A Study of the Correlation between Functional Size Measures and Object-oriented Measures from UML Requirements Models[*]

Luigi Lavazza[1][0000−0002−5226−4337] and Geng Liu[2][0000−0003−4686−0834]

[1] Università degli Studi dell'Insubria, Varese, Italy luigi.lavazza@uninsubria.it
[2] Hangzhou Dianzi University, Hangzhou, Zhejiang, China
liugeng@hdu.edu.cn

Abstract. *Background.* Functional size measurement methods aim at measuring the size of functional user requirements of software applications. Functional user requirements can be represented via different notations, including UML diagrams.
Objectives. In this paper, the relationship between functional size measures (namely IFPUG Function Points and COSMIC Function Points) and object-oriented measures of UML diagrams representing functional requirements are investigated.
Method. A set of functional requirement specifications was modeled via UML diagrams. The functional size measures of user requirements were derived via the standard IFPUG and COSMIC processes; the corresponding UML models were measured using a set of object-oriented metrics that are applicable to UML models representing requirements. Functional size measures were then compared to object-oriented measures.
Results. Statistically significant linear regression models were found. It was also found that object-oriented measures of UML requirements models can be used to estimate functional size measures with good accuracy.
Conclusions. The obtained results suggest that object-oriented measures –which tools can automatically extract from UML models– provide indications concerning requirements size that are substantially equivalent to those provided by functional size measures.

Keywords: Functional Size Measures · COSMIC · Function Point Analysis · Object-oriented measures · UML measures.

1 Introduction

Functional Size Measurement (FSM) methods [2, 13, 15, 8] aim at measuring the size of functional user requirements of software applications. Being available in the early phases of development, these measures are widely used to estimate the effort required to develop software applications.

[*] This work was partly supported by the "Fondo di ricerca d'Ateneo" funded by the Università degli Studi dell'Insubria.

Organizations that develop software are thus interested in FSM processes that are reliable, rapid and cheap, and that fit well in software development processes.

There are several different ways for organizing functional size measurement activities. The usual process starts from Functional User Requirements (FUR), which are generally expressed via a set of heterogeneous documents, such as data flow diagrams, Entity/Relationship diagrams, tables, formulas, text, etc. The FUR are examined by a measurer (usually a certified one), who has to identify the elements on which FSM is based. The elements to be considered are specified in the official manuals of the method being used: the IFPUG counting manual for Function Point Analysis [13] and the COSMIC manual for the COSMIC method [8]. Once the elements mentioned in the manuals have been identified, the counting is fairly easy.

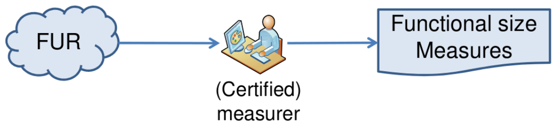

Fig. 1. The traditional functional size measurement processes.

The process (Fig. 1) is entirely manual, being carried out by the (certified) measurer. This fact has a few negative consequences. First, the process is slow. FPA performed by a certified function point consultant proceeds at a relatively slow pace: between 400 and 600 function points (FP) per day, according to Capers Jones [18], between 200 and 300 function points per day according to experts from Total Metrics [38]. Consequently, measuring the size of a moderately large application can take too long, if cost estimation is needed urgently. Second, the process is expensive, since the process takes some time and the work time of certified counters is expensive. Finally, the measurement has proved to be prone to some variability. Empirical data show that different counters can yield quite different measures of the same set of software requirements, even in the same organization [20]: a 30% variance was observed within an organization, while the observed difference was even greater across organizations [30]. Even according to the IFPUG, the difference between counts yielded by different certified experts for the same application may be around 10% [6].

The problems described above are not specific of FPA. In principle, using the COSMIC FSM method involves similar problems (although with quantitatively different effects).

An alternative process can be carried out when FUR are specified using UML (Fig. 2). In fact, in object-oriented development processes, requirements are often modeled via UML diagrams. In this way, two advantages are achieved: 1) requirements are modeled via a well defined set of diagrams, which are written in a standard and expressive language, 2) the transition from the requirements specification phase to the design phase is smooth. In such conditions, the UML diagrams representing FUR are the input of the FSM process.

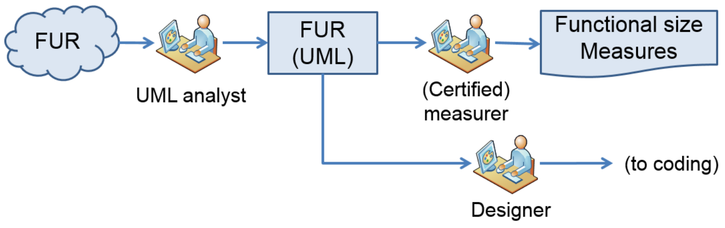

Fig. 2. Functional size measurement applied to UML models of FUR.

