Università degli Studi dell'Insubria
Dipartimento di Scienze Teoriche e Applicate

Dottorato di Ricerca in Informatica
XXVI Ciclo

Towards Making Functional Size Measurement Easily Usable in Practice

Ph.D Thesis of
Geng Liu

Advisor
Prof. Luigi Lavazza

Supervisor of the Doctoral program
Prof. Claudio Gentile

This page intentionally left blank.

In memory of my mother

To my family, especially to my father, for their love

This page intentionally left blank.

Acknowledgement

First of all, I owe heartfelt thanks to my supervisor Prof. Luigi Lavazza, not only for accepting me to research with him, but also for his patient guidance, stimulating suggestions and insightful comments throughout my research. I have benefited greatly from his profound knowledge and consistent encouragement. From the beginning to the end, he has guided my wide interest into a coherent thesis and has always replied my questions in time no matter when.

I give sincere gratitude to Prof. Sandro Morasca and Prof. Elena Ferrari for their guidance during my Ph.D. research, especially for helping me select the research field. I express my appreciation to Prof. Giovanni Denaro from University of Milano-Bicocca for his valuable and expert suggestions on my PhD thesis. My genuine thanks extend to Dr. Vieri del Bianco and Dr. Abedallah Zaid Abualkishik who have been working together with me.

I would also like to acknowledge Prof. Claudio Gentile, supervisor of our doctoral program, for the support he gave to all of us Ph.D. students, and to Prof. Elisabetta Binaghi, Prof. Simone Tini, Dr. Vallentina Pedoia for their help of improving my academic knowledge through lectures.

Moreover, I am deeply grateful to all my colleagues in the Lab, in particular Dr. Pietro Colombo, Dr. Stefano Braghin, Dr. Paolo Brivio, Lorenzo Bossi, Cuneyt Gurcan Akcora, Michele Guglielmi, Marco Daddeo, and Tran Hong Ngoc, for their friendship, help and encouragement. We encouraged each other, and of course, had a lot of fun while struggling for a brilliant future.

I would like to give very special thanks to my girlfriend Cui Liu for her endless help, support, and encouragement.

Last but not least, I'm deeply indebted to my family - my father, my sister, and my brother – for their love, understanding, patience, and unshakable faith in me.

Geng. Liu
Varese, Italy
December 18, 2013

This page intentionally left blank.

Abstract

Functional Size Measurement methods –like the IFPUG Function Point Analysis and COSMIC methods– are widely used to quantify the size of applications. However, the measurement process is often too long or too expensive, or it requires more knowledge than available when development effort estimates are due. To overcome these problems, simplified measurement methods have been proposed.

This research explores easily usable functional size measurement method, aiming to improve efficiency, reduce difficulty and cost, and make functional size measurement widely adopted in practice.

The first stage of the research involved the study of functional size measurement methods (in particular Function Point Analysis and COSMIC), simplified methods, and measurement based on measurement-oriented models.

Then, we modeled a set of applications in a measurement-oriented way, and obtained UML models suitable for functional size measurement. From these UML models we derived both functional size measures and object-oriented measures. Using these measures it was possible to:
1) Evaluate existing simplified functional size measurement methods and derive our own simplified model.
2) Explore whether simplified method can be used in various stages of modeling and evaluate their accuracy.
3) Analyze the relationship between functional size measures and object oriented measures.

In addition, the conversion between FPA and COSMIC was studied as an alternative simplified functional size measurement process.

Our research revealed that:

1) In general it is possible to size software via simplified measurement processes with acceptable accuracy. In particular, the simplification of the measurement process allows the measurer to skip the function weighting phases, which are usually expensive, since they require a thorough analysis of the details of both data and operations. The models obtained from out dataset yielded results that are similar to those reported in the literature.

All simplified measurement methods that use predefined weights for all the transaction and data types identified in Function Point Analysis provided similar results, characterized by acceptable accuracy. On the contrary, methods that rely on just one of the elements that contribute to functional size tend to be quite inaccurate. In general, different methods showed different accuracy for Real-Time and non Real-Time applications.

2) It is possible to write progressively more detailed and complete UML models of user requirements that provide the data required by the simplified COSMIC methods. These models yield progressively more accurate measures of the modeled software. Initial measures are based on simple models and are obtained quickly and with little effort. As

models grow in completeness and detail, the measures increase their accuracy. Developers that use UML for requirements modeling can obtain early estimates of the applications' sizes at the beginning of the development process, when only very simple UML models have been built for the applications, and can obtain increasingly more accurate size estimates while the knowledge of the products increases and UML models are refined accordingly.

3) Both Function Point Analysis and COSMIC functional size measures appear correlated to object-oriented measures. In particular, associations with basic object-oriented measures were found: Function Points appear associated with the number of classes, the number of attributes and the number of methods; CFP appear associated with the number of attributes. This result suggests that even a very basic UML model, like a class diagram, can support size measures that appear equivalent to functional size measures (which are much harder to obtain). Actually, object-oriented measures can be obtained automatically from models, thus dramatically decreasing the measurement effort, in comparison with functional size measurement.

In addition, we proposed conversion method between Function Points and COSMIC based on analytical criteria.

Our research has expanded the knowledge on how to simplify the methods for measuring the functional size of the software, i.e., the measure of functional user requirements. Basides providing information immediately usable by developers, the researchalso presents examples of analysis that can be replicated by other researchers, to increase the reliability and generality of the results.

Keywords:
Functional Size Measurement; FPA; COSMIC; Measurement-Oriented Model-based Methods; Simplified measurement processes; FSM conversion; UML models; Object oriented measurement.

Contents

This page intentionally left blank.

List of Figures

List of Tables

Chapter 1 Introduction

Measurement is a basic activity in everyday life, since it is necessary for understanding the objects and the activities of interest. In every scientific and technical discipline, especially in the engineering filed, measurement is essential, sometimes it is at the very core of development activities. In software engineering, software measurement has become a key aspect of good software management and engineering practices.

1.1 Functional Size Measurement

Software metrics can be classified into three categories: product metrics, process metrics, and project (resource) metrics. Function Size Measurement (FSM) is a product metrics, which characterizes the size of a software application. Functional size measures are often used in conjunction with metrics addressing complexity, design features, quality, etc. For instance, the number of faults found in a software product is hardly meaningful in itself, while the number of faults divided by the size provides a fault density indication, which is a clear indicator o software quality.

However, the main role of FSM in software development consists in providing the input data required by effort estimate models and tools. In general, FSM helps improving the software development process, predicting faults and fault-prone software units, allocating resources during the development, and checking requirements completeness. In conclusion, function size measurement is an essential component of software development.

1.1.1 Software Size Measurement in the old days: Lines of Code

The oldest and most commonly used measure of software products is Lines of Code (LoC), sometimes named Source Lines of Code (SLoC) or Delivered Source Instruction (DSI). There are two major types of LoC measures: physical LoC and logical LoC (also known as “effective LoC”). The former is defined to count lines in the text of the program's source code including comment lines; the latter attempts to measure the number of executable statements (thus excluding comments, blank lines and often also lines containing only syntactic elements). This measurement was first introduced in the 60s and was used for economic, productivity, and quality studies.[22]

A measure in LoC has the problem that the same function generally requires a different number of LoC when coded with different language. For example in Figure 1, the same function programmed with a basic assembly language needs 3 lines of code, while it needs only one line when programmed with COBOL.

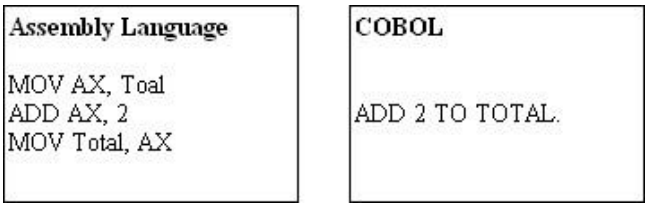

Figure 1 Comparison of two programming languages coding same function

Even when the same programming language is used, different developers can produce implementations of the same function having different sizes in LoC. In general, the size in LoC depends on the technology, programming language, and programmers' attitudes. Given the above observations, it is easy to conclude that the LoC measure focuses only on the "physical" dimension of software, and does not represent the net functionality provided by software.

1.1.2 Functional Size Measurement

The Function Point method was originally introduced by Albrecht [8] to overcome the limits of LoC measurement. The basic idea is that measuring software size is not carried out in the term of its physical component (LoC), but in terms of its 'functionality'. The viewpoint of measuring software size was changed from the internal attribute to the external functional viewpoint of the end-user. The user functions requested and recognised by the user are defined in Function User Requirements (FURs) that describe what the software should do to fulfil user's needs.

This idea makes software size measurement independent from technology, programming language and programmer's attitudes. Functional Software size measurement can also be started earlier in the software development lifecycle.

1.1.3 Functional Size Measurement methods

In the field of functional size measurement, many methods have been proposed, including IFPUG FPA [10], NESMA FPA [17], Mark II FPA [36], FiSMA [113] and COSMIC [33]. Among them, we focus only on IFPUG FPA and COSMIC.

IFPUG FPA

The Functional Point method was originally introduced by Albrecht to measure data-processing systems by quantifying the functionality the software provides to the user, from the information view, by quantifying the volume of data flow and the storage[8][9].

The basic idea of FPA is that the ''amount of functionality'' released to the user can be evaluated by taking into account the data used by the application to provide the required functions, and the transactions (i.e., operations that involve data crossing the boundaries of the application) through which the functionality is delivered to the user. Data are user identifiable groups of logically related data, and are classified as Internal Logical Files (ILF) or External Interface Files (EIF). A transaction is a set of actions seen as one cohesive unit of work. FPA differentiates three types of transactions: External Input (EI), External Output (EO), and External Inquiry (EQ).

The size of each data function depends on the function type and contents; the size of each transaction depends on the number of data files used and the amount of data exchanged with the external.

The sum of the sizes of data and transactions is the size of the application in Unadjusted Function Points (UFP). Based on 14 general system characteristics the value adjustment factor (VAF) is computed; the "adjusted" size of the application is obtained by multiplying the size in UFP by the VAF. The adjusted size was introduced to improve

the correlation of the size in FP to the development effort. However, adjustment is generally not considered a sound practice. Accordingly, UFP have been recognized as an ISO standard, while adjusted FP did not. Accordingly, in this Thesis, only UFP are considered.

COMSIC Function Points

The COSMIC method measures the functional size of a piece of software based on its functional user requirements, which are broken down into a number of functional processes, which are independently executable sets of elementary actions that the software should perform in response to a triggering event. The elementary actions that software can perform are either data movements or data manipulations. As a reasonable approximation, COSMIC assumes that each data movement has an associated constant average amount of data manipulation. Accordingly, in the COSMIC model of software FUR are broken down into a number of functional processes, which in turn involve only of data movements.

Data movements are the basic functional components that are used for establishing the size of the software. A data movement moves a unique data group, i.e., a set of data attributes (each attribute describes a complementary aspect of an object of interest a thing or concept about which the software is required to store and/or process data).

The COSMIC method distinguishes four different types of data movements, namely Entry, Write, Read, and Exit. Writes and Reads move a data group to and from persistent storage, respectively. An Entry moves a data group into the software from a functional user and an Exit moves a data group out. The size of a piece of software is then defined as the total number of data movements (Entries, Exits, Reads, and Writes) summed over all functional processes of the piece of software.

1.1.4 Benefits and limits of Functional Size Measurement

Functional size measurement has a long history and its effectiveness make it very popular, so many measurement procedures have arisen to support it. [23]

Functional size measurement is used for two main purposes: to help estimating the effort of a development or maintenance projects or measuring the actual productivity of a finished development endeavour. Several studies have highlighted pros and cons of FSM, as described below.

Estimate
It has been shown that the functional size of a software application is highly correlated with the amount of work needed to develop the application. So, functional size is the input of several software development estimation models and tools.

Requirements understanding and Completeness Checks
Understanding user functional requirements and evaluating whether requirements are sufficiently complete before beginning design and coding is most relevant and tough problem. The functional size measurement is helpful to deal with this problem.[24][25][26]

Excellent way to excellent software product
Function size measurement is an excellent instrument to identify potential problems and to improve the development process; it is also a powerful tool for managing the software development process, since function points can be used as an indicator of requirements creep and quality.[27][28][29][30]

Early measurement
Functional size measurement can be applied early in the software development life cycle, namely as soon as FUR are available, while the size in LoC can be measured only after the conclusion of development. In addition, FSM can also be used in the phases of the software development lifecycle following requirement specification (e.g., design, coding, etc.)

Failure to capture the Non-functional requirements
In the literature three types of software requirements are mentioned: functional user requirements, non functional user requirements, and technical requirements [23]. FSM only aims at measuring the functional user requirements: non-functional properties and technical requirements are not taken into account.
Complementary software metrics can be defined and used along with function points to measure also other aspects of the software that FPA does not consider.[31]

Failure to capture the "amount of elaboration"
Most FSM methods proposed until now (including FPA and COSMIC) fail to capture the "amount of elaboration" required. The consequence is that two applications that differ only in the amount of elaboration required are considered of the same size, even though in general the more elaboration intensive application is bound to require more effort to be developed. An example of this problem is mentioned in [21], where the incapacity of FP to capture the amount of elaboration leads to underestimating both the physical size (in LoC) and the development effort of the considered software application [22]. An exception is represented by Mark II FP, which to some extent take into account the amount of elaboration performed by software.[36]

1.1.5 Simplified FSM methods

The measurement of Function Points can be expensive and time consuming. The measurement process involves (among others) the following activities:
− Identifying logic data;
− Identifying elementary processes;
− Classifying logic data as internal logic files (ILF) or external interface files (EIF);
− Classifying elementary processes as external inputs (EI), outputs (EO), or queries (EQ);
− Weighting data functions;
− Weighting transaction functions.

Simplified measurement processes allow measurers to skip –possibly in part– one or more of the aforementioned activities, thus making the measurement process faster and cheaper.

Early & Quick Function Point

The best-known approach to simplifying the process of FP counting is probably the Early & Quick Function Points (E&QFP) method [16]. E&QFP descends from the consideration that estimates are sometimes needed before the analysis of requirements is complete, when the information on the software to be measured is partial or not sufficiently detailed.

Since several details for performing a correct measurement following the rules of the FP manual [10] are not used in E&QFP, the result is a less accurate measure. The trade-off between reduced measurement time and costs is a reason for adopting the E&QFP method even when full specifications are available, but there is the need for completing the measurement in a short time, or at a lower cost. An advantage of the method is that different parts of the system can be measured at different detail levels: for instance, a part of the system can be measured following the IFPUG manual rules [10] [11], while other parts can be measured on the basis of coarser-grained information. In fact, the E&QFP method is based on the classification of the processes and data of an application according to a hierarchy (see Figure 2. (from [16])).

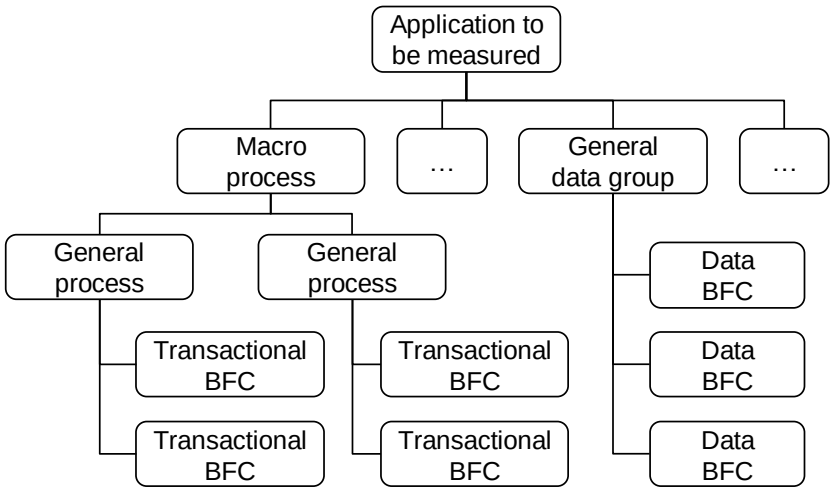

Figure 2 Functional hierarchy in the Early & Quick FP technique (from [16])

Transactional Base Functional Components (BFC) and Data BFC correspond to IFPUG's elementary processes and LogicData, while the other elements are aggregations of processes or data groups. The idea is that if you have enough information at the most detailed level, you count FP according to IFPUG rules; otherwise, you can estimate the size of larger elements (e.g., General or Macro processes) either on the basis of analogy (e.g., a given General process is "similar" to a known one) or according to structured aggregation (e.g., a General process is composed of 3 Transactional BFC). By considering elements that are coarser-grained than the BFC of Functional Point Analysis, the E&QFP measurement process leads to an approximate measure of size in IFPUG FP.

In the E&QFP manual[16], some tables taking into account the previous experiences with the usage of E&QFP are provided to facilitate the task of assigning a minimum, maximum and most likely quantitative size to each component. For instance, Table 1 provides minimum, maximum and most likely weight values for generic (i.e., not weighted) functions as given in [16]. The time and effort required by the weighting phases are thus saved. Such saving can be relevant, since weighting requires analyzing every data or transaction function in detail.

Table 1 E&QFP: Function type weights for generic functions

Function type	Weight		
	Low	Likely	High
Generic ILF	7.4	7.7	8.1
Generic EIF	5.2	5.4	5.7
Generic EI	4.0	4.2	4.4
Generic EO	4.9	5.2	5.4
Generic EQ	3.7	3.9	4.1

The size of unspecified generic processes (i.e., transactions that have not been yet classified as inputs, outputs or queries) and unspecified generic data groups (i.e., logical files that have not been yet classified as ILF or EIF) as given in [16] are illustrated in Table 2. When using this method, only the identification of logical data and elementary processes needs to be done: both the classification of data and transaction functions and their weighting are skipped. Consequently, sizing based on unspecified generic processes and data groups is even more convenient –in terms of time and effort spent– than sizing based on generic (i.e., non weighted) functions.

Table 2 E&QFP: Function type weights for unspecified generic processes and data group

Function type	Weight		
	Low	Likely	High
Unspecified Generic Data Function	6.4	7.0	7.8
Unspecified Generic Processes Function	4.3	4.6	4.8

NESMA indicative and estimated methods

The Indicative NESMA method [17] simplifies the process by only requiring the identification of LogicData from a conceptual data model. The Function Point size is then computed by applying the following formulae, whose parameters depend on whether the data model is normalized in 3rd normal form:

Non normalized model:

$$\text{Function Points} = \#^{1}\text{ILF} \times 35 + \#\text{EIF} \times 15 \quad (1)$$

Normalized model:

$$\text{Function Points} = \#\text{ILF} \times 25 + \#\text{EIF} \times 10 \quad (2)$$

The process of applying the NESMA indicative method involves only identifying logic data and classifying them as ILF or EIF. Accordingly, it requires less time and effort than the E&QFP methods described above, in general. However, the Indicative NESMA method is quite rough in its computation: the official NESMA counting manual specifies that errors in functional size with this approach can be up to 50%.

The Estimated NESMA method requires the identification and classification of all data and transaction functions, but does not require the assessment of the complexity of each

[1] Here and hereafter # represents "Number of…"

function: Data Functions (ILF and EIF) are all assumed to be of low complexity, while Transactions Functions (EI, EQ and EO) are all assumed to be of average complexity.

Tichenor method
The Tichenor ILF Model [15] bases the estimation of the size on the number of ILF via the following formula for transactional system (for batch systems, Tichenor proposes a smaller multiplier):

$$UFP = \#ILF \times 14.93 \quad (3)$$

This model assumes a distribution of BFC with respect to ILF as follows: EI/ILF = 0.33, EO/ILF = 0.39, EQ/ILF = 0.01, EIF/ILF = 0.1. If the considered application features a different distribution, the estimation can be inaccurate.

The fact that a method based only on ILF requires a given distribution for the other BFC is not surprising. In fact, the size of the application depends on how many transactions are needed to elaborate those data, and the number of transaction cannot be guessed only on the basis of the number of ILF, as it depends on the number of ILF just very loosely. Instead of allowing the user to specify the number of transactions that are needed, the Tichenor method practically imposes that the number of transactions complies with the distribution given above.

ISBSG distribution model
The analysis of the ISBSG dataset yielded the following distribution of BFC contributions to the size in FP:

ILF 22.3%, EIF 3.8%, EI 37.2%, EO 23.5%, EQ 13.2%

The analysis of the ISBSG dataset also shows that the average size of ILF is 7.4 UFP. It is thus possible to compute the estimated size on the basis of the number of ILF as follows:

$$UFP = (\#ILF \times 7.4) \times 100 / 22.3 \quad (4)$$

The same considerations reported above for the Tichenor model apply. If the application to be measured does not fit the distribution assumed by the ISBSG distribution model, it is likely that the estimation will be inaccurate.

Simplified FP
The simplified FP (sFP) approach assumes that all BFC are of average complexity [18], thus:

$$UFP = \#EI \times 4 + \#EO \times 5 + \#EQ \times 4 + \#ILF \times 10 + \#EIF \times 7 \quad (5)$$

ISBSG average weights
This model is based on the average weights for each BFC, as resulting from the analysis of the ISBSG dataset [19], which contains data from a few thousand projects. Accordingly, the
ISBSG average weights model suggests that the average function complexity is used for each BFC, thus

$$UFP = \#EI \times 4.3 + \#EO \times 5.4 + \#EQ \times 3.8 + \#ILF \times 7.4 + \#EIF \times 5.5 \quad (6)$$

Table 3 provides a quick overview of the activities required by FP measurement and estimation methods. Of course, the IFPUG method requires all the activities listed in Table 3, while simplified methods require a subset of such activities.

