COSMIC Functional Measurement of Mobile Applications and Code Size Estimation

Loris D'Avanzo, Filomena Ferrucci, Carmine Gravino, Pasquale Salza
University of Salerno
Via Giovanni Paolo II, 132, Fisciano (SA)
Italy
{fferrucci, gravino, psalza}@unisa.it

ABSTRACT
The paper presents the application of the COSMIC functional size measurement method in mobile environment. In particular, we describe how COSMIC has been applied to Android mobile applications, also through an example of measurement, and the identification of some possible recurrent patterns. Moreover, we report the results of an empirical study carried out to verify the ability of the COSMIC measure to estimate mobile applications code sizes, i.e., the amount of needed memory. The results show that in the considered domain it is possible to get early and accurate prediction of the needed memory space in bytes.

Categories and Subject Descriptors
[D.2.8] Metrics; [D.2.9] Management;

General Terms
Measurement, Management, Experimentation

Keywords
Functional size measurement, COSMIC, Mobile application, Empirical study

1. INTRODUCTION
Functional Size Measurement (FSM) methods measure software size in terms of the functionality provided to the users and have been introduced to overcome the limitations of the LOCs [8]. Among FSM methods, Function Point Analysis (FPA) was the first to be introduced in 1979 [1] and since then several variants have been defined (all known as 1st generation of FSM methods) with the aim of improving size measurement or extending the applicability domain. COSMIC is a 2nd generation FSM method, being the first to be conceived for complying to the standard ISO/IEC14143/1 [8]; it is based on fundamental principles of software engineering and measurement theory, and it was developed to be applicable to business, real-time, and infrastructure software (or hybrids of these) [6]. FSM methods have been widely applied both in software engineering research field and industry for sizing software systems and then employing the obtained functional size as independent variable in estimation models (e.g., effort estimation models), for productivity benchmarking, quality evaluation, etc.

Mobile applications domain is growing and in the near future specific software engineering processes, including functional size measurement and estimation techniques, could be used to improve the quality of those applications. As a matter of fact, the International Function Point User Groups (IFPUG) has recently proposed a sort of guidelines for the application of IFPUG FPA to mobile applications [16], [17] and some software companies tried to apply it [18]. As for COSMIC, at the best of our knowledge, only a preliminary example of application of the method for the mobile context has been reported in the literature [15]. In this paper, we show how we have used the COSMIC method to measure the functional size of a data set of 8 mobile applications, trying also to derive some common cases that can be useful as a first draft of guidelines for software measures. We took into account applications that manage information and have a database and we were supported by the Business domain specifications of the COSMIC guide. Moreover, we focused on the issue of memory size used by applications into mobile devices: in the current era mobile user can obtain a lot of applications out of the box, but a wide range of device is not able to maintain all of them. Code size with respect to functionality can also provide an indication of efficiency of the mobile application. So, it is important to control and predict the needed memory size in the early phases of the application development process. To this end, we propose to exploit the COSMIC functional sizes of Android mobile applications to estimate the corresponding compiled code sizes (i.e., the amount of memory used). This approach is similar to the one recently employed in the context of embedded applications [12][13][14]. We present the results of a preliminary study we conducted employing 8 Android mobile applications and the linear regression as estimation techniques.

Paper structure: Sec. 2 briefly describes the method COSMIC and how it has been applied on mobile applications in our study. The performed empirical study and the achieved results are shown in Sec. 3. Conclusions and future work conclude the paper.

2. Measuring Mobile Applications
In this section we first describe the COSMIC size measurement method and then how we applied it on mobile applications.

2.1 COSMIC FSM
Functional size is defined as 'size of the software derived by quantifying the Functional User Requirements (FURs)' [6]. FURs describe what the software is expected to do for its users. Some examples are data transfer, data transformation, data storage and

SAC'15, April 13-17, 2015, Salamanca, Spain.

http://dx.doi.org/10.1145/2695664.2695948

data retrieval. COSMIC defines a standardized measure of software functional size expressed in COSMIC Function Point (CFP) units. The measurement is designed to be dependent by only the FURs of the software to be measured and independent from any requirements/ constraints about their implementation.