Several researchers addressed the problem of applying FSM methods to UML requirements specifications, as we report in Section 6. Unfortunately, these methods are applied to UML built according to whatever analysis methodology or point of view, so that the suitability of UML models for FSM is not guaranteed: models could omit some piece of essential information required by the counting rules, or could provide information at the wrong granularity level. In these cases the measurer has to integrate and adjust the available information: in practice, UML models become just another type of artifact in the (usually already crowded) set of documents used to specify user requirements. UML models that do not provide the correct information required by FPA open the door to subjectivity.

The need for a clear representation of FUR, upon which carrying out the measurement, is widely mentioned. Even the IFPUG newsletter mentioned this issue: *"As Function Points are counted based on the requirements, then it may be possible to identify a standardized structure for capturing requirements which will facilitate in identifying the FP context entities"* [34].

To overcome the problems described above, the process described in Fig. 3 was proposed: UML models are built having FSM rules in mind, so that the resulting models provide exactly the information required for FSM [24]. "Measurement-oriented" UML models can be derived from the heterogeneous set of documents that describe the FUR, or from non-measurement-oriented UML models that specify the FUR. In the former case, defining the measurement-oriented UML models can be relatively expensive, while in the latter case the UML to UML transformation is generally easy.

As highlighted in Fig. 3, once the measurement-oriented UML models are available, performing the measurement is easy, so that even an inexperienced measurer can carry out the measurement rapidly and correctly [9]. It could even be possible to automate the derivation of functional size measures from measurement-oriented UML models.

Several types of object-oriented measures have been proposed, including simple structural measures (like the number of classes, number of methods, etc.) as

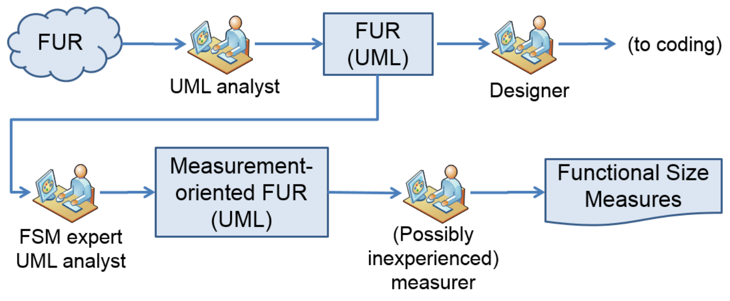

Fig. 3. FSM process based on Measurement-oriented models.

well as sophisticated measures (like the well-known suite by Chidamber and Kemerer [7]). A few tools are available to automatically derive several object-oriented measures from UML models. Such tools can be applied to functional size measurement-oriented UML models –which are in first place UML models– to obtain "regular" (i.e., not functional) measures (Fig. 4).

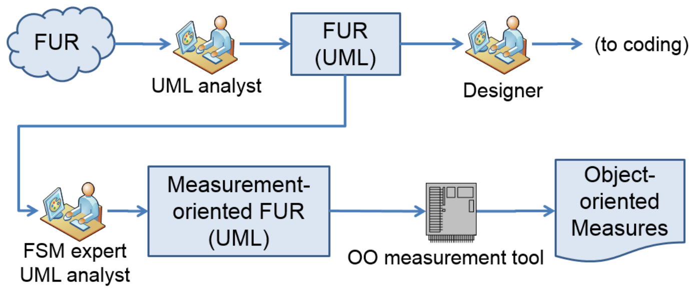

Fig. 4. Object-oriented measurement of FMS-oriented models.

The possibility of automatically deriving measures from UML models as shown in Fig. 4 leads to the following research questions:

RQ1 Are functional size measures and object-oriented measures of FSM-oriented UML models correlated? I.e., do the processes depicted in Fig. 3 and Fig. 4 yield correlated measures?

RQ2 If the answer to RQ1 is positive, what are the object-oriented measures that are correlated to functional size measures? More specifically: are FSM-oriented UML models needed, or do UML models built according to the usual object-oriented analysis practices [35] provide measures that are correlated to functional size measures?

In RQ1 and RQ2, we not only look for evidence of statistically significant correlations, we also look for regression models that fit sufficiently well to support accurate estimation of functional size measures based on a small set of object-oriented measures.

If the answer to RQ1 is positive, once you get a software model that is compliant with the FSM standards, object-oriented measures provide information that is equivalent to functional size measures, as far as sizing is concerned. Therefore, we could use the process described in Fig. 4 instead of any of the processes described in Fig. 1, Fig. 2 or Fig. 3.