Table 3 Activities required by different simplified measurement process

Measurement activities	FPA	NESMA indic.	NESMA estin.	E&QFP Generic func.	E&QFP Unspec. Generic func.	Tichenor ILF Model	ISBSG distribution	sFP	ISBSG average weights
Identifying logic data	√	√	√	√	√	√	√	√	√
Identifying elementary processes	√		√	√	√			√	√
Classifying logic data as ILF or EIF	√	√	√	√		√	√	√	√
Classifying elementary processes as EI, EO, or EQ	√		√	√				√	√
Weighting data functions	√								
Weighting transaction functions	√								

This page intentionally left blank.

1.2 Problems of Functional Size Measurement addressed in this thesis

1.2.1 Problems, limits, and challenges of FSM

It's well known that Function Point Measurement suffers from several problems, such as:

1. Both data and transaction functions' sizes have upper limits. For instance, no External Input has size greater than 5 FP, even if it is "very very big".
2. Function points are not well formed metrics because their constituent elements are correlated.[32]
3. Function point counts are expected to be obtained early in the development cycle. Unfortunately, since the measurement requires too much detailed information, measurement is often not achievable a very early phase of development.
4. The measurement criteria and procedure are not defined in a thoroughly precise way. Accordingly, the FPA counting involves judgment on the part of the counter, it requires human interpretation. The same product is usually sized differently by different counters, even within the same organization.
5. Function point counts are considered not equally applicable to all kinds of software. They have not enjoyed widespread success in embedded systems or heavily computational applications.
6. Counting often requires a relevant effort to analyze several heterogeneous requirements documents in order to identify BFCs (Basic Functional Components). In fact, the identified BFCs are often not easy to trace back to elements of the requirements. Moreover, the effort done to understand the requirements is not exploited to build any artefacts that can be useful in the design and implementation phases.
7. Lack of formal language description for measurement process. Usually the analyst that defined the requirements and the measurer are two different persons, who perform separate tasks. The lack of the formal language description for measurement process makes it hard to assure that the right functionalities are measured, and that they are measured correctly. [3]
8. After the COSMIC method has been proposed, the issue of convertibility between traditional FP (Function Points) and CFP (COSMIC Function Points) has arisen. The organizations that have historical data in FP and wish to adopt the COSMIC method face the problem of converting data from FP into CFP. FSM conversion is in itself a quite difficult problem.
9. Although the methods are technology independent, their use in object oriented development is quite difficult. Each method uses its own abstraction to represent a software system in a convenient way, so as to perform size count.
10. In COSMIC, the data movements of a software component also contain the data manipulation of the software component. This is strength of the method, since you can obtain the size measure based on data movements alone. On the other hand, this is also one of the limitations with the COSMIC method, and the consequence is that the method does not capture complex calculations, or treatment of large amounts of data.

1.2.2 Problem Analysis

In subsection 1.2.1, the main problems of FSM were summarized. In this subsection the problems addressed in this thesis are highlighted.

Table 4 Analysis of the problems, challenges and the problems addressed in this thesis

Problems ID	Summarization	FPA	CFP	Addressed In this thesis
1	Function point of FPA has upper size limit	√		
2	Correlation exists among actors of BFC	√	√	
3	Difficult to use in early phase	√	√	√
4	Criteria not precisely described	√	√	
5	application scope	√	√	
6	difficult to identify and obtain BFCs	√	√	√
7	Measurement process not precisely described	√	√	√
8	Conversion between FPA e CFP is not easy	√	√	√
9	Does not capture complex calculation or treatment or large amounts of data		√	
10	Difficult to use in OO methods	√	√	√
11	Difficult to use for new product and process form	√	√	
12	Does not involve quality or technical aspect	√	√	

The problems addressed in this thesis are 3, 6, 7, 8, and 10.

1.3 Research objectives

1.3.1 Research objectives

The goal of this thesis is to identify and test an easily usable method for functional size measurement in practice.

Numerous experts conducted research on this topic. For example, the already mentioned E&QFP [16] and NESMA [17] methods addressed the simplification FSM through

analogy-based classification and structured aggregation; the Tichenor, ISBSG, and ISBSG average weights methods exploited statistical analysis to avoid the most expensive phases of FSM.[4]

1.3.2 Research methods

A few strategies are possible to tackle the complexity of FSM. Among them:

1. Supporting and easing the standard measuring processes. The FSM processes described in official manuals are not changed. Instead, the activities required by the process are made easier. Ideally, one could think of totally automating the FSM process; however, the partial subjectivity of all FSM methods makes automation extremely hard. An interesting alternative consists in moving the complexity of the process from the actual measurement phase to the requirements modelling phase, and making measurement a sort of "by product" of requirements specification. This is reasonable, because requirements have to be specified anyway, even if one does not intend to measure their functional size. However, to pursue this approach, i.e., measurement based on requirements models, it is necessary to build models so that they actually contain the information required by the FSM method of choice: requirement model building must be done in a measurement oriented way [2][3].
2. Simplify the FSM process, while preserving the definition of the measure. This strategy consists in simplifying the measurement process by skipping or downgrading some of the more expensive and lengthy activities involved in the measurement process (see section 1.1.5). The result is an approximation of the real measure in FP or CFP.
3. Simplify the definition of the measure, which results in simplifying the measurement process, while preserving a clear compatibility between the full-fledged and the simplified measures. So, the latter a) are substantially equivalent to standard FP or CFP, and b) are easier to measure, e.g., because they are based on a smaller amount of information [4].

This thesis concentrates on the above point 2. In fact points 1 and 3 had already been partly explored when the PhD work reported here started.

As to point 2, the work reported here addresses the following main points:

- Evaluating current proposals.[1][5][6]
- Exploring the possibility of establishing statistical correlations between functional size measures and measures of object-oriented (UML) models.
- Exploring the analytical convertibility of functional size measures [7]. This activity is loosely correlated with the main topic, but is justified in that obtaining a measure as a conversion of another measure is much easier than performing the measurement.

1.4 Thesis structure

The remainder of this thesis is organized into the following chapters:

- Chapter 2. Functional Size Measurement Methods. In this chapter the methods of FSM, FPUG FPA and COMSIC FP are introduced and compared.
- Chapter 3. Simplified Functional Size Measurement. This chapter presents the Simplified Functional Size Measurement methods, namely E&QFP, NESMA and the other existing methods.

- Chapter 4. Model-based measurement. This chapter is about the Model-based measurement-oriented method proposed by Lavazza et al. We first talk about the fundamentals of the method; then the modeling method and the procedure of modeling are separately presented according to IFPUG FPA and COSMIC; at the end of this chapter, we also compare both methods, IFPUG FPA and COSMIC FFP, from the view point of the measurement-oriented model-based method.
- Chapter 5. Evaluation of Simplified FSM processes. This chapter empirically assesses and justifies the Simplified FSM proposals. The Model-based Simplified Functional Size Measurement methods are also empirically evaluated.
- Chapter 6. Conversion between FPA and CFP. In this chapter the analytical conversion between FPA and CFP is discussed and evaluated.
- Chapter 7. Investigation of statistical correlations between FSM and Object-Oriented Measures of Requirements models. In this chapter we discuss and investigate the statistical correlation between FSM and Object-Oriented Measures of UML Requirements models.
- Chapter 8. Related work. This chapter contains a review of the state-of-the-art about the simplified FSM.
- Chapter 9. Conclusion and future work. This chapter presents the main contributions of this thesis and the plan for future work.
- Appendix. Glossary

The structure of this thesis is illustrated in Figure 3

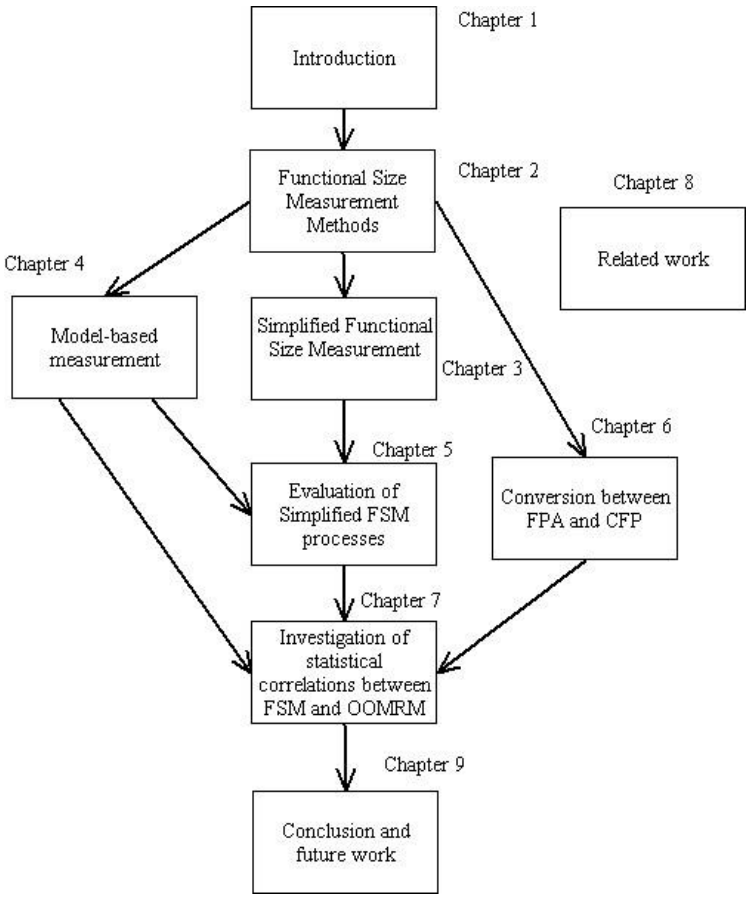

Figure 3 The structure of the thesis

Chapter 2 Functional Size Measurement Methods

In this chapter two main FSM methods, FPA and COMSIC, are introduced and compared. Before that, we also explored the methodology abut the FSM.

2.1 Methodology

The exploration starts by quoting a sentence by Albrecht [8], which describes the beginning of the first generation of FSM.

The basis for this method was developed over the last 5 years from the Data Processing Service projects estimating experience. As part of that estimating we validated each estimate with a series of weighted questions about the application function and the development environment. We found that ***the basic value of the application function was consistently proportional to a weighted count of the number of external use inputs, outputs, inquires and master files***."

Another sentence provides a clear insight into FSM [9]: ***"The thesis of this work is that the amount of function to be provided by the application can be estimated from the itemisation (itemization) of major components of data to be used or provided by it. …"***

From these two quotations, the following observation can be easily derived: First, the FSM method is based on engineering practical experience. "*… in the last 5 years from the DP Service projects estimating experience…"*; second, based on the datasets accumulated in these engineering practices, a consistent correlation exists. "*…a consistently proportional to…"*; Third, since a consistent correlation exists, naturally, the objects involved in this correlation should be clearly and firstly identified. *"....****the basic value of the application function… a weighted count of the number of external use inputs, outputs, inquires and master files****."*

Figure 4 is used to explain the core part of the FSM methods (FPA and COSMIC) and the other simplified methods in this chapter and the next chapter. Through the comparison of their factors, the key difference between the traditional methods and the simplified methods is discovered.

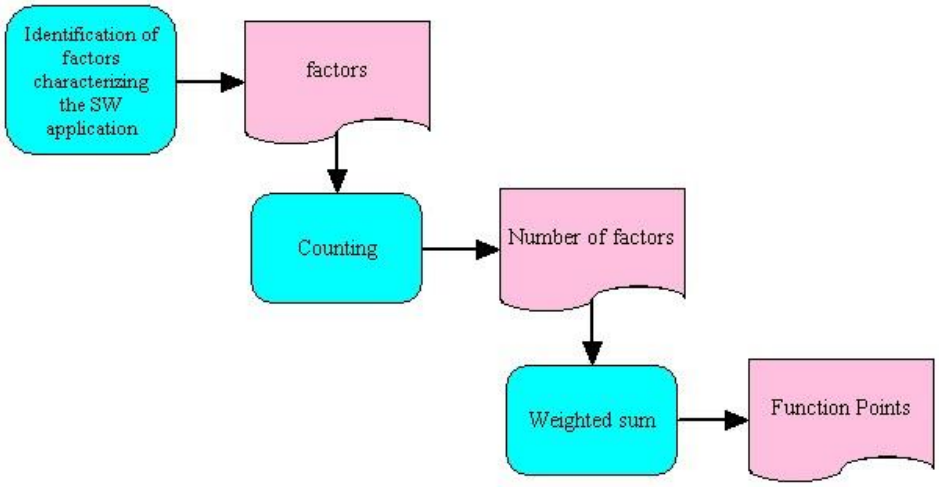

Figure 4 High level abstract model of FSM methodology

Here, the 3 "things", namely, factors, counting, and weighting are defined. The Factors represent the basic elements of a software application taken into account by a FSM method. For example, in the FPA method, factors are the elementary process, the DETs, RETs, FTRs, the types of transaction functions and data functions. In a word, factors represent the elements that characterize each function of an application to be measured and the rules to identify them; Counting represents how to count the above basic factors; Weighting represents the relation between the final function point and the above basic elements.

While the Factors and Counting are always present, some methods do not include Weighting: for instance the COSMIC method does not involve any weighting of elements, just counting.

2.2 IFPUG FPA

2.2.1 The brief history about IFPUG FPA

Function Point Analysis was developed first by Allan J. Albrecht in the mid 70s. It was an attempt to overcome difficulties associated with LoC as a measure of software size, and to measure the size of a data-processing system from the end-user's point of view, in order to estimate the development effort, i.e. to assist in developing a mechanism to predict effort associated with software development. The method was first published in 1979 [8], then Albrecht refined the method in 1983-84 [9] [37]. Since 1986, when the International Function Point User Group (IFPUG) was set up, several versions of the Function Point Counting Practices Manual have been published by IFPUG.

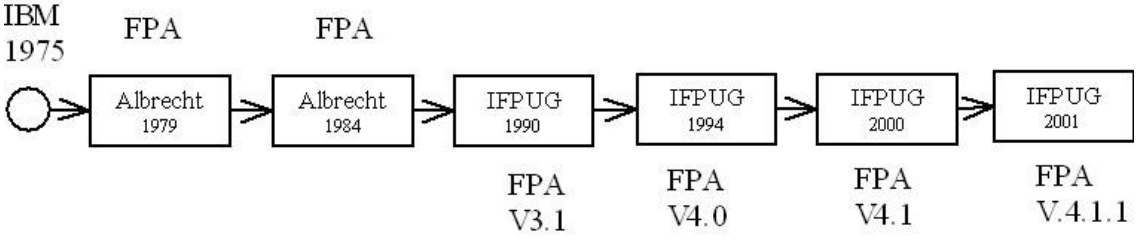

Figure 5 Evolution of FPA method

The current version of the IFPUG Manual is version 4.1.1. The IFPUG counting manual is now an ISO standard in its "unadjusted" version.

2.2.2 The basic principles of FPA

The basic idea of FPA is that the "amount of functionality" released to the user can be evaluated by taking into account the data used by the application to provide the required functions, and the transactions (or processes) through which the functionality is delivered to the user. Figure 6 illustrates the schematic view of FPA, where the "Factors" that characterize the software application (as defined in section 2.1) are highlighted. In FPA jargon, these factors are named "Based Functional Components" (BFC).

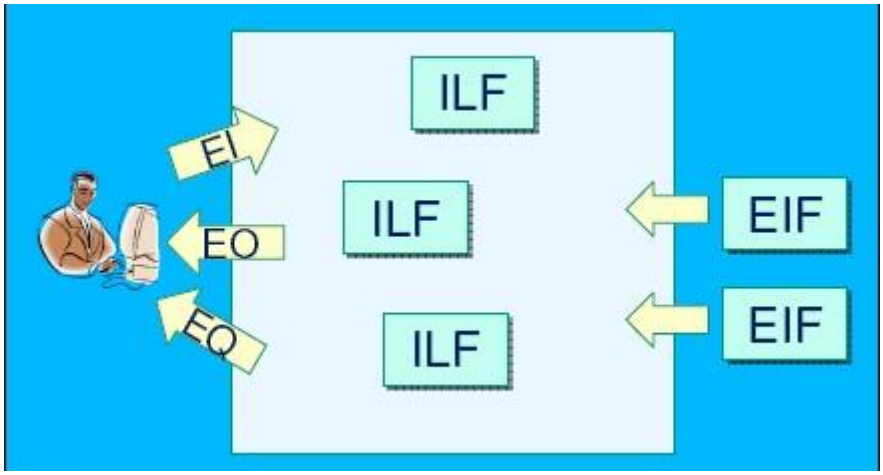

Figure 6 Schematic view of FPA base functional components

2.2.3 Basic functional components

Data functions: ILF and FIF

Data functions represent data that are relevant to the user and are required to perform some function, and are classified into internal logical files (ILF), and external interface files (EIF).

An ILF is a user identifiable group of logically related information managed within the boundary of the application. Its primary intent is to hold data maintained through one or more elementary processes of the application being counted.

An EIF is similar to an ILF, but is maintained within the boundary of another application, i.e., it is outside the application being measured, for which an EIF is read-only.

The term "file" in the FPA does not indicate a file in the traditional. In FPA, it refers to a logically related group of data and not the physical implementation of those groups of data.

ILF and EIF are characterized on the basis of their Record Element Types (RET) and Data Element Types (DET). A RET is a user recognizable subgroup of data elements within an ILF or EIF. A DET is a unique user recognizable, non-repeated field (non-repeated means that if the same filed appears multiple times in a RET, it counted only once).

In Figure 7, we present the FPA meta-model, which illustrates the information we need to identify and capture for representing a software system to be measured.

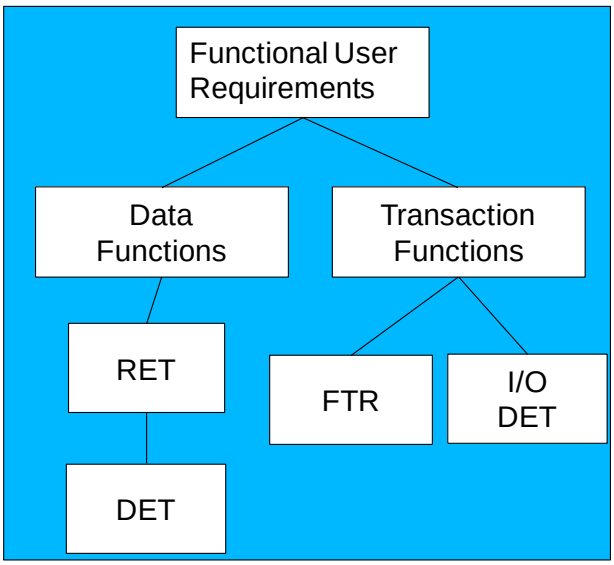

Figure 7 FPA software model

From the definitions of the data elements, we can draw Figure 8 to display the relative conceptual granularity among them.

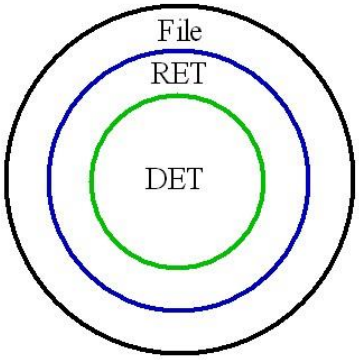

Figure 8 Relative conceptual granularities of FPA data elements

Transaction Functions

Transaction functions represent operations that are relevant to the user and cause input and/or output data to cross the application boundary. Transaction functions represent elementary processes. An elementary process is the smallest unit of activity that is

meaningful to the user(s). An elementary process must be self-contained and leave the state of the application being counted in a consistent state.

Transactional functions are classified into external inputs (EI), external outputs (EO), and external inquiries (EQ) according to the main intent of the process: updating ILF for EI, computing and outputting results for EO, retrieving and outputting data for EQ.

- External Inputs: An external input (EI) is an elementary process that processes data or control information that comes from outside the application boundary. The primary intent of an EI is to maintain one or more ILFs and/or to alter the behavior of the system.
- External Outputs: An external output (EO) is an elementary process that sends data or control information outside the application boundary. The primary intent of an external output is to present information to a user through processing logic other than, or in addition to, the retrieval of data or control information. The processing logic must contain at least one mathematical formula or calculation, create derived data, maintain one or more ILFs or alter the behavior of the system.
- External Inquiry: An external inquiry (EQ) is an elementary process that sends data or control information outside the application boundary. The primary intent of an external inquiry is to present information to a user through the retrieval of data or control information from an ILF of EIF. The processing logic contains no mathematical formulas or calculations, and creates no derived data. No ILF is maintained during the processing, nor is the behavior of the system altered.

The main difference between the transactional function types is their primary intent. They are characterized on the basis of their file type referenced (FTRs) and data element type (I/O DET). A FTR is an internal logical file read or maintained by a transactional function or an external interface file read by a transactional function. An I/O DET is a unique user recognizable, non-repeated field which flows through the boundaries of the application being measured.

