

A Case Study on the Evaluation of COSMIC-FFP and Use Case Points

Çigdem Gencil*, Luigi Buglione, Onur Demirors*, Pinar Efe

Abstract

The size attribute of software has been measured by classifying different types of externally observable features of a software entity, such as inputs and outputs, and then by measuring those features. Each measurement method measures different types of features in a different way. Among various approaches to software size measurement, the methods based on measuring the amount of “functionality” the software system provides have become widely used due to their earlier applicability during the software development life cycle. After the introduction of ISO/IEC 14143 - a standard developed by International Organisation for Standardisation (ISO) on Functional Size Measurement (FSM) - IFPUG (International Function Point Users Group) Function Point Analysis (FPA), Mark II FPA, COSMIC (Common Software Measurement International Consortium) Full Function Points (FFP) and NESMA FSM methods have been approved by ISO as being international FSM standards. As object-oriented systems emerge, new size measurements methods and metrics have been developed to address the issues related to traditional measurement methods when applied to those systems.

This paper presents the results of a case study on implementing COSMIC FFP and Use Case Points (UCP) methods to an industrial project. The effort estimates obtained from the size figures are compared with the actual development effort utilized for the system. The difficulties faced during the measurement processes of both methods are discussed as well.

1. Introduction

During last years FSM has been applied more and more worldwide. Indirectly, also the number of projects included in well known and recognized benchmarks such as the one by ISBSG [18] (currently in its release 9, with more than 3000 projects included and new, parallel benchmark on enhancement maintenance projects) shows this dramatic increase. Usually, an organisation applies a single FSM method for measuring the functional size of its software projects. An interesting aspect stressed in [61] is about the refinement of software productivity measurement using further data for improving the measurement. But a significant question could be: are we sure that the methods we are using are the best one for our projects? Could we try to apply also another method for a trial? And what about the application of two sizing methods on a project in order to gather the information useful to achieve an improved effort & cost estimation process? Main counter-answer from the managers' viewpoint will be: how much does it cost? How many changes should I make in the internal processes? And the training on the new method for involved people? Will it be possible to automate the measurement process? For those organisations with object-oriented (OO) projects, a possible combination could be the one represented by COSMIC-FFP, accompanied by UCP, where this one – when UML documentation is yet available – could be relatively quick and cheap.

The objectives for this paper are therefore to propose a case study with a double size measurement on the same project and looking in detail to the obtained results, stressing all the information useful for improving the measurement process.

* The work of these authors are partially supported by Turkish Undersecretariat for Defense Industries

The paper is structured as follows. Section 2 discusses the evolution of FSM methods, with a particular attention to the OO domain. Section 3 proposes a case study, providing the background and results from the measurement according to COSMIC-FFP and UCP methods. Furthermore, the analyses of results as well as the comparison of effort estimation results are discussed. Finally, Section 4 draws the conclusions and the prospects for future works.

2. Literature Survey

2.1. A brief history of FSM methods

Fenton stated identifying the entities and attributes wished to be measured as the first obligation of any software measurement activity [13]. Accordingly he classified the software entities to be measured as processes, products, and resources.

Being a product attribute, “software size” has been measured by measuring the artifacts, deliverables or documents produced during a software process activity. However, software development practitioners do not have socially accepted basic size measures or on what constitutes product size. It has been associated to other attributes such as; the length of the code, functionality delivered to the users, amount of reuse and complexity of the development [13][45]. Therefore, software size measurement process has involved a wide range of metrics and methods.

The first approach to measure size is to associate this attribute with the “length” attribute of code. The length of code can be measured in terms of Lines of Code (LOC), number of characters, etc. LOC is the oldest and most widely used traditional size metric, which is the key input to most software cost/effort, productivity and quality measurements. It has also been used for normalisation of other metrics [13]. Although the oldest one, it is still the most popular size metric since it is objective, easy to understand and measure. However, since LOC is language-dependent, programs written in different languages cannot be directly compared. Accurate measurement of LOC is possible only at the later stages of a project when the code is written. Measurement in the early phases of a project when no code is available can only be done by experts in measurement, remembering it is an estimate and not a measurement, with a potential high variability against final values.

The second and widely known approach is to associate the size with the “functionality” attribute. The methods that use this approach measure the size of software in terms of the amount of functionality to be delivered to the users. Initially in 1979, Allan Albrecht – an IBM researcher - designed the “Function Points” (FP) metric and related FPA method in order to measure the functional size of software systems [5], improving it later in 1984 with John Gaffney [6]. After that, variants have been developed. During the ‘80s and ‘90s, several authors have suggested new FSM techniques that intended to improve the original FPA or extend its application field, typically intended for MIS environments [51]. These methods, which we found in the literature that measure the “functionality” attribute, are listed in Table 1 and briefly introduced in the following sections.

Table 1: Software Size Measurement Methods Based on “Functionality”

Year	Method	ISO Certification	Developer(s)
1979	Albrecht/IFPUG FPA	√ [22] (20926:2003)	Albrecht, IBM [5] [6] [17]
1982	DeMarco’s Metrics	Bang	DeMarco [12]
1986	Feature Points		Jones, SPR [29]
1988	Mark II FPA	√ [23] (20968:2002)	Symons [53] UKSMA[56]
1990	NESMA FPA	√ [24] (24570:2003)	Netherlands Software Metrics Users Association [43]
1990	ASSET- R		Reifer [46]
1992	3-D Function Points		Boeing [57]
1994	Object Points		Banker, Kauffman, and Kumar [8] [34]
1994	FP by Matson, Barret and Mellichamp		Matson, Barret and Mellichamp [37]
1997	Full Function Points		University of Quebec in coop. with the Software Eng. Laboratory in Applied Metrics [4], [51]
1997	Early FPA		Meli [38][39]
1998	Object Oriented Function Points		Caldiera, Antoniol, Fiutem, and Lokan [10]
1999	Predictive Object Points		Teologlou [54]
1999	COSMIC Full Function Points	√ [21] (19761:2003)	COSMIC [3][55]
2000	Early&Quick COSMIC FFP		Meli, Abran, Ho, Oligny [11], [40]
2001	Object Oriented Method FP		Pastor et al. [44]
2000	Kammelar’s Component Object Points.		Kammelar [30]
2004	FiSMA FSM		Finnish Software Metrics Association [14]

Due to the variation of these methods, inconsistencies among the methods have developed. Thus, in 1996, the International Organisation for Standardisation (ISO) started a working group on FSM to establish common principles for those methods. The first part of ISO/IEC 14143, which defines the fundamental concepts of FSM such as “Functional Size”, “Base Functional Components (BFC)”, “BFC Types” and the FSM method characteristics requirements that should be met by a candidate method to be conformant to this standard [19]. The standard promoted the consistent interpretation of FSM principles. Table 2 shows the parts of this standard.