Main concepts. A *functional process* is one of the main concepts defined in COSMIC. It is a set of data movements representing an elementary part of the FURs. A *functional user* is defined as a (type of) user that is a sender and/or an intended recipient of data in the FURs; this means a functional user can be a human or for instance an external device as well. Moreover, a *boundary* is a conceptual interface between the software being measured and its functional users. With these definitions, it is possible to focus about the four different data movement types: Entry (E) types move data from a functional user to a functional process; Exit (X) types move data from a functional process to a functional user; Write (W) types move data from a functional process to persistent storage; Read (R) types move data from persistent storage to a functional process.

1 CFP unit is given per each data movement and their sum represents the measurement size. COSMIC method defines a measuring process, which consists of three phases: the Measurement Strategy Phase, the Mapping Phase, and the Measurement Phase. Each of them is explained in the following.

Measurement Strategy Phase. This is the preliminary work phase in which the key parameters of the measurement are defined. Some of them are: the *purpose*, which defines what the measurement result will be used for; the *scope* defining which pieces of software (in terms of FURs) have to be measured; the *level of granularity* which describes how much detailed the documentation about the software is (e.g., in terms of the requirements description or also the structure description). All parameters are defined in the *COSMIC Context Software Model* and it is extremely necessary to define them carefully. In addition, when different functional sizes are compared and then the purposes of their measurements are equal, as in our work, it is essential to define this phase in a consistent way for ensuring the results being safely compared. COSMIC defines several Measurement Strategy Patterns for many commonly situations in which the COSMIC method needs to be consistent for different measurements [5]. A Measurement Strategy Pattern defines a standard combination of parameters to be determined in the Measurement Strategy Phase of a COSMIC measurement process. It also defines the possible types of data movements and provides a template for drawing the Context Diagram for the software to be measured [4]. Measurement Strategy Pattern is a concept introduced in the last current version (4.0) of COSMIC [6]. Previous versions do not refer to any strategy patterns.

Mapping Phase. In this phase the measurer extrapolates the functional processes from the available FURs. It is a technical work in which the principles and, above all, the rules of the COSMIC method (reported in the *COSMIC Generic Software Model*) have to be carefully complied with. The measurer identifies the potential functional processes inside the FURs remembering that each functional process is started by a triggering E and shall comprise at least two data movements: an E plus either an X or a W. The triggering E is the E of the functional user that starts the functional process. A functional process cannot have more than one triggering E. In some cases there could not be a one-to-one relation between a FUR and the functional processes. *Data manipulations* inside a functional process are not counted as CFP [6], thus COSMIC is not able to size data manipulation intensive systems. The *object of interest* is defined as any 'thing' that is identified from the point of view of the FURs; it may be any physical thing, as well as any conceptual object or part of a conceptual object in the world of the functional user about which the software is required to process and/or store data. Objects of interest should not match with terms related to specific software engineering methods (e.g., Object Oriented). Each E, X, R, or W is a movement of data group of a single object of interest. There are only two exceptions: the triggering E which can start a functional process without data movement, e.g., in specific enquiry for a list of items; the error/confirmation message which is defined as an X for the attention of a human user that either confirms only that entered data has been accepted, or only that there is an error in the entered data.

Measurement Phase. It defines how to count data movements, consisting in associating a CFP to each data movement. The amount of all data movements represents the functional value of the measurement. It is worth noting that in cases (differently from our work) of aggregating measurement sizes (software stratified into different layers) or when measuring the size of software changes, this phase may become more complex [6].

2.2 Applying COSMIC on the analyzed mobile applications

COSMIC defines two main domains of applicability [6]:

- *Business Application Software* that typically supports business administration, such as banking, insurance, accounting, personnel, purchasing, distribution or manufacturing, etc.
- *Real-time Software* that is typically employed to control events happening in the real world. Examples are software for telephone exchanges and message switching, software embedded in devices to control machines such as domestic appliances, elevators etc.

Mobile environment consists of several types of applications. In this paper, we focus on applications fable to manage data and information exchanged with a persistent storage inside the Android device. Those applications include CRUDL (Create, Read, Update, Delete, List) functionality, import/export data, sharing info, etc. They fall in the Business Application Software domain [4]. In the following, we describe our work methodology, by first presenting the employed Measurement Strategy Pattern and then describing how the measurement is made on a mobile application and some recurrent cases during the Mapping phase.