2.2.4 Measurement procedure

Although the measurement is essential for cost estimation, it is very important to the management of software development as mentioned in the subsection 1.2.1. But it is too difficult in practice, too boring during the measurement, and too costly to carry out. The strict, integrated and official measurement procedure of FPA is illustrated in the Figure 9.

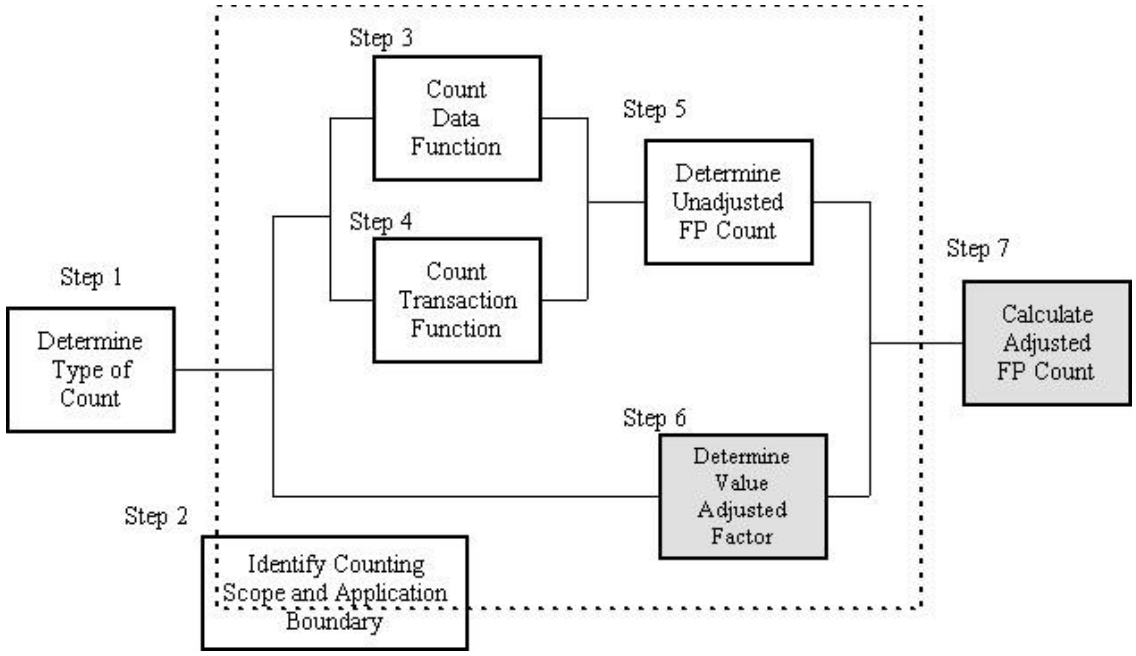

Figure 9 Procedure of the FPA measurement

Type of function point count

The function point method is suitable to be used in three types of projects, namely development project, enhancement project, and application. So, the first step in the function point count procedure is to determine the type of function point count. In this thesis, only development project measurement is taken into account (although it is expected that the presented results can be extended to other types of measurements rather easily)

Identify the Counting Scope and Application Boundary

The application boundary indicates the border between the software being measured and the user. The boundary is identified according to the counting scope, which defines the functionality that will be included in a particular function point count.

Determine the Unadjusted Function Point Count (step 3, 4, 5)

The application's specific user functionality is evaluated in terms of what is delivered by the application, not how it is delivered. Since the basic idea of FPA is that the ‘‘amount of functionality’’ released to the user and the unadjusted function point count (UFPC) reflect the specific countable functionality provided to the user by the project or application, only user-requested and defined components are counted. In this thesis, only unadjusted FP is considered. This is coherent with the usage of function points purely as a size measure.

Count Data Functions (step 3)

For counting data function, in practice, we must first identify the ILFs and EIFs from the software artefacts. For each ILF and EIF, the RETs and DETs must be further

identified and the numbers of RETs and DETs must be counted. This is the process of counting data function of an application to be measured.

How to identify and count them? Certainly the FPA manual defines the counting process and the related concepts. But the FPA counting process does not make reference to a rigorous representation or model of the application to be measured.

Count Transactional Functions (step 4)

For counting transactional functions, the first thing is to define and identify all the elementary processes that represent the functionality provided to the user to process data. The type of each transaction function is needed to be classified into External Input, External Output, or External Inquiry. Although classifying is not always easy to carry out, there is a table to be referenced. An FTR can be an ILF referenced or maintained by the transaction or an EIF read by the transaction. The DETs are considered to be those that cross the application boundary when the transaction is performed.

The size of a software application is given by the sum of the sizes of its data and transaction functions.

Weight data and transaction functions (step 5)

Each data function is sized according to its "complexity".

The complexity of a data functions depends on its type (ILF or EIF), and the number of DETs and RETs it includes, as specified in Table 5.

The complexity of the data function and its type determine the size in UFP of the function; for instance, an ILF having 3 RETs and 25 DETs is classified into average complexity and contributes 10 UFP.

Table 5 FPA reference table (the part of ILF and EIF)

Function Type	Weight					
	Low		Average		Hight	
ILF	7		10		15	
EIF	5		7		10	
	[0, 1]	[1,50]	[0,1]	[51,∞)		
	[2,5]	[1,19]	[2,5]	[20,50]	[2,5]	[51, ∞)
			[6, ∞)	[1,19]	[6, ∞)	[20, ∞)
	N_{RETs}	N_{DETs}	N_{RETs}	N_{DETs}	N_{RETs}	N_{DETs}

Transaction functions are sized in a similar way. Their complexity (as specified in Table 6) depends on the File Type Referenced (i.e., the number of ILF and EIF potentially accessed during the execution of the transaction), the number of DETs that cross the boundary of the application, and the type (input, output or inquiry) of the transaction.

Table 6 FPA reference table (the part of EI, EO, and EQ)

Function Type	Weight		
	Low	Average	Hight
EI	3	4	6

	[0, 1]	[1,15]	[0, 1]	[15,∞)		
	2	[1,4]	2	[5,15]	2	[16, ∞)
			[3, ∞)	[1,4]	[3, ∞)	[5, ∞)
EO	4		5		7	
	[0, 1]	[1,19]	[0, 1]	[29, ∞)		
	[2, 3]	[1,5]	[2, 3]	[6, 19]	[2, 3]	[20, ∞)
			[4, ∞)	[1,5]	[4, ∞)	(6, ∞)
EQ	3		4		6	
	[0, 1]	[1,19]	[0, 1]	[20, ∞)		
	[2, 3]	[1,5]	[2, 3]	[6, 19]	[2, 3]	[20, ∞)
			[4, ∞)	[1,5]	[4, ∞)	[6, ∞)
	N_{FTRs}	N_{DETs}	N_{FTRs}	N_{DETs}	N_{FTRs}	N_{DETs}

The UFP value of an application

The weighted sum of transaction functions and data functions is the size of the application in unadjusted function points.

a) Albrecht found that the development effort depends not only on the functional size of an application, but also on several other factors. Accordingly he devises to "adjust" the size measurement so that it takes into account all the factors that affect effort;
b) It is a better predictor of development effort via equations of type Effort = K × Size.

Determining Value Adjustment Factor and calculating the FP
The size in FP is calculated using a specific adjustment formula.

$$FP = UFP \times VAF \quad (7)$$

The value adjustment factor (VAF) takes into account 14 characteristics of the application to be measured and is calculated as follows:

$$VAF = 0.65 + [\ (\sum_{i=1}^{14} 14C_i)/100] \quad \textbf{(8)}$$

Where: C_i is the degree of influence of the i^{th} General System Characteristic

In (8), each C_i represents the degree of influence of one General System Characteristic (GSC). GSC is evaluated at six scales (from zero to five) according to its degree of influence in the given application. The GSCs are listed in the following table.

Table 7 14 General System Characteristics (GSC)

Ord.	General System Characteristic	Brief Description
1	Data communications	How many communication facilities are there to aid in the transfer or exchange of information with the application or system?
2	Distributed data processing	How are distributed data and processing functions handled?
3	Performance	Did the user require response time or throughput?
4	Heavily used configuration	How heavily used is the current hardware platform where the application will be executed?

5	Transaction rate	How much frequently are transactions executed daily, weekly, monthly, etc.?
6	On-Line data entry	What percentage of the information is entered On-Line?
7	End-user efficiency	Was the application designed for end-user efficiency?
8	On-Line update	How many ILF's are updated by On-Line transaction?
9	Complex processing	Does the application have extensive logical or mathematical processing?
10	Reusability	Was the application developed to meet one or many user's needs?
11	Installation ease	How difficult is conversion and installation?
12	Operational ease	How effective and/or automated are start-up, back up, and recovery procedures?
13	Multiple sites	Was the application specifically designed, developed, and supported to be installed at multiple sites for multiple organizations?
14	Facilitate change	Was the application specifically designed, developed, and supported to facilitate change?

Table 8 Degrees of influence of the GSCs

Or.	Degrees of influence	Scale Value
0	Not present, or no influence	0
1	Incidental influence	1
3	Moderate influence	2
4	Average influence	3
5	Significant influence	4
6	Strong influence throughout	5

Brief summary

The FPA measures functional user requirements, but a requirement specification is a structured document, which is often written in natural languages. It is very difficult to identify the BFCs and to count them. The standard method for counting function points is to count the BFCs, adjustments, and weighting factors for several kinds of complexity. The process is long and expensive. In principle the FPA is ideal to measure the software size, and then to estimate the effort of development. But the drawback is that although it can be used for effort estimation, FPA itself needs more effort to be done. According to literature [65], a certified counter can only count 400 -600 function points using the normal function point analysis per day. The cost of counting every point runs $6.00.

It is obvious that, concerning the use in practice, if a measurement method would be more complex and expensive than the customer can accept and afford, let us say, such a method would have relatively low value for the project management and control during the cycle-life of development. We surely need the simplified method in practice.

2.3 COSMIC

2.3.1 Brief story about COSMIC

The COSMIC method developed by the Common Software Measurement International Consortium (COSMIC) has emerged as the second generation of the FSM methods. It is a recognized international standard (ISO 19761 [35]). It aimed at addressing some of the major weaknesses of earlier methods, like FPA.

In 1996, the industry sponsored the development of an IFPUG extension for real-time and embedded software, which was put into the public domain under the name of FFP (Full Function Points). Then, the COSMIC -formed in 1998- after reviewing existing methods (IFPUG, Mark II, NESMA and Full Function Point 1.0), published version 2.0 of COSMIC-FFP in 1999. Extensive field trials were carried out in 2000 and 2001. COSMIC published the latest definition of the method (Version 3.0.1) [33], in May 2009.

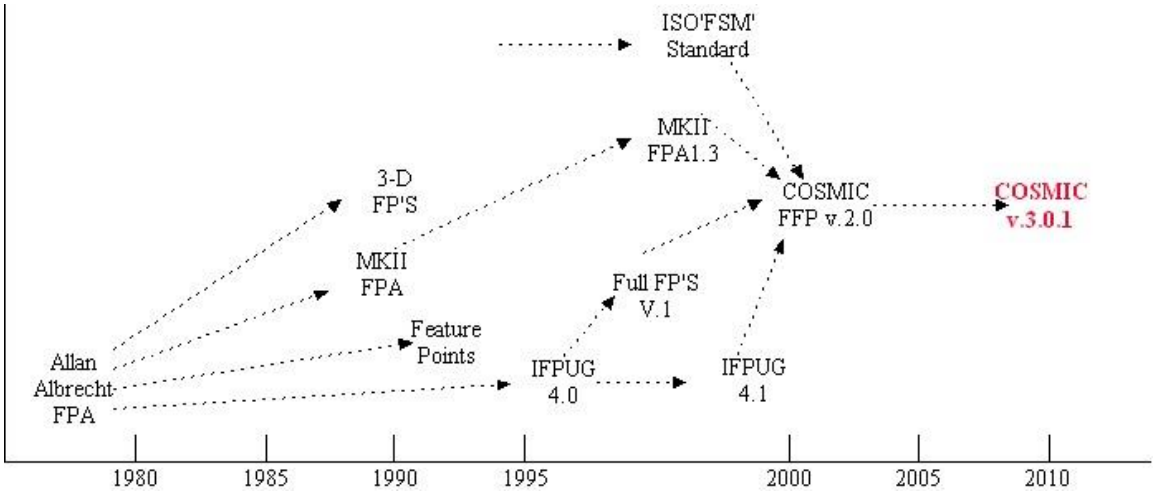

Figure 10 Evolution history of COSMIC (from [66])

2.3.2 COSMIC basic principles

The COSMIC method is used to measure the functional size of a piece of software from the viewpoint of end users. This method is based on the COSMIC Generic Software Model (see Figure 11), which assumes that the functional user requirements of a piece of software can be decomposed into unique functional processes, which are further classified into either data movements or data manipulations.

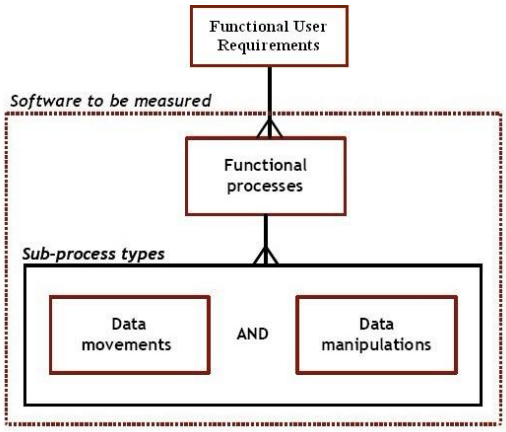

Figure 11 COSMIC generic software model

The COSMIC method assumes that each data movement has an associated constant average amount of data manipulation; therefore only the data movements are measured. The assumption that the amount of data manipulations is proportional to the number of data movements is often violated. For instance, applications belonging to different domains can easily be characterized by different data manipulation/movements ratios. In this respect, COSMIC is not better than FPA: both tend to overlook the amount of elaboration involved in processes.

Each movement is considered as one COSMIC function point; the size of a functional process is the number of its data movements; the COSMIC functional size of this piece of software is the sum of the sizes of its processes.

2.3.3 Functional process

Functional user requirements are known early in the development process; therefore they are a good starting point for estimation. They can be broken down into a number of functional processes; independently executable sets of elementary actions that the software should perform in response to a triggering event.

The COSMIC method defines a functional process as "*an elementary component of a set of Functional User Requirements comprising a unique, cohesive and independently executable set of data movements*". [33]

Each functional process is triggered by an "Entry" data movement, which comes from a functional user and aims to activate a functional process identified by the end-user and carried out by the piece of software to be measured. Figure 12 (from [33]) illustrates clearly the relation between triggering event, functional user and functional process.

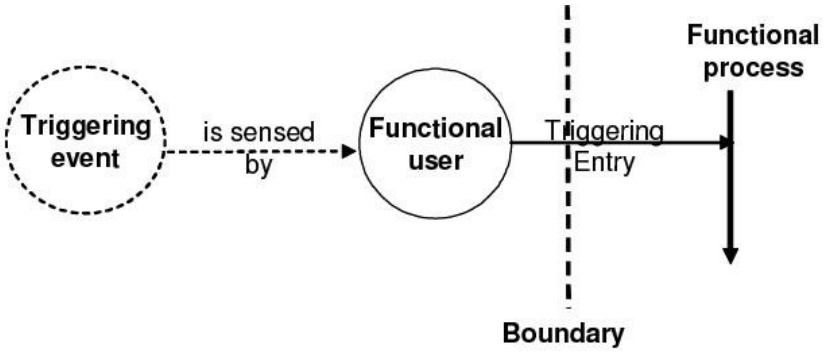

Figure 12 Relation between triggering event, functional user and functional process

The data movements are the base functional components that are used for establishing the size of the software. The COSMIC recognizes four types of data movement, namely Entry, Exit, Write, and Read (see Figure 13).

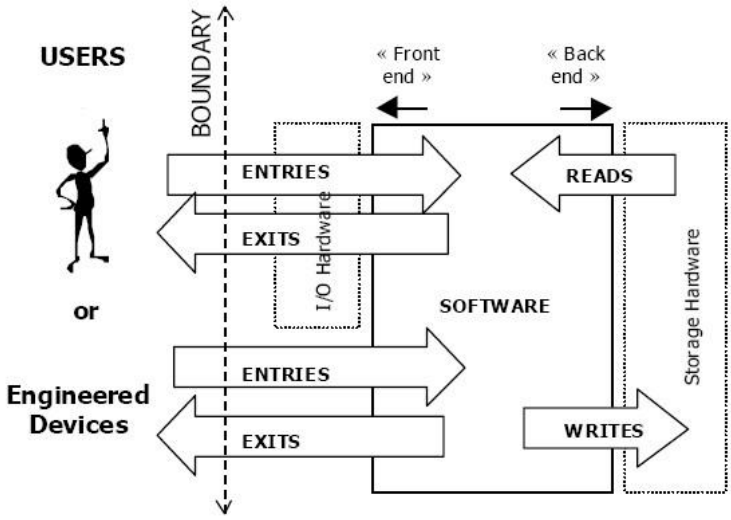

Figure 13 COSMIC view of software [38]

As illustrated in Figure 13 (from [38]), an Entry moves a data group from a user across the software boundary into the functional process where it is required. An exit is a data movement that moves a data group from a functional process across the software boundary to the user that requires it. A Write movement updates (possibly creates or deletes) data group that is stored within the boundary of the application being measured. Similarly, a Read movement involves reading a data group. Obviously the Entry and Exit movements do not involve in updating the data they move, but every Entry or an Exit is considered to include certain associated data manipulations (for example validation of the entered data or formatting and routing associated with the data to be exited).

In COSMIC, the functional processes are characterized on data group. The movement of a data group can be of type Entry, Exit, Read, or Write. In Figure 14, we present the COSMIC meta-model, which illustrates the information we need to identify and capture for representing a software system to be measured.

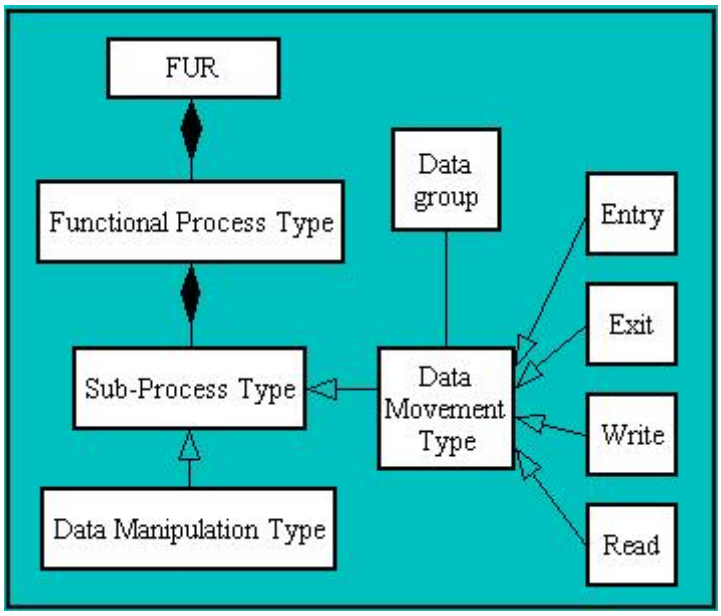

Figure 14 COSMIC software model

Each data movement, no matter the movement type, is counted as one COSMIC Function Point (CFP).

To identify data movements, data groups have to be identified first. A data group is defined "*a distinct, non empty, non ordered and non redundant set of data attributes where each included data attribute describes a complementary aspect of the same object of interest*."[33] From the characters of the data group and attribute, we can draw Figure 15 to display the relative conceptual granularity among them.

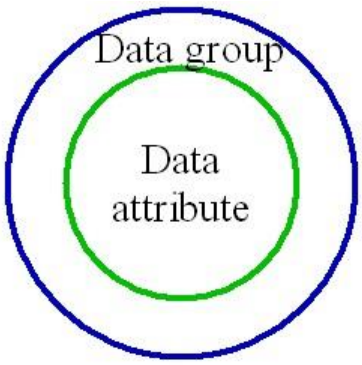

Figure 15 Relative conceptual granularities of COSMIC data elements

An object of interest is defined as any 'thing' that is identified from the point of view of the Functional User Requirements. It can be any physical thing, any (part of) conceptual object about which the software is required to process and/or store data.

The definition and principles of objects of interest and of data groups are intentionally broad in order to be applicable to the widest possible range of software. This quality sometimes results in it being difficult to apply the definition and principles when measuring a specific piece of software.

Only movements that involve persistent or transient data groups are considered. A data group is persistent if its value is preserved between two functional process activations; so, temporary variables used in a computation within a functional process are not persistent; a data group that is set by a process and read by another one is persistent, even if the value is lost when the program terminates.

Transient data groups are typically created for output: when you have an exit that involves some attributes taken from a data group and some other taken from another data group, you consider that the exit involves a transient data group.

Similarly, in an ad hoc enquiry, the selection parameters to derive the required data are considered a transient data group associated with the Entry consisting of the query execution request. Transient data groups that do not survive the execution of the functional process; nevertheless, moving them counts as a legal data movement. They always involve data that cross the boundary between the software and its user(s).

2.3.4 Measurement process

The measurement strategy phase

In this phase, the purpose and scope of the measurement, the identification of functional users and the level of granularity are considered, before actually starting to measure, because they define and help us to clarify which size should be measured, how should we interpret this measurement, what is the artefact to be measured, from which view point is this measurement carried out, etc.