If the answer to RQ2 is positive, there would be evidence that the measures of the FUR expressed as UML models built according to regular OO analysis practices [35] provide size information that is essentially equivalent to what is achieved via much longer and more expensive functional size measurement processes. This result would have important practical consequences in development environments that use UML, since measures that represent adequately the size of functional requirements could be achieved as a byproduct of the normal development activities (as shown in Fig. 5) at no additional costs (except possibly the cost of the UML measurement tool). Note that the work flow represented in the top part of Figure 5 is the regular object-oriented development: we have just added the automated measurement of UML requirements models.

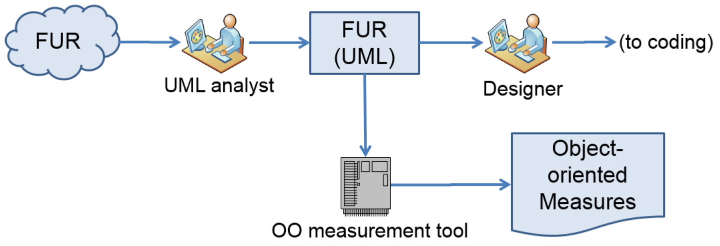

Fig. 5. Object-oriented measurement of UML requirements models as a byproduct of the regular development process.

This paper describes an empirical study aiming at answering the research questions defined above.

The paper is structured as follows. Section 2 describes the empirical study. The results of the statistical analyses are given in Section 3 and discussed in Section 4. Section 5 discusses the threats to the validity of the study, while Section 6 accounts for related work. Finally Section 7 draws some conclusions and outlines future work.

2 The Empirical Study

The empirical study was organized as follows:

1. We collected a set of UML models of software applications' FUR. The set is sufficiently large to support statistical analysis.

2. Functional size measures (both IFPUG FP and COSMIC FP) were derived from the FUR expressed via UML models. In this phase, the fact that FUR were expressed via UML is irrelevant: standard FSM processes were used, and measurers looked for the required elements (e.g., elementary processes and data movements) in the proper UML diagrams.
3. We used the SDmetrics tool [41] to get OO measures of UML models.
4. Possible correlations between the measures obtained at steps 2 and 3 were studied, using statistical methods, as described in Section 3.

Steps 1 and 2 were carried out as described in Fig. 3; all the applications' FUR were modeled and measured by a PhD student with the supervision of an expert. Step 3 was carried out as described in Fig. 4.

Our dataset includes the measures of 15 software applications: 7 applications were defined by students of a software project management course and were table game playing applications; 3 miscellaneous applications proposed in the literature for illustrating FSM principles and problems; a small information system defined to explain FSM to students; a tool to support measuring functional size according to multiple standards [25]; 3 open source project planning and measurement applications.

Descriptive statistics of the measured projects are given in Table 1.

Table 1. Descriptive statistics of the analyzed projects.

	COSMIC FP	IFPUG UFP	Num. classes	Num. methods	Num. use cases
Mean	93	114	9.1	32	85
st. dev.	28	29	4.4	14	26
Median	86	107	8	33	75
Min	50	73	3	13	38
Max	154	163	17	65	128

Object-oriented measures were obtained via the SDmetrics tool. After a brief survey of the available tools, we selected SDMetrics as the most complete, mature, usable, and easily available tool [41]. SDMetrics accepts as input XMI files and is able to measure several UML diagrams, including those providing the information needed for functional size measurement-oriented modeling. After the quite straightforward initial configuration of the tool, measurement was performed automatically.

Not all the measures provided by SDmetrics are relevant for our purposes: for instance, several SDmetrics measures are meant to represent the quality of OO design. Therefore, among the many measures supported by SDmetrics, we chose only those most likely related to the properties considered by functional size measurement. They are:

– Num_Class: The total number of classes in the model.
– Num_Attr: The total number of attributes in the model.
– Num_Met: The total number of methods in the model.

- AvMetperClass: The average number of methods per class.
- AvAttperClass: The average number of attributes per class.
- Num_UseCase: The total number of Use cases (measured from use case diagrams).
- Num_Msgs: The total number of messages in sequence diagrams.
- AvMsgsperClass: The average number of messages (from sequence diagrams) per class.
- AvMsgsperSD: The average number of messages (from sequence diagrams) per sequence diagram.

It is important noticing that in our UML models, we used functional size measurement criteria to identify use cases. Specifically, we have one use case for each IFPUG elementary process or COSMIC functional process. This makes our use cases and the related measures completely different from and not comparable to other measures based on the notion of use case, like use case points [19].

Sequence diagrams are used to describe the dynamic behavior of use cases, hence of IFPUG elementary process or COSMIC functional process. Therefore, AvMsgsperSD indicates the average number of messages per process.

