On the Relationship between Different Size Measures in the Software Life Cycle

Cigdem Gencel
Blekinge Institute of Technology,
Ronneby, Sweden
cigdem.gencel@bth.se

Rogardt Heldal, Kenneth Lind
Chalmers University of Technology,
Gothenburg, Sweden
heldal@chalmers.se, kenneth.h.lind@se.saab.com

***Abstract*—Various measures and methods have been developed to measure the sizes of different software entities produced throughout the software life cycle. Understanding the nature of the relationship between the sizes of these products has become significant due to various reasons. One major reason is the ability to predict the size of the later phase products by using the sizes of early life cycle products. For example, we need to predict the Source Lines of Code (SLOC) from Function Points (FP) since SLOC is being used as the main input for most of the estimation models when this measure is not available yet. SLOC/FP ratios have been used by the industry for such purposes even though the assumed linear relationship has not been validated yet. Similarly, FP has recently started to be used to predict the Bytes of code for estimating the amount of spare memory needed in systems. In this paper, we aim to investigate further the nature of the relationship between the software functional size and the code size by conducting a series of empirical studies.**

Keywords-Software Size; Functional Size; Source Lines of Code; Function Points; Bytes of Code; IFPUG; COSMIC; ISBSG

I. INTRODUCTION

Software size measurement involves a wide range of measures and methods [1][2]. This variation is mainly due to different size attributes of software entities produced throughout the software development life cycle. The major entities, the sizes of which are usually measured are the software requirements specification (SRS), design and the code. In the scope of this paper we mainly focus on the SRS and the code sizes.

The code size is the oldest attribute for which a number of measures are defined such as; Source Lines of Code (SLOC), number of characters, number of executable statements, bytes, etc. [2].

Among those, SLOC is the most widely used traditional code size measure which is the key input to most software effort estimation models as well as to performance measurements. It has also been used for the normalization of other measures. In order to better define and to enable the consistent usage of SLOC, a number of studies were also made such as [3].

Bytes of code is another code size measure which is used for other purposes as for measuring the amount of spare memory needed in a system [4].

Measuring the size of SRS involves many measures and methods [2]. The older ones focused on measuring the number of pages, the number of requirements etc. Recent measures and the methods attempt at measuring size by trying to capture the amount of functionality laid out on projects' functional user requirements (FUR) which are available earlier in the project lifecycle.

The original measure; Function Points (FP) and the method were introduced by Albrecht and Gaffney [5][6]. This new measure aimed at overcoming some of the shortcomings of the code size measures for estimation purposes and performance analysis, such as their availability only fairly late in the development life cycle and their technology dependence.

After then, the topic of Function Point Analysis (FPA) evolved quite a bit [7][8]. Many variations and improvements on the original idea were suggested some of which proved to be the milestones in the development of Functional Size Measurement (FSM). In 1996, the International Organization for Standardization (ISO) established the common principles of FSM methods and published ISO/IEC 14143 standard family in the following years [9][10][11][12][13][14][15].

The methods which conform to ISO/IEC 14143-1 standard [9] are accepted as international standards for FSM. Currently, the Common Software Measurement International Consortium Function Points (COSMIC FP) [16][17], the International Function Point Users Group (IFPUG) FPA [18][19], MarkII FPA [20][21], the Netherlands Software Metrics Association (NESMA) FSM [22][23] and the Finnish Software Metrics Association (FiSMA) FSM [24][25] methods are the ones accepted as FSM standards.

In spite of the fact that rigorous and well-defined measures and methods have been developed to measure the size for software entities throughout the software development process, one of the major issues still requires further investigation: the nature of the relationship between different size measures defined to measure different software entities [8].

One of the main reasons for such a need is the ability to estimate the size of the software entities produced at the later phases of the life cycle such as the code, by using the sizes of the entities produced early in the life cycle such as the SRS.

For example, the majority of software cost and effort estimating parametric tools such as COCOMO II [26], Putnam's Model/SLIM [27], SoftCost [28], Price-S [29] are based on code size as the primary input. Similarly, the amount of memory to be integrated in various kinds of hardware such as in cars, televisions, mobile phones, etc. requires bytes of code to be estimated when the code is not available yet.

In this study, we investigate the nature of the relationships between functional size measures and the code size measures by conducting empirical studies. For the functional size measures, we chose two of the widely-used ones: COSMIC FP (CFP) and IFPUG FP. For the code size measures, we chose SLOC and Bytes of Code. This paper is organized as follows: the related research is briefly summarized in the second section. We discuss the empirical studies we performed as well as the results obtained in the third section. Finally, conclusions are given in the fourth section.