Table 2: ISO/IEC 14143 Parts

Part Name	Year of Publication	Title
ISO/IEC TR 14143-1 [19]	1998	Definition of concepts
IEEE Std. 14143- 1 [16]	2000	Adoption of ISO/IEC 14143-1:1998
ISO/IEC TR 14143-2 [20]	2002	Conformity evaluation of software size measurement methods to ISO/IEC 14143-1:1998
ISO/IEC TR 14143-3 [25]	2003	Verification of functional size measurement methods
ISO/IEC TR 14143-4 [26]	2002	Functional size measurement - Reference model
ISO/IEC TR 14143-5 [28]	2004	Determination of functional domains for use with functional size measurement
ISO/IEC FCD 14143-6 [29]	2005	Guide for the Use of ISO/IEC 14143 and related International Standards

Currently, four methods have been approved by ISO to become an international standard (see Table 1):

- COSMIC Full Function Points (ISO/IEC 19761, 2003) [21].
- IFPUG Function Point Analysis (ISO/IEC 20926, 2003) [22].
- Mark II Function Point Analysis (ISO/IEC 20968, 2002) [23].
- NESMA Function Point Analysis (ISO/IEC 24570, 2005) [24].

The methods and metrics discussed above have been designed to measure “length” and “functionality” attributes. There developed other size measurement metrics and methods such as “Use Case Points”, “Web Points”, etc. These are measuring other size related attributes of software.

2.2. Size Measurement and Object-Oriented Software Development

As Object-Oriented (OO) software systems emerged, various approaches have been developed to measure these kinds of software. These involve the methods, which adapt the FSM methods to the needs of OO software development (see Table 1) as well as totally newly developed ones.

A widely referenced method is the “Object Points (OP)” method [8] based on an earlier work by Kauffman and Kumar [33]. The underlying concepts for this method are very similar to of those in FPA, except that software objects, instead of functions, are being counted [34]. A software object may be a Rule Set, a 3GL Module, a Screen Definition, or again a Report.

Caldiera et al. [10] presented the “Object Oriented Function Points” (OOF) method for estimating the size of object oriented software development projects. The central concept in FPA are logical files and transactions whereas in OOFs the classes and their methods [62]. This method maps data functions of FPA to classes and transactional functions of FPA to class methods.

The “Predictive Object Points (POPs)” method [54] includes a series of metrics based on the three dimensions of the OO size, i.e. functionality, complexity and reuse. The metrics involved in POPs count are: number of top-level classes, weighted methods per class, average depth of inheritance tree, and average number of children per base class. Later, this method is embedded in Price Systems tool, a commercial product [42].

Kammelar described another approach, which applies the idea behind FPA to the OO concepts with new counting rules rather than mapping the OO concepts to FPA [30]. In this approach, the size is defined in terms of Component Object Points (COPs). It takes into

account reusability and takes use cases as a base in its calculations. In the counting process, User Domain Elements (FUR - Functional User Requirements) which include the use cases and business objects and System Domain Elements (BFCs) which include services, classes, operations and transformations, are determined. Then three different measurements (domain model count, analysis count and design count), are conducted.

“Object Oriented Project Size Estimation (OOPS)” is a statistical technique [9], where the main idea is to obtain the object’s “point value” depending on the objects’ names, number of attributes, methods and parameters to those methods. Then, this value is used to determine the days required to develop that object.

The “OO-Method Function Points” (OOmFP) was designed to be conformant to the IFPUG FPA counting rules for OO systems [1][44]. The IFPUG counting rules are redefined in terms of the concepts used in OO-Method. As in IFPUG-FPA, the data and transactional functions are taken into account [2]. All information that exists in the OO-Method conceptual model views is used for measurement.

The “Distance-Based Approach” is a mathematical method in which definition of distance is used to measure the size of OO specifications [45]. The method defines the size of an object as the distance between this object and a reference object.

The “Vector-Based Approach” introduced by Hastings and Sajeev [15], is based on two concepts: “Vector Size Measure (VSM)” to size the system and “Vector Prediction Model (VPM)” to estimate the corresponding effort. The approach attempts to measure the system size from the algebraic specifications described in the Algebraic Specification Language (ASL). The algebraic specifications are based on abstract data types (ADTs) and ASL provides a mathematical description of the system. The approach accepts Fenton’s multidimensional definition of size. To measure VSM, first system functionality and complexity is calculated. Then, the system length is derived from these values. Similar to Halstead’s method, the ADT properties are defined in terms of operators and operands.

Laranjeira’s “Statistical Object Model (SOM)” was developed to measure software size for OO systems [36]. SOM tries to provide the estimators more accurate size estimates by using statistical theory. Non-functional requirements and low biasing are taken into consideration in the model. In addition, various cost measurement models (e.g. COCOMO) uses the results of SOM as an input. SOM is a statistical approach to estimate the size of software within a specified confidence interval. Its logic comes from Boehm’s previous cost measurement studies.

The “Shepperd and Cartwright Size Prediction System” was developed in 1997 [50]. By using the data from the empirical investigation of an industrial OO real time C++ system, they found that the count of states per class in the state model could be a good predictor of size in SLOC.

“Use Case Points (UCP)” method was developed by Gustav Karner [31][32]. UCP method is based on FPA method [7]. The measurement process involves the weighted count of the number of actors of the use case model and the number of use cases. Details for this method are provided in Section 3.2.2

3. A Case Study

In this section, we discuss the results of a case study with a double size measurement on the same project.