Statistical analysis was carried out according to commonly accepted rules and conditions. So, we set a 0.05 statistical significance threshold throughout the paper, as is customary in Empirical Software Engineering studies. We used standard statistical significance tests when studying the statistical dependence between variables with Kendall's tau, and Spearman's rho and when building OLS regression models. Outliers were identified according to Cook's distance.

3 Analysis and Results

We looked for statistical models that could account for possible relationships between OO measures and the measure of functional size. To this end, we used Ordinary Least Square (OLS) linear regression.

We looked for models with one or two independent variables. We could not use more than two independent variables, because –given the size of the dataset– we would have risked overfitting.

3.1 Models of Object-oriented Measures vs. Functional Size Measures

We found that the measure of functional size expressed in Function Points is statistically related to several (sets of) object-oriented measures. The statistically significant models found are given in Table 2, together with the adjusted R^2 and the number of outliers that were eliminated from the dataset to derive the model.

The analysis performed to evaluate the correlation between object-oriented measures and IFPUG measures was repeated for COSMIC measures.

Also in this case we found fairly good correlations : the statistically significant models found are given in Table 3.

Table 2. Summary of Significant OLS models found for IFPUG FP.

Model	**Adj.** R^2	**#Outl.**
FP = 36.5 + 1.4 Num_Messages	0.75	0
FP = 3.8 + 1.04 Num_Attr + 3.9 Num_UseCase	0.82	1
FP = 14.5 + 1.106 Num_Met + 3.5 Num_UseCase	0.85	1
FP = 21 + 2.4 Num_UseCase + 0.87 Num_Messages	0.84	1

Table 3. Summary of Significant OLS models found for COSMIC FP.

Model	**Adj.** R^2	**#Outl.**
CFP = 18.9 + 1.4 Num_Messages	0.64	1
CFP = 7.1 + 2.4 Num_UseCase + 0.78 Num_Messages	0.65	0
CFP = -28.4 + 1.34 Num_Messages + 5.2 AvMessagesperClass	0.67	1

3.2 Evaluation of Models' Accuracy

To evaluate how well models based on object-oriented measures estimate functional size measures, we needed to avoid the risks connected with using accuracy statistic like the Mean Magnitude of Relative Errors (MMRE), which has been shown to be flawed, in that it is a biased estimator of central tendency of the residuals of a prediction system because it is an asymmetric measure [21][11][32].

Instead of MMRE, we used the Mean Absolute Residual (MAR) [37]: $MAR = \frac{1}{n}\sum_{i=1..n}|y_i - \hat{y}_i|$, where y_i and $\hat{y}_i$ are the i^{th} actual and estimated value of interest, in our case, the actual and estimated functional size of the i^{th} software application. Unlike MMRE, MAR is not biased, as discussed in [37].

When a new estimation model P is proposed, it is necessary to verify if it is a "good enough" model. To establish if P satisfies minimum accuracy conditions, we compare the proposed model with a "baseline" model, which requires little or no knowledge of the phenomena being estimated. To stay on the safe side, we used two baseline models: the random model and the constant model [37, 26].

When no obvious baseline model exists, Shepperd and MacDonell suggest to use as a reference model random estimation, based solely on the known (actual) values of previously measured applications. A random estimation $\hat{y}_i$ is obtained by picking at random y_j, with $j \neq i$. Of course, in this way there are n–1 possible estimates for y_i; therefore, to compute the MAR of the random model rnd we need to average all these possible values. Shepperd and MacDonell suggest to make a large number of random estimates (e.g., 1000), and then take the mean $\overline{MAR_{rnd}}$. Langdon et al. showed that this is not necessary, since the average of the random estimates can be computed exactly [22].

Shepperd and MacDonell observed also that the value of the 5% quantile of the random estimate MARs can be interpreted like α for conventional statistical inference, that is, any accuracy value that is better than this threshold has a less than one in twenty chance of being a random occurrence. Accordingly, the MAR of model P, MAR_P, should be compared with the 5% quantile of the random estimate MARs, rather than with $\overline{MAR_{rnd}}$, to be reasonably sure that P is actually more accurate than rnd.

Lavazza and Morasca [26] observed that the comparison with random estimation is not always effective in supporting the evidence that P is a good estimation model. Instead, they proposed to use a "constant model" ($const$), where the estimate of the size of the i^{th} application is given by the average of the sizes of the other applications; that is, $\hat{y}_i = \frac{1}{n-1} \sum_{j \in Y - \{y_i\}} y_j$.

In conclusion, to consider the models given in Tables 2 and 3 acceptable, we need that the MAR of the given model is less than both the 5% quantile of the random estimate MARs and MAR_{const}.

When considering IFPUG FP measures, with our dataset, the 5% quantile of the random estimate MARs is 28.3 FP and MAR_{const}=26 FP. Hence, we shall accept models of functional size expressed in FP that feature MAR $<$ 26 FP.