II. RELATED RESEARCH

Jones [30] developed an approach called 'backfiring' to convert the length of code measured in SLOC and the functional size in IFPUG FP to one another. Backfiring is the direct mathematical conversion of SLOC and IFPUG FP to one another. It is based on the assumption that the functional size in IFPUG FP can be converted to SLOC by multiplying the former with an average ratio figure derived from earlier project data.

COCOMO II [26], which is a widely known effort estimation method, utilizes the backfiring approach to obtain SLOC which is the primary input in the estimation model. However, in a number of studies such as in [31][32], this kind of conversion were criticized for introducing extra uncertainty by adding another level of estimation.

Henderson [33] and then Desharnais and Abran [34] presented the results of their analysis on the SLOC to IFPUG FP ratio. Both studies concluded that the published conversion ratios should be used with caution due to large range of variations in the ratio figures.

Rollo [35] studied the reliability of SLOC/FP ratio by showing the results of an empirical study conducted on 20 applications and concluded that the use of backfiring functional size to SLOC is inherently inaccurate and that only homogeneous data allows for acceptable results.

Dekkers and Gunter [36] discussed the uncertainties and risks associated with using the backfiring method based on the fundamentals of these two measures.

On the other hand, Lind and Heldal [4][37][38] found a significant correlation between software code size of components in Bytes and functional size in CFP. They developed a linear model to estimate the amount of memory required in Electronic Control Units (ECU) in cars from CFP before the software is available.

The results of the literature survey shows few studies on the conversion of different size measures designed to measure different entities. More research is necessary to understand and explain the 'true' nature of these relationships so that they can be used reliably.

III. EMPIRICAL STUDIES

We designed and conducted empirical studies in order to investigate the nature of the relationship between functional sizes (in CFP and IFPUG FP) and code sizes (in SLOC and Bytes) at four different granularity levels; from higher to lower levels. The aim of conducting these empirical studies is to bring into light the factors which influence these relationships.

At the first and highest granularity level, we made an empirical study using the International Software Benchmarking Dataset (ISBSG) Release 10 [39] to explore the nature of the relationship between the length of code (in SLOC) and functional size (in CFP and in IFPUG FP).

Then, we conducted a multiple-case study which involved two projects from an organization. We looked at the SLOC and CFP relationship in more detail.

Later, at a lower granularity level, we further observed the relationship between SLOC and CFP among three modules of one of the case projects from the previous case study.

Finally, we conducted another case study to explore the nature of the relationship between SLOC and CFP as well as Bytes of Code and CFP among the components of a project. In the following sub-sections we discuss each of the empirical studies and the results we obtained.

A. Level-1: Projects from a Benchmarking Dataset

ISBSG 2007 Repository, CD Release 10 [39] contains data from 4,106 completed projects collected from the software organizations all over the world.

Before making any analysis, we first filtered the data in the ISBSG repository to obtain the projects which reported the sizes in CFP, IFPUG FP and SLOC - excluding therefore those where these information are missing. We used the data of the projects, which have high Quality Data Rating and Function Point Rating (see Table 1).

ISBSG rating code of A, B, C or D applied to the Data Quality and Function Point Count data by the ISBSG quality reviewers. Data Quality Rating 'C' is given to the projects for which it was not possible to assess the integrity of the submitted data due to significant data not being provided. Data Quality Rating 'D' is given to the projects to which little credibility should be given to the submitted data due to one factor or a combination of factors.

As for the Function Point Rating 'D', this is given to the project data to which little credibility should be given to the unadjusted function point data due to one factor or a combination of factors.

TABLE 1. FILTRATION OF ISBSG DATASET 2007 RELEASE 10

Attribute	Filtering Value
Count Approach	= 'COSMIC', 'IFPUG'
Data Quality Rating	= {A | B}
Function Point Rating	= {A | B}

After the filtration process by selecting the projects measured by COSMIC, 29 projects remained whose SLOC values are reported. Since, the 'Primary Programming Language" is one of the requirements of the backfiring technique [30], we did not include the projects in our analysis for which the language type is not reported.

After the filtration, 14 projects remained; with all having a primary language type as C++. The data quality rating of all of these projects are 'B'. The application types are all 'Customization to a Product Data Management System' and they are all enhancement projects.

