Journal of Zhejiang University-SCIENCE C (Computers & Electronics)
ISSN 1869-1951 (Print); ISSN 1869-196X (Online)
www.zju.edu.cn/jzus; www.springerlink.com
E-mail: jzus@zju.edu.cn

An experimental study on the conversion between IFPUG and UCP functional size measurement units#

Juan J. CUADRADO-GALLEGO[1,2], Alain ABRAN[1], Pablo RODRÍGUEZ-SORIA[†‡2], Miguel A. LARA[2]

([1]*École de Technologie Supérieure - ETS 1100 Notre-Dame Ouest, Montréal QC H3C 1K3, Canada*)

([2]*Computer Science Department, University of Alcala, Madrid 28805, Spain*)

[†]E-mail: pablo.rsoria@uah.es; pablo_roso@hotmail.com

Received Apr. 20, 2013; Revision accepted Dec. 11, 2013; Crosschecked Feb. 19, 2014

Abstract: The use of functional size measurement (FSM) methods in software development organizations is growing during the years. Also, object oriented (OO) techniques have become quite a standard to design the software and, in particular, Use Cases is one of the most used techniques to specify functional requirements. Main FSM methods do not include specific rules to measure the software functionality from its Use Cases analysis. To deal with this issue some other methods like Kramer's functional measurement method have been developed. Therefore, one of the main issues for those organizations willing to use OO functional measurement method in order to facilitate the use cases count procedure is how to convert their portfolio functional size from the previously adopted FSM method towards the new method. The objective of this research is to find a statistical relationship for converting the software functional size units measured by the International Function Point Users Group (IFPUG) function point analysis (FPA) method into Kramer-Smith's use cases points (UCP) method and vice versa. Methodologies for a correct data gathering are proposed and results obtained are analyzed to draw the linear and non-linear equations for this correlation. Finally, a conversion factor and corresponding conversion intervals are given to establish the statistical relationship.

Key words: Software engineering, Requirements analysis, Functional size measurement, Use cases analysis, Object oriented, Function point analysis, Use cases points

doi:10.1631/jzus.C1300102 **Document code:** A **CLC number:** TP311

1 Introduction

There have been many attempts to measure the software size based on the amount of functionalities laid out on functional user requirements (FUR). Albrecht (1979) defined the measurement unit function points and function point analysis (FPA) method to measure the size of software product based on the amount of functionalities to be provided to the users.

After its introduction, FPA-like methods evolved quite a bit. Many variations and improvements on the original idea were suggested (Symons, 2001; Efe *et al.*, 2006); some of them have proved to be milestones in the development of functional size measurement (FSM). Also, with the success of object oriented (OO) software development methodologies (Svetinovic *et al.*, 2007), some methods were used to extend the functional domain of application to OO.

All these methods have their own definition models, measurement processes, and units of measure. Therefore, well established conversion formulas are required to convert functional sizes in different units of measure to one another. The need for conversion might arise due to a number of reasons. To be able to compare the functional sizes of software products measured by different methods, software organizations require their conversion to a common

‡ Corresponding author

Electronic supplementary materials: The online version of this article (http://dx.doi.org/10.1631/jzus.C1300102) contains supplementary materials, which are available to authorized users

format. Or, whenever software organizations need to change the size measurement method they use, they need to convert their past measurement data to the new unit of measure. Additionally, to use the benchmark datasets for estimation, the size measurement data using different methods will be converted.

In this paper, we discuss the results of an empirical study to establish a statistical relationship for conversion between International Function Point Users Group (IFPUG) FPA and Kramer-Smith's use case points (UCP) methods.

2 Background

In this section, we first briefly discuss the FSM methods approved by the International Organization for Standardization (ISO). Then we summarize the literature on the FPA-like methods developed specifically to measure the software developed using OO methodology. Finally, we discuss the related work on conversion.

2.1 Main FSM methods

In 1996, ISO established the common principles of FSM methods and published ISO/IEC 14143 standard family in order to promote the consistent interpretation of FSM principles.

Among those methods, five ones have been recognized as the ISO 'de jure' standards: (1) IFPUG, ISO/IEC 20926:2009 (ISO/IEC, 2009); (2) NESMA, ISO/IEC 24570:2005 (ISO/IEC, 2005); (3) Mk II, ISO/IEC 20968:2002 (ISO/IEC, 2002); (4) COSMIC, ISO/IEC 19761:2011 (ISO/IEC, 2011); (5) FiSMA, ISO/IEC 29881:2010 (ISO/IEC, 2010).

These methods have evolved significantly over the years (Fig. 1). In 1986, the IFPUG was established to promote the use of Albrecht's original method and to control the evolution of the measurement standard definition. This changed the name from Albrecht's function points to IFPUG FPA. Since then, several versions of IFPUG FPA have been published (IFPUG, 1990; 1994; 1999; 2000; 2004; 2010). IFPUG FPA enjoys widespread popularity and large publicly available data sets for those who wish to train their own company-specific IFPUG model or to compare their measurements with others. It was accepted by ISO as an international standard in 2009 (ISO/IEC, 2009).

Mk II FPA was developed by Symons (1988) to improve the original FPA method. This method brought some suggestions to reflect the internal complexity of a system. Currently, the Metrics Practices Committee (MPC) of the UK Software Metrics Association (UKSMA) is the design authority of the method. It was also designed mainly to measure business information systems. Mk II FPA has been