Table 4. MARs of models of IFPUG FP size.

Model	MAR
Num_Messages	13.5
Num_Attr, Num_UseCase	14.1
Num_Met, Num_UseCase	11.0
Num_UseCase, Num_Messages	11.8

Table 4 reports the MARs computed for the FP models given in Table 2. The estimates used for computing the MARs were obtained via a typical leave-one-out procedure, that is, given an application A from our dataset, an OLS linear model was obtained based on the applications from the dataset excluding A, and the resulting model was used to estimate the size of A.

It is easy to see that all the models' MARs are substantially smaller than 26 FP, hence all the found models are acceptable.

We now need to check that the absolute errors of the models found are actually smaller than the errors of the constant model. We use the the Wilcoxon Signed Rank test to test the following Null Hypothesis: "The absolute errors yielded by a model P are not less than those provided by the constant model". We use the Wilcoxon Signed Rank test because it can be safely applied also to not normally distributed data, since it makes no assumptions about data distributions. The Wilcoxon Signed Rank test rejected the null hypothesis for all the models found.

When considering COSMIC measures, with our dataset, the 5% quantile of the random estimate MARs is 23.5 CFP and MAR_{const}=21.6 CFP. Hence, we shall accept models of functional size expressed in CFP that feature MAR<21.6 CFP.

Table 5 reports the MARs computed for the FP models given in Table 3. It can be noticed that the model based on the number of messages does not perform better than the constant model, the MARs of the two models being very close. All the other models' MARs are substantially smaller than 21.6 CFP, hence these models should be considered acceptable. However, when applying the Wilcoxon Signed Rank test to absolute errors, we found that only the model

Table 5. MARs of models found for COSMIC size.

Model variables	MAR
Num_Messages	21.7
Num_UseCase, Num_Messages	12.8
Num_Messages, AvMessagesperClass	15.5

based on the number of use cases and the number of messages provides absolute residuals that are smaller than the constant model's.

4 Discussion of Results

A first quite interesting result is that object-oriented measures appear better correlated to IFPUG FP measures than to COSMIC measures. However, in both cases we found reasonably accurate models, which appear suitable for practical usage. In fact, if you consider that the difference between functional measures yielded by different certified experts for the same application may be around 10% [6], estimates based on UML measures –which feature MAR slightly greater than 10%– appear quite good.

Another interesting observation is that for both IFPUG and COSMIC measures, the number of use cases and the number of messages appear to be the measures of UML models that have more predictive power with respect to functional size measures. It is not surprising that the number of use cases is correlated with the functional size; in fact, in measurement-oriented UML models, use cases are used to represent elementary or functional processes, which are a fundamental concept in functional size measurement. As to the number of messages, we must remember that messages in UML sequence diagram represent phenomena that are conceptually very close to COSMIC data movements, which are the base functional component of the COSMIC size measure. Messages play a primary role in IFPUG measurement as well: DET that cross the system boundaries are message arguments, and FTR are sources or destinations of messages.

On the basis of these observations, RQ1 ("Are the functional size measures and object-oriented measures of FSM-oriented UML models correlated?") can be given a positive answer, in that a small set of object-oriented measures appears quite well correlated to functional size measures via OLS linear regression models.

Concerning RQ2 ("What are the object-oriented measures that are correlated to functional size measures? More specifically: are FSM-oriented UML models needed to get such measures, or could UML models built according to the usual object-oriented analysis practices [35] provide such measures?"), all the statistically significant models found are based on the number of use cases and the number of messages. Actually, models using these variables can be used to predict functional size measures quite accurately. So, the first part of the answer is that the number of use cases and the number of messages appear very well correlated to functional size measures.

Now, we have to consider that the number of use cases and the number of messages depend on the way the measured UML model is built. In fact, the scope and the level of details of use cases depends on the modeling 'style'. Similarly, the number of messages depends on what sequence diagrams are built, and how detailed they are. We made the variability of these measure close to zero by setting strict rules on how use cases are identified (they match functional/elementary processes) and imposing that a sequence diagram is built for every use case.

So the second part of the answer to RQ2 is that a few object-oriented measures from UML models built according to rigorous principles correlate with functional size measures; on the contrary, we have no evidence that measures that can be collected from every UML model –like the number of attributes, the number of methods and their class averages, etc.– correlate with functional size measures.

5 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.

5.1 Threats to internal validity

The limited size of the dataset may be a first threat to internal validity. Despite the relatively small number of data points, we still filtered out outliers, to make sure that the results are not unduly influenced by a very small number of high-leverage points, even though this further reduced the cardinality of the samples. Very few data points of the analyzed dataset proved to be outliers, though.

5.2 Threats to external validity

The dataset used in the study may not be representative of the entire universe of software applications. The relatively small size of the sample may make the models we found of limited external validity.

