Investigating Functional and Code Size Measures for Mobile Applications

Filomena Ferrucci, Carmine Gravino, Pasquale Salza
University of Salerno
{fferrucci, gravino, psalza}@unisa.it

Federica Sarro
University College London
f.sarro@ucl.ac.uk

***Abstract*—This paper investigates the use of the COSMIC functional size measurement method for mobile applications. Some proposals have been recently introduced to size mobile applications in terms of COSMIC. In this work we empirically analyse whether the COSMIC functional size of mobile applications can be exploited to estimate the size of the final applications in terms of lines of code and number of bytes of the source code and bytecode. To this end, we take into account a total of 7 different code size measures collected from 13 Android applications. The results of the empirical study show that the COSMIC functional size is strongly correlated to all the size measures taken into account and that it can be also used to predict the mobile application size in terms of bytes with a high accuracy.**

***Index Terms*—functional size measurement, COSMIC, android mobile application, code size measure, LOC, empirical study**

I. Introduction

Functional Size Measurement (FSM) methods have been widely investigated in software engineering research and also used in industry to size software systems in terms of the functionality provided to the users. They were introduced to overcome the limitations of the Lines of Code (LOCs), one of the most used size measure and the main input to parametric software cost and effort estimation tools [1]. The main issue with LOCs is that they are not available early in the development process when effort/cost estimations are needed. Differently, a functional size measurement method can be applied to software requirements and the obtained functional size can be exploited to estimate LOCs using backfiring FSM/LOCs ratios based on earlier projects. Then, the calculated LOCs can be used in the parametric software cost models, e.g., COCOMO [2].

Among the FSM methods, the Function Point Analysis (FPA) was the first to be introduced in 1979. Several variants have been then proposed (know as 1st generation of FSM methods) to improve the size measurement or extend its application domain. COSMIC [3] is a 2nd generation FSM method, being the first to comply to the standard ISO/IEC14143/1 [1]. It is based on fundamental principles of software engineering and measurement theory, and it is conceived to be applicable to business, real-time, and infrastructure software (or hybrids of these) [3].

Recent studies have investigated the applicability of 1st and 2nd generation FSM methods to mobile applications [4]–[10]. This domain is growing faster and specific software engineering processes, including functional size measurement and estimation techniques [6], might be required to improve the quality of these applications. The International Function Point User Groups (IFPUG) has proposed guidelines for the application of IFPUG FPA to mobile applications [7][8] and some software companies have used them [9]. As for COSMIC, three preliminary investigations to size mobile applications have been reported in literature [4][5][10] focusing on the measurement of mobile games apps [4], apps that use internal data storage [5], and approximated measurement methods [10].

In this paper, we further analyse the use of COSMIC to measure mobile applications by investigating (1) how it relates to some size measures of the source and compiled code, (2) if it can be used to predict the final application code size in terms of LOCs or number of bytes of the source code and bytecode. It is worth noting that the idea of estimating code size in terms of bytes has been recently proposed by Lind and Heldal [11] that presented a practical approach to estimate the size of compiled C code of embedded applications. Their study highlighted a better correlation between bytecode and COSMIC size with respect to LOCs. They argued that this was due to the fact that the compiler behaves always in the same way by filtering differences in programming style (like "condensed" programs with many operations per line, few comments vs. only one operation per line, many comments). They also encouraged further studies considering different types of applications and from other domains in order to conclude that bytes can be used as code size measure [12].

We conducted an empirical study on 13 Android mobile applications that manage information exchanged with a persistent storage inside the Android device. Thus, to measure their COSMIC functional size we followed the guidelines provided in [5]. The design of the study is summarised in Section II while the results are presented and discussed in Section III. Conclusions and future work close the paper.

II. The Empirical Study Design

To assess the relationship of the COSMIC functional size with other size measures of the final mobile application (source/byte) code, we investigated the following questions:

RQ1: Does COSMIC measure relate with code size measures for mobile applications?

RQ2: Can COSMIC measure be used to estimate code size measures for mobile applications?

In the following, we provide details about the employed data set, estimation technique, validation method, and evaluation criteria. Threats that could affect the validity of the empirical study are also discussed.