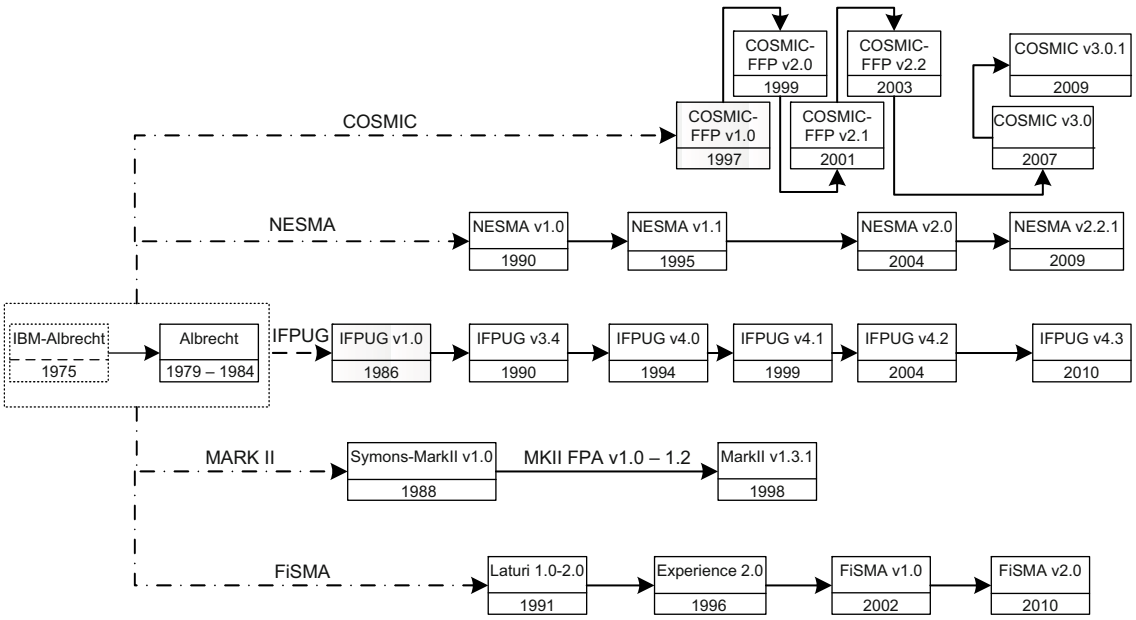

Fig. 1 Evolution of functional size measurement (FSM) methods

accepted as being conformant to ISO/IEC 14143 and became an international ISO standard in 2002 (ISO/IEC, 2002).

NESMA FPA was introduced by the Netherlands Software Metrics Association (NESMA). Since then, five versions have been published (NESMA, 1990; 1995; 2004; 2005; 2009). This method has the same rules as the IFPUG FPA method. However, the NESMA measurement manual provides different guidelines, hints, and examples. It was accepted by ISO as an international standard in 2005 (ISO/IEC, 2005).

The Common Software Measurement International Consortium (COSMIC) introduced COSMIC-FFP (COSMIC, 1997); it has been defined as a 2nd generation FSM method as a result of a series of innovations, such as: a better fit with both real-time and manager information system (MIS) environments, identification and measurement of multiple software layers, different viewpoints from which the software can be observed and measured, and the absence of a weighting system. Since its first publication, the interest developed in both the academic community and industry by the new unit has been enormous, reaching vast diffusion and utilization with three new versions published in a very short time (COSMIC, 1999; 2001; 2003) and two more later (COSMIC, 2007; 2009). It was approved by ISO as an international standard in 2003 as ISO 19761 (ISO/IEC, 2003) and updated in 2011 (ISO/IEC, 2011).

FiSMA FSM was developed by a working group of the Finnish Software Metrics Association (FiSMA). It is a general parameterized size measurement method designed to be applied to all types of software. The difference between FiSMA FSM and other methods is that it is service-oriented rather than process-oriented. It was recently accepted as an international FSM standard in 2010 (ISO/IEC, 2010).

2.2 FSM methods for object-oriented systems

The five methods described above can be applied to any software application, no matter which programming paradigm is used. Even if they have been deployed in a period when structured programming was important, FSM methods have evolved over time adapting their counting rules to the novelties and technology changes, as the OO paradigm.

To better reflect the needs of OO software development methodologies, several approaches have been proposed.

Abrahão *et al.* (2006) grouped these approaches into three categories. The first category involves the ones that are compliant with FPA, such as those in Lehne (1997), Caldiera *et al.* (1998), Fetcke *et al.* (1998), Uemura *et al.* (1999), Condori-Fernandez *et al.* (2004), and Marín *et al.* (2010). The result of the functional size measurement obtained through these approaches is the same as the result that would be obtained by directly applying FPA. The second category includes the ones that are not compliant with FPA, but take a view on functional size related to the FPA such as in Whitmire (1992), ASMA (1994), and Antoniol *et al.* (1998). A third category presented in Laranjeira (1990), Rains (1991), Banker *et al.* (1991; 1994), Zhao and Stockman (1995), Gupta and Gupta (1996), Minkiewicz (1999), Teologlou (1999), Kammelar (2000), and Zhang *et al.* (2009), takes a fundamentally different view on functional size by considering the class definition of an object as the main item that contributes to functional size.

In 1993, Gustav Karner of Objectory (now Rational Software) developed the use case points (UCP) method, as an extension of IFPUG and Mark-II FPA, adopting the same philosophy (Karner, 1993). The philosophy is that the functionality seen by the user is the basis for estimating the size of the software.

In OO analysis, as Ribu (2001) proposed, use case models describe the functional requirements of a future software system. Sizing the system can be done by measuring the size or complexity of the use cases in the use case model. The size can then serve as an input to a cost estimation method or model, in order to compute an early estimate of cost and effort.

OO analysis and design (OOAD) applies the Unified Modeling Language (UML) to model the future system, and the use cases to describe the functional requirements. The use case model serves as the early requirements specification, defining the size of the future product. The size may be translated into a number, which is used to compute the amount of effort needed to build the software.

An important requirement for applying Karner's method is that the declaration of a project's use cases (UC) to be developed must be realized with a proper level of details, in particular UC transactions, defined as a couple of steps where

any action performed by an actor implies a reply by the system, tracked as a system event (Anda *et al.*, 2001), because it is needed to count the number of events for each use case.

According to Robert Biddle's assumptions (Biddle *et al.*, 2002), use cases are the accepted best practice for capturing requirements for OO software development. This is especially true in the OO community where they originated and they are widely supported in modeling languages and in development processes.

Moving from Karner's method, other relevant proposals using use cases for measuring software functional size come from Fetcke *et al.* (1998), through the realization of a mapping between Jacobson's use case method and the IFPUG FPA model (Fetcke *et al.*, 1997). Such a method was formulated as a group of short rules compatible with the IFPUG method; it can be useful as a starting point for measurement using IFPUG FPA, because a large number of parameters must be evaluated by similar rules.