Applying the COSMIC Generic Software Model

Applying the COSMIC Generic Software Model means identifying the set events issued by each of the functional user (types) identified in the FUR, and then identifying the corresponding functional processes triggered in response to those events, together with the associated objects-of-interest, data groups, and data movements.

The COSMIC Generic Software Model shall be applied to the functional user requirements of each separate piece of software for which a separate measurement scope has been defined.

In the literature it has been often noted that "… *FPs are counted according to a set of informal rules that require human interpretation; moreover, the rules are defined in a rather fuzzy way, so that it is not always clear how every element of the requirements should be classified and counted. As a consequence, you need an expert …"* [3], although this sentence refers to FPA, it applies to the COSMIC method as well, even though the COSMIC provides a measurement guide [33] in addition to the official manual to help the measures.

The general COSMIC-FFP procedure consists of three phase, namely identifying data movements, applying the measurement function, and aggregating measurement result.

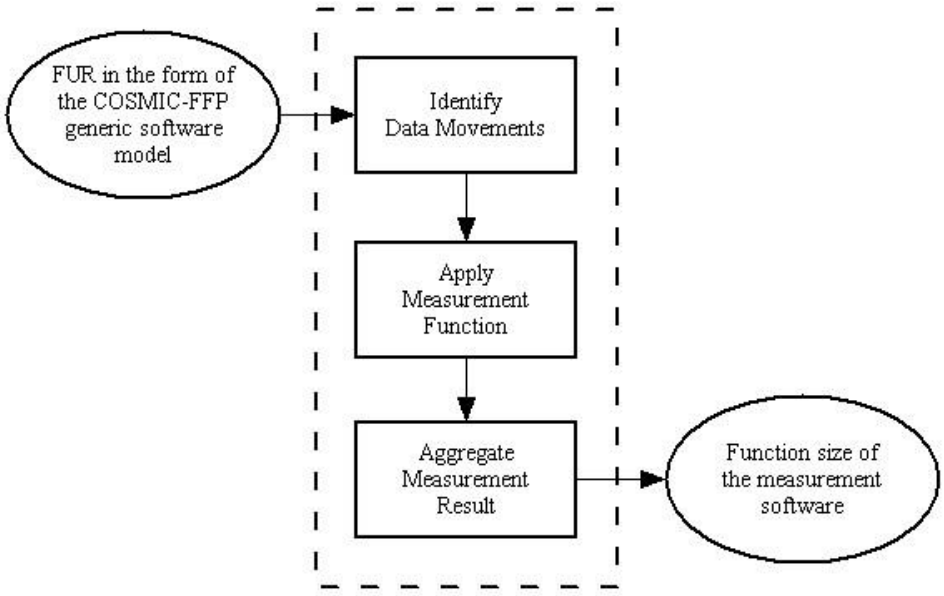

Figure 16 COSMIC general measurement procedure [33]

Identifying data movements
This step consists in identifying the data movement of sub-process types (Entry, Exit, Read, and Write types) of each functional process type.

To identify data movements, we suggest two steps. First, we identify the persistent data groups. Data groups are relatively easy to be identified, since data mentioned in the requirements are always persistent or transient. Second, for each functional process we check what data groups are subject to input, output, reading or writing (i.e., creation, update or deletion).

Applying the measurement function

An important rule is that one data movement has to be counted for each data group that is moved. So, for instance, an input operation that moves attributes from two data groups involves two data movements: one for each data group involved.

Any data appearing on input or output screens or reports that are not related to an object of interest to a functional user should not be identified as indicating a data movement, so should not be measured.

Aggregating measurement function results

In the COSMIC method the aggregation of size measures is straightforward:
- The size of a functional process is given by the number of its data movement types;
- The size of the application is the sum of sizes of its functional processes.

2.4 Comparison between FPA and COSMIC

In literature [48], J. P. Jacquet and A. Abran presented a process model for software measurement methods. The proposed model details the distinct steps, namely design of

the measurement method, measurement method application, measurement result analysis, and exploitation of the result.

The first one of them consists in 4 sub-steps, as illustrated in Figure 17 (from [48]), namely definition of the objectives, design or selection of the meta-model, characterisation of the concept to be measured, and definition of the numerical assignment rules.

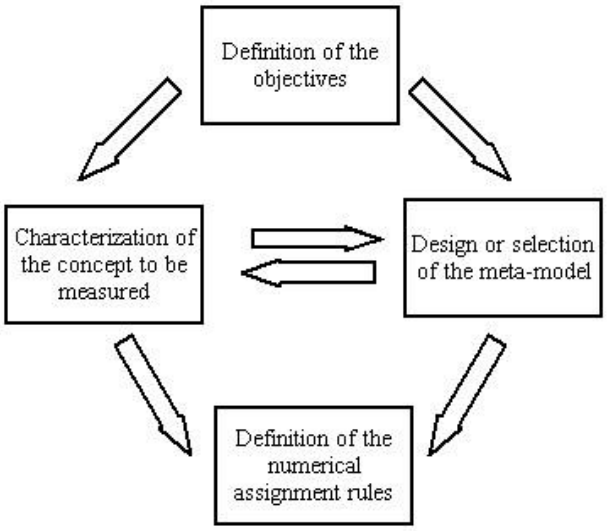

Figure 17 Design of the measurement method

Now, we compare both methods according to the above 4 sub-steps, then we will make a general comparison of the elements of both methods.

2.4.1 Objectives

To define the objectives of measurement, for example, what we want to measure, which attribute should we measure, what the measurement method point of view will be, software user, software designer, etc. Because all of these criteria have a strong influence on the design and the result of the measurement, it is very important to compare them. In Table 9 we list all the criteria.

Table 9 Objectives of measurement of both methods

		FPA	COSMIC
Software kind	MIS	√	√
	Real-time		√
	Embedded system		√
	complex mathematics algorithms	√	
	Other type applications	√	√
Type of count	development project	√	√
	enhancement project	√	√
	application	√	√
viewpoint		end-user	end-user
scope/propose		software function size	software function size
Result		FP	CFP
object of interest/attribute		DET, RET, FTR	Data group

2.4.2 Software model

Though software (also the production of each stage of development) is artefact, is not tangible, however it can be made visible through multiple representations. As described in Figure 7 and in Figure 14, the set of characteristics (or relationship) that can represent a software or a software piece is abstracted and illustrated. All the entity types of meta-model of both methods are listed in Table 10, and they are compared in detail in the following.

Table 10 Entity type of software model

	FPA	COSMIC
Entity type of meta model	Transaction Process(EI,EO, or EQ)	Functional Process
	Data Function (ILE, EIF)	

Processes

Both FPA and COSMIC consider SW applications as composed by processes, namely elementary processes in FPA and functional processes in COSMIC.

In the first approximation, they represent the same concept. FPA defines an elementary process as "the smallest unit of activity meaningful to the user." It must be self-contained, and leave the application in a consistent state. The COSMIC defines a functional process as "*an elementary component of a set of FURs comprising a unique, cohesive, and independently executable set of data movements*" [33]. It is triggered by one or more triggering events and completes when it has executed all that is required to be done in response to the triggering event type."

Elementary processes and functional processes are not "exactly" the same concept since the rules to be fulfilled by a proper COSMIC functional process are slightly more restrictive than the rules for IFPUG transactions, since COSMIC is more demanding on defining what the right granularity of a proper functional process is.

Data function

The elementary process of FPA can be of type EI, EO, or EQ. FPA considers SW application as composed by processes but not just them, the data functions are also part of the measurement. To the contrary, CFP focus only on the part of functional process.

2.4.3 Characterisation of the concept to be measured

In order to enable the measurement method to be built, the concept of measurement must be clearly defined.

In FPA, ILFs and EIFs are characterized on the basis of their Record Element Types (RETs) and Data Element Types (DETs). EIs, EOs, and EQs are characterized on the basis of their file type references (FTRs) and data element type (I/O DETs). In COSMIC, the functional processes are characterized on data group. The movement of a data group can be of type Entry, Exit, Read, or Write.

Now, we arrive at a point where we can compare the data elements of both methods. Data elements are handled and counted in quite a different way in IFPUG and COSMIC methods. This is one of the main differences between FPA and COSMIC.

Both methods group data attributes in larger entities, but the data attributes are used differently in the counting, and the grouping of the attributes into larger entities is done following two different approaches. The differences are:

- Internal and external data elements. In IFPUG you need to discriminate whether a data element is within (ILF) or out of (EIF) the system under development. In COSMIC there is no such explicit distinction (data group).
- Data attributes grouping rules. In IFPUG the data attributes are grouped according to the rules that define DETs and RETs: a logic file (ILF or EIF) is composed by one or more RETs, and each of RET is composed by attributes. In COSMIC the data attributes are grouped according to the rules that define an object of interest (see Figure 18): a data group is composed by attributes (data attributes).

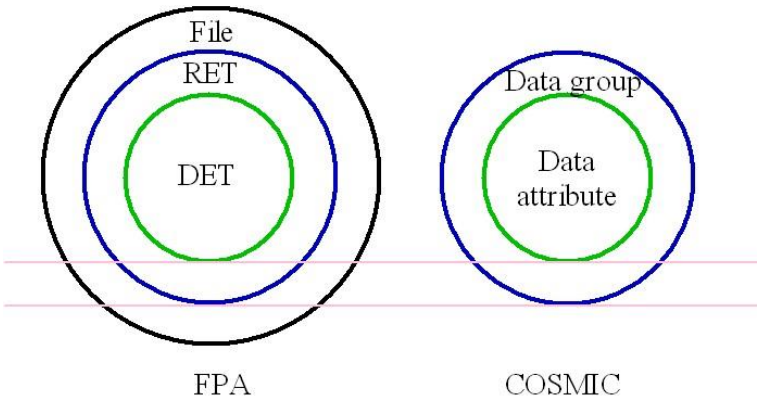

Figure 18 Comparison of conceptual granularity of FPA and COSMIC data elements

- Mandatory elements. In FPA all the mentioned elements need to be modeled: ILF/EIF, RETs and DETs. In COSMIC the data attributes are not considered at all in the counting, the data attributes are used only to be able to properly characterize a data group, thus precisely identifying them is facultative.
- Parameters granularity. The data flows considered by FPA and COSMIC are at different granularity: in FPA DETs are counted, while in COSMIC data groups are counted. The nice direct relationship between RETs and data groups is made fruitless by the different data flow granularity. FPA needs much more detailed software models than COSMIC. From RETs it is possible to determine the corresponding RETs, thus the corresponding data groups. The other way around is unfeasible: it is impossible to automatically extract the information concerning DET flows from COSMIC models.

Are there any commonalities?
- Data attributes are conceptually the same in both methods.
- A RET in IFPUG can be mapped to a data group in COSMIC.

Identifying FPA logic files is as difficult as identifying COSMIC data groups. Although these difficulties can seem to be of different nature, they are actually about the same problem, namely how to deal with the data in the information processing.

Summarizing these facts, it is easy to understand that the level of details of data representation in FPA and COSMIC methods differ, but it is still possible to establish quite clear correspondences.

We can conclude that, even if the two methods are different in treating data elements, FPA models can be used directly to obtain the COSMIC data models, while the COSMIC models need to be augmented of all the missing details (i.e., which data groups belong to the same ILF or EIF, and which attributes of a data group participate in each data flow crossing the boundaries of the system) to be usable as FPA models.

2.4.4 Definition of the numerical assignment rules

For FPA, the numerical assignment is carried on via the relative FPA reference table in the circumstance of knowing the type of the function and the numbers of the relative factors that can characterize the functions (such as DET and RET for data function). The aggregation of all the functions' UFP is the UFP of the application. The final FP of the application is assignment according to the VFA and the formula (8).

For the CFP, one data movement is assigned as 1 CFP. The COSMIC aggregate the numbers of each functional process. It is very simplex.

The units of both methods are not the same, and the ratio between both units is not 1.

2.4.5 A general comparison of the elements of both methods

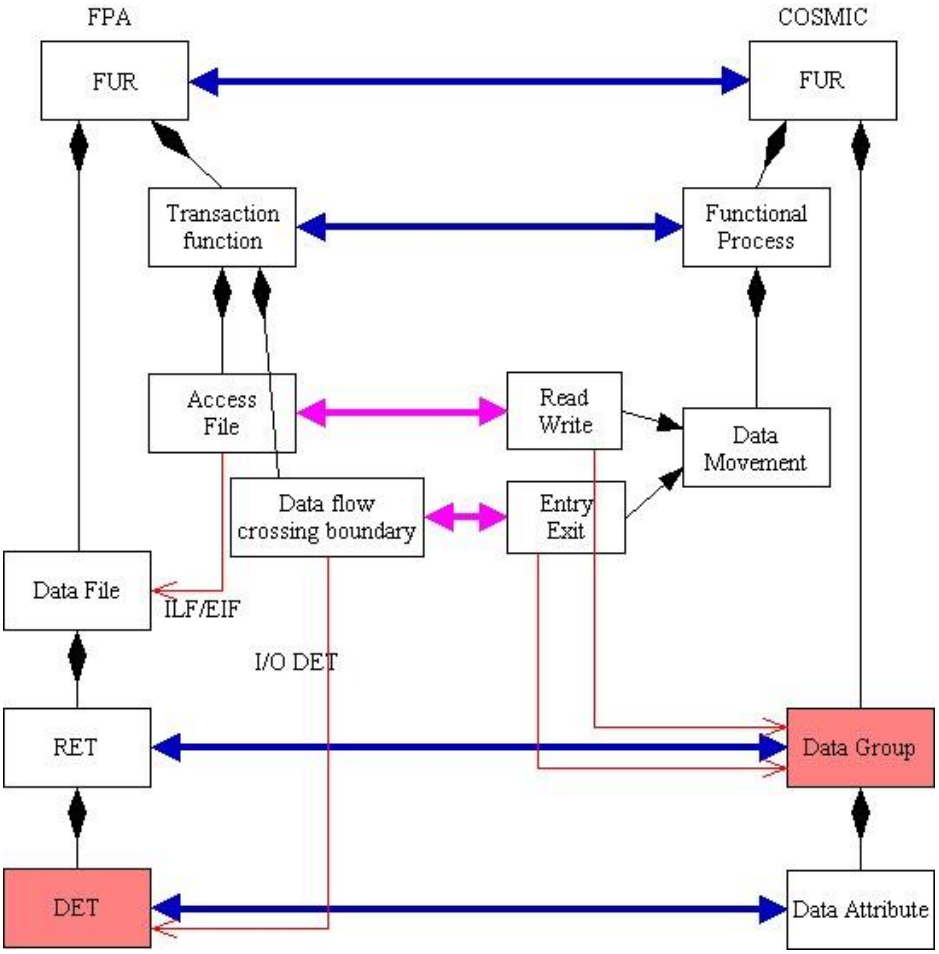

Figure 19 Comparison of the elements of FPA and COSMIC

In Figure 19, the data modeling concepts of FPA and COSMIC are informally illustrated and compared. The level of detail grows from top to bottom: DETs and data attributes are the elements at the fine level of granularity. In the central part of the figure, operations involved in transaction functions or functional processes are also illustrated.

In Figure 19, blue lines connect elements that have essentially the same meaning. On the contrary, pink lines connect elements that represent the same concept, but at different granularities.

In Table 11, we summarize the mapping between the elements of FPA and COSMIC.

Table 11 Mapping of FPA and COSMIC Concepts

FPA	COSMIC
Elementary process	Functional process
DET	Data attribute
RET	N/A
Logical data file (ILF or EIF)	Data group (or set of strictly related data groups)
FTR (Logic data file involved in an elementary process)	Data groups involved in a functional process
Set of DET that cross the boundary of the application	Persistent or Transient Data Groups that cross the boundary of the application

EI	Functional process, necessarily involving a Write movement
EO	Functional process, necessarily involving an Exit movement
EQ	Functional process, necessarily involving Read and Exit movements
Action within an elementary process that involves DET entering the application	Entry
Action within an elementary process that involves DET exiting the application	Exit
Action within an elementary process that involves reading from a FTR	Read
Action within an elementary process that involves modifying a FTR	Write

2.4.6 Comparison about the measurement process

The measurement process of both methods involves the same macro phases. The three phases mentioned in Figure 4 in Section 2.1, Factor identification, Factor counting e Factor weighting, are the core of the FSM methodology.

All the elements involved in both methods and the detailed activities of the last two phases are listed in Table 12. From this table, it is clear to see that FPA is relatively long, expensive, and difficult, while applying the COSMIC method is faster, simpler, and cheaper.

Table 12 Analysis of all the elements involved in FPA and COSMIC

ID	Element	Basic element	Process activity	FPA	COSMIC
1	DET	√		√	
2	RET	√		√	
3	Type of Data function(ILF vs. EIF)	√		√	
4	Complexity of each Data Function		√	√	
5	UFP of each Data Function		√	√	
6	I/O DET	√	√	√	
7	FTR	√	√	√	
8	Type of transaction function	√	√	√	
9	Complexity of transaction function		√	√	
10	UFP of application to be measured		√	√	
11	VAF		√	√	
12	Data group			√	√
13	Summation of a piece of software to be measured		√	√	√

Since during the measurement process identifying the factors that compose the FUR model is the core and most difficult part of FSM, and the factors needed to be identified in the first macro affect the next two phases, so all those who intend to invent new FSM methods or need to improve or simplify existing methods try to either specify new sets

of the factors that define the FUR model or devise new procedures for identifying these factors. In the next chapter this observation is confirmed through the study of several simplified FSM methods.

Different FSM methods use different FUR models. These models affect the measurement principle and the measurement process and activities. In the forth chapter we will illustrate the model-base measurement-oreinted method proposed by Lavazza et al. to facilitate the measurement activity, especially facilitate the identification and counting of the factors.

Chapter 3 Simplified Functional Size Measurement

As we have seen in Chapter 2, performing FSM requires a thorough exploration of FUR, to identify and possibly weight basic functional components. Therefore, the measurement process can be quite long and expensive. In fact, FPA performed by a certified function point consultant proceeds at a relatively slow pace: between 400 and 600 function points (FP) per day, according to Capers Jones [12], between 200 and 300 function points per day according to experts from Total Metrics [13]. Consequently, measuring the size of a moderately large application can take even long time. Also the cost of estimation is often considered excessive by software developers.

In addition, cost estimates may be needed when requirements have not yet been specified in detail and completely. This is a problem, since often at the beginning of a project FUR are known only in an approximate and incomplete way. Instead, the accuracy of a measure (i.e. the closeness to its "thoric" value) grows with the completeness and precision of FUR specifications.

In practice, in the early phases of the software development lifecycle, size estimations would be necessary for bidding and planning. But the available information is often incomplete and insufficient. So the customer only wants or is only able to do approximate measurements. When we can measure with the greatest accuracy, we no longer need that measure. The situation is described in a paradox illustrated in Figure 20 (from [16]).

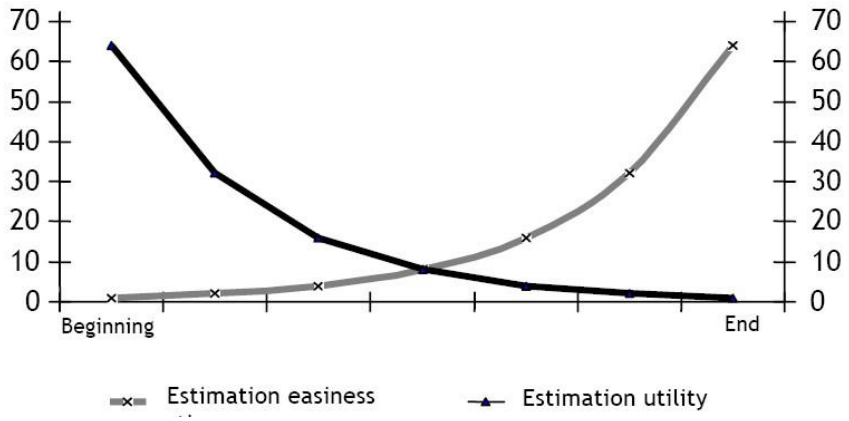

Figure 20 Estimation paradox (from [16])

Given the above situation, many simplified function point methods have been proposed. In the following sections, we will discuss the existing simplified function point methods according to their principles.

3.1 E&QFP

The most well-known approach for simplifying the process of FP counting is probably the Early & Quick Function Points (E&QFP) method [16].

3.1.1 Theoretical basis and characters

The definition of E&QFP is based on the consideration that estimates are sometimes needed before the analysis of requirements is completed, when the information on the software to be measured is incomplete or not sufficiently detailed.

The method is based on analogy-based classification, structured aggregation, and statistical data. The method aims at providing an approximate measure of size in FP. In other words, the process is simplified, but the unit of measure size result is IFPUG FP.

The E&QFP manual provides a description of the "Functional hierarchy" according to which FUR can be decomposed and measured (see Figure 21, corresponding to Figure 2 in the E&QFP manual, subsection 1.1.5)

The idea is that if you have enough information at the most detailed level (and enough time to apply the standard process) you count FP according to IFPUG rules (see the level 1 in Figure 22); otherwise, you can estimate the size of larger elements (e.g., General or Macro processes) either on the basis of analogy (e.g., a given General process is "similar" to a known one) or according to the structured aggregation (e.g., a General process is composed of a few Transactional BFC). By considering elements that are coarser-grained than the BFC_{FPA}, the EQFP measurement process leads to an approximate measure size result in IFPUG FP.