A. Data Set and Variables

The data set employed in this study includes 13 Android mobile applications randomly downloaded from Google Play Store. For each application one of the authors collected the requirements in a Functional User Requirements (FURs) document. Then, following the guidelines proposed by D'Avanzo *et al.* [5], COSMIC was applied to the FURs document obtaining the data movements summarized in Table I (where E, X, R, and W denotes the number of Entry, Exit, Read, and Write data movements obtained by applying COSMIC [3]). CFP is the independent variable in our study and it denotes the number of COSMIC Function Points characterizing an application.

Table I
FUNCTIONAL SIZE IN TERMS OF COSMIC

Descriptive statistics	Data Movements				CFP
	E	X	R	W	
Min	4.00	4.00	2.00	2.00	15.00
Max	39.00	52.00	32.00	22.00	145.00
Mean	15.23	18.38	7.92	7.77	49.31
Median	13.00	14.00	5.00	7.00	40.00
Std Dev	9.63	13.17	8.35	5.12	35.03

The information about the mobile application code sizes considered in our empirical study is summarized in Table II. Since in Android the Java code is mainly used to develop functionalities, while XML is used to design the user interfaces [13], we considered both Java and XML size, to analyse apart the components constituting the interface of the mobile apps. We did not measure any other raw asset files, such as images or local database files, because they do not directly relate to the application functionalities. We considered each Java class involved (except for external libraries) and only the XML layouts that are needed to visualize the application. For example, we discarded those XML files that describe additional resolutions for other devices, such as tablets, since these are optimizations and do not describe functionalities. As for the LOC, the variables Java_{LOC}, XML_{LOC}, and $\text{Total}_{\text{LOC}}$ represent the lines of code for the Java code, XML code, and their sum, respectively. As for number of bytes, we considered both the source and the compiled code. The variables Java_{kB}, XML_{kB}, and Total_{kB} represent the Java size, XML size, and their sum in terms of kilobytes, respectively. The variable $\text{Bytecode}_{\text{kB}}$ denotes the size of the compiled code in terms of kilobytes. The above measures were collected by using the apps APK files downloaded from the official store; the code size was measured with the "du" UNIX command and the lines of code with the "CLOC" tool.

Table II
MOBILE APP CODE SIZES

Descriptive statistics	Java		XML		Bytecode
	kB	LOC	kB	LOC	kB
Min	19.00	473.00	12.00	167.00	23663.00
Max	322.00	7514.00	94.00	1695.00	272775.00
Mean	122.50	2786.58	42.33	590.67	111892.33
Median	91.50	2295.5	38.00	487.50	94298.50
Std Dev	96.85	2157.30	23.20	427.38	73823.36

B. Correlation Test and Estimation Technique

To assess (RQ1) the relationship among the independent variable (i.e., CFP) and the dependent variables (i.e., Java_{kB}, Java_{LOC}, XML_{kB}, XML_{LOC}, Total_{kB}, $\text{Total}_{\text{LOC}}$, and $\text{Bytecode}_{\text{kB}}$) we applied the nonparametric association statistics Spearman's rho [14], which is widely employed in the literature. This statistic ranges from $+1$ to -1, where $+1$ indicates perfect correlation and -1 indicates a perfect inverse correlation, while 0 indicates no correlation.

Since we are also interested in verifying whether or not the functional size of a mobile application can be exploited to predict the corresponding code size expressed both in terms of bytes and lines of code (RQ2), we verified the strength of the relationship between each dependent variable described in the previous section (e.g., $\text{Total}_{\text{LOC}}$) and the variable CFP, by performing a Linear Regression (LR) analysis. To evaluate the goodness of fit of a regression model, several indicators can be considered. Among them, the square of the linear correlation coefficient, R^2, shows the amount of the variance of the dependent variable explained by the model related to the independent variable. Other useful indicators are the F value and the corresponding p-value (denoted by Sign F), which high and low values, respectively, denote a high degree of confidence for the prediction. For all the statistical tests performed, we decided to accept a probability of $5\,\%$ of committing a Type-I-Error [14].