Another proposal for estimation using use cases was realized by IBM's John Smith (Smith, 1999). In such a proposal, use cases are used for estimating the total number of lines of code (LOC) that represent the base for the following measurement. In this work, Karner's UCP method is used. The method by Smith quantifies functional and non-functional system characteristics, through the following steps:

1. Classify the actors by complexity type:

Simple: external system interfaced through an application program interface (API) (weighting factor=1);

Average: external system that interacts using a text-based protocol as TCP/IP (weighting factor=2);

Complex: physical end user interfacing through a graphical user interface (GUI) or a website (weighting factor=3).

This classification returns the total number of unadjusted actor weights (UAW) as follows:

$$\mathrm{UAW} = \sum_{i=1}^{3} \mathrm{AT}_i \cdot \mathrm{WF}_i, \tag{1}$$

where AT_i is the number of type i actors and WF_i is the weighting factor of type i actors.

2. Classify use cases by complexity type:

Simple: three or less transactions (weighting factor=5);

Average: between four and seven transactions (weighting factor=10);

Complex: more than seven transactions (weighting factor=15).

This classification returns the total number of unadjusted use case weights (UUCW) as follows:

$$\mathrm{UUCW} = \sum_{i=1}^{3} \mathrm{UCT}_i \cdot \mathrm{WF}_i, \tag{2}$$

where UCT_i is the number of type i cases and WF_i is the weighting factor of type i cases.

Moving from the obtained results, it is possible to obtain the unadjusted use case points (UUCP) value, returned as follows:

$$\mathrm{UUCP} = \mathrm{UAW} + \mathrm{UUCW}. \tag{3}$$

3. Determine adjusted use case points (AUCP):

Using a list of technical and environmental factors (pertaining to the non-functional dimension), each one to be rated with a value in the 0–5 range, it is possible to obtain the technical complexity factor (TCF) and environmental complexity factor (ECF) by

$$\mathrm{TCF} = 0.6 + 0.01\,\mathrm{TFactor}, \tag{4}$$

$$\mathrm{ECF} = 1.4 - 0.03\,\mathrm{EFactor}, \tag{5}$$

where TFactor and EFactor are given by the summation of the product of each factor by the assigned weight, exactly as did in IFPUG FPA with the 14 general system characteristics (GSC) for obtaining the technical degree of influence (TDI). See Tables S2 and S3 in the supplementary materials for the list of complexity factors and related weights. Finally, it is possible to obtain the AUCP value:

$$\mathrm{AUCP} = \mathrm{UUCP} \cdot \mathrm{TCF} \cdot \mathrm{ECF}. \tag{6}$$

Much research on the relevance of UCP has been published in recent years. Such studies can be classified into two groups: (1) those ones studying Karner's method applying UCP to several software projects, which will be described here; (2) those ones comparing UCP with other measurement methods largely diffused and accepted by the international scientific community, which will be described in Section 2.3.

Referring to the first group, Nageswaran (2001) realized a detailed UCP study using several Web applications for a test, with the conclusion that estimation results can be improved and become more

affordable when using UCP, offering evident advantages against FPA-based extended methods.

Another experience came from Cohn (2005), who evaluated a project using UCP, trying to make the count automatic under a software tool. It was stated to be more precise in establishing an average time for implementing a use case, according to an established level of details for the writing.

Another valuable study was produced by Clemmons (2006), using a Web application developed by his team, stressing the value of applying UCP also on closed projects. Conclusions of this study led to a 20% error range in estimates against the actual project effort. Again, Clemmons (2006) affirmed that, different from other estimation and sizing methods, technical and environmental factors are properly weighted, with the possibility during time to adjust their values with local calibrations, with the aim of obtaining more precise measurements. As in the previously cited studies, Clemmons stressed the versatility, extensibility, ease of learning, and speed in applying the UCP method.

Nevertheless, some authors stressed the opposite and did not recommend the use of this counting method for several reasons:

1. Many variations of the use case style can make it difficult to measure the complexity of a use case (Smith, 1999).

2. Free textual descriptions may lead to ambiguous specifications (Arnold and Pedross, 1998).

3. Since there are a large number of interpretations of the use case concept, Symons (2001) concluded that one way to solve this problem was to view the Mk II logical transaction as a specific case of a use case, and that using this approach leads to requirements which are measurable and have a higher chance of unique interpretation.

4. The coauthor of this paper, Alain Abran, identified with his colleague J. Ouwerkerk some relevant aspects of UCP that make this method unreliable and with lack of consistency (Ouwerkerk and Abran, 2006).

5. The UCP method does not describe any means to ensure the consistency of granularity from one use case to another. The impact is the following: quantities as the outcome of UCP are not necessarily comparable and a poor basis for benchmarking and estimation.

6. The UCP measurement method measures several entities and attributes. The impact of combining these different concepts is that the end-result is of an unknown and unspecified entity type; that is, we do not know what has been measured.

7. The UCP measurement method is based on the constants and weights of Albrecht's function point value adjustment factors without supporting justification.

8. The attributes are evaluated by a measurer without criteria or a guide to interpretation—a 5 for one measurer could be a 2 for another. Therefore, repeatability and reproducibility could be very poor.

9. UCP method calculations are based on several algebraically inadmissible scale type transformations. It is unfortunate that this has not yet been challenged, by either UCP users or the designers of subsequent UCP variants.

However, the measurement of functional size using use cases (therefore, very early in the development life cycle) represents an interesting opportunity for an industry which is increasingly using the rational unified process (RUP) and agile use-case-oriented methodologies. The aim of this research is not to justify or recommend the use of UCP, but to propose a statistical relationship that allows the conversion of the software sizes measured by IFPUG FP and UCP methods, due to their widespread use in the industrial environment.

2.3 Related work on conversion

After describing the main software functional sizing methodologies and, in more depth, the ones to be used in this study (IFPUG and UCP), the next step will be to describe the issue of the conversion between such functional size units.