2.2.1 Measurement Strategy Pattern

The *purpose* of our measurements is to obtain the size of the FURs of the delivered Android applications, to be employed for estimating the needed amount of memory in the Android system. Then we define the *scope* of the measurement that consists of all FURs executed inside the developed application. This means that all the data movements inside the application are counted in the measurement. Whenever there is an external application used to execute a functional process (e.g., email application for sharing data), only the data movements between the application being measured and this external application are counted, as defined by the Context Diagram in Figure 1. Note that the diagram symbols are consistent with what is stated in [6].

Thus, the further amount of memory needed for external applications is not of interest for the aim of our study, and then their data movements cannot added to the data movements of the application being measured. On the contrary, we took into account

the data movements to/from external applications that are relevant for the amount of memory the application needs.

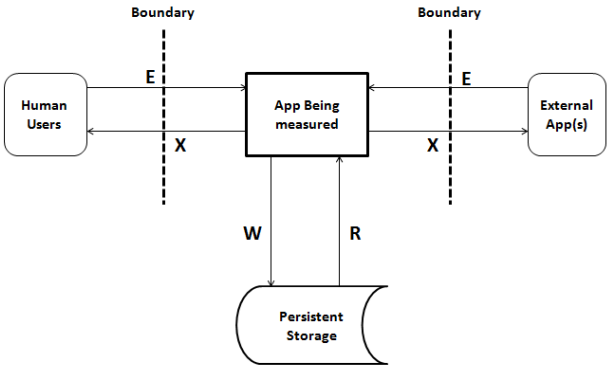

Figure 1. The Context Diagram

Example. Table 1 contains the FURs of an application considered in our empirical study. It implements a digital academic transcript handled autonomously by the user. Table 2 shows the results we obtained by applying the Mapping phase of the proposed process.

Table 1. Functional requirements of academic transcript

#	Functional User Requirements
R1	User opens the application to see on home screen the principal info included in his transcript, i.e., the list of exams, the number of exams, the number of credits and the average mark.
R2	User clicks on the icon button 'new' to insert data about a new exam in the database. The system provides error/confirmation messages.
R3	User selects an exam from the list in the home screen and clicks on the button 'delete' to delete it from the database. The system provides error/confirmation messages.
R4	User clicks on the button 'delete all' to delete all the exams data from the database. The system provides error/confirmation messages.
R5	User selects an exam from the list in the home screen and clicks on the button 'update' to update its data in the database. The system provides error/confirmation messages.
R6	User selects an exam from the list in the home screen and clicks on the button 'details' to see detailed info.
R7	User clicks on the icon button 'projection average' and the system shows a new box containing the current average mark and a form. The user selects in the form the number of expected exams, their number of credits and their expected mark. The system provides the expected average mark given by the input data values and the current exams.
R8	User clicks on the button 'export exams' to export exams from database to SD. The system provides error/confirmation messages.
R9	User clicks on the button 'import exams' to import exams from SD to the database. The system provides two error/confirmation messages, one for the input from SD and another for the writing on the database. The system shows the list of exams after importing.
R10	User sets the lode value for the statistics (30+0, 30+1 etc.). The system provides error/confirmation messages
R11	User sets maximum credits value. The system provides error/confirmation messages
R12	User clicks on the button to read change log.
R13	User clicks on the button to read FAQ.
R14	User clicks on the button to read application license.
R15	User clicks on the button to read info to donate a payment to the developer.

R1 consists of a basilar reading requirement that is mapped into a single functional process with six data movements, 1 triggering E of the human user, 1 R to the persistent storage for the required information and 4 Xs, one for each of information data sent to the user (the list of exams, the number of exams, the number of credits, and the average mark). Other requirements also consist in providing information to the user (e.g., R6, R12-R15) requested by a simple click. So, for them we have 3 data movements (as shown in Table 2). R2 consists in adding a new element in the database, thus we have 1 E for requesting the functionality and providing the corresponding parameter values, 1 W for create the new exam, and 1 X for possible error message. W data movements characterize also R3 and R4 that require a deletion, and R10 and R11 that consist in a setting operation.