3.1. Organisational and Project Context

The case project is a military inventory management project integrated with a document management system. It is a data-strong system, which also involves a number of algorithmic operations. The software development organisation is a Turkish independent supplier. The Organisation received the ISO 9001:2000 and AQAP 110 certifications and during an internal assessment conducted by the Quality Department was recognized conformant to CMMI/SW Maturity Level 3. The annual turnover is around 4 million Euros, with nearly seventy software developers. The organisation focus on mostly web based projects and has its own framework to develop web applications rapidly. The project was started in October 2004 and completed in December 2005.

The project staff consisted of 6 people, profiled as in the following:

- 1 project manager: 5 years project management experience.
- 1 senior software engineer (development team – full time): 5 years software development experience, expert in object oriented analysis, design with UML, development with Java, database design, familiar with Internal Development Framework, good at Oracle.
- 1 software engineer (development team – full time): 3 years software development experience; expert in object oriented analysis, design with UML, development with Java, database design, familiar with Internal Development Framework, good at Oracle.
- 1 software engineer (development team - part time): 2 year software development experience; expert in object oriented analysis, design with UML, development with Java, familiar with Internal Development Framework, good at Oracle.
- 1 software engineer (development team - part time): familiar with Java, Internal Development Framework and Oracle.
- 1 software engineer (test team – part time).

The effort needed for the development and management activities of the project are:

Table 3: Efforts Utilized for the Life Cycle Processes of the Case Project

Software Development Life Cycle Phase	Effort (person-hours)
<i>Development</i>	6,308.0
Software Requirements Analysis	911.0
Software Design (Architectural-Detailed)	698.0
Software Coding & Unit Testing	3,111.0
Testing	1,588.0
<i>Management</i>	225.0
<i>Training</i>	240.0
<i>Support</i>	720.0
Total	7,493.00

The types of software products and programming language(s) used for the project are:

- Analysis and Design tool; Rational Rose Enterprise Edition Release - v2003.06.13.402
- Development: IBM WebSphere Application Developer v4.0
- Java Development Kit (JDK) v1.3
- Tomcat Application Server v4.0.6
- Oracle 9i Database Management System v9i – 9.2.0.5
- Internal Development Framework v2.1

3.2. Case Study Conduct and Data Collection

We implemented COSMIC-FFP v2.2 and UCP methods to measure the same software system. We selected COSMIC-FFP as being an ISO certified FSM method and since the authors of this paper are experienced in using this method. UCP was selected since it is specifically designed to measure OO software

COSMIC-FFP method was published by COSMIC in 1999 [3]. This method is designed to measure the functional size of data-strong and control-strong software based on the FURs. In this method, each FUR is decomposed into its elementary components, called Functional Processes. A Functional Process is defined as “an elementary component of a set of FURs comprising a unique cohesive and independently executable set of data movements. Each of these Functional Processes comprises a set of sub-processes which perform either a data movement or a data manipulation. The BFCs of this method are “Data Movement Types”: Entry, Exit, Read, and Write. The functional size of each Functional Process is determined by counting the Entries, Exits, Reads and Writes in each Functional Process. Then, the functional sizes of all Functional processes are aggregated to compute the overall size of the system.

Use Case Points (UCP) method was developed by Gustav Karner [31][32]. The idea behind this method is quite close to the FPA method [47]. First, the actors of the use case model are categorized depending on their properties and assigned weights. Then, the number of actors in each category is counted. Each of these counts is multiplied with the corresponding weight factors, and then summed to obtain the Unadjusted Actor Weight (UAW). Depending on the number of transactions included, the use cases are categorized and weights are assigned. The number of use cases in each category is counted. Each of these counts is multiplied by the corresponding weight factors, and then summed in order to obtain the Unadjusted Use Case Weights (UUCW). Summing UAW and UUCW, the Unadjusted Use Case Points (UUCP) is obtained. By using a list of 13 technical complexity factors and 8 environmental factors, UUCP are adjusted, obtaining the final UCP value.

In order to measure the size of the case project, we used the SRS document of the case project, which involves 125 Use Cases (UC). The company uses an SRS standard developed by the company itself.

3.2.1. Implementation of the COSMIC-FFP Method

Two people performed the COSMIC-FFP count. One of them works for the development organisation and was involved in this project. The other one is one of the authors of this paper. Although both of them are experienced in using the COSMIC-FFP method, they are not certified by COSMIC5.

5 As well as in the IFPUG practice with the CFPS (Certified Function Point Specialist) exam, also the COSMIC-FFP community is moving toward this direction.

By applying COSMIC-FFP v.2.2, the number of functional size unit for the project was equal to 1,029.0 Cfsu (Table 4). The effort needed to perform the COSMIC-FFP measurement was equal to 12.58 person-hours.

Table 4: Case Study COSMIC-FFP Size Measurement Details

Number of Functional Processes	Number of Entries	Number of Exits	Number of Reads	Number of Writes	Functional Size (Cfsu)
125	157	381	336	155	1,029.0

Logical SLOC values for the case projects are given in Table 5. Since the user interface and the database components of this project are developed by using the Internal Development Framework, the SLOC values for these components are not be directly comparable, since XML files are generated by this tool. Internal Development Framework is a tool to reuse CRUDL processes in standard web applications. By this tool, the interface and database components are generated in parallel. For the processing part, Java is as the primary programming language. The SLOC values for the processing part are obtained by using Borland Together Architect, a multi-platform UML modeler. This tool measures the logical un-commented SLOC values.

Table 5: SLOC Values of the Case Project

GUI SLOC (XML)	Process Logical SLOC (Java)	Permanent Data Storage SLOC (XML)
11,760	11,817	23,550

3.2.2. Implementation of the UCP Method

Two people performed the FSM using the UCP method. One of them works for the development organisation and was involved in this project. The other one is one of the authors of this paper. Although they are experienced in implementing COSMIC-FFP, Mk II FPA and IFPUG FPA methods, this was the first time they implemented UCP.

By applying UCP method, the size of the project was equal to 592.994 UCP (Table 6). A UC based software development effort estimation tool named EzEstimate (see Annex A for a list of UCP tools) was used for the measurement. The effort needed to perform the measurement was equal to 3 person-hours.

Table 6: UCP Method Measurement Summary

Elements	Values
Number of Use Cases	125 (all simple)
Number of Actors	9 (all complex)
Unadjusted Actor Weight (UAW)	27
Unadjusted Use Case Weights (UUCW)	625
Unadjusted Use Case Points (UUCP)	652
TFactor	25
EFactor	11
Technical Complexity Factor (TCF)	0.85
Environmental Factor (EF)	1.07
Use Case Points (UCP)	592.994

The total *UAW* is obtained by counting the number of actors in each category, multiplying each one by its specified weight, as shown in Table 7, and then summing the partial results.