Nowadays, one of the most important issues to be solved in the field of software measurement is the conversion between different functional size units (FSU) from various FSM methods. This need can be verified, as previously stated, from many software development companies worldwide using IFPUG for a long time as their software measurement unit; as a result, there are large databases used suitably to plan future software projects. Although most of these companies are satisfied with using IFPUG and do not want to abandon it, others are actually interested in starting to use, for some product lines, other FSM-based measures. However, since they collected IFPUG measurement data in their databases, they

cannot use these data for estimation purposes for the software projects to be measured by UCP. To obtain confident results from those databases, many years of data collection are required, and the only realistic solution to encourage UCP usage is to find a statistical relationship for conversion between the existing measures obtained by IFPUG and UCP.

This work deals with this issue and wants to solve this problem through the use of a large number of trustable estimates using UCP and a statistical comparison with IFPUG FPA data.

Referring to the second group of studies as described in Section 2.2 (comparative studies), it is possible to obtain some experience. One notable study by Gencel *et al.* (2006) compared the effort estimation accuracy using a regression model based on UCP and a 'closed' parametric model as COCOMO. The estimation error produced using COCOMO was equal to −384%, while using a UCP-based regression model it was equal to 88%, stressing the complementarity among the methods and urging the realization of a larger number of comparative papers.

Damodaran and Washington (2002) produced a theoretical comparison between the two methods, concluding that the level of difficulty in classifying UC according to the categories proposed by Karner is low. Again, they considered UCP a method with the same level of trustworthiness as the IFPUG one, when it is used by projects where UC is produced. The most complete study right now on the IFPUG-UCP conversion was the one by Minkiewicz (2004), using 15 projects (the results obtained are listed in Table S6).

As in some of the aforementioned studies, this work also presents conversion derived on the minimum square method, where the independent proxy is given by the IFPUG FP value I and the dependent proxy is given by the UCP value U. Therefore,

$$U = a + bI. \tag{7}$$

The statistical analysis methodology applied was the one in Cuadrado-Gallego *et al.* (2010), related to the experimental study for conversion between IFPUG and COSMIC.

All the results from the aforementioned papers have been revised during the writing of this paper and new results are presented, obtained from Minkiewicz's work which brings a new testing field to be explored, theoretically feasible.

Minkiewicz (2004) discussed the UC software system requirements, making easier the communication between developers and end users, since they can be available before relevant decisions are taken on the architecture and implementation of the system. Furthermore, UC offers the possibility to size a software system at early stages of the software life cycle. A statistical measure for measuring the proper fit between UUCP and UFP was given by the determination coefficient:

$$R^2 = 0.6. \tag{8}$$

In Minkiewicz (2004), no conversion factors were presented. Assuming such results to be promising, our aim will be—through the use of new data sets from Minkiewicz (2004)—to determine such a ratio.

3 Data gathering

In this section four different aspects of data gathering are treated, each in a different subsection: The first one describes the main problems that the practitioners find when they try to obtain these kinds of data; Section 3.2 analyzes the data used in previous studies; in Section 3.3 the data gathering methodology is stated; and finally Section 3.4 describes the new databases specifically obtained in this research.

3.1 Data gathering problems

One of the main problems of this kind of research, if not the topmost, relies on data gathering, both in size and quality, enabling a trustable and rigorous analysis. The source of the problem comes from data gathering costs, since the fees of the specialists are usually high, up to 200 €/h in Europe. However, companies that have already selected a specific measurement unit for their projects cannot easily see the return on investment derived from analyzing all the measures on the same software once again with a different unit, considering the costs involved. As a consequence, data gathering in industrial environments is practically impossible.

To solve this problem, there could be two solutions: (1) the researchers, coming from other institutions, usually a university of research center, do the measurements with different units by themselves, using a set of applications provided by companies

under agreements subscribed to between both organizations; or (2) groups of students, once they finalize their courses on software measurement studying these units, are asked to do a final project performing the required measurements with the different units studied. In both cases, the cost of the measurements is reduced significantly, effectively making the data gathering process feasible.

A second problem is the quality of data obtained from the measurements. Two possible causes of poor quality data could be: (1) an incorrect procedure in performing measurements, due to inadequate training received by the measurers or an inadequate way of doing the work by the measurers (in that case, students) because of three main reasons, including (a) they do not work full time because they take several courses at the same time, (b) they are not getting paid, and (c) they work less seriously on course work than a software practitioner would do on a real piece of work; (2) the homogeneous building of a project data repository, due to the contribution of different measurers. About this second issue, no publication or work quantifying the impact of the study done by a measurer has been devoted to analysis of the quality of the obtained data, even if some results were achieved by a University of Alcalá (UAH) team that can help draw some preliminary conclusions, at least on the qualitative side, determining that such a kind of impact exists. In order to solve this second issue related to the quality of data, all the students were trained intensively in the methods used; the measurements were done on the dates between two semesters on which the students did not have to attend courses and they were asked to work on the measurements full time; the measurements done had a serious impact on the final marks of the students, making sure that the students took their work very seriously. The final solution requires the participation of expert measurers who in the first case will revise the measurements done by the students and in the second case will revise and make homogeneous the measurements previously obtained by other less experienced measurers.

3.2 Data used in previous studies

When realizing an analysis of the data gathering method used in Gencel *et al.* (2006) and Cuadrado-Gallego *et al.* (2010) and moving from the possible problems now introduced, solutions were as described above: use of a single, expert measurer coming from another organization, very often from a university, or university students without proper skill and experience with the evaluated methods.

It is important to stress that, even if the case from Gencel *et al.* (2006) seems to be different from others coming from an industrial experience where the organization provides data, the performed count was part of an internal research work.

About the heterogeneity of data, the assumption is that it can be considered of the same value as counting performed by a unique measurer with another one done by two largely experienced measurers working together.

3.3 Data gathering methodology

The data gathering procedure used in this study has been very different from the procedures used in previous studies. The way chosen to solve the aforementioned two problems was, for the aspect related to data gathering costs, the second available solution, and data was collected by a final assignment in a measurement course on IFPUG and UCP on real, industrial applications. This solution to the first problem led to a data quality issue in the second problem, both in correctness, since all measurers lack experience, and in heterogeneity, since every measure is obtained by a different, inexperienced measurer.

