%</td>
<td>1</td>
<td>1.33%</td>
</tr>
<tr>
<td>-27%</td>
<td>2</td>
<td>4.00%</td>
</tr>
<tr>
<td>-18%</td>
<td>9</td>
<td>15.00%</td>
</tr>
<tr>
<td>-10%</td>
<td>24</td>
<td>48.00%</td>
</tr>
<tr>
<td>-1%</td>
<td>14</td>
<td>66.67%</td>
</tr>
<tr>
<td>7%</td>
<td>18</td>
<td>90.67%</td>
</tr>
<tr>
<td>16%</td>
<td>2</td>
<td>93.33%</td>
</tr>
<tr>
<td>24%</td>
<td>3</td>
<td>97.33%</td>
</tr>
<tr>
<td>&gt;24%</td>
<td>2</td>
<td>100.00%</td>
</tr>
</tbody>
</table>

Fig. 12. Learnability of SiFP method: an empirical experiment.

The results of the experiment suggest that the SiFP method is much simpler to be learnt than the IFPUG one and it can be effectively used even by novices.

B. SiFP is quick and easy to apply.

The SiFP and IFPUG FP measurement process activity flows are represented in Fig. 13. As already mentioned, the SiFP process is a subset of the IFPUG process, since the detailed analysis and weighting of UGEP and UGDG are not required. In Fig. 13, the approximate fraction of effort required by each activity is indicated. Note that some activities, like the collection of the relevant available documentation, have a cost that hardly depends on the size of the application (i.e., the cost of the activity is nearly constant); the given percentages apply to the average case. At any rate, it is quite clear that the SiFP process does not include the longest and most expensive phase, thus allowing measurers to save around 60% of the total effort.

In productivity terms we may say that measurement average pace could vary from 400 FP per person-day to 1000 SiFP per person-day. Field experience seems to suggest that actual productivity is greater than that, as the learnability experiment shows: 267.8 SiFP measured in 40 minutes corresponds to a productivity of around 2600 SiFP per person-day (assuming 6.5 productive hours per day).

C. IFPUG data can be reused as SiFP data

In general, when an organization switches from a measurement unit to another (e.g., from IFPUG FP to COSMIC FP), it faces the problem of converting historical data into the new measure. Otherwise, the valuable knowledge contained in the historical datasets would be lost.

Organizations owning historical datasets with size measures expressed in IFPUG FP face a very easy problem:

- if they retained the number of EI, EO, EQ, ILF and EIF in the datasets, they can calculate SiFP measures very easily;
- otherwise, they can take advantage of the fact that –as described in Section III and demonstrated in Section V– SiFP and IFPUG FP measures are equivalent, i.e., the size expressed in IFPUG FP can be assumed as a (very good) approximation of the size in SiFP.

D. Sizing Portfolios

In many cases, organizations are interested in knowing the size of their software portfolio, i.e., the total size of the software they own. Of course, the software portfolio size is obtained by summing up the sizes of the applications. Therefore, the considerations that apply to sizing applications apply to portfolios as well.

To test the equivalence of SiFP and IFPUG FP in sizing portfolios, we summed up the sizes of the applications represented in the ISBSG dataset. The difference between the sum of all measures made with the IFPUG method and the sum of all measures made using the SiFP method is equal to -1123 FP out of 284,005 FP (-0.4%). This result shows that absolute errors are compensated by combining the counts as if they were belonging to a large application portfolio.

This result is not surprising. In fact –being the distribution of the difference between SiFP and IFPUG FP close to normal (see Fig. 5) with null mean– the sum of such differences from a large set of projects will generally be close to zero.

In practice, the two metrics are nearly coincident when applied to a typical portfolio of initiatives or a real organization (but measuring via SiFP is quicker and cheaper).

X. RELATED WORK

Several approaches to simplifying IFPUG FP measurement were proposed in the literature [11] [12]. Here we briefly discuss the most popular ones.

The most well-known approach for simplifying the process of FP counting is probably the Early & Quick Function Points (EQFP) method [13][14].

The Early & Quick (E&Q) functional size estimation method is a consistent set of concepts and procedures that, even when applied to non-detailed system or project
information, maintains the overall structure and the essential concepts of standard functional size measurement methods.

The E&Q method combines different estimating approaches in order to provide better estimates of a software system functional size: it makes use of both analogical and analytical classification of function types (transactions and data). Moreover, it allows the use of different levels of detail for different branches of the system (multilevel approach): the overall global uncertainty level in the estimate (which is a range, i.e. a set of minimum, most likely, and maximum values) is the weighted sum of the individual components’ uncertainty levels. The “core driver” of the method is an analytically and statistically originated table of UFP (Unadjusted Function Points) values to be used in making functional size estimation.

The starting point of the process is the logical product breakdown structure of the system being estimated, and the mapping of FURs on the E&QFP elements. The basic E&QFP elementary components are the following software objects:

- logical data groups, and
- elementary functional processes,

that is, the Base Functional Components (BFC) of the IFPUG measurement method.

Further aggregations, as depicted in Fig. 1, are provided:

- data BFC can be grouped into general data groups;
- transactional BFC can be grouped into “general” logical processes;
- general processes can be grouped into “macro” logical processes.