For this sub-dataset, the median, minimum and maximum SLOC values are 509, 96 and 2261, respectively. In Table 2, we show the ratios of SLOC/CFP for this sub-dataset.

TABLE 2. THE RATIOS OF SLOC/CFP

No of Projects	SLOC / CFP			
	Min	Med	Max	Std. Dev.
14	2.95	6.03	20.6	6.04

Then, we investigated the nature of the relationship between SLOC and CFP (see Fig.1).

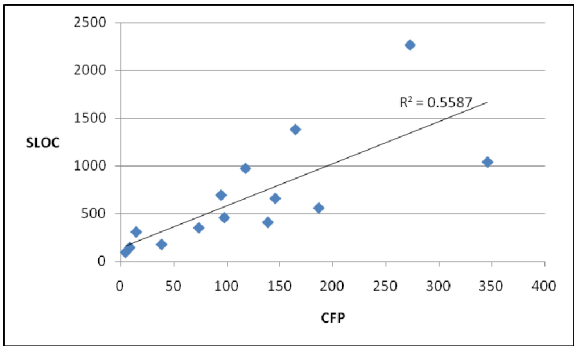

Figure 1. The Relationship between SLOC and CFP

R^2 was found to be 0.559 for this sub-dataset. Since R^2 is between 0.5 - 0.7, we concluded that this relationship should be used with caution although there is an adequate correlation for many purposes [40]. In fact, this is also reflected to SLOC/CFP ratios by great variation between the ratios (see Table 2).

Next, we selected the new development projects in the ISBSG dataset, which were measured by IFPUG FP and for which SLOC values reported (see Table 3).

TABLE 3. THE RATIOS OF SLOC/IFPUG FP

Progr. Lang.	# of Prj	SLOC / IFPUG FP			
		Min	Med	Max	Std. Dev.
C	14	6.7	66	500	125.1
Visual Basic	14	10.4	54.9	149.6	48.0
SQL	9	78.6	226.0	329.8	81.0

The SLOC per IFPUG FP values of the projects are given for three of the sub-datasets formed by grouping the projects with respect to the primary programming languages. This table does not include the sub-datasets which have less than 9 data points.

During the analyses, three projects from the first sub-dataset (C); three projects from the second sub-dataset (Visual Basic) and one project from the third sub-dataset (SQL) were excluded as being outliers. They all have very low functional size values whereas very high SLOC values. The development efforts for these projects are also very high. The reason for these differences from the other projects might be due to high amounts of data-manipulation rich requirements which cannot be measured by IFPUG FP. Unfortunately, the application types are not reported for most of these projects. Therefore, we could not conclude for a definite reason.

The relationships between IFPUG FP and SLOC for three sub-datasets are shown in Fig.2, Fig.3 and Fig.4, respectively.

For sub-dataset (C), sub-dataset (Visual Basic) and sub-dataset (SQL), the R^2 values were found as 0.26, 0.66 and 0.61, respectively.

For the first sub-dataset, the correlation is very weak. This is also observed in the very high standard deviation figure (125.1) in SLOC/IFPUG FP ratios in this subset where the median values is 66 (see Table 3). For the other two subsets, we concluded that these relationships should be used with caution since the correlation coefficients are not very high. Moreover, the standard deviations are significantly high; 48 for the second sub-dataset and 81 for the last one.

It is not known whether the SLOC values for the projects in the ISBSG dataset are the count of logical or physical SLOC, uncommented or commented SLOC, which can affect both the correlation coefficient and the variation between the ratios significantly. Therefore, we made further case studies for which we have more detailed information regarding the projects as well as the development organization to be able to understand what influences the nature of the relationships.