The way to approach these two problems was twofold: first, a rigorous selection of measurers among those students with more knowledge in the matter and better grades; second, by reviewing all works by an experienced measurer. Below are the steps followed:

1. Training in the UCP measurement process

The audience for such training were students from the Management Information System courses, within a 3-credit 3rd-year mandatory course entitled Laboratory on Software Planning & Management. The theoretical training was 24 hours long introducing IFPUG v4.1 and UCP methods, using eight hours (two hours per class) for each method. The students were subdivided into four small groups: the first two groups were introduced first to IFPUG and then to UCP methods, and the other two groups received the training in opposite order, to avoid any possible influence on the method learned before. Versions chosen were IFPUG CPM v4.1 and Karner-Smith's UCP method.

2. Selection of the students participating in the measurements

To participate in the measurement process, the following were required:

(1) To attend at least 90% of the classes (i.e., students could be absent only in one class).

(2) To have a grade better than 8/10 in a written test of knowledge performed in the 11th week of classes. This test was applied to all students, even if they did not participate in the measurements.

3. Measurement of a real world application with the UCP method

Each selected student received the specifications of a different real software application which was previously measured by the IFPUG FPA method and previously developed by UAH during the implementation of software in a collaborative project with external companies. The university owns or shares the copyrights for such projects.

All the projects documentation used came from finished projects, in which all the design models, done with UML, were at the measurer's disposal. For that reason, the level of granularity was very low, with specification at a high level of details, which allows application of the UCP measurement method with high precision. The use cases were all written in detail in UML with textual notes.

Each student should then proceed to measure the assigned application in UCP units.

4. Correction and standardization of the measurements performed by students

A revision of each counting was done by an expert measurer with UCP. Reviewing tasks consisted in verifying the eventual existence of errors:

(1) In the proper identification of system's actors and use cases.

(2) In the proper identification of transactions within use cases.

(3) In the proper calculation of UUCP.

(4) In the proper determination of the weighting factors, technical and environmental complexity and in the UCP calculation.

3.4 Data obtained

The steps described in the previous section were applied during one academic year with the following results:

1. Generation of the UCP sample U1 (results obtained are given in Table S4)

(1) Moving from 74 students, they were subdivided into 4 groups, 3 with 18 students each, and the last with 20 students.

(2) Sixty-three students out of 74 attended the course and 90% of them completed the exercise.

(3) Fifty-six students out of 63 passed the exam, with the following results: 16 with Fair, 23 with Good, 11 with Excellent (at least 8 points out of 10), and 6 with Super.

(4) As a result of this process, 17 students were selected to perform measurements. These 17 students were assigned 17 corresponding real software applications from UAH's own database, to be measured in UCP measurement units.

2. Generation of the IFPUG sample I1 (Table S5)

As 17 students were selected in the above casting process, 17 real software applications that have already been measured in IFPUG units were chosen to form the dataset. These 17 software applications correspond to the 17 items in sample U1.

3. Generation of the sample MK (Table S6)

In this sample, as said before, a 15-project database collected by Minkiewicz was used, with the final results measured in IFPUG and UCP measurement units. This sample was added to this research in order to improve the accuracy of the results obtained.

4. Generation of the testing sample I1-U1 (Table S7)

Combining samples I1 and U1 led to a third, new sample called I1-U1, matching IFPUG and UPC methods, since projects measured in I1 with the IFPUG method were the same as those used in the U1 sample. As a result, this sample is made of 17 projects evaluated properly with both IFPUG and UCP methods.

5. Generation of the testing sample U1-I1-MK (Table S8)

Combining samples MK and I1-U1, a fourth sample was obtained, labeled U1-I1-MK, joining values referred to by IFPUG and UCP methods.

With the aim of establishing, if possible, the merger of the two original data samples, both qualitative issues and the measurement data gathering process were taken into account, and these two samples were performed in a similar manner. In particular, during the last phase the expert measurer was employed, introducing a homogeneity element.

For quantitative issues, a statistical analysis verifying different hypotheses on the variance of both samples was run, to establish if they pertained to the same dataset.

In both cases, the sample size is quite small, and both $n_{\text{U1-I1}}$ and n_{MK} are lower than 25. Therefore, in order to analyze variance, assume the following hypotheses:

$$H_0 : \sigma^2_{\text{MK}} = \sigma^2_{\text{U1-I1}},$$
$$H_1 : \sigma^2_{\text{MK}} \neq \sigma^2_{\text{U1-I1}}, \tag{9}$$

where σ_i^2 is the variance for sample i. The measurement parameter is given by the ratio between the simple variance, which has a distribution in Fisher's coefficient (FC), with $n_{\text{U1-I1-1}}$ freedom degrees being the numerator and $n_{\text{MK-1}}$ the denominator:

$$d = \frac{S^2_{\text{U1-I1}}}{S^2_{\text{MK}}} = \frac{5139.3}{2108.7} = 2.4, \tag{10}$$

where the acceptance criterion is $d <$ FC. Therefore, the variance comparison will be as follows:

$$F_{14,16}(0.95) = 2.46 > 2.4, \tag{11}$$

and it is possible to obtain an acceptance level equal to 95% where the common data series from the two samples have the same variance. Therefore, 32 data points were obtained from the combination of 17 data points of U1 and I1 and 15 data points of MK.

4 Data analysis

4.1 Finding linear equations

The first step in the analysis of data gathered was about finding the linear equations to allow a conversion between IFPUG function points and UCP for each sample under the scope of the analysis. The coefficients of the regression line and intervals were calculated and summarized in Table 1.

1. Sample U1-I1

The linear equation obtained by applying the ordinary least square (OLS) method on the 17 data points is

$$\text{UCP} = -0.62 + 0.94\, I, \tag{12}$$

with $R^2 = 0.8$.

2. Sample MK

The linear equation obtained by applying the OLS method on the 15 data points is

$$\text{UCP} = 20.23 + 0.39\, I, \tag{13}$$

with $R^2 = 0.6$.

3. Sample U1-I1-MK

The linear equation obtained by applying the OLS method on the 32 data points is

$$\text{UCP} = -15.26 + 0.93\, I, \tag{14}$$

with $R^2 = 0.86$.