It must be noted that within the same application, some parts can be measured at a fine granularity level (possibly the IFPUG level), while other parts can be estimated at a much coarser level.

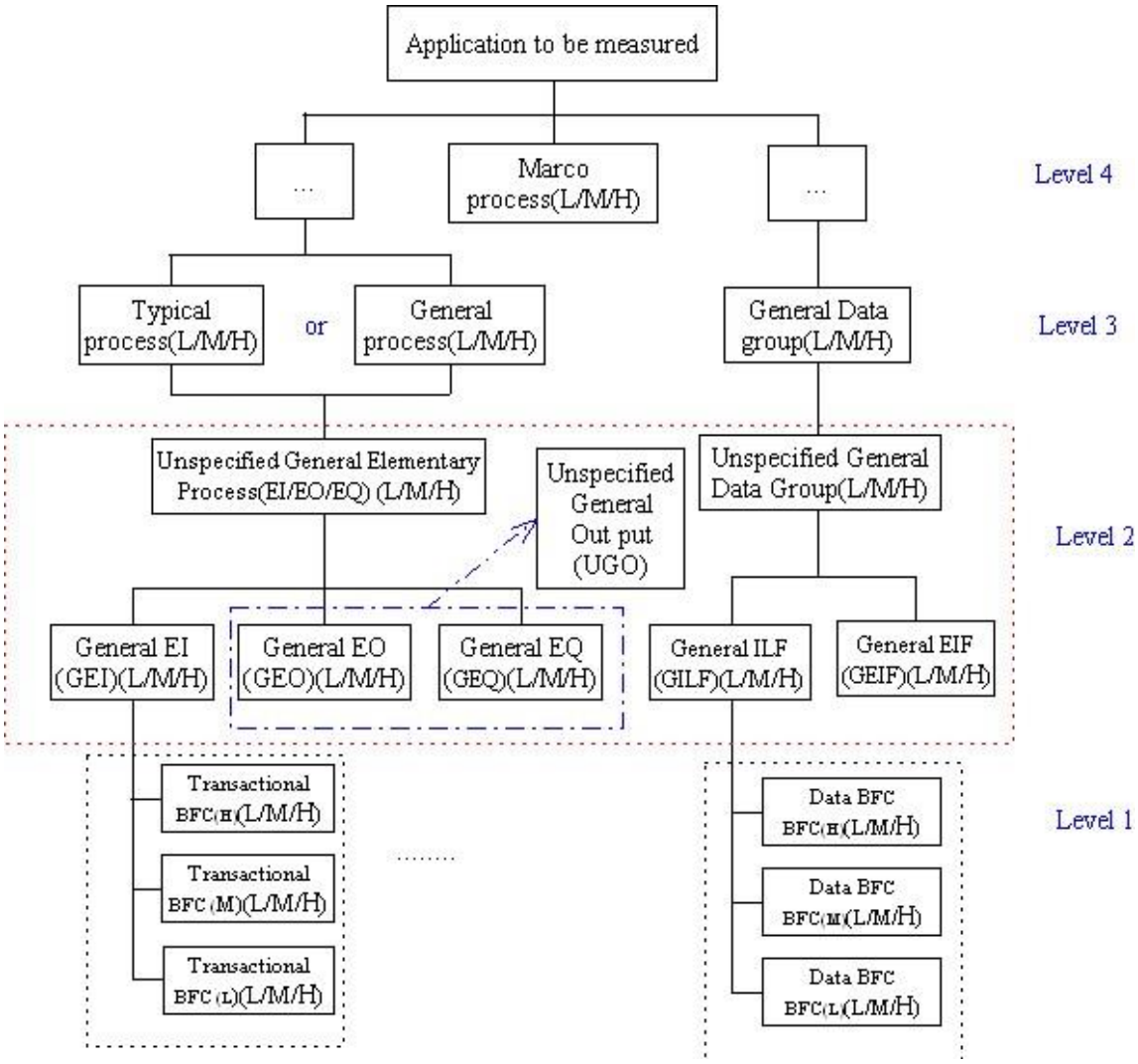

Figure 21 Functional hierarchy in the E&QFP technique

1st aggregation level

The 1st aggregation level is exactly the same as FPA.

2nd aggregation level

The 2nd aggregation level has been introduced to deal with cases when data and transaction functions have been identified, but there is no time or not enough detail to weight them properly. At this aggregation level, there are 3 existing cases:

- In the first case, it is possible to identify exactly the type of IFPUG BFC, but not its complexity.
- In the second case, it is not possible to identify exactly the type of BFC nor its complexity, "doubtful" or "uncertain" elementary process for which there are no details available to differentiate between EO and EQ.
- In the third case, it is not possible to identify exactly the type of BFC nor its complexity, "doubtful" or "uncertain" elementary process for which there are no details available to single out the primary goal, namely the presence of an EI, EO or EQ.

The first and the last case have been introduced in the subsection 1.1.5. So, we only introduce the second case, namely "Un specified Output", as following. This method provides weights derived from the statistical analysis of many projects (see Table 13)

Table 13 Components and Values of Unspecified data group, generic EI, and Unspecified Generic Output at the 2nd aggregation level

Function type	Weight		
	Low	Likely	High
Unspecified Generic Data Function	6.4	7.0	7.8
Generic EI	4.0	4.2	4.4
UGO-Unspecified Generic Output(EO/EQ)	4.1	4.6	5.0

3rd aggregation level

When user requirements are insufficient to identify specific BFCs but only groups of unspecified BFCs, the aggregations of individual BFCs are taken into account (see Level 3 of Figure 21).

E&QFP method defines "typical processes", which consist of a set of 4 CRUD (create, read, update, and delete) elementary process, each generally deals with a special logic data file. There are 3 Typical Processes, as displayed in the Table 14.

Table 14 Components and Values of Typical Process at the 3rd aggregation level

Function Type	Weight		
	Low	Likely	High
Typical Process Small(CRUD)	14.1	16.5	19.0
Typical Process Medium (CRUD + List)	17.9	21.1	24.3
Typical Process Large (CRUD + List + Report)	22.3	26.3	30.2

If a set of functional processes cannot be classified as typical processes because they involve additional operations, they can be generally classified into 3 general process types according to the number of involved Unspecified Elementary Processes (UEP).

Table 15 Components and Values of General Process at the 3rd aggregation level

Function Type	Weight		
	Low	Likely	High
General Process Small(6-10 UEP'S)	26.4	35.2	44.0
General Process Medium (11-15 UEP'S)	42.9	57.2	71.5
General Process Large(16-20 UEP'S)	59.4	79.2	98.9

Concerning data, 3 general data group (GDG) typologies are recognized as different aggregation levels, which depend on the amount of Unspecified Logic File (ULF) belonging to the GDG. An ULF is a file whose size and type (i.e., ILF or EIF) is not known.

Table 16 Components and Values of General Data Group at the 3rd aggregation level

Function Type	Weight		
	Low	Likely	High
General Data Group Small(2-4 ULFs)	15.0	21.4	27.8
General Data Group Medium(5-8 ULFs)	32.4	46.3	60.2
General Data Group Large(9-13 ULFs)	54.8	78.3	101.8

4th aggregation level

If the levels of 1, 2, or 3 can not be used, the 4th aggregation level is suitable. The type of macro process level is defined according to the number of general processes that are expected to be included in the macro process.

Table 17 Components and Values of Macro Process at the 4th aggregation level

Function Type	Weight		
	Low	Likely	High
Macro Process Small(2-4 Generic GPS's)	111.5	171.5	231.5
Macro Process Medium(5-7 Generic GPS's)	185.8	285.9	385.9
Macro Process Large(8-10 Generic GPS's)	297.3	457.4	617.4

3.1.2 Estimation procedure

The official procedure for estimating with E&QFP is illustrated in Figure 22 (from [16]). Most steps are similar to the FPA counting procedure.

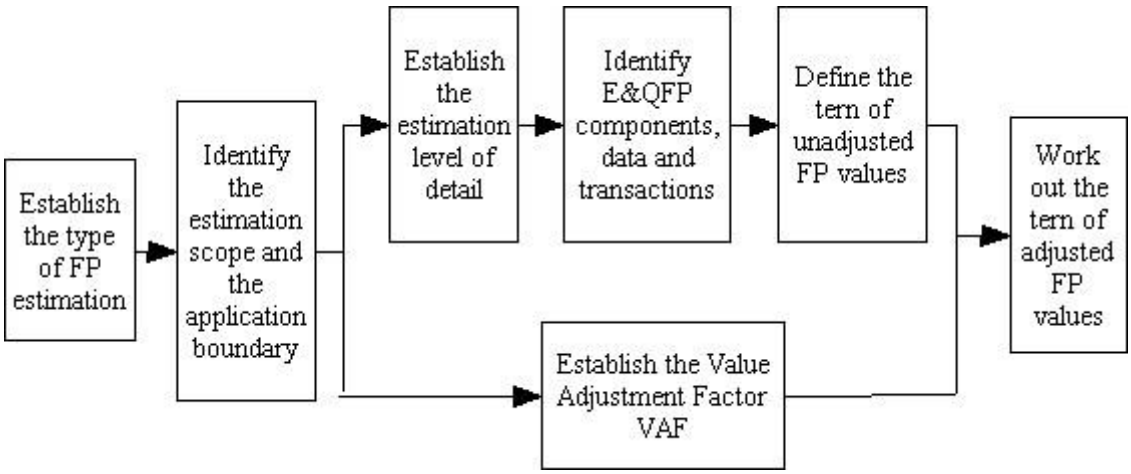

Figure 22 Diagram of the E&QFP estimation procedure (from [16])

However - unlike in IFPUG counting- the values of each component of the above E&QFP tables are made up of a tern of values labelled with Low, Likely and High. Therefore the estimated size s not a single vales; rather it is made of a likely value and an expected variability range.

3.1.3 Characteristics of E&QFP

Multi level approach
An advantage of the method is that different parts of the system can be measured at different detail levels: for instance, a part of the system can be measured following the IFPUG manual rules [10][11], while other parts can be measured on the basis of coarser-grained information. In fact, the E&QFP method is based on the classification of the processes and data of an application according to a hierarchy (see Figure 21).

Time and cost savings
The trade-off between reduced measurement time and costs is also a reason for adopting the EQFP method even when full specifications are available, but there is the need for completing the measurement in a short time, or at a lower cost.

Limits
We have to remember that in general, applying E&QFP involves ignoring some details of the FUR specifications that should be considered according to the standard IFPUG manual. As a result, the obtained size estimate is generally less accurate than the measure performed according to the manual. According to the authors of the E&QFP, the error is no greater than 10%, on average.

3.2 Average complexity (weight) values

In this section we describe methods that adopt average weights. These methods do not require the weighting of functions; instead each function is weighted with average values.

3.2.1 Estimated NESMA method

The NESMA (Netherlands Software Metrics Association) recognizes three types of function point counts: detailed function point count, estimated function point count, and indicative function point count. The latter 2 methods have been developed to enable function point counting early in the system life cycle.

The Estimated NESMA method requires the identification and classification of all data and transaction functions, but does not require the assessment of the complexity of each function: Data Functions (ILF and EIF) are assumed to be of low complexity, EI, EQ and EO are assumed to be of average complexity. So the weights of the functions - ILF, EIF, EI, EO, and EQ - are respectively valued as 7, 5, 4, 5 and 4 [41].

$$UFP = \#ILF \times 7 + \#EIF \times 5 + \#EI \times 4 + \#EO \times 5 + \#EQ \times 4 \quad (9)$$

The procedure of counting the Estimated NESMA function points is as following:
• Determine the numbers of ILF and EIF (# ILF and #EIF, respectively);
• Determine the numbers of EI, EO and EQ (# EI, #EO, #EQ, respectively);
• Compute the function points by Equation (9):

The Estimated NESMA method is expected to be more approximated than the E&QFP method based on generic functions, as the latter uses likely values for transactions of unknown complexity, derived from statistic analysis.

3.2.2 ISBSG average weights

This model is based on the average weights for each BFC, as resulting from the analysis of the ISBSG dataset [15], which contains data from a few thousand projects.
The mean weight of the transaction and data functions in the ISBSG dataset is reported in the following table.

Function	ILF	EIF	EI	EO	EQ
Mean weight	7.4	5.5	4.3	5.4	3.8

The estimated size in UFP is then computed assuming that each function has mean weight:

$$UFP = \#EI \times 4.3 + \#EO \times 5.4 + \#EQ \times 3.8 + \#ILF \times 7.4 + \#EIF \times 5.5 \quad (10)$$

3.2.3 Simplified FP

The simplified FP (sFP) approach simply assumes that all BFC are of average complexity [18], thus:

$$UFP = \#EI \times 4 + \#EO \times 5 + \#EQ \times 4 + \#ILF \times 10 + \#EIF \times 7 \quad (11)$$

3.2.4 Prognosis of CNV AG

The Prognosis of CNY AG method [42] was defined by the CNV AG (the outsourced non-insurance part of AXA Colonia Insurance) based on the average complexities resulting from a historical dataset. The version defined in 1998 is the following:

$$UFP = \#EI \times 4.6 + \#EO \times 5.5 + \#EQ \times 4.3 + \#ILF \times 8.0 + \#EIF \times 5.9 \quad (12)$$

In 1999, considering new historical data, the simplified model was updated as following:

$$UFP = \#EI \times 4.6 + \#EO \times 5.7 + \#EQ \times 4.3 + \#ILF \times 8.2 + \#EIF \times 6.1 \quad (13)$$

3.3 Size estimation based on a single component

With this technique an estimation model is built using one type of components (usually ILFs).According to this model, the FP of the whole system can be calculated by the component and the given model.

This technique is based on the statistically significant correlation between the number of ILFs (for example) in an application and the application's unadjusted function point co unt.
This technique is very simple and is very easy to be developed locally. To build an ILF-based
model, it is only necessary to collect UFPs and ILFs of all your applications, and derive the model (e.g., using regression).

In the following subsections, a few methods - Indicative NESMA FP, Tichenor ILF Model, Prognosis by CNV AG, and ISBSG distribution model – are given.

3.3.1 Indicative NESMA method

The Indicative NESMA method [17] is well known and is often referred to as "the Dutch method". It simplifies the process by only requiring the identification of Logic Data from a data model. The Function Point size is then computed by applying predefined weights, whose values depend on whether the data model is normalized in 3rd normal form:

Non normalized model: Function Points = #ILF × 35 + #EIF × 15 (14)
Normalized model: Function Points = #ILF × 25 + #EIF × 10 (15)

The process of applying the NESMA indicative method involves only identifying logic data and classifying them as ILF or EIF. Accordingly, it requires less time and effort than several of the methods described above, in general. However, it is quite clear that the Indicative NESMA method is quite rough in its computation. The official NESMA counting manual specifies that errors in functional size with this approach can be up to 50%.

3.3.2 ILF Model

The Internal Logical File Model (sometimes named “ILF Model,” or “One File Model”) was developed in 1994 by the IRS function point team and was presented at the fall 1997 IFPUG Conference [15]. It bases the estimation of the size on the number of ILF via the following formula for transactional system (for batch systems, Tichenor proposes a smaller multiplier):

$$UFP = \#ILF \times 14.93 \quad (16)$$

This model assumes a distribution of BFC with respect to ILF as follows: EI/ILF = 0.33, EO/ILF = 0.39, EQ/ILF = 0.01, EIF/ILF = 0.1. If the considered software application features a different distribution, the estimation can be inaccurate.

The fact that a method based only on ILF requires a given distribution for the other BFC is not surprising. In fact, the size of the application depends on how many transactions are needed to elaborate those data, and the number of transaction cannot be guessed only on the basis of the number of ILF, as it depend on the number of ILF just very loosely. Instead of allowing the user to specify the number of transactions that are needed, the Tichenor ILF model practically imposes that the number of transactions complies with the distribution given above.

3.3.3 ISBSG Distribution model

In the very early phases of a software development project it is not practical or even possible to know in detail all of the items that make up all of the function point components. However, it is often possible to detail one of the components - such as the Internal Logical Files or External Inputs - with a fair degree of certainty.

However, to estimate the size of an application on the basis of a single component, it is necessary to know the average contribution of that component, at least on average.

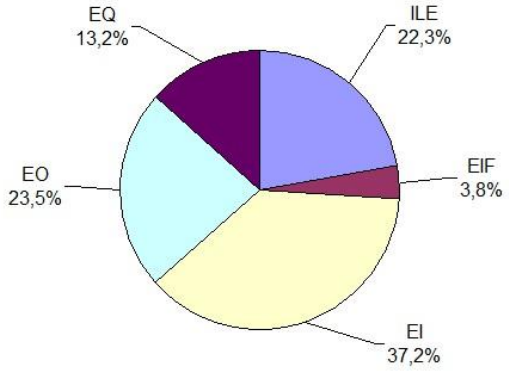

Figure 23 Relationships among IFPUG Functional Component Types

Figure 23 shows the relationships among the five components of the IFPUG functional size method from the project data in the ISBSG repository. These relationships can be used to estimate the functional size of a project.

The analysis of the ISBSG dataset yielded the following distribution of BFC contributions to the size in FP:

ILF 22.3%, EIF 3.8%, EI 37.2%, EO 23.5%, EQ 13.2%

The analysis of the ISBSG dataset also shows that the average size of ILF is 7.4 UFP. It is thus possible to compute the estimated size on the basis of the number of ILF as follows:

$$UFP = (\#ILF \times 7.4) \times 100 / 22.3 \quad (17)$$

The same considerations reported above for the Tichenor model apply. If the application to be measured does not fit the distribution assumed by the ISBSG distribution model, it is likely that the estimation will be inaccurate.

Note: The techniques discussed above are only valid only if your application or development project is loosely coupled from other applications and fits the profile of projects currently in the ISBSG Repository. Early research indicates that the above relationships may not hold for the domains of real-time, control, scientific or embedded software.

3.3.4 Prognosis of CNV AG

The CNV Prognosis [42] method, defined in 1998 by CNV AG (the outsourced non-insurance part of AXA Colonia Insurance) uses the following model:

$$FP = 56 + \#IO \times 7.3; (R^2=0.9525) \quad (18)$$

Where #IO = number of (EI + EO).

In 1999, this model was revised as

$$FP = 39 + \#IO \times 7.6; (R^2=0.9509) \quad (19)$$

3.3.5 Early Function Point Method (EFPM)

Asensio et al. [44] proposed a method called Early Function Point Method (EFPM) for the need of estimates at the early stage of software development when the required documentation is not available yet. Based on a set of 30 projects in his work, the following regression equation were found and proposed to calculate the FP by ILE, ILE + EIF, or EI + EO + EQ.

$$FP = 130.327 + \#ILE \times 15.90 \quad (20)$$

$$FP = 66.905 + (\#ILE + \#EIF) \times 13.035 \quad (21)$$

$$FP = 50.784 + (\#EI + \#EO + \#EQ) \times 6.28 \quad (22)$$

3.4 Approximation technique and estimation technique

3.4.1 "Smart" Approximation Technique

In [14], Santillo suggested probabilistic approaches, where the measurer can indicate the minimum, medium and maximum weight of each BFC, together with the expected probability that the weight is actually minimum, medium or maximum. This leads to estimate not only the size, but also the probability that the actual size is equal to the estimate.

The measurement procedure consists of the following steps:

- Preparing the requirements to an acceptable level of description (e.g. "lists" rather than "grouped statements");
- Further specifying the requirements at the level of "single functions", which typically resembles the concept of elementary/functional process in FSM);
- Based on the measure's expertise, assessing functions with the minimum, medium, and maximum weights and their related probabilities.

Table 18 (from [14]) shows a form to collect the data required by the method.

Table 18 Smart FP assessment (Only for FPA)

Function	Weights&Probability	Low Min	Avg Mid	High Max	Weighted Value
	Weight value				
	Related Probability				
Total FP					
Estim/prob					
Estim. Range					

− For each function, the weighted value is calculated by the functions (23) and (24). The size of the i^{th} function is computed as follows:

$$WeightedSize_i = LowMin\_value_i \times LowMin\_probability_i +$$

$$AvgMid\_value_i \times AvgMid\_probability_i + HighMax\_value_i \times HighMax\_probability_i \quad (23)$$

The size of the whole application is computed –as usual– as the sum of functions' sizes.

$$TotalSize = WeightedSize_1 + \ldots + WeightedSize_n \quad (24)$$

This method also supports the computation of the confidence probability for the total size.
With this method, the measurer can choose to spend more time in the analysis of each function to get more probable values, or speed up the process, indicating a smaller confidence in the provided values. In any case, the probability associated with the result reflects this trade-off.

3.5 Comparison of simplified methods

In this section, we compare all the simplified methods mentioned in the sections 3.1, 3.2, 3.3, and 3.4. The comparison is summarized in Table 19. We compare methods mainly with respect to the techniques used, the factors measured, and the measurement processes.

This page intentionally left blank.