The number of actors counted in the use case model was equal to ‘9’. All actors were people interacting with the system through a graphical user interface (GUI). Therefore, they were categorized as complex, with a related weight equal to ‘3’. The *UAW* was therefore equal to 27 (Table 06).

Table 7: Actor Categories and Corresponding Weight Factors

Actor Categories	Properties	Weight Factors
Simple	Another system with a defined API	1
Average	Another system interacting through a protocol such as TCP/IP	2
Complex	Such a person indicating through a GUI or a webpage.	3

Second step was to count the number of use cases and the number of transactions in each use case. The use cases were classified depending on the number of transactions they included (Table 8).

Table 8: Use Case Categories and Corresponding Weight Factors

Use Case Categories	Number of Transactions Included	Weight Factors
Simple	≥ 3	5
Average	4-7	10
Complex	< 7	15

In this case study, all use cases categorized as ‘simple’ since they have 3 or fewer transactions. The total number of use cases found was 125. This figure is multiplied with the corresponding weight factors and then summed to obtain the final *UUCW* value (Table 9).

Table 9: UUCW Calculation Details

Use Case Categories	Number of Use Cases	Weight Factors	Total Weight
Simple	125	5	625
Average	0	10	0
Complex	0	15	0
<i>UUCW</i>			625

The *UUCP* is calculated applying the following formula:

$$\mathbf{UUPC} = \mathbf{UAW} + \mathbf{UUCW} = 27 + 625 = \mathbf{652}$$

The *UCP* were adjusted according to the values assigned to technical complexity factors (Table 10) and environmental factors (Table 11).

Table 10: Technical Complexity Factors (TCF)

Factor #	Description	Weight
T1	Distributed system	2
T2	Response or throughput performance objectives	2
T3	End-user efficiency	1
T4	Complex internal processing	1
T5	Reusable code	1
T6	Easy to install	0.5
T7	Easy to use	0.5
T8	Portable	2
T9	Easy to change	1
T10	Concurrent	1
T11	Includes security features	1
T12	Provides access for third parties	1
T13	Special user training facilities are required	1

Table 11: Environmental Factors (EF)

Factor#	Description	Weight
E1	Familiar with Rational Unified Process (RUP)	1.5
E2	Application experience	0.5
E3	OO experience	1
E4	Lead analyst capability	0.5
E5	Motivation	1
E6	Stable requirements	2
E7	Part-time workers	-1
E8	Difficult programming language	-1

A value between 0 and 5 was assigned to each factor by the estimator, depending on their rate of influence on the system.

The TCF is calculated by using the following formula;

$$TCF = 0.6 + (0.01 * TFactor)$$

$$TCF = 0.6 + \left(0.01 * \sum_{n=1}^{13} T_n \right)$$

In this case study, TFactor was equal to '25' and therefore TCF to '0.85'.

The EF is calculated by using the following formula;

$$EF = 1.4 + (-0.03 * EFactor)$$

$$EF = 1.4 + \left(-0.03 * \sum_{n=1}^8 E_n \right)$$

In this case study, Efactor was equal to '11' and therefore EF to '1.07'.

Finally, the adjusted UCP is obtained applying the following formula.

$$\text{UCP} = \text{UUCP} * \text{TCF} * \text{EF}$$

$$\text{UCP} = 652 * 0.85 * 1.07 = 592.994 \cong 593$$

3.3. Analysis of Results

After presenting the results from the two functional counts, here in the following some thoughts and reflections are presented, starting from points and assumptions in common between the two methods as well as the differences, trying after to read them from a measurement process viewpoint. This can lead to have more information to reduce the probability that a risk due to effort estimation errors will occur in the future projects.

3.3.1. Commonalities

- **Input documentation:** both methods use (or can use) some UML diagrams as the main input allowing the FSM for a software system, even if COSMIC-FFP goes more into details for calculating Cfsu [21] and can use UC as a high-level documentation.

3.3.2. Differences

A series of differences is evidenced between the two methods as follows:

- **Input documentation:** although both methods can measure software systems, the FURs of which are defined by UC, COSMIC-FFP does not have such limitation to use UC as UCP, but requires only the FURs defined by any notation.
- **Adjustment Factors:** COSMIC-FFP is defined as a 2nd FSM method and has rejected – according to the ISO 14143-1 viewpoint – the usage of adjustment factors, while UCP – created in 1993 – as Mk II FPA method proposes two adjustment factors (TCF and EF). The UCP method does not have a measurement manual describing in detail TCF and EF. Therefore, we encountered some difficulties in interpreting the factors when assigning weights.. This is significant since TCF may change a projects size in the order of a range [0.6-1.35] whereas EF may change it between [0.425-1.7]. Thus, their total effect would be in the [0.255-2.295] range, with possible adjustment of the projects size that can change with a factor of 9.
- **Layers:** COSMIC-FFP introduces the concept of *layers*; therefore each software system can be evaluated taking into account different software layers while UCP – as the other 1st generation FSM methods – consider only a single layer in its measurement process.
- **Viewpoints:** COSMIC-FFP extends the traditional End-User viewpoint towards a broader view, including several possible viewpoints. A use case describes the system's behavior under various conditions, responding to the requests from one of the stakeholders, called the *primary actor*, but considering at the same time different possible scenarios.
- **Metrics:** COSMIC-FFP measures the functional size by counting the Entry, Exit, Read and Write whereas UCP counts the number actors and UC to derive the size. These two methods use different metrics, which makes the size figures obtained by these methods incomparable directly.

3.3.3. Productivity

According to the above presented differences between the two methods, in Table 12, Productivity Delivery Rate (PDR) as well as Productivity rates for the case project are provided. In order to compare the productivity against two (or more) methods, the comparison should be done by the common element (in this case, the effort in man-days).

Because there is no a consolidated correlation value between UCP and COSMIC-FFP based on a huge historical dataset, the solely comparison by PDRs can offer the risk to be meaningless or offer a little informative value.