Table 1 Overall parameters

Sample	a	b	Interval(b)
U1-I1	−0.62	0.94	(0.53, 1.35)
MK	20.23	0.39	(0.11, 0.68)
U1-I1-MK	−15.26	0.93	(0.71, 1.14)

In the simple linear regression model there are three parameters that we must estimate: coefficients of the regression line, a and b, and the variance in the normal distribution, σ^2. There are different ways to obtain these estimators; we use the least square method.

Variance σ^2 is defined as the average of the quadratic differences of n ratings with regard to its arithmetic average:

$$\sigma^2 = \frac{1}{n}\sum_i (\hat{y}_i - \bar{y})^2. \tag{15}$$

Table 2 shows other statistical indicators for the data points analyzed at a confidence level of 95%.

Table 2 Statistical indicators for the linear equations

Sample	σ^2	Interval(a)	Interval(b)
U1-I1	5139.29	(−133.7, 132.37)	(0.53, 1.35)
MK	2108.67	(−18.97, 59.43)	(0.11, 0.68)
U1-I1-MK	17 824.38	(−70.14, 39.62)	(0.71, 1.14)

Fig. 2 shows the data plotted and the regression equation obtained.

4.2 Analysis of linear equations

A first analysis performed on the statistical indicators from the equations shows that they all fall within acceptable thresholds. If it is true that U1 data fall slightly outside their thresholds, distances are not so large to discard the results obtained.

Analyzing the equation parameters in Fig. 2 and Table 1, it is possible to observe that, even if they significantly differ in the independent variables, they show a marked tendency on the straight line with a

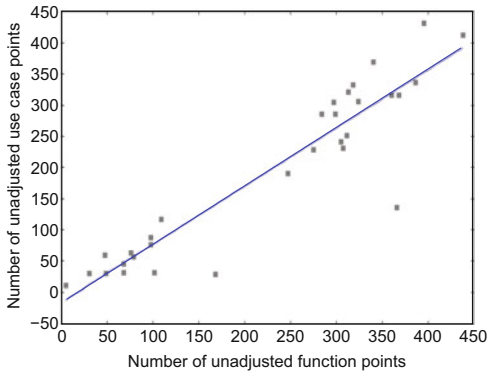

Fig. 2 Linear equation for sample U1-I1-MK

slope close to 1. Considering the three more likely intersection intervals, the (0.7, 0.8) interval is obtained. This is very significant because it can be noted as a positive trend of the straight line according to a higher number of projects measured by both methods.

Nevertheless, as can be appreciated, if we analyze data from the variance σ^2, lineal equations do not conform to the required model and therefore we move to seek an appropriate non-linear model.

4.3 Finding non-linear equations

Although there is no published reference related to the conversion between functional size units applying non-linear equations, two relevant reasons indicated that this study could lead to interesting results:

1. U1-I1 and U1-I1-MK data samples presented similarly sloped straight lines, with the slope quite close to 1. This means that a conversion factor between IFPUG FP and UCP would be equal to 1, when an independent variable is absent.

2. A basic issue in this finding for a conversion factor is the need for verifying the coordinates origins, since software of size 0 IFPUG FP (therefore with a null functional size) means a null UCP value. This means that the linear equations would be represented only by the independent variable. A further complication is introduced when such an independent variable is a negative value, which will generate a negative UCP size, conceptually absurd.

These two reasons were taken into account in this study, to analyze the use of non-linear equations in obtaining the conversion factor. The studied equations have therefore this form:

$$\mathrm{UCP} = aI^b. \tag{16}$$

To analyze the equations using the collected data, an initial transformation of data was done, performing an OLS adjustment for the obtained logarithmic equation:

$$\ln \mathrm{UCP} = \ln a + b \ln I. \tag{17}$$

The new analysis was conducted with non-linear equations not only for the sample data obtained within the framework of this investigation, but also for samples from previous works. Table 3 shows the results obtained after performing the reverse conversion of parameters.

Table 3 Parameters for non-linear equations

Sample	a	b	R^2
U1-I1	1.5	0.92	0.84
MK	3.86	0.59	0.63
U1-I1-MK	1.05	0.94	0.82

Table 4 shows the same statistical indicators calculated at a confidence level of 95%; in this case, we have introduced a new statistical variable to show the relationship between our selected datasets, i.e., residual variance or unexplained variance, S_{e}^2. This statistical value can be defined as the sum of the squares of the residuals:

$$S_{\mathrm{e}}^2 = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{a} - \hat{b}x_i)^2 = \frac{1}{N}\sum_{i=1}^{N} e_i^2. \tag{18}$$

If the residual variance is high (far from 0), residuals will be high and the dependency between values will be low; the adjustment will be bad. If the residual variance is low (close to 0), dependency between values will be strong and the adjustment will be correct.

Table 4 Statistical indicators for non-linear equations

Sample	σ^2	S_{e}^2	Interval(a)	Interval(b)
U1-I1	0.08	0.001	(0.24, 9.67)	(0.59, 1.24)
MK	0.49	0.18	(0.69, 21.85)	(0.19, 0.98)
U1-I1-MK	1.08	0.19	(0.28, 3.86)	(0.69, 1.20)

4.4 Analysis of non-linear equations

A first analysis performed on the statistical indicators from the three equations shows that they

all fall within acceptable thresholds. Analyzing the equation parameters, it can be observed that these samples present behavior of the exponent, toward a value close to 1. Also, it would lead to a positive result because an exponent with such characteristics makes easier an unambiguous conversion between two units of measure. It is also noteworthy that the tendency assumed by the value of the independent variable determines the proportionality of the conversion. Such a tendency is close to the unit, being very significant and interesting. Considering the intersection, the more likely interval for obtaining the exponent is (0.7, 1.2), which is quite good; it is possible to observe that the dispersion is narrowed with respect to what was observed for the independent term of the linear equations.

If what is said above is combined with the specific results of sample U1-I1-U2, which owns the largest amount of data, then it might be considered as the most significant one. Considering the apparent trend in the independent term and the exponent towards their compensation close to the unit, it will be possible to observe that these two variable units of measurement are quite aligned, and a first possible conversion would be suggested:

$$\mathrm{UCP} \cong I. \tag{19}$$

This means that one IFPUG FP would be approximately equal to one UCP, with a 25% error range.