There are also requirements more articulate (in terms of functional processes), e.g., R5 'update exam', R7 'average mark projection', and R8 'export exams'. In R5, the human user can select the exam from the list shown on the principal screen of the application, then data about the selected exam are shown in a new screen in order to allow user to update them. The data movements are:

- 1 (triggering) E - Exam data (ID) selected from list
- 1 R - Exam data (all) to update
- 1 X - Exam data to update
- 1 E - Exam data updated
- 1 W - Exam data
- 1 X - Error/confirmation messages.

Table 2. Data Movements of academic transcript FURs

#	E	X	R	W	Cases	CFP
R1	1	4	1	0	(i)	6
R2	1	1	0	1	(iii)	3
R3	1	1	0	1	(iii)	3
R4	1	1	0	1	(iii)	3
R5	2	2	1	1	(iv)	6
R6	1	1	1	0	(ii)	3
R7	4	2	1	0	(v)	7
R8	1	2	1	0	(viii)	4
R9	2	3	0	1	(vii)	6
R10	1	1	0	1	(iii)	3
R11	1	1	0	1	(iii)	3
R12	1	1	1	0	(ii)	3
R13	1	1	1	0	(ii)	3
R14	1	1	1	0	(ii)	3
R15	1	1	1	0	(ii)	3
	21	22	9	7		**59**

As for R7 'average mark projection', the functional process starts after a triggering E of the human user who clicks on the button 'projection average', then the functional process reads the list of exams from the persistent storages, computes the marks average and provides this information to the user with the new form. The user selects the values for number of exams, number of credits, and total mark and the system provides the new estimated average. So, data movements are:

- 1 (triggering) E - Click on the button
- 1 R - List of exams data
- 1 X - Current average mark
- 3 E - one for each input value selected by the user
- 1 X - Estimated average mark.

As for R8 'export exams', the list of exams is exported to the Secure Digital (SD) memory which is identified as a functional user. The functional process starts after a triggering E of the human user who clicks on the button 'export', then the functional process reads the list of exams from the persistent storage. Finally, the functional process executes the export operation to the

functional user SD. Data movements are:

- 1 (triggering) E - Click on the button
- 1 R - List of exams data
- 1 X - List of exams data
- 1 X - Error/confirmation messages.

2.2.2 Some Common Cases: Towards Guidelines for COSMIC Measurement of Mobile Applications

As we can observe from the Mapping phase of the academic transcript measurement, there are some kinds of FURs that are mapped into similar functional processes, i.e., with the same data movements (e.g., R12-R15). So, we tried to abstract those measurement processes giving a sort of guidelines for those common cases (considering that our data set falls in the Business domain and the analyzed applications manage data with an internal database) that are listed below:

(i) For mapping FUR as 'User opens the application to *see* info listed on home screen', the data movements are:
- 1 (triggering) E - The enquiry (represented by the opening application
- 1 R - Data (List of items)
- 1 X - Data (List of items)

For each other expected data shown on the screen an additional couple that includes 1 R and 1 X is added. But if the data is given by a *data manipulation* on the list of items, only 1 X is added (because there are no additional Rs to the persistent storage).

(ii) For mapping FUR as 'User clicks on the button to *read* info' or 'User selects an item from the list to see its detailed info', the data movements are:
- 1 (triggering) E - The only enquiry (or) the item ID
- 1 R - Data (Info)
- 1 X - Data (Info)

(iii) For mapping FUR as 'User clicks on the button to *Create/Set/Delete/Delete* all data', the data movements are:
- 1 (triggering) E - The only enquiry (Delete/Delete all) or data (Create/Set)
- 1 W - Data to delete/insert from/into the persistent storage
- 1 X - Error/Confirmation messages

(iv) For FUR as 'User clicks on the button to *update data*', the data movements are represented by the functional process to enquire the data (ii) and the functional process to update them (iii), giving rise to six data movements.

(v) For FURs where the system *processes* input data and data present in the database to provide an output to the user, the data movements are:
- 1 (triggering) E - The enquiry
- 1 R - Data on the persistent storage to be processed
- 1 E - Input data to be processed
- 1 X - The result shown on the screen

If there are other values to elaborate given in input by the user or retrieved from the persistent storage as well, additional E/R are mapped (e.g., in R7 we had 2 more Entries for the corresponding data). Additional Xs are also possible whenever different data are shown to the user (e.g., in R7 we had 1 more X, for the current average mark). If the system provides Error/Confirmation message, the correspondent X is also added.