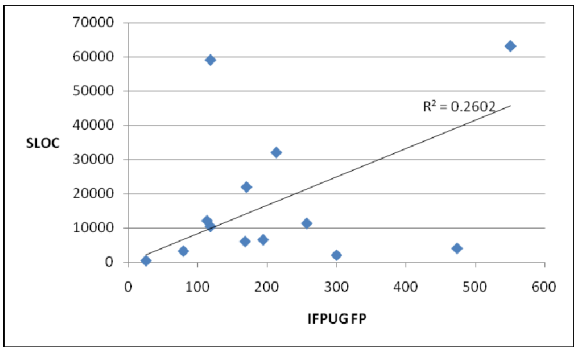

Figure 2. The Relationship between SLOC and IFPUG FP for C projects

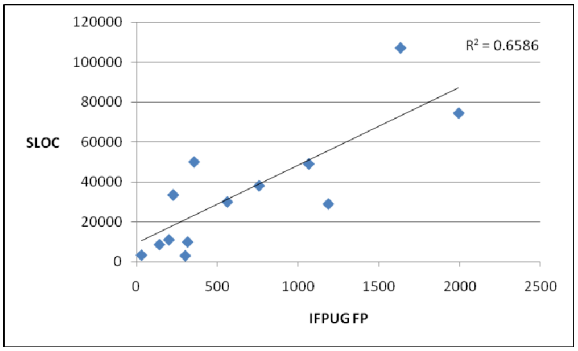

Figure 3. The Relationship between SLOC and IFPUG FP for Visual Basic projects

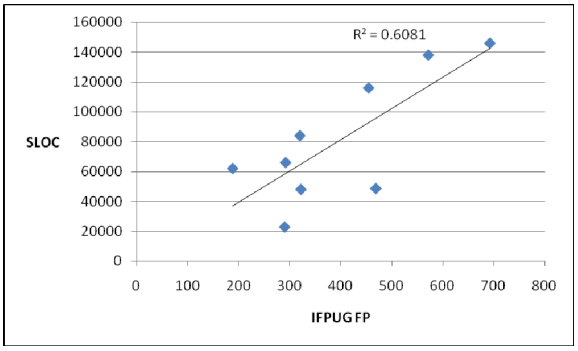

Figure 4. The Relationship between SLOC and IFPUG FP for SQL projects

B. *Level 2: Projects from a Software Organization*

We conducted a multiple case study using two projects data to further investigate the relationship between SLOC and CFP for the projects completed by a specific organization.

Project-1 is one of the subsystems of an avionics managements system for small to medium size commercial aircrafts on a Flight Display System. It is certified by Federal Aviation Administration.

Project-2 is a Collision Avoidance Subsystem of the Traffic Alert and Collision Avoidance System (TCAS). In this study, we measured the size of CAS-Own Aircraft Algorithm. Own Aircraft function determines the TCAS operational mode, effective sensitivity level and other operation parameters used by the collision avoidance logic.

According to CHAR Method [15], the functional domains of Project-1 and Project-2 are 'Complex Data Driven Control System' and 'Complex Control System', respectively.

We obtained the uncommented logical SLOC values for Project-1 and Project-2 by using "Understand for C++" which is a source code analyzer [41]. The ratios of SLOC/CFP values for the case projects are given in Table 4.

Both Project-1 and Project-2 are developed within the same organization by the project teams of very similar characteristics using the same methodology. However, for the first project, SLOC/CFP ratio is about 15 times greater than the second. One of the reasons might be attributed to the fact that first project involves high numbers of algorithmic operations which cannot be measured by COSMIC. Therefore, the size of this project obtained in CFP is smaller with respect to SLOC which measures all types of processes.

TABLE 4. THE RATIOS OF SLOC/CFP

Prj. No	Functional Size (CFP)	SLOC	SLOC /CFP
1	4,036.0	19,506	4.83
2	945.0	289	0.31

C. *Level-3: Sub-systems of a Project*

In this study, we further investigated the relationship between CFP and SLOC for the sub-systems of Project-1 discussed in the previous section (see Table 5).

TABLE 5. THE RATIOS OF SLOC/CFP FOR THE SUB-SYSTEMS OF PRJ-1

Prj. No	Subsy. Name	Functional Size (CFP)	SLOC (Logical)	SLOC/CFP
1	A	3,505.0	12,143	3.46
	B	279.0	3,449	12.36
	C	252.0	3,914	15.53
	Total	4,036.0	19,506	4.83

All subsystems of Project-1 are developed by the same people and all the sub-systems are of the same application type. However, the values of SLOC per CFP still have a wide degree of variation for sub-system A and the other sub-systems. For the sub-systems B and C, which involve very similar types of functionalities, the variation is not that significant.

Therefore, in this case study we concluded that for the same amounts of functionality (in CFP), similar types of functionalities might require similar amounts of code to be written. In order to further investigate this hypothesis, we conducted the next case study.

D. Level-4: Components of Modules

We performed this case study to further observe the relationship between the functional size and code size at a lower granularity level; i.e. for the components of modules.

In this case study we investigated two types of software components used in vehicles from the automotive company General Motors (GM). For our experiment we had both the specifications of the components in the form of UML component diagrams and the source code of the components. For each UML component, we measured the functional size in CFP and obtained the software size in SLOC and bytes by using the tool "Understand for C++" and the C compiler "Green Hills Optimizing C Compiler" [42].