Table 12: PDR (Development Effort / UCP) Rates for the Case Project

Development Effort (person-hours)	Functional Size		PDR ⁶		Productivity rate	
	UCP	Cfsu	person-hours/UCP	person-hours/Cfsu	UCP/person-hours	Cfsu/person-hours
6,308.00	593	1,029.0	10.7 (1.3375)	6.13 (0.76625)	0.094 (0.752)	0.163 (1.304)

3.4. Effort Estimation

About the usage of Cfsu as the size unit for predicting the project effort, more and more attention is paid and there are several papers and technical reports in this issue. First of all, Fetcke [63] proposed a comparative study among different FSMM and versions counting 5 applications stressing commonalities and differences. More recent studies tried to derive estimation formulas on reduced datasets by linear regression as [64].

About the usage of UCP as the primary predictor for effort estimations, there is a little literature on it, and no evidence of estimation studies with consistent datasets, also due also to the not so huge application of the method⁷.

Since there are not enough studies with consistent datasets used for estimation purposes for both methods, we decided to compare the results from the case study with some external sources.

3.4.1. Effort Estimation by COSMIC-FFP

For COSMIC-FFP, we considered the data from the ISBSG r9 dataset [18], including 3024 projects. The ISBSG dataset was used to determine the normalized PDR and the correlation between the functional size and the related normalized effort utilized for developing the projects, measured by COSMIC-FFP. The normalized work effort is an estimate of the full development life-cycle effort for projects covering less than a full development life-cycle. Projects were grouped according to their application type.

In the analyses, the selection from the ISBSG repository was done by the following attributes, on all projects reporting the functional size (excluding therefore those where this number is missing):

Table 13: Filtering the ISBSG Dataset

Step	Variable	Filter	Excluded Elements	Remaining
1	Count Approach ⁸	'COSMIC'	2951	73
2	Development Type	'New Development'	17	56
3	Data Quality Rating (DQR)	{A B C}	5	51
4	Application Types	see Table 14	see Table 14	see Table 14

⁶ In parenthesis, the same values are also expressed into person-days.

⁷ For instance, Minkievitz [41] proposed a comparative analysis of PDR for IFPUG FPA and UCP, but applied to 15 use cases (and not to complete projects including a certain number of use cases).

⁸ No further filter has been considered about the COSMIC-FFP versions.

The impact of these restrictions is to eliminate any project for which ISBSG has doubts about the data validity and ensure that all selected projects have been sized using the same technique⁹. After applying these filters to the initial database (including 3024 projects), we obtained in return 51 projects, reported as in Table 14.

In Table 14, the maximum, average, median, minimum and standard deviation values for the Normalized PDR of the selected projects are also given, calculated by subdividing the Normalized Work Effort by the Unadjusted Function Point (UFP) count. We used normalized effort and unadjusted count to have the comparable rates.

Table 14: Normalized PDR for the Projects Measured by COSMIC-FFP in ISBSG r9 Dataset (n =51)

Appl.Type	No.Prj	Normalized PDR (person-hrs/function size)					R ²	Regr Formula (y*=a+bx)
		Max	Avg	Med	Min	Std Dev		
<i>Management Information System</i>	15	4.20	1.89	1.49	0.69	1.17	0.3942	a: 123.65 b: 0.9286
<i>Operating system or software utility</i>	7	8.75	3.25	1.47	0.92	3.16	0.5539	a: 124.47 b: 0.7025
<i>Financial transaction process/accounting;</i>	6	330.56	95.33	46.33	11.28	123.18	0.8910	a: -972.78 b: 67.314
<i>Network Management</i>	3	34.79	23.57	18.80	17.13	9.75	0.8868	a: -1141.2 b: 46.76
<i>Others</i>	20	139.35	28.57	7.51	0.20	36.41	0.1429	a: 3056.5 b: 17.078
All	51	330.56	25.27	4.70	0.20	54.41	0.2653	a: -504.76 b: 24.121

Considering that the case study project was similar to a pure MIS project, but involving also some algorithmic operations¹⁰, we applied the linear regression formula derived for such subset in order to verify such values. Starting from the initial dataset of 15 projects, we deleted 3 outliers, improving R² up to 0.732. Simply applying the new linear regression formula (y*=1.2147x+52.788) to the current Cfsu, we would obtain 1302.71 person-hours.

⁹ 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 UFP Rating, ‘C’ is given to the projects data for which it was not possible to provide the unadjusted function point data due to unadjusted function point or count breakdown data not being provided and ‘D’ 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.

¹⁰ It could lead to obtain a PDR greater than for the solely MIS projects.

3.4.2. Effort Estimation by UCP

Actually there is not a public, consistent project dataset presenting UCP data as the ISBSG one does for the ISO/IEC 14143 compliant ones. Some suggestions in the technical literature about average productivity figures come from the UCP creator, Karner [31] that proposed a PDR of c.a. 20hrs/UCP; again Ribu [47] suggested a range between 15 and 30 hrs/UCP; and finally Schneider & Winters [49] proposed an average of 20-28 hrs/UCP.

In order to verify such values, we applied the suggestions by Schneider and Winters [49]¹¹ to our case study project, simulating an effort estimation calculation; simply applying a value of 20 person-hours/UCP, we would obtain an estimated development effort for this software system equal to 11,859.88 person-hours.

3.4.3. Relative Errors (RE)

The actual total effort utilized for the development and management activities of the case project was equal to 6,308 person-hours (as shown in Table 3). Supposing to apply as-is the above presented information for estimation purposes, the results and relative errors are those presented in Table 15.

Table 15: Comparison of the Estimated and Actual Effort

Actual Dev. Effort (person-hours)	Estimated Effort		% Relative Error between the Actual and Estimated effort	
	COSMIC-FFP Size (person-hours)	UCP Size (person-hours)	COSMIC-FFP	UCP Size
6,308	1,302.71	11,859.88	-384.221%	+88.013%

The main problem in both estimation – leading to wrong results – states in wrong assumptions and more info should be explored before using such estimates for planning purposes. Thus, going more into details, the size of the case study project (1,029 Cfsu) is greater than the maximum value for the MIS projects subset (470 Cfsu). In this case, the usage of the ISBSG repository would be of no help, with the risk to lead to dangerous results.