(vi) For FUR as 'User clicks on the button to *share data* with an external application', the data movements are:
- 1 (triggering) E - The enquiry
- 1 R - Data
- 1 X - Data to external application

Additional error/confirmation messages after the X are not handled by the application being measured. If they occur, they are managed by the external application or by the Android system.

(vii) For FUR as 'User clicks on the button to *import data* from the SD to the database', the data movements are:
- 1 (triggering) E - The enquiry
- 1 E - Data from SD
- 1 X - Error/confirmation messages
- 1 W - Data
- 1 X - Error/confirmation messages

If the imported data are shown on the screen, as in academic transcript (see R9), another X is counted.

(viii) For FUR as 'User clicks on the button to *export data* from the database to the SD', the data movements are:
- 1 (triggering) E – The enquiry
- 1 R - Data
- 1 X - Data to SD
- 1 X - Error/confirmation messages

In a way similar to the previous mapping, if data are shown on the screen before the export step another E is counted.

The above cases are listed near to each requirement in Table 2.

3. The empirical study

In this section we present the empirical study we performed to assess whether the functional size, in terms of COSMIC, of mobile applications can be used to estimate the application code size, in terms of kilobytes (i.e., the amount of needed memory). To this end, we defined the following research question:

RQ: *Can COSMIC measure be used to estimate the mobile application code size in kilobytes of compiled code?*

In the following, we first describe the design of the study (Sec. 3.1) and then present and discuss the achieved results (Sec. 3.2).

3.1 Design of the Study

We describe the design of the study by providing 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.

Data set. It was related to 8 mobile applications downloaded from the Android applications market (google play). For each application one of the authors derived the FURs document. The COSMIC FSM was applied to that FURs document obtaining the data movements reported in Table 3. In the table we reported for each application also the code size (CodeSize column) representing the amount of memory needed by the application. Table 4 summarizes this information, reporting the descriptive statistics of the variables employed in our analysis.

Estimation technique. The goal of our study was to verify whether or not the functional size of a mobile application can be exploited to predict the corresponding code size (i.e., the amount of memory needed by the application). To this end, we verified the strength of the relationship between the variable CodeSize and the variable CFP, by performing a Linear Regression (LR)

analysis. LR is one of the most commonly used statistical techniques for exploring the relationship between a dependent variable and one or more independent variables, providing a prediction model described by an equation [2]:

$$y = b_1x_1 + b_2x_2 + ... + b_nx_n + c \quad (1)$$

where y is the dependent variable (the size), x_1, x_2, ..., x_n are the independent variables (the predictors) with coefficient b_i, and c is the intercept. In our empirical study we have exploited simple LR to obtain linear regression models that use CodeSize as dependent variable and only one variable, i.e., CFP, as independent variable.

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 (i.e, Sign F), which high and low values, respectively, denote a high degree of confidence for the prediction. We have also considered the p-values and t-values for the corresponding coefficients and the intercept. The p-values give an insight into the accuracy of the coefficients and the intercept, whereas their t-values allow us to evaluate their importance for the built model. In particular, p-values less than 0.05 are considered acceptable, meaning that the variables are significant predictors with a confidence of 95%. As for the t-value, a variable is significant if its value is greater than 1.5.

Table 3. Data set

Application name	CodeSize (KB)	Data Movements				CFP
		E	X	R	W	
Academic transcript	584	21	22	9	7	59
FuelStat	444	14	20	10	5	49
Simple Money	404	18	18	5	11	52
JustNote	292	15	19	5	9	48
Check list	208	13	13	4	4	34
Store products	164	10	12	3	5	30
Shopping note	160	10	11	4	6	31
Simple note	60	6	6	2	4	18

Table 4. Descriptive statistics of the variables

Variables	Obs	Min	Max	Mean	Median	Std.Dev
CodeSize	8	60	584	289.5	250	175.479
CFP	8	18	59	40.12	41	13.892

Validation method. 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=8 different subsets (8 is the size of the original data set) of training and validation sets, where each validation set has one observation.