![Fig. 1 - Functional hierarchy in the E&Q estimation method](image)

Each “software logical object” is assigned a set of UFP values (minimum, most likely, maximum) based on an analytical/statistical table, then the values are summed up to provide the overall estimation result (minimum, most likely, maximum). Simple Function Point BFC (UGEP, UGDG) are already present in the latest version of E&QFP as the detailed level.

The Indicative NESMA method dramatically simplifies the functional size measurement process, by only requiring the identification of LogicData from a conceptual data model; however, it provides quite roughly approximated estimates [12].

The Estimated NESMA method [15] requires the identification and classification of all data and transaction functions (differentiated by the primary intent), but does not require the assessment of the complexity of each function: Data Functions (ILF and EIF) are all assumed to be of low complexity, while Transactions Functions (EI, EQ and EO) are all assumed to be of average complexity. The Estimated NESMA method provides a fairly good level of accuracy [12].

With the Tichenor ILF Model [16] the estimate of size is obtained by multiplying the number of ILF by a constant. However, this model assumes a given distribution of EI, EO, EQ and EIF with respect to ILF. If the considered application features a different distribution, the estimation can be inaccurate.

The ISBSG distribution model is based on the same considerations that inspire the Tichenor model. In fact, the method assumes that Basic Functional Components are distributed as in the ISBSG dataset. The size estimate is obtained by multiplying the number of ILF by a constant. The same considerations reported above for the Tichenor model apply: if the application to be measured does not fit the distribution assumed by the ISBSG distribution model, it is likely that the estimation will be inaccurate.

The simplified FP approach [17] assumes that all BFC are of average complexity.

Finally, the ISBSG average weights method assumes that each Base Functional Component has the average weight observed in the ISBSG dataset. This model provides fairly good estimates [12].

**XI. Threats to Validity**

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

A. Threats to Internal Validity

The datasets used in the empirical study are large enough to assure fairly good internal validity. Besides, we filtered out outliers, to make sure that the results are not unduly influenced by a very small number of high-leverage points; even so, the number of the remaining data points was comfortably large.

B. Threats to external validity

The equivalence of SiFP and IFPUG FP was studied in two different and large datasets, which are representative of various types of software applications. Nevertheless, application domains or typology (operational, decisional, multichannel, GIS, etc.) could influence the quality of correlation between SiFP measures and IFPUG measures. For instance, applications much more data-intensive than the ISBSG applications have probably a SiFP size different from their IFPUG FP size.
It is however worthwhile noticing that equivalence of functional size measures is hardly important per se. What is really valued by most practitioners is the possibility of using functional size measures for the estimation of the development effort. To this end, SiFP proved essentially equivalent to IFPUG FP (see Section VII). Experimentation with additional datasets is advisable, to confirm and strengthen the obtained results.

C. Threats to Construct Validity

A first construct validity threat is due to the inherent subjectivity of counting FSM methods. We relied on the ISBSG data, but only selected the projects with the two higher categories of data quality, as is usually done in research studies that use ISBSG data.

Another threat may come from interpreting MMRE, MdMRE, and Pred(25) as accuracy indicators, as pointed out by criticisms in the previous literature. At any rate, we provided other accuracy indicators (like boxplot representing the distribution of residuals) to provide a more complete picture about the accuracy of our results.

XII. CONCLUSIONS

Before the availability of a large public data set of complete size measurements and effort, it was reasonable to think that functional complexity based on primary intents and quantity of attributes and file references might be relevant to determine better effort models. We have shown, here, that this assumption is not actually true, at least on the entire ISBSG data set related to IFPUG projects which is one of the most popular and used resource to derive effort and productivity models.

On the contrary, it is true that the accuracy of a model of correlation between actual work effort and the software functional size does not decrease when considering only the number of BFC in each of the two classes (data and transactions).

This finding makes the whole system of rules aimed at the differentiation between EL, EO, EQ, ILF and EIF and at the determination of the complexity of the single BFC (DET, RET, FTR), useless in business terms.

The consequences of this finding are truly relevant in terms of impact on the method and process of Function Points measurement.

Other important results are that adopting SiFP it is possible to reuse all the cost models, productivity rates, unitary costs, contract templates based on IFPUG FP since the two methods are numerically equivalent. Software measurement asset conversion is immediate; there is no need to go back to the requirements interpretation. It is an operation that may be done in few minutes for hundreds of data points using a simple formula in a spreadsheet. ISBSG IFPUG data are immediately usable for Simple FP when you know the number of ELEO, EQ, ILF and EIF identified in an IFPUG project (these data are obtainable by ISBSG on demand). It is now possible to reduce the costs of measurements, detailed analysis of documentation, discussions, interpretations, litigations but also of training and maintaining measurement experts.

Measurements may be done as part of the production process by the same analysts that produce functional specifications with a quality increase of measures. In this case, independent audits may assure the absence of intentional biases if needed.

Further researches will explore the convertibility between SiFP and IFPUG FP using different data sets and filtering on domain areas and application profiles.

REFERENCES