And about UCP, it would be meaningless to apply as-is a PDR value in a start-up phase without performing some tests and experiments for determining an approximate internal productivity level using UCP. In this case, the application of suggestions from technical literature, would also be of no help, leading to an effort estimate quite double of the actual one. So, the estimation process is a very relevant issue, to be treated with great care.

3.4.4. Automating the FSM counting: some UCP Tools

One of the main interesting issues from the Industrial viewpoint is about tools for automating the count. In particular, the goal of managers would be to make the functional count automatic in order to reduce the time devoted to counting (several days for each project). During the last 20 years, more than a time it was experimented; some references are, for instance the UPAAC tool [58], the FUN project [59] and an approach based on analogy [60]. Although the different approaches, this issue represents a shortcoming in FSMs, because these methods are requirement-based methods, therefore requiring a human-intensive

¹¹ The number of factors F1-F6 with values below 3 have been counted and added to the number of factors F7-F8 that are above 3. If the total is 2 or less, a value of 20 person hours per UCP is used; if the total is equal to 3 or 4, 28 man-hours [0]per UCP is used. If the number exceeds 4, it is recommended that changes should be made to the project so the number can be adjusted, or alternatively that the number of man-hours should be increased to 36 per UCP.

activity and it is not possible to make it automatic. Among reasons, one is the definition and related level of granularity for a user requirement to be refined into functional specifications. An evidence for it can be found looking at the several commercial tools managing FP: they typically work as FP repositories with all possible elements details for estimation purposes.

In the case of UCP, the working products from which retrieving the information for the count are yet ready (UML diagrams) when an organisation applies UML as an internal standard. So, as in [58] it is possible to use a tool, and the main desired advantage (to count the number of FP from UR) cannot be achieved (in fact, the automation comes after the human-based activity, passing from the elicitation of Customer desiderata into UR and then into SRS and FUR). That's why the suggestion to apply UCP into an organisation yet producing analysis documents UML-based should be evaluated quite cheap, in place of other ones. Refer to Annex A for a list of tools with UCP features.

4. Conclusion & Prospects

During years several FSM methods (not strictly ISO 14143 compliant) have been proposed to the Software Measurement community for a best fit with object-oriented projects more than the IFPUG FPA method, and COSMIC-Full Function Point is one of them. A further proposal in 1993 was the one by Karner, called UCP. UCP moves from some UML diagrams (Use Cases and Scenarios) and can be easily calculated and, differently from other methods, be automatized. A list of tools is in Annex A.

A case study was introduced and the project (a MIS project with an actual development effort of 6308 person-hours) size was measured against both methods (1,029 Cfsu; 593 UCP), presenting the detailed figures. Average PDR from external sources were compared to our values, revealing consistent differences on which investigate more.

Differently from the comparisons between COSMIC-FFP and other ISO/IEC 14143-compliant methods, these two functional size methods can be seen as complementary and not necessarily alternative¹². In fact COSMIC-FFP presents a series of advantages not present in UCP (i.e. multi-layer; not UML-dependent; ISO/IEC standard method) but on the opposite needs a certain amount of time and knowledge of the measurement rules to be calculated. UCP, being UML-based, can be a Trojan horse in an organisation helping to spread UML as a reference standard for analysis and from a management perspective could be attractive because it can allow saving time for counting the size through automatic tools. So, with a little additional effort (3 person-hours were needed for measuring the case study project against more than 12 person-hours for the COSMIC-FFP count) for gathering both measures from projects and building a double-size historical series, it will be possible for an organisation to establish its R^2 value between the two (obviously considering the UUCP and not the final UCP values, for a sake of consistency), as done in [41] between IFPUG FPA and UCP. This will allow:

To verify if UCP have been consistently applied to projects in terms of granularity in the UC and Scenarios definitions

This revision will be therefore beneficial for a better definition of analysis documentation and UML diffusion inside the organisation

Next steps will be related in gathering a considerable numbers of projects to be sized against both measures, in order to validate the hypothesis presented in the current paper.

¹² An example – even if from a different perspective – is in [65].

5. References

- [1] Abrahão S., Pastor, O., “Estimating the Applications Functional Size from Object-Oriented Conceptual Models”, In: International Function Points Users Group Annual Conference (IFPUG'01), Las Vegas, USA, 2001
- [2] Abrahão, S., Poels, G., Pastor, O. “Functional Size Measurement Method for Object-Oriented Conceptual Schemas: Design and Evaluation Issues”, Working Paper, Faculty of Economics and Business Administration, Ghent University, 2004
- [3] Abran, A., “COSMIC FFP 2.0: An Implementation of COSMIC Functional Size Measurement Concepts”, FESMA'99, Amsterdam, 7 October 1999
- [4] Abran, A., St-Pierre, D., Maya, M., Desharnais, J.M., “Full Function Points for Embedded and Real-Time Software”, UKSMA Fall Conference, London (UK), October 30-31, 1998.
- [5] Albrecht, A. J., “Measuring application development productivity”, in Proceedings of IBM Applications Development Symposium, Monterey, California, 14-17 October 1979
- [6] Albrecht, A.J. and Gaffney J. E., “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] Anda, B., Dreiem, H., Sjoberg, D. I. K., Jorgensen, M., “Estimating Software Development Effort based on Use Cases-Experiences from Industry”, 4th International Conference on the Unified Modeling Language (UML2001) Toronto, Canada, October 1-5, 2001, pp. 487-502, LNCSE 2185, Springer-Verlag, 2001.
- [8] Banker, R., Kauffman, R.J., Wright, C., Zweig, D., “Automating Output Size and Reuse Metrics in a Repository Based Computer Aided Software Engineering (CASE) Environment”, IEEE Transactions on Software Engineering, Vol.20, No.3., 1994.
- [9] Bradine D., “Oops, There It Is”, Enterprise Systems Journal, March 2000.
- [10] Caldiera, G., Antonioli, G., Fiutem, R., Lokan, C. “Definition and Experimental Evaluation for Object Oriented Systems”, Proceedings of the 5th International Symposium on Software Metrics, Bethesda, 1998.
- [11] Conte, M., Iorio, T., Meli, R., Santillo, L., “E&Q: An Early & Quick Approach to Functional Size Measurement Methods”, Proceedings of the 1st Software Measurement European Forum (SMEF2004), Rome, Italy, 2004.
- [12] DeMarco, T., “Controlling Software Projects”, Yourdon press, New York, 1982.
- [13] Fenton, N.E. and Pfleeger, S.L., Software Metrics: A Rigorous and Practical Approach, Second Edition, International Thomson Computer Press, 1996
- [14] Forselius, P.. Finnish Software Measurement Association (FiSMA), FSM Working Group: FiSMA Functional Size Measurement Method, version 1.1, 2004
- [15] Hastings T.E. & Sajeev A.S.M., “A Vector-Based Approach to Software Size Measurement and Effort Estimation”, IEEE Transactions on Software Engineering, vol. 27, no. 4, April 2001.
- [16] IEEE Std. 14143.1-2000, Implementation Note for IEEE Adoption of ISO/IEC 14143-1:1998 - Information Technology- Software Measurement- Functional Size Measurement -Part 1: Definition of Concepts, 2000.
- [17] IFPUG: Counting Practices Manual - Release. 4.1, International Function Point Users Group, Westerville, OH, 1999.
- [18] ISBSG (International Software Benchmarking Standard Group) Dataset (r9), <http://www.isbsg.org>, 2004.
- [19] ISO/IEC 14143-1:1998 Information technology -- Software measurement -- Functional size measurement -- Part 1: Definition of concepts, 1998.
- [20] 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.
- [21] ISO/IEC 19761:2003: COSMIC Full Function Points Measurement Manual v. 2.2. (2003).
- [22] ISO/IEC 20926, Software engineering - IFPUG 4.1 Unadjusted FSM Method - Counting Practices Manual, 2003.