Evaluation criteria. The accuracy of the obtained estimations was assessed by using summary measures *MMRE*, *MdMRE* and *Pred(l)* [3], which have been widely used in empirical studies to assess the accuracy of estimation models (see e.g., [7][10]). We have calculated summary measures as described in the following. The Magnitude of Relative Error can be defined as:

$$MRE = |CodeSize_{real} - CodeSize_{predicted}| / CodeSize_{real} \quad (2)$$

where $CodeSize_{real}$ and $CodeSize_{predicted}$ are the actual and the predicted amount of memory needed by an application, respectively. MRE has been calculated for each observation in the data set. All the MRE values were aggregated across all the data points using the mean and the median, giving rise to the Mean of MRE (MMRE) and the Median MRE (MdMRE).
The prediction at level l [3], defined as:

$$Pred(l) = p / n \quad (3)$$

where p is the number of observations whose MRE is less than or equal to l, and n is the total number of observations. Pred(l) is a quantification of the percentage of predictions whose error is less than l%. In the context of effort estimation, where these measures were proposed [3], l is widely set to 0.25 and a good estimation model should have a MMRE≤0.25 and Pred(0.25)≥ 0.75, that is, the mean error should be less than 25%, and at least 75% of the estimated values should fall within 25% of their actual values [3]. In the current study, which can be considered a preliminary investigation, we decided to use l=0.25.

Threats to validity. The construct validity can be biased by the collection of the information to determine the size measure. 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. A second researcher cross-checked the information obtained. Reliability of the data and lack of standardization should be taken into account for the internal validity [9]. We did our best to collect information in a uniform fashion. Instrumentation effects in general do not occur in this kind of studies. As for the conclusion validity, we carefully applied LR 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" [11]. Furthermore, our study can contribute to provide useful indications to be further validated in subsequent studies. So, other investigations should be performed to verify/confirm our results, possibly with different kinds of mobile applications.

3.2 Results of the Study

In order to apply LR we first verified the normality of distributions (i.e., of CFP and CodeSize). Furthermore, we verified the assumptions underlying the application of LR: *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). The performed statistical tests revealed that the residuals cannot be considered uncorrelated. Thus, we decided to perform a log-transformation of the employed variables since it is widely applied in this kind of studies (see e.g., [7][10]). We also verified the presence of influential observations (i.e., extreme values which might unduly influence the models obtained from the regression analysis). As suggested in [10], we analyzed the residuals plot and used Cook's distance to identify possible influential observations. No observation was removed.

Table 5 presents the results of the LR analysis with statistics on useful indicators to verify the quality of the obtained models. We can observe that the obtained model is characterized by a high R^2 value (98%). Furthermore, a high F value (232.9) and a low p-value (<0.01) were obtained, indicating that the prediction is possible with a high degree of confidence. The t-values and p-

values for the coefficient of CFP are greater than 1.5 and less than 0.05, respectively, meaning that CFP is a significant predictor with a confidence of 95%. The Equation as read from the final model's output, when transformed back to the raw scale, is:

$$CodeSize = CFP^{1.836} \cdot 0.302 \quad (4)$$

Table 5. The results of LR using CFP and CodeSize

	Value	Std.Err.	t-value	p-value
Coefficient of CFP	1.836	0.439	15.261	<0.01
Intercept	-1.196	0.12	-2.275	0.034
	R^2	Std.Err.	F	Sign. F
	0.975	0.125	232. 9	<0.01

To evaluate the accuracy of the obtained estimates we performed a leave-one-out cross validation and computed the values of MMRE, MdMRE, and Pred(0.25). The results are reported in Table 6. We can conclude that CFP was a good indicator of the mobile application code size, when used in combination with LR, since the values of MMRE and MdMRE are lower than 0.25 and the value of Pred(0.25) is greater than 0.75 (i.e., the thresholds of Conte et al. [3] are satisfied).

Thus, we can positively answer our research question, i.e., "*COSMIC measure can be used to estimate the mobile application code size in kilobytes of compiled code"*.