In our previous case studies [37][4], we observed a very strong relationship between CFP and bytes for software components. In this study we specifically investigated whether we also have strong relationships between CFP and SLOC for the same components.

We chose distributable type components which cannot be split into smaller components, and therefore have similar granularity levels. The reason for considering similar granularity level was to have a one to one mapping between the distributable components and the Generic Software Model, which is the basis of COSMIC measurement. And yet another reason was that for each of the experiments, we ensured that components involve similar functionalities.

We first show our previous results of CFP and bytes of code relationship and then show our new results for the nature of the relationship between CFP and SLOC.

In our previous study [4] we used 12 software components of Display & Indication type. They were randomly picked from a set of 60 components. They typically perform small calculations and display information of vehicle data such as vehicle speed, engine speed, etc. This type of functionality is representative of at least 20-25% of the features in a typical vehicle today.

All the components were developed by the same team using the same methods and tools. We removed one of them being an outlier; extremely large CFP with respect to others. For these components, the median Bytes of code value is 1122, with a minimum 388 and a maximum 2182. In Table 6, the ratios of Bytes/CFP are given.

TABLE 6. THE RATIOS OF BYTES/CFP

No of components	Bytes / CFP			
	Min	Med	Max	Std. Dev.
11	72.7	83.1	97.0	7.6

Fig.5 shows the relationship between Bytes of code and CFP.

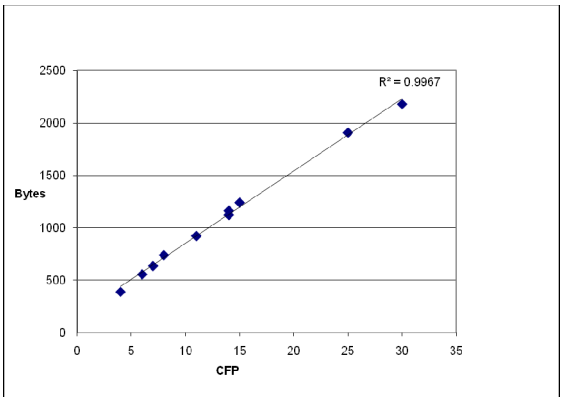

Figure 5. The Relationship between CFP and Bytes

R^2 is found to be 0.99 which shows a very high correlation between CFP and Bytes of code for the components of type Display & Indication.

Our second experiment [37] was a replication of the first one using a different team and a different component type. Here we used the software components of the type Comfort & Convenience. We randomly picked 15 components out of a set of 100 components.

They are characterized by a combination of event-based user inputs causing changes of one or several digital or analogue output(s), and represent at least 25-35% of the features in a typical vehicle today.

For this second set of components, the median Bytes value is 2202, with a minimum 932 and a maximum 4530. In Table 7, the ratios of Bytes/CFP are given.

TABLE 7. THE RATIOS OF BYTES/CFP

No of components	Bytes / CFP			
	Min	Med	Max	Std. Dev.
15	162.0	178.9	233.0	21.6

Fig.6 shows similar results for the relationship between bytes and CFP as in our first experiment. In this case the R^2 value is found as 0.992.

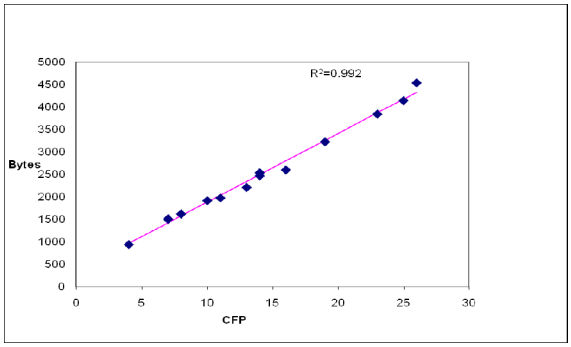

Figure 6. The Relationship between CFP and Bytes

Then, we investigated the relationship between CFP and SLOC for the same two types of components. The median,

minimum and maximum SLOC values for these components are 216, 115 and 832, respectively. In Table 8, the ratios of SLOC/CFP for the components of type Display & Indication are given.

TABLE 8. THE RATIOS OF SLOC/CFP

No of components	SLOC / CFP			
	Min	Med	Max	Std. Dev.
11	6.8	25.7	36.9	8.2

Fig.7 shows the relationship between SLOC and CFP. R^2 value was found to be 0.4855 which shows a weak correlation.