- [23] ISO/IEC 20968, Software engineering - Mk II Function Point Analysis - Counting Practices Manual, 2002.
- [24] ISO/IEC 24570: Software engineering - NESMA Functional Size Measurement Method v.2.1 - Definitions and counting guidelines for the application of Function Point Analysis, 2005.
- [25] ISO/IEC TR 14143-3:2003 Information technology -- Software measurement -- Functional size measurement -- Part 3: Verification of functional size measurement methods, 2003.
- [26] ISO/IEC TR 14143-4:2002 Information technology -- Software measurement -- Functional size measurement -- Part 4: Reference model, 2002.
- [27] 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.
- [28] ISO/IEC FCD 14143-6:2005: Guide for the Use of ISO/IEC 14143 and related International Standards, 2005.
- [29] Jones, T. C., A Short History of Function Points and Feature Points, Software Productivity Research Inc., USA, 1987.
- [30] Kammelar, J., "A Sizing Approach for OO-environments", 4th International ECOOP Workshop on Quantitative Approaches in Object-Oriented Software Engineering, 2000.
- [31] Karner, G., "Resource Estimation for Objectory Projects", White Paper, Objective Systems, 17 September 1993.
- [32] Karner, G., Metrics for Objectory. Diploma thesis, University of Linköping, Sweden. No. LiTHIDA-Ex-9344:21, 1993.
- [33] Kauffman, R. and Kumar, R., "Modeling Estimation Expertise in Object-Based CASE Environments", Stern School of Business Report, New York University, 1993.
- [34] Kauffman, R. and Kumar, R., "Investigating Object-Based Metrics for Representing Software Output Size", Conference on Information Systems and Technology (CIST), in the INFORMS 1997 Annual Conference, San Diego, 1997.
- [35] Kusumoto S., Matukawa F., Inoue K., Hanabusa S. & Maegawa Y., "Estimating Effort by Use Case Points: Method, Tool and Case Study", 10th IEEE International Symposium on Software Metrics (METRICS'04), Chicago, IL (USA), pp. 292-299, 2004.
- [36] Laranjeira, L., "Software Size Estimation of Object Oriented Systems", IEEE Transactions on Software Engineering, Vol. 16, No. 5. pp.510-522, 1990.
- [37] Matson, J. E., Barret, B. E., Mellichamp, J. M., "Software Development Cost Estimation Using Function Points", IEEE Transactions on Software Engineering, Vol. 20, No. 4, pp. 275-287, 1994.
- [38] Meli, R., "Early and Extended Function Point: A New Method for Function Points Estimation", IFPUG-Fall Conference, September 15-19, Scottsdale, Arizona, USA, 1997.
- [39] Meli, R., "Early Function Points: a new estimation method for software projects", ESCOM 97, Berlin, 1997
- [40] Meli, R., Abran, A., Ho, V.T., Oligny, S., "On the Applicability of COSMIC FFP for Measuring Software Throughout Its Life Cycle", Proc. of the ESCOM-SCOPE 2000, April 2000, Munich, Germany, Shaker Publ., pp. 289-297, 2000.
- [41] Minkiewicz A. "Use Case Sizing", 19th International Forum on COCOMO and Software Cost Modeling, Los Angeles, CA (USA), October 2004.
- [42] Minkiewicz, A. F., "In Measuring Object Oriented Software with Predictive Object Points", PRICE Systems, L.L.C, 2000.
- [43] Netherlands Software Metrics Association (NESMA), Definitions and Counting Guidelines for the Application of Function Point Analysis, Version 2.0, 1997.
- [44] Pastor, O., Abrahão, S.M., Molina, J.C. and Torres, I., "A FPA-like Measure for Object Oriented Systems from Conceptual Models", In Proceedings of the 11th International Workshop on Software Measurement - IWSM'01, Montréal, Canada, Shaker Verlag, pp. 51-69, 2001.
- [45] Poels, G., "Towards a Size Measurement Framework for Object Oriented Specifications", In Proceedings of the FESMA'98, Antwerp, Belgium, May 6-8, pp. 379-394, 1998.