Table 19 Comparison of the simplified methods

Simplified Function point method	Technique used	Factors									Process	
		Basic Functional Components					BFC Granularity	BFC identification Difficulty	Weight			
		ILF	EIF	EI	EO	EQ			Fixed	Probability	changed	Difficulty
E&QFP(Level 1)	Analogy-based classification, structured aggregation	√	√	√	√	√	*	****				*****
E&QFP(Level 2) General	Idem	√	√	√	√	√	*	****	√		√	****
E&QFP(Level 2) Unspecified General Output	Idem	√	√	√	√ (UGO)		*	***/****	√		√	***/****
E&QFP(Level 2) Unspecified	Idem	√ (UGDG)		√ (UGEP)			**	***	√		√	***
E&QFP(Level 3)	Idem	√ (GDG)		√ (TP)			***	**	√		√	**
E&QFP(Level 4)	Idem	√ (MP)					***	*	√		√	*
Estimated NESMA method	Average complxities/values	√	√	√	√	√	*	****	√		√	**/***
ISBSG average weights	Idem	√	√	√	√	√	*	****	√		√	**/***
Simplified FP	Idem	√	√	√	√	√	*	****	√		√	**/***
Prognosis of CNV AG	Idem	√	√	√	√	√	*	****	√		√	**/***

Indicative NESMA method	Extrapolation	√	√				*	**	√		√	**
ILF(Tichnor's) Model	Idem	√					*	****	√		√	*
ISBSG Distribution model	Idem	√					*	****	√		√	*
Prgnosis of CNY AG	Idem			√			*	****	√		√	*
Early FP method(Model 1)	Idem	√					*	****	√		√	*
Early FP method(Model 2)	Idem	√					**	***	√		√	*
Early FP method(Model 3)	Idem			√			**	***	√		√	*
Smart FP	Approximation technique	√	√	√	√	√	*	****		√	√	****

3.5.1 Techniques

Four techniques are used in these methods:

The E&QFP method uses the techniques of analogy-based classification and structured aggregation technique.
The Estimated NESMA method, ISBSG average weights, Simplified FP, and Prognosis of CNV AG method use average complexities or average values.
The Indicative NESMA method, ILF (Tichnor's) Model, ISBSG Distribution model, and Prognosis of CNV AG, and Early Function Point method use the technique of extrapolation.
The Smart FP method employs the approximation technique.

3.5.2 Factors

Factors being used
Most of the methods use the basic functional components (BFCs).

The Estimated NESMA method, Prognosis of CNY AG, and Smart FP use all the basic functional components (ILF, EIF, EI, EO, and EQ);

The Indicative NESMA method, ILF (Tichnor's) Model, ISBSG Distribution model, and Early Function Point method (Models 1, 2, and 3) only use a subset of the BFCs.

About the E&QFP method, the situation is more complex. The level 1 and level 2 (general E&QFP) still use the basic elements components (ILF, EIF, EI, EO, and EQ). From the level 2 (unspecified general Output), the E&QFP uses the new aggregation components, for example, EO and EQ are aggregated as Unspecified General Output(UGO). At level 3, the ILF and EIF are aggregated as General Data Group (GDG). EI, EO, and EQ are aggregated as Typical Process (TP). At level 4, all the five factors are aggregated as Macro Process (MP).

The granularity of factors being used

Since identifying the factors that compose the FUR model is the core and most difficult part of FSM, all those who intend to invent new FSM methods or need to improve or simplify existing methods try to either specify new sets of the factors that define the FUR model or devise new procedures for identifying these factors.

Here, we indicate the granularity of factors as follows:
- Granularity level of BFC (*);
- Granularity level of unspecified general type of transaction function or data function (**);
- Granularity level of unspecified function (***).

The level of difficulty in capturing factors
We also compare the level of difficulty in capturing the factors being used in the measurement process. Obviously, the finer granularity of the factory, the more difficult

to capture it. In other words, the level of difficulty in capturing a factory is inversely proportional to the level of granularity of the factor.

Weight values
There are two methods for getting the weight value, namely using the fixed value and subjective estimating. The Smart FP method uses the latter method, namely subjective estimation. The other methods use the fixed values supplied by the relative method.

3.5.3 The aspect of measurement process

If the factor(s) of a simplified method are different from those of the standard method, the measurement process will also be different. So, all the processes of simplified methods, except the E&QFP (only Level 1) method, are different from the standard.

3.5.4 Brief summary

Although the above comparisons (especially in the aspects of factory granularity, capturing difficulty, and measurement process difficulty) are quit rough, the results of the comparison improve our knowledge of simplified methods.

Chapter 4 Model-based measurement

FSM aims at providing a measure of functional user requirements(FURs). The FSM methods do not specify how to model FURs. From the traditional model written in a mixture of E/R diagrams, data flow diagrams, tables, text, formulas, etc., it is very difficult to identify BFC and all those elements that contribute to size measures. So how to model FURs, in other words, how to deal with FURs and get the "object" of measurement before starting the measurement becomes a focus of FSM. In this chapter the model-based measurement methods, especially the measurement-oriented model-based methods [3] proposed by Lavazza et al, are introduced.

4.1 Fundamentals

The idea of model-based measurement stems from the observation that the most difficult part of FSM consists in extracting from functional user requirements the elements that need to be identified according to the method being used (e.g., elementary processes, logic data files, RET, DET, etc. for FPA; functional processes, data groups, data movements for COSMIC).

So, model-based measurement requires that models of the functional user requirements are built, so that the aforementioned elements can be easily identified and measured. To this end, we cannot just rely on the fact that requirements models are available, since they could be incomplete with respect to the required information (or they could provide much more information than needed), and they could provide such information at a too detailed or too coarse granularity level. Therefore, models must be measurement-oriented. The fact that model-based measurement is performed on measurement-oriented models makes this method conceptually very different from other proposals concerning the measurement of the functional size of UML models. Measuring models that have been built with measurement in mind is easy and reliable. This is the spirit of model-based measurement.

Because of the popularity of object-oriented modeling techniques, model-based measurement is actually Object-Oriented (OO) model-based measurement. Accordingly, in the following sections, a brief introduction to OO modeling techniques is given.

4.1.1 Object Oriented Modeling Technique

Before we choose or establish the OO model-based measurement method, we must look back on the technology-based OO model, because the sizing method should go in line with the approach chosen for development, for the users to adopt it and apply it consistently. [47]

With regard to OO model-based measurement, we must clearly define two main aspects: first, which conceptual modeling patterns is adopted for the analysis and modeling; second, which notation is used for capturing the conceptual modeling patterns.

Among object-oriented analysis and modelling methods, the Object-Modeling Technique (OMT) model [53] is one that is used more frequently. This conceptual approach comes from the ANSI's 4 frame schema, namely External Schemas, Conceptual Schema, Logical Data Model, and Physical Data Model (Figure 24). This pattern uses a set of schemas to describe the system from different views. Each one of those represents one person's view of the world. These schemas are then consolidated into a single conceptual schema, which can entirely, accurately, and correctly represent the application being measured.

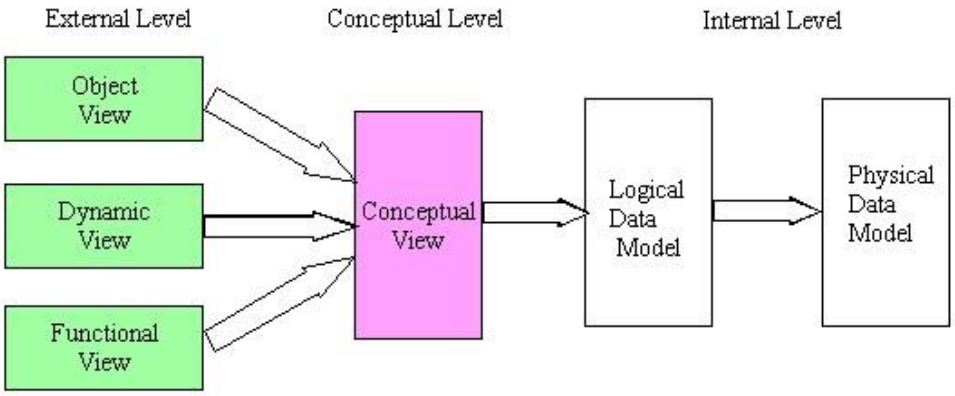

Figure 24 ANSI's conceptual schema

With regard to the notation, there are a number of different notations for representing OO models, such as the Unified Modeling Language (UML). UML incorporates OMT principles. Its static diagrams, dynamic diagrams, and functional diagrams have a good ability of capturing and representing OMT's three external conceptual schemas. The collection of these three types of diagrams can entirely, accurately, and correctly represent the application being measured. UML was designed with the characteristics of simplicity and expressiveness, and achieved a good popularity; for these reasons it was selected as the notation for model-based measurement.

4.1.2 Object-based measurement-oriented reference model

Now, the problem of measuring the application's FURs is changed to measure the conceptual model of the application represented via UML diagrams.

In order to measure the size of UML models in Function Points, three issues must be tackled. First, the mapping between FSM concepts (mainly, the BFC) and UML elements must be established; second, UML modeling rules must be defined on the basis of the rules of FSM, and the UML diagrams to be used must be identified; finally, the measurement rules must be defined.

On the basis of the considerations reported above [3] and the literature on OO software functional measurement process [48] [49], we define the model-based measurement process shown in Figure 25. This model-based measurement-oriented FSM method consists of two phases. The first phase is to specify the FURs using appropriate UML diagrams, according to the modeling rules; the second phase is to identify, count and calculate the function point according to the mapping rules and measure rules.

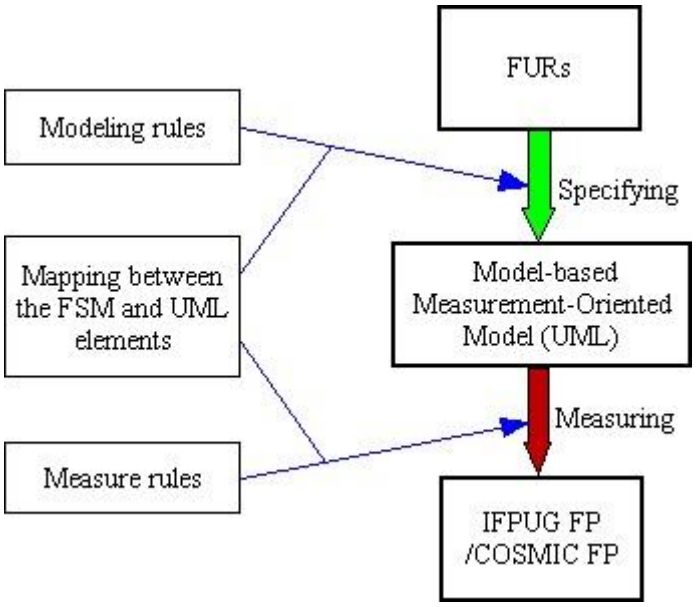

Figure 25 Process of model-based measurement

In order to carry out the specifying phase for FSM methods, a FUR must have certain properties [14]. By tracing the measurement process of both methods, we derived 13 elements (see Table 20) of a FUR that must be extracted, identified and mapped to the FSM elements shown in Table 11.

Table 20 Mapping of FPA and COSMIC Concepts

FUR element	FPA	COSMIC
Elementary operation (function)	Elementary process	Functional process
Elementary piece of information	DET	Data attribute
Data sub-group	RET	N/A
Cohesive data group	Logical data file (ILF or EIF)	Data group (or set of strictly related data groups)
Data involved in an operation (function)	FTR (Logic data file involved in an elementary process)	Data groups involved in a functional process
Elementary pieces of information that cross the boundary of the application during an operation (function)	Set of DET that cross the boundary of the application	Persistent or Transient Data Groups that cross the boundary of the application
Operation (function) whose main purpose is data input	EI	Functional process, necessarily involving a Write movement
Operation (function) whose main purpose is outputting computed results	EO	Functional process, necessarily involving an Exit movement
Operation (function) whose main purpose is retrieving data and outputting them	EQ	Functional process, necessarily involving Read and Exit movements
Action within an operation (function) that involves data input	Action within an elementary process that involves DET entering the application	Entry

Action within an operation (function) that involves data output	Action within an elementary process that involves DET exiting the application	Exit
Action within an operation (function) that involves reading stored data	Action within an elementary process that involves reading from a FTR	Read
Action within an operation (function) that involves writing (i.e., storing) data	Action within an elementary process that involves modifying a FTR	Write

Developing systems using OO paradigm requires new development approach, it is common view in academe and practice. Over the many years, although a number of OO software development approaches are discussed and prescribed, two themes are common, the first is that "*the distinction between analysis, design, and implementation often blurs in object-oriented system development*." [50]; and the second is that "*The iterations are, therefore, a key aspect of the development process*. [50]". Compared with the traditional development processes these two recurring themes change, in several ways, the form and shape of development process for object-oriented systems.

According to the COSMIC Measurement Manual [33] and other literature [48] on the OO software measurement process definition, the proposed OO-model measurement process is shown in Figure 26.

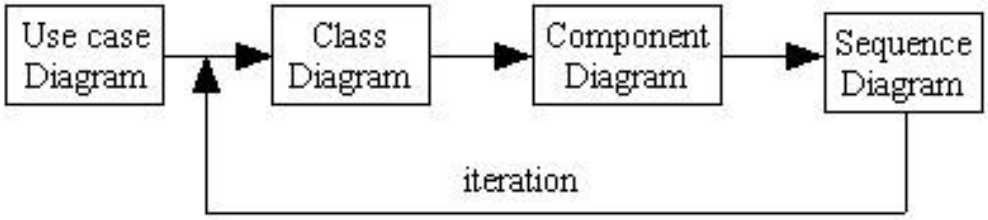

Figure 26 Specification process of OO

We assume that the functional user requirements for an OO system are descried through the following steps:

- The first step: construct Use case diagram;
- The second step: construct class diagram;
- The third step: construct component diagram;
- The last step: construct sequence diagrams.

This model-based measurement-oriented FSM method consists of two steps. The first step is to specify the FURs using appropriate UML diagrams, according to the modeling rules; the second step is to identify, count and calculate the function point according to the mapping rules and measure rules. [21] [22]

4.2 The Case of Warehouse Software Portfolio

We use as an example the Warehouse Software Portfolio (WSP) by Fetcke [55]. The version of the WSP used here includes a few marginal changes with respect to Fetcke's version.

The WSP is a collection of overlapping applications for warehouse management. The Entity/Relationship diagram representing the entities involved in the WSP is given in Figure 27. The entities and their attributes are described in Figure 28. Both figures are from [55]. Attributes Owner and Storage place are references to entities Customer and Place, respectively.

Figure 27 Entity/Relationship diagram of the WSP

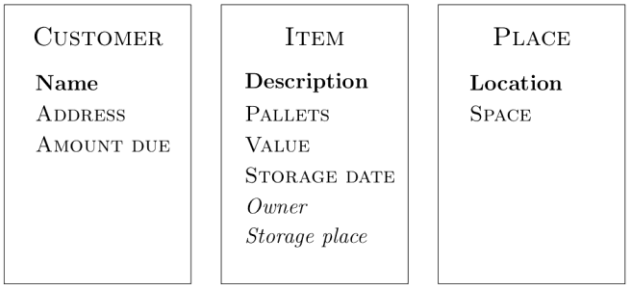

Figure 28 Entities of the WSP

The transactions supported by the WSP are:

- Add customer
- Change customer data
- Delete customer
- Receive payment
- Deposit item
- Retrieve item
- Add place
- Change place data
- Delete place
- Print customer item list
- Print bill
- Print stored items list
- Query customers
- Query customer's items
- Query places
- Query stored items

Because of limited space we cannot give the detailed requirements for these transactions. The complete FUR of the WSP can be found in [55].

4.3 Model-based measurement of Function Points

As already described in 2.2, Function Point Analysis assumes that user requirements are composed of Data and Transaction functions (see Figure 6), and the latter are characterized in terms of RET, DET and FTR (see Figure 7).

4.3.1 Representing data function

In the IFPUG manual [11] there are a few relevant indications concerning data functions:

- Logical data files are *"logically related groups of data"*.
- The RET is defined as "*a record element type (RET) is a user recognizable subgroup of data elements within an ILF or EIF*."

– The DET is defined as follows "*a data element type (DET) is a unique user recognizable, non-repeated field*."

There are similarities between FPA and object-oriented concepts. For instance, a logical file in the function point approach is a collection of related data (which are user-identifiable, if the class is defined in a model of user requirements).

Therefore, the class is the natural candidate for representing logical files using the object-oriented paradigm, but a class tends to represent information at a lower level of granularity than a FPA logic file. In some cases, it is possible to identify a class as a logical file, in some cases a set of classes should be identified as a logical file, i.e., objects that are instances of a class correspond to records (RETs) of a logical file in data processing applications. Lavazza et al. found a good way of presenting logical file using UML component [3].

So, FPA concepts are mapped onto object-oriented concepts as follows:
– Logical data files (either ILF or EIF) are represented as (conceptual) components that include data (and the methods that are needed to manipulate those data).
– RETs can be represented by classes within components. Since each RET belongs to a data file, each class representing a RET belongs to the component representing the corresponding data file.
– DETs can be represented as class attributes.

Within the system component (i.e., the component representing the application to be measured), a subcomponent has to be introduced for every logically related group of data that are managed (i.e., created, updated, deleted, etc.) by the application and that are user identifiable (i.e., that have a precise meaning for the user, according to the user requirements) [3].

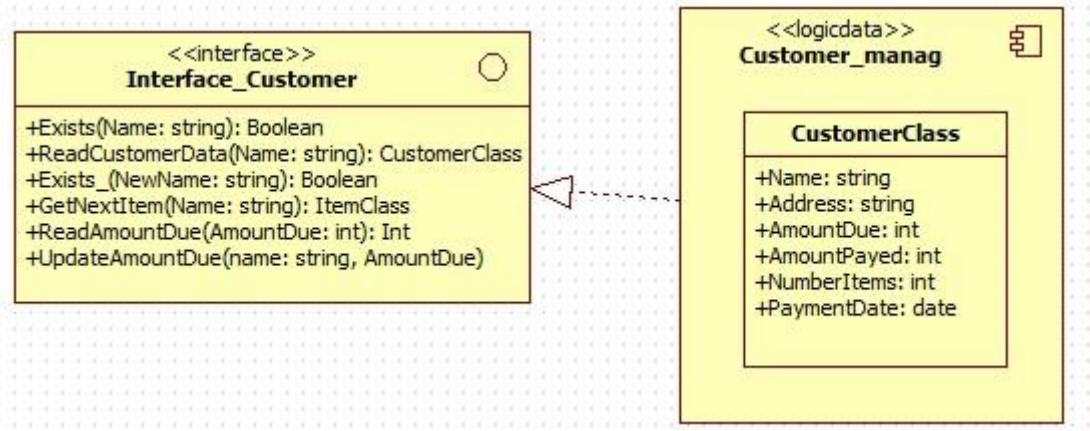

Figure 29 Component of Customer_manag

4.3.2 Representing elementary process

According to the IFPUG manual [11], "*An elementary process is the smallest unit of activity that is meaningful to the user(s). The elementary process must be self-contained and leave the business of the application being counted in a consistent state."*

In the specification of UML [52] use cases are described as follows: "*Each use case specifies some behavior, possibly including variants, that the subject can perform in*

collaboration with one or more actors. […] These behaviours, involving interactions between the actor and the subject, may result in changes to the state of the subject and communications with its environment. […] Each use case specifies a unit of useful functionality that the subject provides to its users […]. This functionality, which is initiated by an actor, must always be completed for the use case to complete. It is deemed complete if, after its execution, the subject will be in a state in which no further inputs or actions are expected and the use case can be initiated again or in an error state.

So, it seems that use cases are compatible with the concept of elementary processes. Actually, a use case could be used to represent a set of related elementary processes. We impose a modeling discipline in order to make sure that there is a one-to-one relationship between use cases and elementary processes.

However, it is easy to observe that the amount of information reported in UML use case diagrams is not sufficient to measure them, since the FTR involved in a process and the DET that cross the boundaries of the application are not explicitly mentioned in use case diagrams.

In UML, sequence diagrams represent interactions taking the form of sequences of messages exchanged among objects within collaborations to effect a desired operation or achieve a result. These collaborations match quite closely the definition of elementary processes. In any case, it is possible to model an elementary process by means of a sequence diagram.
Sequence diagrams are suitable for modeling the information required to measure elementary processes, because:

- FTR can be represented as instances of components that take part in the collaboration.
- DET crossing the boundaries of the application are represented by the parameters of messages that cross the boundaries of the application.
- A sequence diagram shows the meaning of the process at a quite detailed level, therefore it is possible to evaluate the main purpose of the process and classify it as EI, EO or EQ.

In conclusion, we represent transaction functions by means of sequence diagrams.

4.3.3 Sequence diagrams

The sequence diagram has two axes. The horizontal axis identifies the participants and the corresponding lifelines. According to our definition, users, the system (i.e., the application to be measured), logic data (both internal and external ones) and external systems are the participants of sequence diagrams. An example is given in Figure 30: *:operator* is the user of the application, *:System* is the application itself (more precisely, the component that represents the application), *:Customer_manag* is a FTR, that is, a data file used within the application.