5 Conclusions and prospects

The final planned goal in this research was the realization of a statistical comparative analysis for the convertibility between measurements done using the UCP and IFPUG methods. Therefore, the search for a statistical relationship for converting software functional size measurements was done with these units of measure. Thus, we started from the knowledge acquired in previous works on UCP, proposing its main characteristics, strengths, and weaknesses, and the proper manner to handle a software measurement with it.

It was concluded that the conversion factor between IFPUG FP and UCP is very close to a 1:1 ratio. It was not possible to affirm that it is an exact conversion. In some cases, for instance, the variation range could be close to 25%. Furthermore, conversion intervals were determined to draw some first-level conclusions useful for starting to use datasets containing both IFPUG and UCP functional size data, even if future research must be done.

As part of this research, we also faced the issue of obtaining data from these kinds of studies, proposing a repeatable procedure, discussed with reliable data from an academic environment. Furthermore, such procedures were tested throughout its execution twice.

Future work will be about: (1) further analysis on new, improved datasets, obtained with the double objective of verifying results achieved in this study and reducing the confidence interval for the conversion factor; (2) comparison of results achieved from this kind of investigation based on empirical data for the conversion between functional size measurement units and other units based on qualitative analysis.

References

Abrahão, S., Poels, G., Pastor, O., 2006. A functional size measurement method for object-oriented conceptual schemas: design and evaluation issues. *J. Softw. Syst. Model.*, **5**(1):48-71. [doi:10.1007/s10270-005-0098-x]

Albrecht, A.J., 1979. Measuring application development productivity. Proc. Joint SHARE, GUIDE, and IBM Application Development Symp., p.83-92.

Anda, B., Dreiem, H., Sjøberg, D.I.K., *et al.*, 2001. Estimating software development effort based on use cases—experiences from industry. *LNCS*, **2185**:487-502. [doi:10.1007/3-540-45441-1_35]

Antoniol, G., Calzolari, F., Cristoforetti, L., *et al.*, 1998. Adapting function points to object oriented information systems. Proc. 10th Conf. on Advanced Information Systems Engineering, p.59-76. [doi:10.1007/BFb0054219]

Arnold, M., Pedross, P., 1998. Software size measurement and productivity rating in a large-scale software development department. Proc. 20th Int. Conf. on Software Engineering, p.490-493. [doi:10.1109/ICSE.1998.671613]

ASMA, 1994. Sizing in Object-Oriented Environments. Australian Software Metrics Association (ASMA), Victoria, Australia.

Banker, R.D., Kauffman, R.J., Kumar, R., 1991. An empirical test of object-based output measurement metrics in a computer aided software engineering (CASE) environment. *J. Manag. Inform. Syst.*, **8**(3):127-150.

Banker, R.D., Kauffman, R.J., Wright, C., *et al.*, 1994. Automating output size and reuse metrics in a repository based computer aided software engineering (CASE) environment. *IEEE Trans. Softw. Eng.*, **20**(3):169-187. [doi:10.1109/32.268919]

Biddle, R., Noble, J., Tempero, E., 2002. Essential use cases and responsibility in object-oriented development. Proc. 25th Australasian Conf. on Computer Science, p.7-16.

Caldiera, G., Antoniol, G., Fiutem, R., *et al.*, 1998. Definition and experimental evaluation of function points for object-oriented systems. Proc. 5th Int. Software Metrics Symp., p.167-178. [doi:10.1109/METRIC.1998.731242]

Clemmons, R.K., 2006. Project estimation with use case points. *J. Def. Softw. Eng.*, **19**(i2):18-22.

Cohn, M., 2005. Estimating with use case points. *Methods & Tools*, **13**(3):3-13.

Condori-Fernandez, N., Abrahão, S., Pastor, O., 2004. Towards a functional size measure for object-oriented systems from requirements specifications. Proc. 4th Int. Conf. on Quality Software, p.94-101. [doi:10.1109/QSIC.2004.1357949]

COSMIC, 1997. COSMIC Measurement Manual version 1.0. Common Software Measurement International Consortium, Montreal, Canada.

COSMIC, 1999. COSMIC Measurement Manual version 2.0. Common Software Measurement International Consortium, Montreal, Canada.

COSMIC, 2001. COSMIC Measurement Manual version 2.1. Common Software Measurement International Consortium, Montreal, Canada.

COSMIC, 2003. COSMIC Measurement Manual version 2.2. Common Software Measurement International Consortium, Montreal, Canada.

COSMIC, 2007. COSMIC Measurement Manual version 3.0. Common Software Measurement International Consortium, Montreal, Canada.

COSMIC, 2009. COSMIC Measurement Manual version 3.0.1. Common Software Measurement International Consortium, Montreal, Canada.

Cuadrado-Gallego, J.J., Buglione, L., Domínguez-Alda, M.J., *et al.*, 2010. An experimental study on the conversion between IFPUG and COSMIC functional size measurement units. *Inform. Softw. Technol.*, **52**(3):347-357. [doi:10.1016/j.infsof.2009.12.001]

Damodaran, M., Washington, A.N.E., 2002. Estimation using use case points. Proc. ISECON.

Efe, P., Demirors, O., Gencel, C., 2006. Mapping concepts of functional size measurement methods. *In*: Dumke, R., Abran, A. (Eds.), COSMIC Function Points: Theory and Advanced Practices. CRC Press, USA.

Fetcke, T., Abran, A., Nguyen, T.H., 1997. Mapping the OO-Jacobson approach into function point analysis. Proc. Technology of Object-Oriented Languages and Systems, p.192-202.

Fetcke, T., Abran, A., Nguyen, T.H., 1998. Function point analysis for the OO-Jacobson method: a mapping approach. Proc. FESMA, p.403-410.

Gencel, C., Buglione, L., Demirors, O., *et al.*, 2006. A case study on the evaluation of COSMIC-FFP and use case points. Proc. 3rd Software Measurement European Forum, p.121-140.

Gupta, R., Gupta, S.K., 1996. Object Point Analysis. IFPUG Fall Conf.

IFPUG, 1990. Function Points Counting Practices Manual 3.0. International Function Point User Group, Westerville, Ohio.