5.3 Threats to construct validity

In principle, the inherent subjectivity of FSM methods is a first main construct validity threat. In fact, different measurers could compute different sizes for the projects in our sample; this would lead to different correlations. However, it has been showed [9] that model-based measurement is less prone to subjectivity than FSM carried out according to traditional practices; therefore, we are quite confident that the variability of measures does not appreciably affect our results.

We also note that the tool we used counts messages only when they concern instances of classes in sequence diagrams, while our models also contains instances of components that send or receive messages. We tried to correct this issues, but the number of messages used in the analysis is still only a good approximation of the actual number of messages that appear in sequence diagrams (i.e., in elementary/functional processes).

6 Related Work

Several researchers addressed the issue of deriving functional size measures from UML models. For instance, measurement procedures that are compliant with FPA are defined in [28, 10, 39, 33, 40, 1]. The aim of the mentioned papers is to propose a correct interpretation of the FPA measurement principles and rules in the object oriented context, thus facilitating the application of FPA and improving its performances in object-oriented development processes.

Unfortunately, these methods suffer from a relevant drawback: they propose counting practices that are applied to a UML model after it has been built according to whatever analysis methodology or point of view (as in Fig. 2). Therefore, correctness and completeness of UML models with respect to FPA are not guaranteed, since the models were built without having FPA in mind. In the worst cases, the counting rules are just not applicable because the model lacks some piece of essential information required by the counting rules. Of course, the lacking information has to be provided in some way, e.g. "the person performing the count [...] has been requested to [...] integrate the information which may be available [in UML diagrams] using interviews or other documentation." [14].

To base functional measurement on UML diagrams, the information provided by the latter must be well defined and univocally understood; to this end, UML models should be formalized (or semi-formalized, at a minimum). Several scholars recommend that the object of measurement should be formalized [39, 12]. Jones Capers even states that "... from a technical point of view, it is feasible to automatically obtain a functional point (or other measure) from the demand, if the demand is represented by a structured language, HIPO, use case, CRC, or UML" [17].

Based on the precise mapping relationship between the functional elements and the UML structures, UML models have the ability of expressing the information needed by FPA and COSMIC measurement methods [24, 23].

A first UML based measurement procedure and tool were developed for Function Point Analysis (with a few simplifications) by Uemura et al. [39]. Improvements were proposed by Živkovič et al. [43].

The usage of UML as a notation to model the applications to be measured is fairly common in the COSMIC community. A survey of such approaches was published by Marín et al. [31]. One of the first among such techniques is due to Bévo et al. [5]. They map COSMIC concepts on a few UML diagrams: use cases, sequence diagrams, and classes. However, triggering events are not represented with UML concepts. A tool named Metric Xpert supports the automatic application of the measurement procedure [4]. The experimental application of the tool showed that it is able to produce measures that differ between 11% and 33% from measures obtained by experts.

The techniques proposed to derive COSMIC measures from UML models suffer from the problem mentioned above for FPA applied to UML models: for instance, Bévo observed that counting based on scenarios could give a much larger functional size then counting based on use cases. The problem, recognized later by Jenner [16] is that UML models can represent requirements at various levels of abstraction. As a solution, Jenner proposed using UML sequence diagram as the primary diagram to count COSMIC FP.

Levesque et al. also applied the COSMIC method to measure functional size from use case diagrams and sequence diagrams, with data movements mapped to messages in UML sequence diagrams [29].

Lavazza and Robiolo used measures from UML sequence diagrams in conjunction with functional size measures for effort estimation [27]. Sellami et al. further developed this idea: they used measures from UML models to complement COSMIC size measures, to derive more accurate effort estimation model [36]. In these models, measures of the structure of the sequence diagram account for data manipulations, which are not thoroughly represented by COSMIC measures.

Bagriyanik and Karahoca [3] proposed to automatically measure the functional size of software by using an ontology that formalizes the information created during the requirements engineering process.

Researchers also investigated the relationship between UML measures and code size: Zhou et al. found that measures from UML class diagrams and objective class points metric (an object-oriented version of functional size measures) are able to accurately predict source code size of object-oriented systems [42].

However, to our knowledge, nobody published studies of the correlation between object-oriented measures of UML models of FUR and functional size measures derived from the same UML models.

7 Conclusions

Performing functional size measurement according to the traditional process (Fig. 1) is expensive and prone to variability, because of requirement misinterpretations and measurement errors by human measurers.

Many software developers use UML, hence they are interested in basing functional size measurement on UML models. More precisely, they are interested in integrating FSM in the development process, and possibly in making FSM less expensive and less subject to variability. To this end, several approaches to extracting both IFPUG and COSMIC size measures from UML were proposed (Fig. 2).