C. Validation Method and Evaluation Criteria

To verify whether or not the obtained prediction values are useful estimations of the actual values we carried out a cross validation, which means that the original data set is divided into different subsets of training and validation sets. Training sets are used to build models with LR and validation sets are used to validate the obtained models. In particular, we exploited a leave-one-out cross validation, which means that the original data set is divided into $n = 13$ different subsets (13 is the size of the original data set) of training and validation sets, where each validation set has one observation.

In order to establish whether the predicted size is a useful estimation of the actual size we applied again the nonparametric association statistics Spearmans rho, while to assess the accuracy of the obtained estimations we exploited summary measures, namely MMRE, MdMRE and Pred (l) [15], which have been widely used in empirical studies to assess the accuracy of estimation models (see e.g., [16]). In the context of effort estimation, where these measures were proposed [15], l

is widely set to 0.25 and a good estimation model should have a MMRE ≤ 0.25 and Pred $(0.25) \geq 0.75$, that is, the mean estimation error should be less than $25\,\%$, and at least $75\,\%$ of the estimated values should fall within $25\,\%$ of their actual values [15]. In this study we used $l = 0.25$. In the future, we will further analyse this point.

D. Threats to Validity

The construct validity can be biased by the collection of the information used to determine the size measures. The measurement task of the functional size is crucial. One of the authors, with previous experiences in measuring software in terms of COSMIC, performed the measurement task. Another author cross-checked the information obtained. As for the measurement of the code sizes, we manually inspect Java classes and XML files to remove noisy content (e.g., third part libraries). Reliability of the data and lack of standardization should be taken into account for the internal validity [17][18]. We did our best to collect information in a uniform fashion. Instrumentation effects in general did not occur in this kind of studies. As for the conclusion validity, we carefully applied the estimation method and the statistical tests, verifying all the required assumptions. Another threat to conclusion validity could be the few number of applications composing the data set. However, observe that "a rule of thumb in regression analysis is that 5 to 10 observations are required for every variable in the model" [19]. Furthermore, this kind of studies can contribute to provide useful indications that can be further validated in subsequent studies. So, other investigations should be performed to verify/confirm the results of our study, possibly taking into account different kinds of mobile applications.

III. Results of the Study

Table III shows the results achieved by applying the Spearmans rho test to answer RQ1. We reported the statistics and the p-value of the test for each considered dependent variable. The independent variable is CFP. We can observe that all the employed code size measures are positively associated with CFP. Furthermore, the results of the tests are statistical significant as the p-values of the statistics reveal. This means that when the value of the CFP increases, the value of the code size measures (e.g., Java_{kB}) increases as well. We can also note that the dependent variables characterized by the highest association with independent variable CFP is $\text{Bytecode}_{\text{kB}}$ that is the size in terms of bytes of the compiled code since their Spearman's rho statistics is 0.952. High value were also obtained for Total_{kB} and $\text{Total}_{\text{LOC}}$ that are the total code size expressed in terms of bytes and lines of code, respectively. The variables Java_{kB}, Java_{LOC}, and XML_{LOC} also achieved statistics greater than 0.85, while in only one case the statistics is less than 0.8 (i.e., XML_{kB}).

According to the above results we can positively answer our first research question: for the considered mobile applications COSMIC measure well relates to the considered code size measures.

Table III
Correlation among the independent variable and each dependent variable

Dependent variable	Spearman	
	rho	p-value
Java_{kB}	0.905	< 0.001
Java_{LOC}	0.894	< 0.001
XML_{kB}	0.751	0.003
XML_{LOC}	0.872	< 0.001
Total_{kB}	0.922	< 0.001
$\text{Total}_{\text{LOC}}$	0.922	< 0.001
$\text{Bytecode}_{\text{kB}}$	0.952	< 0.001

To answer RQ2 we built size estimation models by exploiting LR. To this aim, we first verified the assumptions underlying its application: linearity (i.e., the existence of a linear relationship between the independent variable and the dependent variable); homoscedasticity (i.e., the constant variance of the error terms for all the values of the independent variable); residual normality (i.e., the normal distribution of the error terms), and residual uncorrelation (i.e., error terms are uncorrelated for consecutive observations). It is worth noting that we also verified the presence of influential observations (i.e., extreme values which might influence the models obtained from the regression analysis) by using the residuals plot and Cook's distance. According to this analysis no transformation of the original data was performed and no observation was removed.