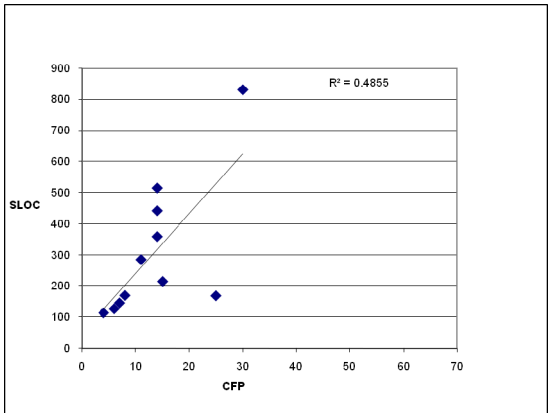

Figure 7. The Relationship between CFP and SLOC

Then, we looked at the ratios CFP/SLOC for the components of type Comfort & Convenience. The median, minimum and maximum SLOC values for these components are 339, 160 and 1061, respectively. In Table 9, the ratios of SLOC/CFP are given.

TABLE 9. THE RATIOS OF SLOC/CFP

No of components	SLOC / CFP			
	Min	Med	Max	Std. Dev.
15	9.2	26.1	65.8	15.4

Fig.8 shows the relationship between SLOC and CFP.

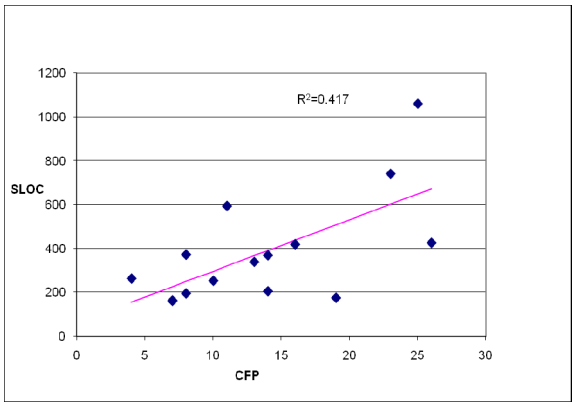

Figure 8. The Relationship between CFP and SLOC

We found R^2 as 0.417 which shows a weak relationship between SLOC and CFP.

E. Discussion of the Results

In this study, we first investigated the relationship between the functional size (CFP and IFPUG FP) and the code size (SLOC) by means of empirical studies.

The results of our empirical studies showed that there is a wide variation between the values of these ratios, especially at the higher granularity levels such as at benchmarking dataset or organization levels. At the project-subsystem or component levels, the variations did not show significant improvement and we still observed weak correlation and significant variation for the SLOC/CFP ratios.

One major reason for the weak relationship might be the fact that the assumption of linear relationship between these two measures is inherently inaccurate. Moreover, the correlation measures only the quality of linear relationships. If the data have a strong quadratic or exponential relationship, the correlation may not be a good measure of the strength of this relationship.

Another reason may be related to the functional domains of the projects and the applicability of the FSM method to measure their size. The existence of projects which involve functionalities that cannot be measured by the applied FSM method may even result in lower functional size values corresponding to higher SLOC values when comparing the projects.

For example, FSM methods are not able to measure the functional sizes of algorithmic operations and manipulations and result in smaller sizes in terms of FP with respect to SLOC. Although we minimized this effect by choosing the components which have very similar functionalities, still even at a component level, there still exist variations in the SLOC/CFP ratio. Therefore this cannot be attributed to this reason in these cases.

One another possible reason might be due to different viewpoints. In FSM methods, the amount of functionality is measured at a fixed level of abstraction, i.e. at the transaction level, which is defined from the user's point of view. The sizes of each of the transactions are measured separately in FP and then they are summed up to obtain the total size figure. SLOC, on the other hand, represents the size from the designer's point of view. The amounts of functionality in very similar transactions are measured separately in FP, although the corresponding SLOC might not reflect the change in the functionality at the same rate.

In this study, we also investigated the relationship between the functional size (CFP) and the code size (Bytes of Code) by means of empirical studies.

We observed that the degree of variation for the ratios Bytes of code per FP is significantly smaller than the SLOC per FP. Moreover, the R^2 values are surprisingly high. One of the reasons why bytes of code better correlates to CFP might be due to how SLOC and Bytes of code are

measured. We obtained SLOC values by using a tool such as Understand [41] for this case whereas we obtained Bytes of code using a C compiler. The tool depends on the user to point out exactly the parts of the library files that are needed, and reports a SLOC result even if there is any code missing. Thus, SLOC values might contain too much or too little code. The compiler on the other hand always does a uniform job, linking in all code used by the components. In addition, compiler optimization will most likely remove some of the different programming styles.