Unfortunately, basing FSM on models that were built without having FSM in mind is not straightforward, since the models often do not provide the information required by FSM methods. A solution to this problem involves a two phases process [24, 23]:

1. First, UML models are built, taking care of incorporating all the information required by FSM. So, IFPUG elementary processes and COSMIC functional processes are represented as use cases or as operations of the interface offered by system component; data are modeled via component or class diagrams, and the details of processes are modeled via sequence diagrams. Organizations that already use UML for requirements modeling can derive measurement-oriented models quickly and at little additional cost.
2. Then, the measurement is carried out entirely on the basis of models. Identifying the elements to be counted in diagrams and computing the measures is straightforward, so that even an inexperienced measurer can carry out the measurement rapidly and correctly [9].

Measurement-oriented modeling and model-based measurement are meant to let organizations that already use UML extend the usage of UML models to FSM as well. No certified counter is necessary for FSM: this makes the procedure easier and cheaper to apply. Also the variability of measures is decreased [9].

Our empirical study showed that there is a correlation between object-oriented measures of UML models –which can be computed automatically– and functional size measures. This fact suggests that in principle object-oriented measures of UML models can be used to estimate functional size measures.

Using object-oriented measures of UML models involves two important advantages: measurement is automatic (so it can be performed at virtually no cost) and it is not subject to any variability. However, building measurement-oriented models requires some effort, even if FUR are already represented via UML models (typically, built according to traditional object-oriented analysis methodology).

Additional research is necessary to support the conclusions reported in this paper. In particular, a larger dataset would be needed. So, our plans for future work include collecting data from additional software applications. Moreover, we have to consider that the most common and relevant application of FUR measures is in effort estimation. We need to find data that include development effort measures, to verify whether effort estimation models based on object-oriented measures are more or less accurate than effort estimation models based on functional size measures.

References

1. Abrahao, S., Insfran, E.: A metamodeling approach to estimate software size from requirements specifications. In: Software Engineering and Advanced Applications, 2008. SEAA'08. 34th Euromicro Conference. pp. 465–475. IEEE (2008)
2. Albrecht, A.J.: Measuring application development productivity. In: Proc. of IBM Applic. Dev. Joint SHARE/GUIDE Symposium, Monterey, CA, 1979. pp. 83–92 (1979)
3. Bagriyanik, S., Karahoca, A.: Automated cosmic function point measurement using a requirements engineering ontology. Information and Software Technology **72**, 189–203 (2016)

4. Bévo, V.: Analyse et formalisation ontologique des procédures de mesure associées aux méthodes de mesure de la taille fonctionnelle des logiciels: de nouvelles perspectives pour la mesure. Ph.D. thesis, UQAM, Montréal (2005)
5. Bévo, V., Lévesque, G., Abran, A.: Application de la methode FFP a partir dune specification selon la notation UML: Compte rendu des premiers essais dapplication et questions. In: 9th IWSM, Lac Supérieur, Canada (1999)
6. Buglione, L.: Misurare il software 3/ed. Franco Angeli Editore, Milan (2008)
7. Chidamber, S.R., Kemerer, C.F.: A metrics suite for object oriented design. IEEE Transactions on software engineering **20**(6), 476–493 (1994)
8. COSMIC – Common Software Measurement International Consortium: The COSMIC Functional Size Measurement Method - version 4.0.2 Measurement Manual (December 2017)
9. Del Bianco, V., Gentile, C., Lavazza, L.: An evaluation of function point counting based on measurement-oriented models. In: EASE (2008)
10. Fetcke, T., Abran, A., Nguyen, T.H.: Function point analysis for the oo-jacobson method: a mapping approach (1998)
11. Foss, T., Stensrud, E., Kitchenham, B., Myrtveit, I.: A simulation study of the model evaluation criterion MMRE. IEEE Transactions on Software Engineering **29**(11), 985–995 (2003)
12. Galorath, D.D., Ferens, D.V., Fischrnan, L.: Automated Software Sizing From Use Case Points and Requirements Repositories. In: 18th International Forum on COCOMO and Software Cost Modeling (2003)
13. International Function Point Users Group: Function Point Counting Practices Manual - Release 4.3.1 (January 2010)
14. Iorio, T.: IFPUG Function Point analysis in a UML framework. Proceedings of SMEF 2004 (2004)
15. ISO: ISO/IEC 20926: 2003, Software engineering – IFPUG 4.1 Unadjusted functional size measurement method – Counting Practices Manual (2003)
16. Jenner, M.: Cosmic-ffp and uml: Estimation of the size of a system specified in uml–problems of granularity. In: Fourth European Conference Soft. Measurement and ICT Control. pp. 173–184 (2001)
17. Jones, C.: Estimating software costs: Bringing realism to estimating. McGraw-Hill Companies New York (2007)
18. Jones, C.: A new business model for function point metrics (2008), http://www.itmpi.org/assets/base/images/itmpi/privaterooms/capersjones/FunctPtBusModel2008.pdf
19. Karner, G.: Resource estimation for objectory projects. Objective Systems SF AB **17** (1993)
20. Kitchenham, B.: Counterpoint: the problem with function points. IEEE software **14**(2), 29 (1997)
21. Kitchenham, B.A., Pickard, L.M., MacDonell, S.G., Shepperd, M.J.: What accuracy statistics really measure. IEE Proceedings-Software **148**(3), 81–85 (2001)
22. Langdon, W.B., Dolado, J., Sarro, F., Harman, M.: Exact mean absolute error of baseline predictor, MARP0. Information and Software Technology **73**, 16–18 (2016)
23. Lavazza, L., Del Bianco, V.: A case study in cosmic functional size measurement: The rice cooker revisited. In: International Workshop on Software Measurement. pp. 101–121. Springer (2009)
24. Lavazza, L., Del Bianco, V., Garavaglia, C.: Model-based functional size measurement. In: Proceedings of the Second ACM-IEEE international symposium on Empirical software engineering and measurement. pp. 100–109. ACM (2008)