Table 6. Accuracy results

MMRE	MdMRE	Pred(0.25)
0.112	0.071	0.875

4. Conclusions and Future work

In this paper we applied the COSMIC measurement method to calculate the functional size of mobile applications. This is one of the first cases reported in the literature of application of that method to mobile applications, a rapidly growing type of applications that soon requires the use of suitable software engineering processes, including functional size measurement and estimation techniques, to improve their quality. As a matter of fact the International Function Point User Groups (IFPUG) has recently proposed a sort of guidelines for the application of IFPUG FPA to mobile applications [16], [17] and some software companies tried to apply it [18]. In the paper we report on the use of COSMIC to 8 Android applications that allows us to derive a sort of draft guidelines that can be used by software measurers.

Moreover, we have presented the result of an empirical study performed to assess whether in the considered domain the COSMIC functional size can be used to get early and accurate code size predictions (in Kb). The study was based on 8 mobile applications and a linear regression on their values was employed to build the prediction models. The results of the validation performed by applying a leave-one-out cross validation show that accurate estimates were obtained taking into account some thresholds widely used in the context of effort estimation.

As future work we intend to replicate the study with larger data sets and considering also different mobile applications to confirm/contradict the results achieved with the preliminary study presented here. We also want to investigate about a possible correlation between functional size and the RAM allocation during execution of Android applications. The collection of effort data could also be useful to derive effort/cost estimation models. Finally, other size measurement approaches could be investigated and compared with COSMIC.

5. REFERENCES

[1] Albrecht, A. Measuring Application Development Productivity, Joint SHARE/GUIDE/IBM Application Development Symposium, 83–92, 1979.

[2] Chatterjee, S., Simonoff, J. Handbook of Regression Analysis. John Wiley & Sons, 2013.

[3] Conte, D., Dunsmore, H., and Shen, V. Software engineering metrics and models. The Benjamin/Cummings Publishing Company, Inc., 1986.

[4] COSMIC: The COSMIC Functional Size Measurement Method, Guideline for Sizing Business Application Software, Version 3.0, 2008.

[5] COSMIC: Guideline for 'Measurement Strategy Patterns', Ensuring that COSMIC size measurements may be compared, Version 1.0, 2013.

[6] COSMIC: The COSMIC Functional Size Measurement Method, Measurement Manual, Version 4.0, 2014

[7] Corazza, A., Di Martino, S., Ferrucci, F., Gravino, C., Mendes, E. Investigating the use of Support Vector Regression for web effort estimation. Empirical Software Engineering 16(2): 211-243, 2011.

[8] ISO/IEC, ISO/IEC 14143-1:2007: Information technology - Software measurement - Functional size measurement - Part 1: Definition of concepts, International Organization for Standardization, Geneva, Switzerland, 2007.

[9] Kitchenham, B., Pickard, L., Pfleeger, S. Case studies for method and tool evaluation. IEEE Software, 12(4): 52–62, 1995.

[10] Kitchenham, B. A., Mendes, E. Software Productivity Measurement Using Multiple Size Measures. IEEE Trans. Software Eng. 30(12): 1023-1035 (2004).

[11] Kliijnen, J. Sensitivity Analysis and Related Analyses: A Survey of Statistical Techniques. J. Statistical Computation and Simulation, 57(1-4): 111-142, 1997.

[12] Lind, K., Heldal, R. A Practical Approach to Size Estimation of Embedded Software Components. IEEE Trans. Softw. Engin., 38(5): 993-1007, 2012.

[13] Lind, K., Heldal, R., Harutyunyan, T., Heimdahl, T. CompSize: Automated Size Estimation of Embedded Software Components. Conference on Software Process and Product Measurement, 86-95, 2011.

[14] Lind, K., Heldal, R. Estimation of Real-Time Software Code Size using COSMIC FSM. IEEE International Symposium on Object/Component /Service-Oriented Real-Time Distributed Computing, 244-248, 2009.

[15] Nitze, A., Measuring Mobile Application Size Using COSMIC FP. DASMA Metrik Kongress, 11/2013.

[16] Preuss, T., Mobile Applications, Functional Analysis, and the Customer Experience. In "The IFPUG Guide to IT and Software Measurement", (ed) Auerbach Publications, 2012.

[17] Preuss, T. Mobile Applications, Function Points and Cost Estimating. International Cost Estimation & Analysis Association Conference, 2013

[18] Sethumadhavan, G. Sizing Android Mobile Applications. International Software Measurement and Analysis, 2011