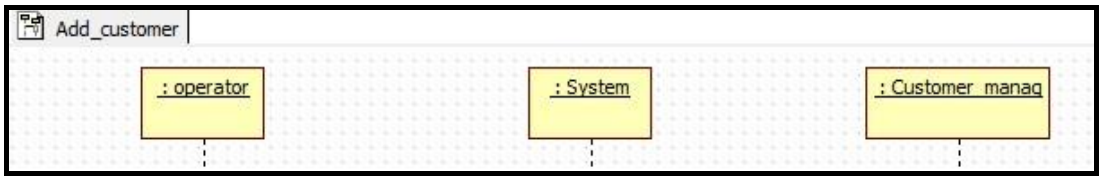

Figure 30 Horizontal axis of a sequence diagram

The other axis of sequence diagrams, the vertical axis, represents time, which increases from top to bottom. A sequence of messages represents a scenario of a use case. A message may have one, multiple, or no parameter. A Message that crosses the boundaries of the system (e.g., a message from the User to the System) carries parameters that represent DETs.

For each sequence diagram, the main intent must be labeled for identifying that the corresponding elementary process is of type of EI, EO, or EQ.

The Add customer transaction adds a record of Customer data to the database. The attributes Name and Address have to be entered. The Amount due is initialized to zero. When the user presses the Add button, the customer record is added into the database. If, however, a customer with the Name entered already exists, the data is not added and an error message is displayed. The user may abort this transaction with the Cancel button. The user interface of the Add customer transaction is illustrated in Figure 31 *on the facing page.*

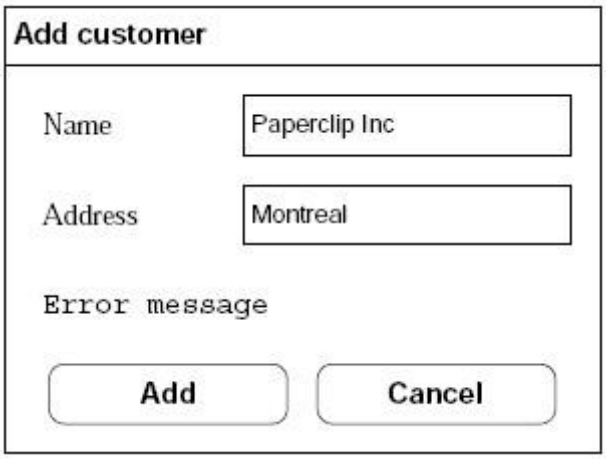

Figure 31 User interface of the Add customer transaction

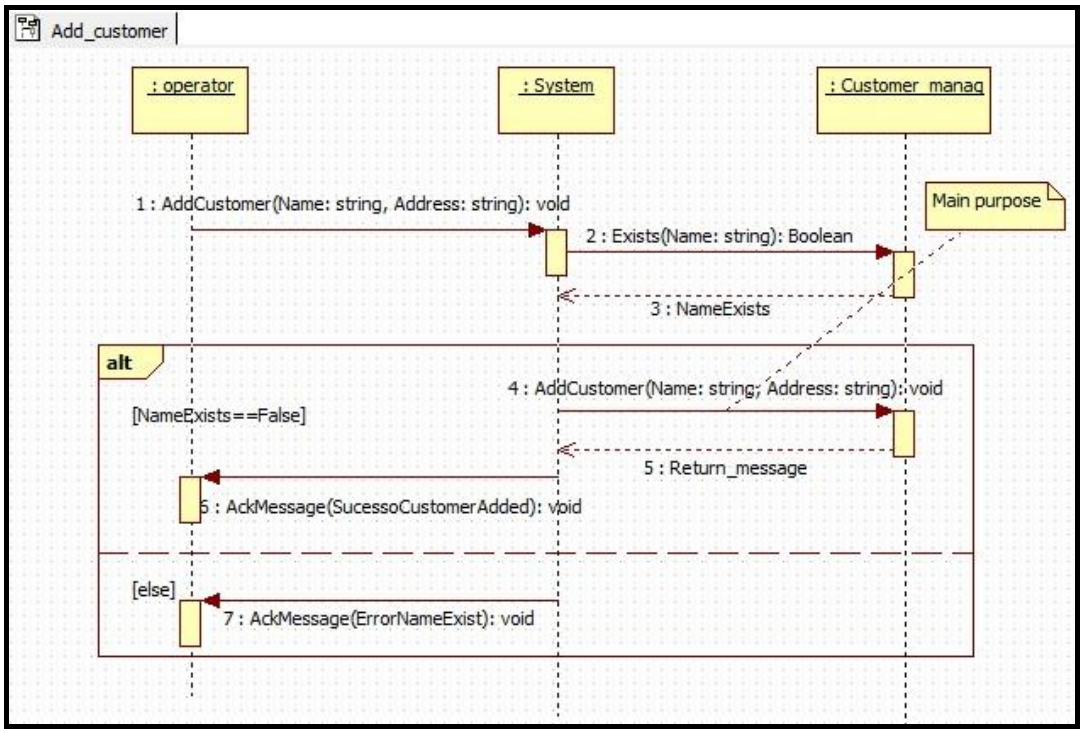

Figure 32 Sequence diagram of the Add customer transaction (FPA method)

The mapping between IFPUG-FPA concepts and UML constructs and elements is summarized in Table 21.

Table 21 FPA-UML element mapping

FPA	UML	
	diagram	**element**
Application boundary	Use case diagram	Boundary of the subject that owns the use cases
Elementary process	Use case diagram	Use case
Users	Use case diagram	Actors
EIF	Use case diagram	Actors
External systems	Use case diagram	Actors
Application being measured	Component diagram	<<system>> component
Logical data file	Component diagram	<<logic data>> component
RET	Component diagram	Class belonging to <<logic data>> component
DET	Component diagram	Class attribute
ILF	Component diagram	<<logic data>> component within system component
EIF	Component diagram	<<logic data>> component outside system component
Elementary process (transaction function)	Component diagram	Operation provided by the system component interface
Elementary process (transaction function)	Sequence diagram	(whole diagram)

FTR	Sequence diagram	Instance of <<logic data>> component
DET crossing boundary in transaction	Sequence diagram	Argument (of attributes granularity) of messages crossing the boundaries

4.3.4 The counting procedure

Here the FPA counting procedure is redefined, in order to be applicable to a UML model which is built according to the rules reported in above subsections.

Counting EIFs and ILFs

Identifying ILFs and EIFs is immediate: both are components stereotyped <<LogicData>>; while ILFs are within the boundaries of the application, i.e., in the application component, EIFs are outside. In order to weight ILFs and EIFs, we need to count their RETs and DETs.

In general we count a RET for every class in the data component.

For data components that contain just a single class we count 1 RET, since there are no data subgroups from the user perspective, but just the main group represented by the class.

For data components including more classes, the number of RETs depends on the relations between classes:
- Classes connected by associations are counted as RETs.
- Composition and aggregation relations are treated like plain associations.
- In a generalization/specialization hierarchy, we count the classes that can be instantiated. I.e., a abstract classes are not counted, since they cannot be instantiated.

Thus, for the ILF *Customer_manag* (Figure 29) we count 1 RET.

Counting the DETs is relatively simple: we count a DET for each attribute of the class(es) belonging to the data component (remember that attributes are nonrepeated by construction).
- For data functions containing just one class, the number of DETs is equal to the number of the class's attributes;
- For classes connected by an association relation, the number of DETs is again equal to the sum of the number of attributes of the classes;
- For classes connected by composition relations, the number of DETs is equal to the sum of the number of attributes of the classes;
- When generalization is involved, according to the FPA counting rules we must take into consideration "nonrepeated" attributes. Therefore the attributes of a super-class are counted just once, regardless how many sub-classes inherit them. We also count an additional DET for each subclass, in order to take into account the specialization criterion.

Thus, for the ILF *Customer_manag* (Figure 29) we count 6 DETs.

Counting transaction functions

Transactions are identified in a straightforward way. The indication of the main intent of the transaction is evaluated by the analyst and made directly available to the measurer, who can take into account this piece of information to classify the given transaction as an EI or as an EO. For this purpose we have to identify the FTR and the DET.

Counting FTR is immediate: we just have to count how many ILF and EIF are referenced, i.e., how many ILF and EIF appear in the sequence diagram that describes the considered transaction. For instance, in the transaction that *Add customer* transaction (Figure 32), only the *Customer_manag* ILF is referenced, thus FTR=1.

DET to be considered are the ones that cross the boundary: in the transaction *Add customer* the 6 parameters of the invoked function and the return message are counted as DET. In addition, the counting rules require that a DET is added for the ability to specify an action to be taken [3]. In conclusion, the *Add customer* transaction is an EI, having 1 FTR and 7 DETs therefore, according to [10], it is a Low complexity EI, which contributes 3 FP.

4.4 Model-based measurement of COSMIC FP

As already described in Section 2.3, the COSMIC method assumes that user requirements are composed of functional processes (see Figure 11), which are characterized in terms of data movements (Write, Read, Entry, and Exit) (see Figure 12).

4.4.1 Representing functional process

According to the COSMIC manual [33], "*A functional process is an elementary component of a set of Functional User Requirements comprising a unique, cohesive and independently executable set of data movements. It is triggered by a data movement (an Entry) from a functional user that informs the piece of software that the functional user has identified a triggering event. It is complete when it has executed all that is required to be done in response to the triggering event."*

The COSMIC method recognizes four types of data movement and defines a data group as the data element that is subject to movements: a data group consists of a non redundant set of data attributes. The concept of data group in the COSMIC method matches very closely the class construct in OO (and UML) model.

It seems that use cases are compatible with the concept of functional process, as they are compatible with the concept of elementary process in FPA. However, it is easy to observe that the amount of information reported in UML use case diagrams is not sufficient to measure them, since the indication of data groups that cross the boundaries of the application are not explicitly mentioned in use case diagrams.

Sequence diagrams match quite closely the definition of functional process. In general, it is possible to model a functional process by means of a sequence diagram.

Sequence diagrams are suitable for modeling the information required to measure functional process, because:

- Data group can be represented as instances of classes that take part in the collaboration.
- Data group crossing the boundaries of the application are represented by the parameters of messages that cross the boundaries of the application, hence indicating entries and exits. Since data movements involve data groups, message parameter must be instances of classes, not single attributes.
- Messages directed to instances of classes within the application represent read or write operations (hence data movements).
- A sequence diagram shows the meaning of the process at a quite detailed level; therefore it is possible to classify data movement as Read, Write, Entry or Exit.

In conclusion, we represent functional process by means of sequence diagrams.

4.4.2 Sequence diagram

The sequence diagrams used to specify functional processes are very much like those used to represent FPA's transactions.

An important difference is that in COSMIC sequence diagrams, the instances of classes are used, instead of the instances of components. An example is given in Figure 33: *:Operator* is the user of the application, *:System* is the application itself (more precisely, the component that represents the application), *:CustomerClass* is an instance of the class CostomerClass (a data group used within the application). (In this example, no external system exists).

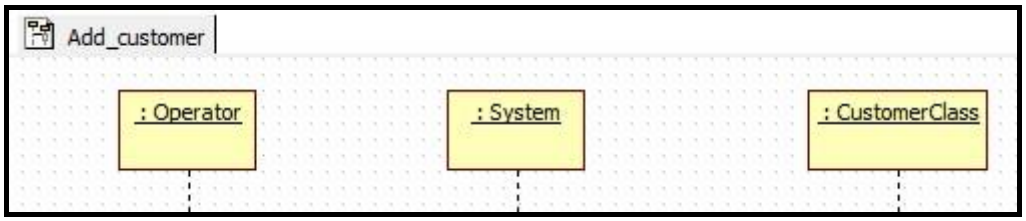

Figure 33 Horizontal axis of a sequence diagram

In COSMIC sequence diagrams, a problem is that if a message has two arguments, it is unknown to which class(es) they belong. If the both arguments belong to the same class, we have one data movement, and one CFP should be counted. If the arguments belong to two different classes, we have two data movements, and two CFPs should be counted. In view of the above reasons, we describe the argument with the prefix of the class involved, for instance, argument = class_ name. attribute_ name. (see the example in Figure 34)

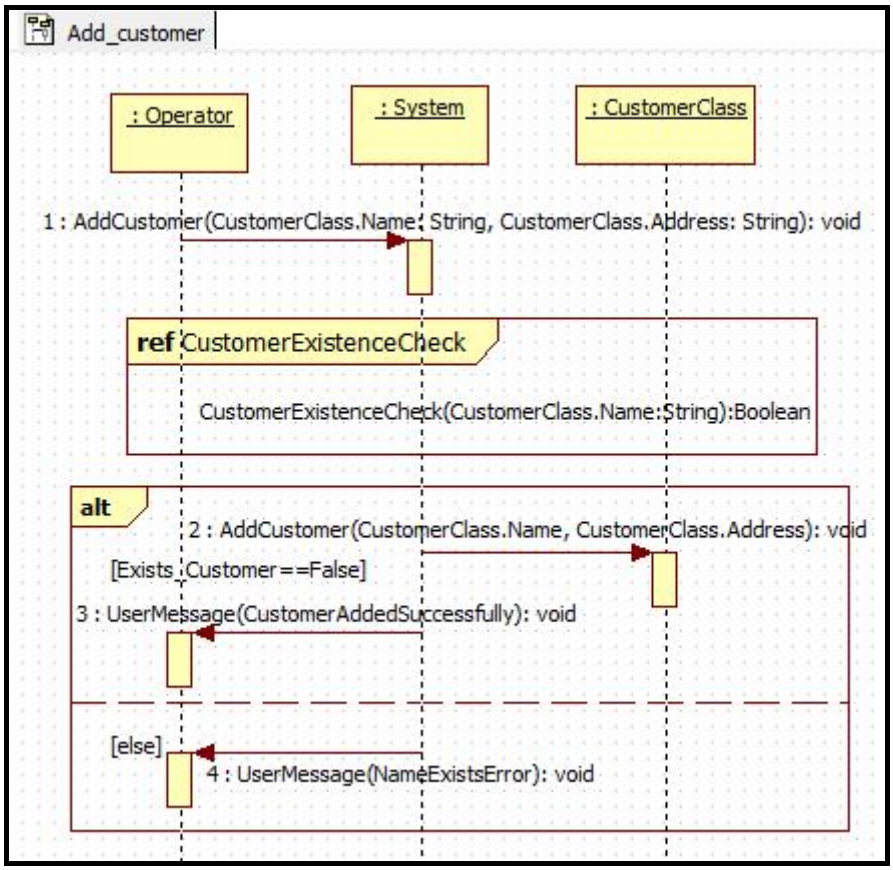

Figure 34 Sequence diagram of the Add customer transaction (COSMIC method)

In Figure 34, the sequence references the "CustomerExistenceCheck" sequence diagram. The "CustomerExistenceCheck" sequence diagram is shown in Figure 35.

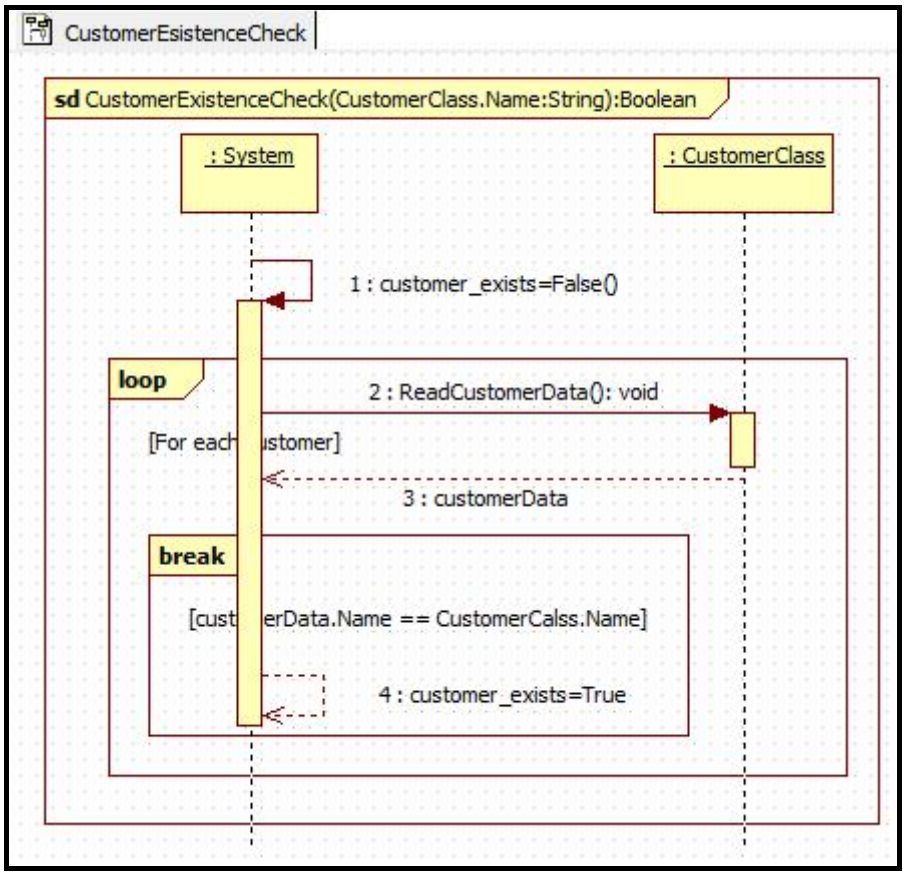

Figure 35 Sequence diagram of CustomerEsistenceCheck

The mapping between COSMIC concepts and UML constructs and elements is summarized in Table 22.

Table 22 COSMIC-UML element mapping

COSMIC	UML	
	Diagram	**element**
Application border	Use case	Boundary of the subject
	component	Boundary of the system component
Functional User	Use case	Agent directly connected with a use case
	component	External component directly connected with the system
Triggering event	component	Operation in interface realized by the system and invoked spontaneously by an active external component
Persistent data group	component	Class
	Class	Class
	Sequence	Class instance
Transient data group	component	Data cross the boundaries of the system: operations of the interfaces, or the parameters of these operation to the interface
Process	Use case	Use case
	Sequence	Sequence diagram
Entry data movement	Sequence	Message from external component to the system
Exit data movement	Sequence	Message from the system to external component
Read data movement	Sequence	Message involving persistent data from system to instance of class within the system

4.4.3 The counting procedure

According to the COSMIC measurement manual [33], the size of an application is given by the sum of the sizes of its functional processes; the size of each functional process is given by the number of its data movements (excluding repetitions). According to our method, a functional process is represented by a Sequence Diagram, thus we must be able to measure the size of a sequence diagram.

Sequence Diagrams include messages, some of which represent data movements. A single message can account for several data movements: in fact, every message argument that is an instance of a Data Group counts as a distinct data movement (except for repetitions, as discussed below). In UML diagrams, Data Groups are represented as classes.

In practice, our method requires that, given a sequence diagram:
- All messages representing data movements are identified. Messages that represent data movements are: the ones that enter or exit the system and the ones that involve reading or writing data stored within the system.
- For each message, the arguments and return values that are instances of data group class are identified (in general all the arguments and returned values should be instances of data group classes). These are the potential data movements.
- The potential data movements are classified as entries, exits, writes or reads, using the indications reported in Table 22.
- Duplicates are eliminated.
- The number of remaining data movements is the size of the sequence diagram representing a functional process.

The procedure described above is quite straightforward. A bit of attention is required in eliminating the duplicate movements: in this phase we are supported by the COSMIC rule “Data movement uniqueness and possible exceptions” (see [33] pp. 49-51).

Transient data groups require a precision: in some cases it may happen that the output of functional process groups attributes from different classes. The latter are treated as a transient data group.

The size of the *Add_customer* functional process (Figure 34) is 4 CFP:
- Entry of Customer.class;
- Write of Customer.class;
- Read of CustomerData
- Exit of user message

4.5 Similarities and differences

The similarities and fundamental differences between the model-based FPA measurement and the model-based COSMIC measurement are described in this section.

In order to properly understand the relation between COSMIC and IFPUG FP a comparative evaluation has to be performed. The evaluation has to be performed not only from a quantitative point of view, but also taking into consideration the differences in the underlying concepts. To this end, the availability of two UML measurement oriented models (UML based), built to ease the IFPUG and COSMIC FP measurement respectively, of the same software system, is clearly beneficial.

In this section, we not only compare the definition and the purpose of these FSM methods, but also compare both methods mainly from the view point of model-based measurement proposed by L. Lavazza et al. [2][3].

4.5.1 Requirements and procedure

Both FPA and COSMIC measure user functional requirements. Use cases are coincident, because their functional processes and elementary processes are essentially the same concept.

4.5.2 Data modeling: Class and Component diagrams

Classes are used in both method models; however, there are big differences in how they are used. In the COSMIC method, each class is directly mapped to a data group; therefore a class diagram is perfectly suited to represent all the data groups.

In the FPA method, one or more classes are possibly grouped into a single logic file. Grouping related classes in a class diagram is difficult (and not “natural”), so components are used to model logic files.

The topmost level component diagram is equal for FPA and COSMIC. A difference is that the system component contains

- The whole class diagram in the COSMIC models.
- Several <<logic data>> components in the FPA models. On their turn, these components include classes.

4.5.3 Process modeling: Sequence diagram

In FPA sequence diagrams, the participants that represent internal parts of the system are instance of <<logic data>> component.

In COSMIC sequence diagrams, the participants that represent internal parts of the system (data groups) are instances of classes.

The format of message arguments is different. In FPA models, method arguments are class attributes, corresponding to DETs, since we are interested in DETs crossing the boundaries of the application. In COSMIC, a data movement involves a data group, i.e., a class. This means that a message having multiple attributes as arguments is potentially ambiguous: if the attributes belong to the same class it is one data movement; if they belong to two distinct classes we have two data movements, etc. To solve this issue, arguments must be prefixed with their class name.