Another significant reason might be the fact that the Bytes of code obtained from the compiler relate to CFP better since CFP also measures the amount of functionality independent of how the functionality is implemented. However, since we obtained these results only for a limited amount of components, further research is required so that this conclusion can be generalized. Moreover, how these two measures relate when measured at the module or project level should also be investigated.

The empirical studies discussed in this paper involve a couple of validity threats. First of all, the case studies are conducted in different contexts due to availability of the data. At a higher level, we used the ISBSG dataset which involves projects data from all over the world. Although we tried to make the analysis on a homogenous dataset as much as possible, still the projects are collected from different organizations. Therefore, it is better to conduct similar types of empirical studies using similar projects data from one organization and which are developed by the same programming language type.

IV. CONCLUSION

Traditionally, it is assumed that the ratios of Code size (in SLOC, Bytes of Code, etc.) to Functional Size (in IFPUG FP or CFP, etc.) can be used to predict the code size at an earlier time in the software life cycle when we are able to measure the functional size of the software.

This paper investigated whether it is reliable to use such ratios when estimating the code size for different purposes. The results of our empirical studies showed that there is a wide variation between the values of the SLOC/IFPUG FP and SLOC/CFP ratios. The variation increases as we measure at higher granularity levels such as at benchmarking dataset or organizational levels.

Most of the software organizations which use parametric effort estimation models use SLOC/FP ratios to predict the code size and then use the value as an input to cost estimation models. However, the results of this study showed that even obtained at the component level, using these ratios can cause significant amount of error. Therefore, we conclude that software organizations should not continue using these ratios unless their local studies show acceptable results.

On the other hand, we observed a very strong relationship between CFP and Bytes of Code at the component level. We believe that it is promising to conduct further research on the nature of this relationship.

REFERENCES

[1] L.M. Laird, M.C. Brennan, Software Measurement and Estimation: A Practical Approach, John Wiley and Sons, Inc., Hoboken, New Jersey, 2006.

[2] N.E. Fenton, S.L. Pfleeger, Software Metrics: A Rigorous & Practical Approach, PWS Publishing Company, 1997.

[3] R. Park, "Software Size Measurement: A Framework for Counting Source Statements", Technical Report CMU/SEI-92-TR-020.

[4] K. Lind, and R. Heldal, "Estimation of Real-Time System Software Size using Function Points", Proc. of the Nordic Workshop on Model Driven Engineering (NW-MoDE), 2008.

[5] A.J. Albrecht, "Measuring Application Development Productivity", in Proceedings IBM Applications Development Symposium, Monterey, California, October 14-17, 1979.

[6] A.J. Albrecht, and J.E. Gaffney, "Software Function, Source Lines of Code, and Development Effort Prediction: A Software Science Validation", IEEE Transactions on Software Engineering, vol. SE-9, no. 6, November 1983.

[7] C. Symons, "Come Back Function Point Analysis (Modernized) – All is Forgiven!)", Proc. of the 4th European Conference on Software Measurement and ICT Control, FESMA-DASMA 2001, Germany, 2001, pp. 413-426.

[8] C. Gencel, and O. Demirors, "Functional Size Measurement Revisited", ACM Transactions on Software Engineering and Methodology (TOSEM), Vol.17, No.3, 2008, 71-106.

[9] ISO/IEC 14143-1: Information Technology -- Software Measurement -- Functional Size Measurement -- Part 1: Definition of Concepts, February 2007.

[10] ISO/IEC 14143-1:1998 Information Technology - Software Measurement - Functional Size Measurement - Part 1: Definition of Concepts, 1998.

[11] ISO/IEC 14143-2:2002: Information Technology - Software Measurement - Functional Size Measurement - Part 2: Conformity Evaluation of Software Size Measurement Methods to ISO/IEC 14143-1:1998, 2002.

[12] ISO/IEC FCD 14143-6:2005: Guide for the Use of ISO/IEC 14143 and related International Standards, 2005.

[13] ISO/IEC TR 14143-3:2003: Information Technology-Software Measurement - Functional Size Measurement - Part 3: Verification of Functional Size Measurement Methods, 2003.

[14] ISO/IEC TR 14143-4:2002: Information Technology-Software Measurement - Functional Size Measurement - Part 4: Reference Model, 2002.