25. Lavazza, L., Del Bianco, V., Liu, G.: Analytical convertibility of functional size measures: a tool-based approach. In: Joint 22nd IWSM and 7th MENSURA. pp. 160–169. IEEE (2012)
26. Lavazza, L., Morasca, S.: On the evaluation of effort estimation models. In: Proceedings of the 21st International Conference on Evaluation and Assessment in Software Engineering. pp. 41–50. ACM (2017)
27. Lavazza, L., Robiolo, G.: Introducing the evaluation of complexity in functional size measurement: a uml-based approach. In: Proceedings of the 2010 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement. p. 25. ACM (2010)
28. Lehne, O.: Experience report: function points counting of object oriented analysis and design based on the ooram method. In: Conference on Object-Oriented Programming Systems, Languages, and Applications (OOPSLA'97) (1997)
29. Levesque, G., Bevo, V., Cao, D.T.: Estimating software size with uml models. In: Proceedings of the 2008 C 3 S 2 E conference. pp. 81–87. ACM (2008)
30. Low, G.C., Jeffery, D.R.: Function points in the estimation and evaluation of the software process. IEEE Transactions on Software Engineering **16**(1), 64–71 (1990)
31. Marín, B., Giachetti, G., Pastor, O.: Measurement of functional size in conceptual models: A survey of measurement procedures based on cosmic. In: Software Process and Product Measurement, pp. 170–183. Springer (2008)
32. Myrtveit, I., Stensrud, E., Shepperd, M.: Reliability and validity in comparative studies of software prediction models. IEEE Transactions on Software Engineering **31**(5), 380–391 (2005)
33. Oudshoorn, R.: Application of functional size measurement on requirements in uml. Ir.-degree Thesis, University of Twente (June 2005)(partly in Dutch) (2005)
34. Radford, P.: The Future of Function Points? IFPUG Metric View (July/August 2013)
35. Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F., Lorensen, W.E., et al.: Object-oriented modeling and design, vol. 199. Prentice-hall Englewood Cliffs, NJ (1991)
36. Sellami, A., Hakim, H., Abran, A., Ben-Abdallah, H.: A measurement method for sizing the structure of uml sequence diagrams. Information and Software Technology **59**, 222–232 (2015)
37. Shepperd, M., MacDonell, S.: Evaluating prediction systems in software project estimation. Information and Software Technology **54**(8), 820–827 (2012)
38. Total Metrics: Methods for Software Sizing – How to Decide which Method to Use (August 2007), http://www.totalmetrics.com/function-point-resources/downloads/R185_Why-use-Function-Points.pdf
39. Uemura, T., Kusumoto, S., Inoue, K.: Function-point analysis using design specifications based on the Unified Modelling Language. Journal of Software: Evolution and Process **13**(4), 223–243 (2001)
40. Van Den Berg, K., Dekkers, T., Oudshoorn, R.: Functional size measurement applied to uml-based user requirements. In: Proceedings of the 2nd Software Measurement European Forum (SMEF2005) (2005)
41. Wüst, J.: SDMetrics: The software design metrics tool for UML (2005)
42. Zhou, Y., Yang, Y., Xu, B., Leung, H., Zhou, X.: Source code size estimation approaches for object-oriented systems from uml class diagrams: A comparative study. Information and Software Technology **56**(2), 220–237 (2014)
43. Živkovič, A., Rozman, I., Heričko, M.: Automated software size estimation based on function points using uml models. Information and Software Technology **47**(13), 881–890 (2005)