All the obtained models were characterized by a high R^2 value, i.e., greater than 0.8 except for the model having XML_{LOC} as dependent variable for which the value is very close to 0.8 (i.e., 0.775). Furthermore, a high F value (except for the model having XML_{LOC} as dependent variable) and a p-value (Sign. F) less than 0.001 were obtained, indicating that the prediction is possible with a high degree of confidence.

To evaluate the accuracy of the obtained estimates we performed a leave-one-out cross validation and applied a Spearman's rho test to establish whether the predicted size is a useful estimation of the actual size. The results of the Spearman's rho test are reported in Table **??**. We can note that the predicted size is always statistically positively correlated with the actual size for all the considered size measures. Thus, we can conclude that the obtained size prediction distributions are a good indication of the actual size distribution.

We also computed the values of MMRE, MdMRE, and Pred (0.25) (see Table V). We can observe that CFP was a good indicator of the total code size in terms of bytes for mobile application, since MMRE is very close to 0.25, MdMRE is lower than 0.25, and Pred (0.25) is greater than 0.75 (i.e., the thresholds of Conte *et al.* [15] are satisfied). In the other cases, MMRE values between 0.33 and 0.46 were obtained. However, the results achieved for the models predicting Java_{kB}, Java_{LOC}, $\text{Total}_{\text{LOC}}$, and $\text{Bytecode}_{\text{kB}}$ are quite interesting since MdMRE values less than 0.25 and Pred (0.25) values close to 0.60 were obtained. The worst results were obtained for the models predicting XML_{kB} and

Table IV
CORRELATION AMONG THE PREDICTED SIZE AND ACTUAL SIZE FOR EACH CONSIDERED DEPENDENT VARIABLE

Size in terms of	Spearman	
	rho	p-value
Java_{kB}	0.901	< 0.001
Java_{LOC}	0.890	< 0.001
XML_{kB}	0.720	0.003
XML_{LOC}	0.872	< 0.001
Total_{kB}	0.918	< 0.001
$\text{Total}_{\text{LOC}}$	0.945	< 0.001
$\text{Bytecode}_{\text{kB}}$	0.918	< 0.001

Table V
ACCURACY RESULTS

Dependent Variable	MMRE	MdMRE	Pred (0.25)
Java_{kB}	0.46	0.46	0.54
Java_{LOC}	0.43	0.24	0.54
XML_{kB}	0.43	0.29	0.31
XML_{LOC}	0.32	0.28	0.46
Total_{kB}	0.29	0.19	0.77
$\text{Total}_{\text{LOC}}$	0.35	0.20	0.62
$\text{Bytecode}_{\text{kB}}$	0.33	0.21	0.54

XML_{LOC}. Thus, we can conclude that CFP based model provide better predictions when predicting Java size with respect to XML size. Furthermore, when predicting the total size (i.e., sum of the Java and XML size) in terms of bytes the predictions are very good according to the considered summary measures. The fact that the functional size allows us to accurately predict the total code size suggests us that the code that realizes the app functionalities and the one realizing its interface are equally important for this kind of software.

Thus, as for the second research question we can state that for the considered mobile applications, COSMIC measure can be used to estimate the source code size in terms of bytes.

IV. CONCLUSIONS AND FUTURE WORK

We have applied the COSMIC measurement method to calculate the functional size of 13 mobile applications by exploiting some guidelines that can be used together with the standard COSMIC measurement manual by software measurers [5]. The functional size obtained in terms of COSMIC has been used to predict some code size measures, namely LOCs and bytes of source and compiled code. Overall, the result of the study highlighted that, in the considered domain, the COSMIC functional size can be used to get early and accurate predictions of the size in terms of bytes, thus, confirming the findings of Lind and Heldal [12][11].

To the best of our knowledge our study is one of the first reported in literature on the application of COSMIC to mobile apps. Previous work investigated the measurement of mobile games [4], apps that do not store data in their application layer [10], and apps that manage information exchanged with a persistent storage inside the mobile device [5]. Future work will compare the existing proposals to investigate whether a unique/unified approach is suitable to measure different kinds of apps, or different approaches are needed. As future work we also intend to replicate this study with larger data sets and considering different kind of mobile applications to confirm/contradict the results reported in the present study. Finally the collection of effort data could be useful to derive effort/cost estimation models [20].