- [46] Reifer, D.J., “Asset-R: A Function Point Sizing Tool for Scientific and Real-time Systems”, *Journal of Systems and Software*, Vol.11, No.3, pp.159-171, 1990.
- [47] Ribu K., “Estimating Object-Oriented Software Projects with Use Cases”, Master Thesis, University of Oslo, Dept. of Informatics, 7 November 2001.
- [48] Santillo, L. and Meli, R., “Early Function Points: some practical experiences of use”, ESCOM-ENCRESS 98, May 18, Roma, Italy, 1998.
- [49] Schneider G. & Winters J., “Applying Use Cases - A Practical Guide”, 2/e, Addison Wesley Longman, ISBN 0201708531, 2001.
- [50] Shepperd, M. & Cartwright, M., “An Empirical Investigation of Object Oriented Software System”, Technical Report No. TR 97/01, Dept. of Computing, Bournemouth University, UK, 1997.
- [51] St-Pierre D., Maya M., Abran A., Desharnais J.-M., Bourque P., « Full Function Points: Counting Practices Manual”, Technical Report, Université du Québec à Montréal, Montréal, Canada, 1997.
- [52] Symons, C., “Come Back Function Point Analysis (Modernized) – All is Forgiven!”, In *Proceedings of the 4th European Conference on Software Measurement and ICT Control, FESMA-DASMA 2001*, Germany pp. 413-426, 2001.
- [53] Symons, C. R., “Function Point Analysis: Difficulties and Improvements”, *IEEE Transactions on Software Engineering*, Vol. 14, No. 1, pp.2-10, 1988.
- [54] Teologlou, G., “Measuring Object Oriented Software with Predictive Object Points”, Shaker Publishing, ISBN 90-423-0075-2., 1999.
- [55] The Common Software Measurement International Consortium (COSMIC), FFP, version 2.2, *Measurement Manual*, 2003.
- [56] United Kingdom Software Metrics Association (UKSMA), *MK II Function Point Analysis Counting Practices Manual Version 1.3.1.*, 1998.
- [57] Whitmire, S.A., “3D Function Points: Scientific and Real-time Extensions to Function Points”, *Pacific Northwest Software Quality Conference*, 1992.
- [58] Cantone, G., Pace, D., Calavaro, G., “Applying Function Point to Unified Modeling Language: Conversion Model and Pilot Study”, *10th IEEE International Software Metrics Symposium (METRICS 2004)*, Chicago, IL, USA. IEEE Computer Society 2004, ISBN 0-7695-2129-0, pp.280-291, 11-17 September 2004.
- [59] Lamma E., Mello P., Riguzzi F., ”A System for Measuring Function Points”, Technical Report, DEIS-LIA-97-006, Laboratory of Advanced Research on Computer Science, University of Bologna, Italy, 1997.
- [60] Frallicciardi, L., Natale, D., “Estimating size using Function Point Analysis and Analogy”, *Proceedings of the 8th ESCOM Conference*, Berlin, May 26-28 1997, pp. 69-73
- [61] Kitchenham, B., Mendes, E., “Software Productivity Measurement Using Multiple Size Measures”, *IEEE Transactions on Software Engineering*, chapter 30 (issue 12), pp. 1023-1035, December 2004.
- [62] Morisio, M., Stamelos, I., Spahos, V., Romano, D., “Measuring Functionality and Productivity in Web-based Applications: A Case Study”, *Proceedings of the sixth IEEE International Symposium on Software Metrics*, November 04 – 06, pp. 111-118, 1999.
- [63] Fetcke T., *The Warehouse Software Portfolio. “A Case Study in Functional Size Measurement”*, University of Berlin, Technical Report, 1999-20, 1999.
- [64] Nagano, Shin-ichi; Mase, Ken-ichi; Watanabe, Yasuo; Watahiki, Takaichi; Nishiyama, Shigeru, “Validation of Application Results of COSMIC-FFP”, in *Australian Software Conference on Measurement (ASCOM)*, Australia, 2001.
- [65] Habela, P., Glowacki E., Serafinski T., Subieta K., “Adapting Use Case Model for COSMIC-FFP Based Measurement”. *Proceedings of the 15th International Workshop on Software Measurement (IWSM2005)*, Montréal, Canada, 12-14 September 2005.

Annex A – Some tools for UCP count

Name	Company / Author	Description http://www.	Pros	Cons
Enterprise Architect 6.0	SparxSystems	sparxsystems.com.au/UCMetrics.htm	<ul style="list-style-type: none"> ◦ Trial version available ◦ Fully functional UML modelling tool including use case points 	
U-EST	Kusumoto et al [35]		<ul style="list-style-type: none"> ◦ Calculate use case point from use case models written in XMI files ◦ Judge the complexity of actors and use cases 	<ul style="list-style-type: none"> ◦ No trial or demo version
EEUC	Duversa Software	duvessa.com/products.htm	<ul style="list-style-type: none"> ◦ Trial version available ◦ Can import use cases and actors in UML diagrams from Rational Rose, XDE, MS Visio 2000 and XMI file ◦ Can import use case name from a use case specification from RequisitePro ◦ Has What-if Capability analysis to create and analyse different configurations of actors, use cases, use case estimation factors and Cocomo II Level 1 parameters 	
EstimatorPal	Chemuturi Consultants	◦ ffortestimator.com	<ul style="list-style-type: none"> ◦ Demo version available ◦ Supports many different size, cost and effort estimation tools, IFPUG FPA, Object Points, Use Case Points, Task Based Effort Estimation, LOC Based Effort Estimation, Intermediate Cocomo ◦ Repository for counted objects 	<ul style="list-style-type: none"> ◦ Do not reuse the given data of one method for another one. ◦ No integration with any other tool

EzEstimate	StriveLogic	openrage.com/main/ezestimate_full.htm	<ul style="list-style-type: none"> ◦ Have free edition to use and distribute freely. ◦ Repository for use cases and actors in text format 	<ul style="list-style-type: none"> ◦ No integration with UML modelling tools
CostXpert 3.0	HALLoGRAM Publishing	hallogram.com/costxpert/	<ul style="list-style-type: none"> ◦ Supports Internet Points and UML Use-Case Points and Class-Method Points ◦ Export and import project size estimates to and from Microsoft Excel ◦ Certified as compliant with the latest COCOMO II.2000 ◦ Provides enhancements to the Work Breakdown Structure (WBS) and the ability to adjust the time-cost tradeoff according a defined maximum team size 	<ul style="list-style-type: none"> ◦ No trial or demo version ◦ No integration with UML modeling tools
PRICE TRUE S	PRICE Systems, L.L.C.	pricesystems.com/products/true_s_price_s.asp	<ul style="list-style-type: none"> ◦ Software development and lifecycle estimating model ◦ Can estimate costs, resources, and schedules for software projects 	<ul style="list-style-type: none"> ◦ No trial or demo version