IFPUG, 1994. Function Points Counting Practices Manual 4.0. International Function Point User Group, Westerville, Ohio.

IFPUG, 1999. Function Points Counting Practices Manual 4.1. International Function Point User Group, Westerville, Ohio.

IFPUG, 2000. Function Points Counting Practices Manual 4.1.1. International Function Point User Group, Westerville, Ohio.

IFPUG, 2004. Function Points Counting Practices Manual 4.2. International Function Point User Group, Westerville, Ohio.

IFPUG, 2010. Function Points Counting Practices Manual 4.3. International Function Point User Group, Westerville, Ohio.

ISO/IEC, 2002. ISO/IEC 20968: Software Engineering Mk II Function Point Analysis - Counting Practices Manual. International Standardization Organization, Geneva, Switzerland.

ISO/IEC, 2003. ISO/IEC 19761: Software Engineering COSMIC: a Functional Size Measurement Method. International Standardization Organization, Geneva, Switzerland.

ISO/IEC, 2005. ISO/IEC 24750: Software Engineering - NESMA Functional Size Measurement Method, Definitions and Counting Guidelines for the Application of Function Point Analysis. International Standardization Organization, Geneva, Switzerland.

ISO/IEC, 2009. ISO/IEC 20926: Software and Systems Engineering - Software Measurement - IFPUG Functional Size Measurement Method. International Organization for Standardization, Geneva, Switzerland.

ISO/IEC, 2010. ISO/IEC 29881: FiSMA 1.1 Functional Size Measurement Method. International Organization for Standardization, Geneva, Switzerland.

Kammelar, J., 2000. A sizing approach for OO-environments. Proc. 4th Int. ECOOP Workshop on Quantitative Approaches in Object-Oriented Software Engineering.

Karner, G., 1993. Resource Estimation for Objectory Projects. Objective Systems SF AB.

Laranjeira, L.A., 1990. Software size estimation of object-oriented systems. *IEEE Trans. Softw. Eng.*, **16**(5):510-522. [doi:10.1109/32.52774]

Lehne, A., 1997. Experience report: function points counting of object oriented analysis and design based on the OOram method. Proc. Conf. on Object-Oriented Programming Systems, Languages, and Applications.

Marín, B., Pastor, O., Abran, A., 2010. Towards an accurate functional size measurement procedure for conceptual models in an MDA environment. *Data Knowl. Eng.*, **69**(5):472-490. [doi:10.1016/j.datak.2010.01.001]

Minkiewicz, A., 2004. Use case sizing. 19th Int. Forum on COCOMO and Software Cost Modeling.

Minkiewicz, A.F., 1999. Measuring object oriented software with predictive object points. Proc. Project Control for Software Quality.

Nageswaran, S., 2001. Test effort estimation using use case points. Proc. Quality Week, p.1-6.

NESMA, 1990. Definitions and Counting Guidelines for the Application of Function Points Analysis: a Practical Manual 1.0. Nederlandse Software Metrieken Associatie, Utrecht, The Netherlands.

NESMA, 1995. Definitions and Counting Guidelines for the Application of Function Points Analysis: a Practical Manual 1.1. Nederlandse Software Metrieken Associatie, Utrecht, The Netherlands.

NESMA, 2004. Definitions and Counting Guidelines for the Application of Function Points Analysis: a Practical Manual 2.0. Nederlandse Software Metrieken Associatie, Utrecht, The Netherlands.

NESMA, 2005. Definitions and Counting Guidelines for the Application of Function Points Analysis: a Practical Manual 2.1. Nederlandse Software Metrieken Associatie, Utrecht, The Netherlands.

NESMA, 2009. Definitions and Counting Guidelines for the Application of Function Points Analysis: a Practical Manual 2.2.1. Nederlandse Software Metrieken Associatie, Utrecht, The Netherlands.

Ouwerkerk, J., Abran, A., 2006. An evaluation of the design of Use Case Points (UCP). Proc. Int. Conf. on Software Process and Product Measurement, p.83-97.

Rains, E., 1991. Function points in an Ada object-oriented design. *OOPS Messenger*, **2**(4):23-25. [doi:10.1145/126983.126986]

Ribu, K., 2001. Estimating Object-Oriented Software Projects with Use Cases. MS Thesis, University of Oslo, Norway.

Smith, J., 1999. The Estimation of Effort Based on Use Cases. Rational Software White Paper, IBM.

Svetinovic, D., Berry, D.M., Day, N.A., *et al.*, 2007. Unified use case statecharts: case studies. *Requir. Eng.*, **12**(4):245-264. [doi:10.1007/s00766-007-0053-1]

Symons, C.R., 1988. Function point analysis: difficulties and improvements. *IEEE Trans. Softw. Eng.*, **14**(1):2-11. [doi:10.1109/32.4618]

Symons, C., 2001. Come back function point analysis (modernized)—all is forgiven! Proc. 4th European Conf. on Software Measurement and ICT Control, p.413-426.

Teologlou, G., 1999. Measuring OO software with predictive object points. Proc. 10th European Software Control and Metrics Conf.

Uemura, T., Kusumoto, S., Inoue, K., 1999. Function point measurement tool for UML design specification. Proc. 6th Int. Software Metrics Symp., p.62-69. [doi:10.1109/METRIC.1999.809727]

Whitmire, S.A., 1992. Applying function points to object-oriented software models. *In*: Keyes, J. (Ed.), Software Engineering Productivity Handbook. McGraw-Hill, p.229-244.

Zhang, Y., Lin, Y., Wang, H., 2009. Study on object-oriented software metrics. *Comput. Dig. Eng.*, **3**:117-119.

Zhao, H., Stockman, T., 1995. Software sizing for OO software development—object function point analysis. Proc. 2nd GSE Int. Conf. on Information Technology and Management, p.413-425.

List of electronic supplementary materials

Table S1 Notations
Table S2 Technical complexity factor (TCF)
Table S3 Environmental complexity factor (ECF)
Table S4 Sample U1
Table S5 Sample I1
Table S6 Sample MK
Table S7 Sample I1-U1
Table S8 Sample U1-I1-MK