REFERENCES

[1] ISO/IEC, *ISO/IEC 14143-1:2007: Information technology - Software measurement - Functional size measurement - Part 1: Definition of concepts*, 2007.

[2] B. W. Boehm, C. Abts, A. W. Brown, S. Chulani, B. K. Clark, E. Horowitz, R. Madachy, D. J. Reifer, and B. Steece, *Software Cost Estimation with COCOMO II*. Prentice Hall Press, 2009.

[3] A. Abran, D. Baklizky, J. Desharnais, P. Fagg, C. Gencel, C. Symons, K. R. Jayakumar, A. Lesterhuis, B. Londeix, S.-I. Nagano, L. Santillo, H. Soubra, S. Trudel, F. Vogelezang, and C. Woddward, *The COSMIC Functional Size Measurement Method, The Measurement Manual, Version 4.0.1*, 2014.

[4] N. A. S. Abdullah, N. I. A. Rusli, and M. F. Ibrahim, "Mobile game size estimation: Cosmic fsm rules, uml mapping model and unity3d game engine," in *IEEE Conference on Open Systems (ICOS)*, 2014.

[5] L. D'Avanzo, F. Ferrucci, C. Gravino, and P. Salza, "Cosmic functional measurement of mobile applications and code size estimation," in *30th ACM/SIGAPP Symposium on Applied Computing (SAC)*, 2015.

[6] A. Nitze, A. Schmietendorf, and R. Dumke, "An analogy-based effort estimation approach for mobile application development projects," in *International Conference on Software Process and Product Measurement (IWSM-MENSURA)*, 2014, pp. 99–103.

[7] T. Preuss, *Mobile Applications, Functional Analysis, and the Customer Experience*. Auerbach Publications, 2012.

[8] ——, "Mobile applications, function points and cost estimating," in *International Conference on Cost Estimation and Analysis Association*, 2013.

[9] G. Sethumadhavan, "Sizing android mobile applications," in *6th IFPUG International Software Measurement and Analysis Conference (ISMA)*, 2011.

[10] H. van Heeringen and E. van Gorp, "Measure the functional size of a mobile app: Using the cosmic functional size measurement method," in *International Workshop on Software Measurement and the International Conference on Software Process and Product Measurement (IWSM-MENSURA)*, 2014.

[11] K. Lind and R. Heldal, "A practical approach to size estimation of embedded software components," *IEEE Transactions on Software Engineering*, vol. 38, no. 5, pp. 993–1007, 2012.

[12] ——, "On the relationship between functional size and software code size," in *ICSE Workshop on Emerging Trends in Software Metrics (WETSoM)*, 2010.

[13] Google. (2015) Android developers guide. [Online]. Available: https://developer.android.com/guide/index.html

[14] J. D. Gibbons, *Nonparametric statistical inference*. Marcel Dekker Inc., 1986.

[15] S. D. Conte, H. E. Dunsmore, and V. Y. Shen, *Software Engineering Metrics and Models*. Benjamin-Cummings Publishing Co., Inc., 1986.

[16] B. Kitchenham and E. Mendes, "Software productivity measurement using multiple size measures," *IEEE Transactions on Software Engineering*, vol. 30, no. 12, pp. 1023–1035, 2004.

[17] B. Kitchenham, L. Pickard, and S. L. Pfleeger, "Case studies for method and tool evaluation," *IEEE Software*, vol. 12, no. 4, pp. 52–62, 1995.

[18] W. Martin, M. Harman, Y. Jia, F. Sarro, and Y. Zhang, "The app sampling problem for app store mining," in *12th Working Conference on Mining Software Repositories (MSR)*, 2015.

[19] T. Menzies, Z. Chen, J. Hihn, and K. Lum, "Selecting best practices for effort estimation," *IEEE Transactions on Software Engineering*, vol. 32, no. 11, pp. 883–895, 2006.

[20] L. Buglione, F. Ferrucci, C. Gencel, C. Gravino, and F. Sarro, "Which cosmic base functional components are significant in estimating web application development? A case study," in *International Conference on Software Process and Product Measurement (IWSM-MENSURA)*, 2010, pp. 205–224.