[15] ISO/IEC TR 14143-5:2004: Information Technology-Software Measurement - Functional Size Measurement - Part 5: Determination of Functional Domains for Use with Functional Size Measurement, 2004.

[16] COSMIC: The Common Software Measurement International Consortium FFP, version 3.0, Measurement Manual, 2007.

[17] ISO/IEC 19761:2003: COSMIC Full Function Points Measurement Manual v. 2.2, 2003.

[18] IFPUG, Function Point CPM, Release. 4.1, IFPUG, Westerville, OH, 1999.
[19] ISO/IEC 20926, Software engineering - IFPUG 4.1 Unadjusted FSM Method - Counting Practices Manual, 2003.
[20] MkII FPA Counting Practices Manual Version 1.3.1, UKSMA: United Kingdom Software Metrics Association, 1998
[21] ISO/IEC 20968, Software engineering - Mk II Function Point Analysis - Counting Practices Manual, 2002
[22] NESMA, Definitions and Counting Guidelines for the Application of Function Point Analysis, Version 2.0, NESMA, 1997.
[23] ISO/IEC 24570:2005: Software engineering - NESMA FSM Method v.2.1 - Definitions and counting guidelines for the application of Function Point Analysis, 2005.
[24] P. Forselius, Finnish Software Measurement Association (FiSMA), FSM Working Group: FiSMA Functional Size, 2004.
[25] ISO/IEC 29881:2008, Software Engineering -- FiSMA functional size measurement method version 1.1, International Organization for Standardization, 2008.
[26] B.W. Boehm, E. Horowitz, R. Madachy, D. Reifer, K.C. Bradford, B. Steece, A. Brown, S. Chulani, C. Abts,, Software Cost Estimation with COCOMO II. Prentice Hall, New Jersey 2000.
[27] L.H, Putnam, "A general empirical solution to the macro software sizing and estimating problem", IEEE Transactions on Software Engineering, July 1978, pp. 345-361.
[28] R. Tausworthe, "Deep Space Network Software Cost Estimation Model", Jet Propulsion Laboratory Publication 81-7, 1981.
[29] R.E., Park, "PRICE S: The calculation within and why", Proceedings of ISPA Tenth Annual Conference, Brighton, England, July 1988.
[30] C. Jones, "Backfiring: converting lines of code to function points", Computer, Vol. 28, Issue: 11, 1995, 87-88
[31] L. Santillo, "Error Propagation in Software Measurement and Estimation", in IWSM/Metrikon 2006 conference proceedings, Potsdam, Berlin, Germany, 2-3 November 2006.
[32] R. Neumann, and L. Santillo, "Experiences with the usage of COCOMOII". In Proc. of Software Measurement European Forum 2006, 2006, 269-280.
[33] G.S. Henderson, "The Application of Function Points to Predict Source Lines of Code for Software Development", An MSc Thesis submitted to Air Force Inst. of Tech., Wright-Patterson AFB, OH, Report Number: AD-A258447, AFIT/GCA/LSY/92S-4, 1992.
[34] J.M. Desharnais, J.M., and A. Abran, "Approximation Techniques for Measuring Function Points", In Proc. of the 13th Inter. Workshop on Software Measurement (IWSM 2003), 23-25 Sept. 2003, Montréal, Canada, Springer-Verlag, 2003, 270-286.
[35] T. Rollo, "Functional Size measurement and COCOMO – A synergistic Approach". In Proc. of Software Measurement European Forum 2006, 2006, 259-267.
[36] C. Dekkers, and I. Gunter, "Using Backfiring to Accurately Size Software: More Wishful Thinking Than Science?", IT Metrics Strategies, Vol. VI, No.11, 2000, 1-8.
[37] K. Lind, and R. Heldal, "Estimation of Real-Time Software Code Size using COSMIC FSM", Proc. of the IEEE Intl. Symposium on Object/component/service-oriented Real-time distributed Computing (ISORC 2009), March 2009.
[38] K. Lind, and R. Heldal, "Estimation of Real-Time Software Component Size", Nordic Journal of Computing (NJC), 2009.
[39] ISBSG (International Software Benchmarking Standard Group) Dataset, ISBSG Dataset 10, 2007, http://www.isbsg.org
[40] K. Maxwell, Applied Statistics for Software Managers, Prentice Hall, 2002, ISBN 0130417890
[41] Understand for C++, http://www.scitools.com/ucpp.html
[42] Green Hills Optimizing C Compiler, http://www.ghs.com