In FPA sequence diagrams, the main intent of transaction function must be indicated via a comment to identify the type of transaction. This is not necessary for COSMIC-oriented sequence diagrams.

4.5.4 Others differences

There are another couple of differences in the definition of FPA and COSMIC that affects the way models are built.

COSMIC counts data movements that “read from” and “write to” the permanent storage. Having methods that both read and write would make the identification of data movements difficult (e.g., a given method could be a read, a write or both). Therefore, when building Measurement-Oriented (namely COSMIC measurement-oriented models) one should be careful to introduce only methods that either read from or write to classes that represent data groups. To easy the counting, methods could be stereotyped as <<read>> or <<write>>. This problem does not occur in FPA-oriented models.

Both COSMIC and FPA do not consider duplicate operations. Accordingly, a measurement-oriented model can safely skip the representation of duplicate operations.

Unfortunately, COSMIC has exceptions to this simple rule: the exceptions rarely take place; nevertheless they have to be considered. Hence the duplicated movements have to be annotated to be correctly recognizable. This issue does not apply to FPA-oriented models.

This page intentionally left blank.

Chapter 5 Evaluation of Simplified FSM processes

As we discussed in Chapter 3, there are many simplified measurement methods. In order to explore our own model-based simplified method, first of all we evaluate the existing simplified methods by exploring the accuracy of sizing with respect to full-fledged Function Point Analysis and their suitable applicable field. This work not only helps us to better understand and use the existing simplified methods, to assess the feasibility of our model-based simplified method, but also provides us a reference to evaluate other (more or less simplified) measurement methods.

5.1 Empirical assessment of Simplified FSM proposals

This section is dedicated to the evaluation of simplified methods aiming at providing size measure (or estimates) in Function Points. That is, the considered methods simplify the IFPUG measurement process.

5.1.1 Method of empirical assessment and procedure of the work

In order to perfom the evalution, we collected 18 projects' FURs which were modeled using UML as described in [11]. These 18 projects are divided into two groups according to the application type. One group consists of 9 "traditional" applications and the other group consists of 9 Real-Time applications.

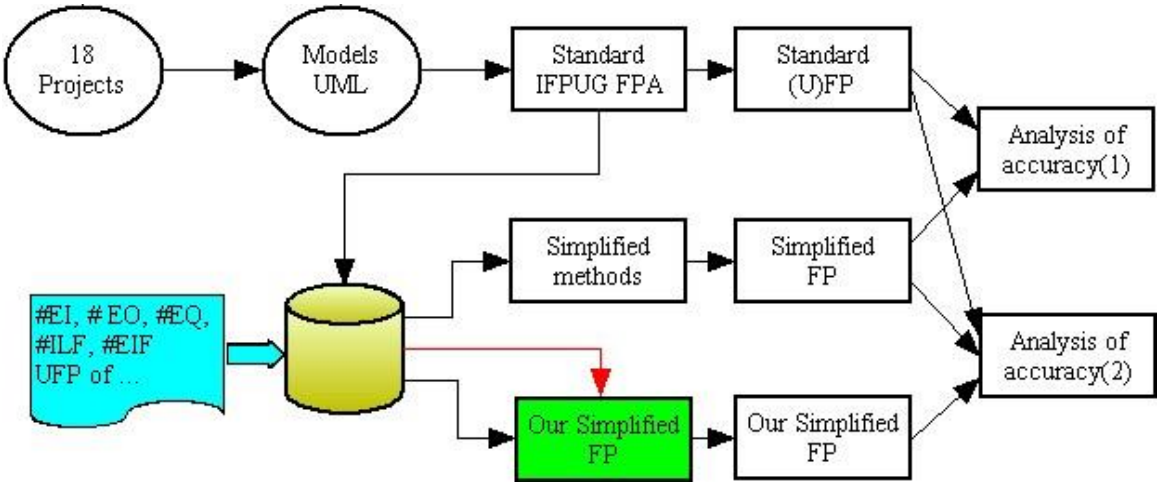

Figure 36 Research Road map of this work

Figure 36 shows how the work was carried out. The process was organized in 7 steps.

- First, a model of each application was built. The models were written in UML and represented the requirements, including all the information needed for the measurement of FPs and excluding the unnecessary details [11].
- Second, on the basis of the above models, we measured the applications according to IFPUG measurement rules [4], thus obtaining the "correct" measures.
- In the third step the 18 projects, which had already been measured by means of Function Point Analysis, have been measured using a few simplified processes,

including those proposed by NESMA, the Early&Quick Function Points, the ISBSG average weights, and a few others.
- In the fourth step, the resulting size measures were compared with those obtained at step 2, for evaluating the accuracy of sizing with respect to full-fledged FPA.
- We also derived a simplified size model on the basis of the measures from the dataset used for experimentations. We also derived simplified size models by analyzing the dataset used for experimentations.
- We used such model to estimate the size of the projects in our dataset.
- Finally, we compared the measurements obtained at step 6 with those obtained at set 2.

5.1.2 The case study and the dataset obtained from the standard FPA measurement

A. Real-Time projects

Most of the Real-Time projects measured are from a European organization that develops avionic applications, and other types of embedded and Real-Time applications. All the measured projects concerned typical Real-Time applications for avionics or electro-optical projects, and involved algorithms, interface management, process control and graphical visualization.

The projects' FURs were modeled using UML as described in Chapter 4 , and then were measured according to IFPUG measurement rules as described in Section 2.2. When the Real-Time nature of the software made IFPUG guidelines inapplicable, we adopted ad-hoc counting criteria, using common sense and striving to preserve the principles of FPA, as described in [56]. The same projects were then sized using the simplified functional size measurement processes mentioned in Section II, using the data that were already available as a result of the IFPUG measurement.

For each project, the measurement of the functional size was carried out in two steps. First, a model of the product was built. The models were written in UML and represented the requirements, including all the information needed for the measurement of FPs and excluding the unnecessary details [57]. Then, the function points were counted, on the basis of the model, according to IFPUG rules.

Table 23 Real-Time Projects' Size (IFPUG method)

Project ID.	ILF	EIF	EI	EO	EQ	FP	TF	DF
1	164	5	90	8	22	289	120	169
	(18)	(1)	(21)	(2)	(5)		(28)	(19)
2	56	0	21	18	6	101	45	56
	(8)	(0)	(6)	(3)	(1)		(10)	(8)
3	73	0	12	47	4	136	63	73
	(7)	(0)	(2)	(8)	(1)		(11)	(7)
4	130	15	44	0	6	195	50	145
	(15)	(3)	(11)	(0)	(1)		(12)	(18)
5	39	0	28	39	0	106	67	39
	(4)	(0)	(8)	(8)	(0)		(16)	(4)
6	71	5	8	139	0	223	147	76

	(9)	(1)	(2)	(28)	(0)		(30)	(10)
7	7	0	3	5	0	15	8	7
	(1)	(0)	(1)	(1)	(0)		(2)	(1)
8	21	0	4	8	0	33	12	21
	(3)	(0)	(1)	(2)	(0)		(3)	(3)
9	21	0	7	16	0	44	23	21
	(3)	(0)	(2)	(4)	(0)		(6)	(3)

Table 23 reports the size in UFP of the measured projects, together with the BFC and – in parentheses– the number of unweighted BFC. For instance, project 1 involved 18 Internal Logic Files, having a size of 164 FP. TF (Transaction Function) and DT (Data Functions) are the sizes of transaction (i.e., EI, EO, and EQ) and data (i.e., ILF and EIF), respectively.

B. Non Real-Time projects

The considered non Real-Time projects are mostly programs that allow users to play board or card games vs. remote players via the internet; a few ones are typical business information systems.

The projects were measured –as the Real-Time ones– in two steps: the UML model of each product was built along the guidelines described in [3]; then, the function points were counted, on the basis of the model, according to IFPUG rules.

Table 24 reports the size in UFP of the measured projects, together with the BFC and – in parentheses– the number of unweighted BFC.

Table 24 Non Real-Time Projects' sizes (IFPUG method)

Project ID.	ILF	EIF	EI	EO	EQ	FP	TF	DF
1	45 (6)	7 (1)	34 (10)	6 (1)	0 (0)	92	40 (11)	52 (7)
2	28 (4)	20 (4)	37 (9)	5 (1)	4 (1)	94	46 (11)	48 (8)
3	21 (3)	5 (1)	27 (7)	8 (2)	18 (6)	79	53 (15)	26 (4)
4	31 (16)	0 (0)	49 (22)	13 (5)	3 (2)	96	65 (20)	31 (4)
5	24 (3)	0 (0)	45 (14)	21 (5)	0 (0)	90	66 (19)	24 (3)
6	49 (7)	0 (0)	36 (9)	0 (0)	6 (2)	91	42 (11)	49 (7)
7	21 (3)	0 (0)	31 (9)	14 (3)	14 (4)	80	59 (16)	21 (3)
8	42 (6)	5 (1)	35 (9)	17 (3)	10 (2)	109	62 (14)	47 (7)
9	21 (3)	0 (0)	38 (11)	15 (5)	8 (2)	82	61 (18)	21 (3)

5.1.3 Application of simplified methods for getting relative results

Simplified measurement processes were applied following their definitions, which require data that can be easily derived from the tables above. So, for instance, the data required for Real-Time project 1 are as following:

- The NESMA indicative method requires the numbers of ILF and EIF. Table 23 shows that the number of ILF is 18, and the number of EIF is 1.
- Similarly, the Tichenor ILF model and the ISBSG distribution models just require the ILF number, i.e., 18.
- The NESMA estimated method, the E&QFP generic functions method, the sFP method and the ISBSG average weights method require the numbers of ILF, EIF, EI, EO, and EQ. Table 23 shows that the numbers of ILF, EIF, EI, EO, and EQ are, respectively, 18, 1, 21, 2, and 5.
- The E&QFP unspecified generic functions method requires the numbers of data groups (that is, the number of ILF plus the number of EIF) and the number of transactions (that is, the sum of the numbers of EI, EO, and EQ). Table 23 shows that the number of data groups is 18+1 = 19, and the number of transactions is 21+2+5 = 28.

A, Applying NESMA indicative

The applications to be measured were modeled according to the guidelines described in [57]. The logic data files – modeled as UML classes– provide a data model that cannot be easily recognized as normalized or not normalized. Therefore, we applied both the formulae for the normalized and non normalized models.

The formulae of the NESMA indicative method were applied to the number of ILF and EIF that had been identified during the IFPUG function point counting process. The results are given in Table 25 for Real-Time projects and in Table 26 for non Real-Time projects.

Table 25 Sizes of Real-Time projects obtained via the NESMA methods

Project ID	IFPUG	NESMA ind. non norm.	NESMA ind. norm.	NESMA estim.
1	289	645	460	245
2	101	280	200	99
3	136	245	175	101
4	195	570	405	168
5	106	140	100	100
6	223	330	235	216
7	15	35	25	16
8	33	105	75	35
9	44	105	75	49

B, Applying NESMA estimated

The formulae of the NESMA indicative method were applied to the number of ILF, EIF, EI, EO, and EQ that had been identified during the IFPUG function point counting process. The results are given in Table 25 for Real-Time projects and in Table 26 for non Real-Time projects.

Table 26 Sizes of NON Real-Time projects obtained via the NESMA methods

Project ID	IFPUG	NESMA ind. non norm.	NESMA ind. norm.	NESMA estim.
1	92	225	160	92
2	94	200	140	93
3	79	120	85	88
4	96	140	100	111
5	90	105	75	102
6	91	245	175	93
7	80	105	75	88
8	109	225	160	106
9	82	105	75	98

C, Applying E&QFP

As described in Figure 21, the E&QFP method can be applied at different levels. Since we had the necessary data, we used the BFC aggregation level. At this level it is possible to use the data functions and transaction functions without weighting them or even without classifying transactions into EI, EO, and EQ and logic data into ILF and EIF. In the former case (generic functions) the weights given in Table 1 are used, while in the latter case (unspecified generic functions) the weights given in Table 2 are used.

The results of the application of E&QFP are given in Table 27 for Real-Time projects and in Table 28 for non Real-Time projects.

Table 27 Sizes of Real-Time projects obtained via the E&QFP method

Project ID	IFPUG	EQFP unspec.	EQFP generic
1	289	262	262
2	101	102	106
3	136	100	108
4	195	181	182
5	106	102	106
6	223	208	229
7	15	16	17
8	33	35	38
9	44	49	52

Table 28 Sizes of NON Real-Time projects obtained via the E&QFP method

Proj ID	IFPUG	EQFP unspec.	EQFP generic
1	92	100	99
2	94	107	99
3	79	97	92
4	96	120	118

5	90	108	108
6	91	100	100
7	80	95	92
8	109	113	113
9	82	104	103

D, Applying Tichenor ILF Model

In order to apply the model we just had to multiply the number of ILF of each of our projects for the constant 14.93 suggested by Tichenor. The obtained results are illustrated in Table 29 and Table 30 for Real-Time and non Real-Time projects, respectively.

When applying this method, it should be remembered that the results are likely to be incorrect if the distribution of BFC in the estimated application does not match the distribution observed by Tichenor. Accordingly, when applying the method, one should also check the distribution of BFC. Unfortunately, this implies making more work, namely, one should count the number of EIF, EI, EO, and EQ in addition to ILF. Even worse, one could discover that the distribution of his/her application is different from the distribution assumed by Tichenor, so that the estimated size is not reliable.

In our case, the projects do not appear to fit well in the distribution assumed by Tichenor: the differences between the measured ratios and the ratios expected by Tichenor are the following:

– For Real-Time projects: 14.3% for EI/ILF, 43.7% for EO/ILF, 3.9% for EQ/ILF, 7.9% for EIF/ILF.
– For non Real-Time projects: 96.7% for EI/ILF, 22.2% for EO/ILF, 27.3% for EQ/ILF, 14.7% for EIF/ILF.

In practice, our projects have a very different distribution of BFC sizes with respect to Tichenor expectations (for instance, in Real-Time projects EI had often a larger size than ILF, while it is expected that the size of EI is about one third of the size of ILF). So, we must expect a quite poor accuracy from Tichenor estimates.

Table 29 Sizes of Real-Time projects obtained via Tichenor ILF model, ISBSG distribution sFP and ISBSG average weights methods

Proj ID	IFPUG	Tichenor ILF model	ISBSG distrib.	sFP	ISBSG average weights
1	289	269	597	301	259
2	101	119	265	123	105
3	136	105	232	122	107
4	195	224	498	219	179
5	106	60	133	112	107
6	223	134	299	245	232
7	15	15	33	19	17
8	33	45	100	44	37
9	44	45	100	58	52

Table 30 Sizes of NON Real-Time projects obtained via Tichenor ILF model, ISBSG distribution sFP and ISBSG average weights methods

Project ID	IFPUG	Tichenor ILF model	ISBSG distrib.	sFP	ISBSG average weights
1	92	90	199	112	98
2	94	60	133	113	100
3	79	45	100	99	91
4	96	60	133	123	118
5	90	45	100	111	109
6	91	105	232	114	98
7	80	45	100	97	92
8	109	90	199	126	112
9	82	45	100	107	104

E, Applying the ISBSG distribution model

We applied the formula UFP = (#ILF × 7.4) × 100 / 22.3 prescribed by the method. Then, we evaluated the differences between the measured percentage contribution of BFC and the ISBSG averages. The differences we found were relatively small:

- For Real-Time projects: 28.7% for ILF, 3.4% for EIF, 19.3% for EI, 21.3% for EO, 13.2% for EQ.
- For non Real-Time projects: 12% for ILF, 4.8% for EIF, 5.6% for EI, 15.4% for EO, 13.2% for EQ.

Accordingly, we expect that the ISBSG distribution model applies well to our dataset, especially as non Real-Time projects are involved.

The obtained results are illustrated in Table 29 and Table 30 for Real-Time and non Real-Time projects, respectively.

F, Applying the sFP and ISBSG average weights

The application of the sFP and ISBSG average weights methods was extremely similar to the application of the NESMA estimated and E&QFP generic methods, only the values of weights being different.

The obtained results are illustrated in Table 29 and Table 30 for Real-Time and non Real-Time projects, respectively.

5.1.4 Summary and lessons learned

In this section, the results of our empirical analysis are reported. First we discuss the quantitative results, and then we analyze the results from a more theoretical point of view.

A. Quantitative analysis

To ease comparisons, all the size measures of RT projects are reported in Table 31 and those of non RT projects are reported in Table 32.

Table 31 Measures of Real-Time Projects obtained via the Various Methods

Proj ID	IFPUG	NESMA ind. non norm.	NESMA ind. norm.	NESMA estim.	E&QFP unspec.	E&QFP generic	Tichenor ILF model	ISBSG distrib.	sFP	ISBSG average weights
1	289	645	460	245	262	262	269	597	301	259
2	101	280	200	99	102	106	119	265	123	105
3	136	245	175	101	100	108	105	232	122	107
4	195	570	405	168	181	182	224	498	219	179
5	106	140	100	100	102	106	60	133	112	107
6	223	330	235	216	208	229	134	299	245	232
7	15	35	25	16	16	17	15	33	19	17
8	33	105	75	35	35	38	45	100	44	37
9	44	105	75	49	49	52	45	100	58	52

Table 32 Measures of NON Real-Time Projects obtained via the Various Methods

Proj ID	IFPUG	NESMA ind. non norm.	NESMA ind. norm.	NESMA estim.	E&QFP unspec.	E&QFP generic	Tichenor ILF model	ISBSG distrib.	sFP	ISBSG average weights
1	92	225	160	92	100	99	90	199	112	98
2	94	200	140	93	107	99	60	133	113	100
3	79	120	85	88	97	92	45	100	99	91
4	96	140	100	111	120	118	60	133	123	118
5	90	105	75	102	108	108	45	100	111	109
6	91	245	175	93	100	100	105	232	114	98
7	80	105	75	88	95	92	45	100	97	92
8	109	225	160	106	113	113	90	199	126	112
9	82	105	75	98	104	103	45	100	107	104

The relative measurement errors are given in Table 33 and Table 34.

Table 33 Relative measurement errors (Real-Time Projects)

Proj ID	NESMA ind. non norm.	NESMA ind. norm.	NESMA estim.	E&QFP unspec.	E&QFP generic	Tichenor ILF model	ISBSG distrib.	sFP	ISBSG average weights
1	123%	59%	-15%	-9%	-9%	-7%	107%	4%	-10%
2	177%	98%	-2%	1%	5%	18%	162%	22%	4%
3	80%	29%	-26%	-26%	-21%	-23%	71%	-10%	-21%
4	192%	108%	-14%	-7%	-7%	15%	155%	12%	-8%
5	32%	-6%	-6%	-4%	0%	-43%	25%	6%	1%
6	48%	5%	-3%	-7%	3%	-40%	34%	10%	4%
7	133%	67%	7%	7%	13%	0%	120%	27%	13%
8	218%	127%	6%	6%	15%	36%	203%	33%	12%
9	139%	70%	11%	11%	18%	2%	127%	32%	18%

Table 34 Relative measurement errors (NON Real-Time Projects)

Proj ID	NESMA ind. non norm.	NESMA ind. norm.	NESMA estim.	E&QFP unspec.	E&QFP generic	Tichenor ILF model	ISBSG distrib.	sFP	ISBSG average weights
1	145%	74%	0%	9%	8%	-2%	116%	22%	7%
2	113%	49%	-1%	14%	5%	-36%	41%	20%	6%

3	52%	8%	11%	23%	16%	-43%	27%	25%	15%
4	46%	4%	16%	25%	23%	-38%	39%	28%	23%
5	17%	-17%	13%	20%	20%	-50%	11%	23%	21%
6	169%	92%	2%	10%	10%	15%	155%	25%	8%
7	31%	-6%	10%	19%	15%	-44%	25%	21%	15%
8	106%	47%	-3%	4%	4%	-17%	83%	16%	3%
9	28%	-9%	20%	27%	26%	-45%	22%	30%	27%

The obtained results show that we can divide the simplified FSM methods in two categories: those which base the size estimation exclusively on some measure of the data (like the NESMA indicative, the Tichenor and ISBSG distribution methods) and those which propose fixed weights for all the BFC of FPA. The former methods yield the largest errors. Although it was expected that estimates based on less information are generally less accurate than estimates based on more information, the really important finding of our experimental evaluation is that the size estimates based exclusily on the measures of data measures feature quite often intolerably large errors, i.e., errors that are likely to cause troubles, if development plans were based on such estimates. For instance, let us consider the Tichenor method (which appears the best of those based on data measures) and assume that only size estimation errors not larger than 20% are acceptable: 10 estimates out of 18 would be unacceptable.

On the contrary, the methods that take into consideration all BFC and provide fixed weights for them yield size estimates that are close to the actual size. Among these methods sFP is an exception, since it regularly overestimates the size of projects, often by over 20%. This seems to indicate that "average" projects are characterized by data and/or transactions whose actual complexity is smaller than the complexity expected by the sFP method.

The accuracy of the used methods is summarized in Table 35, where the mean and standard deviation of the absolute relative errors are given for Real-Time projects, for non Real-Time projects, and for the entire set of projects. The mean value of absolute relative errors is a quite popular statistic, often termed MMRE (Mean Magnitude of Relative Errors).

Table 35 Mean and Standard Deviation of Absolute Relative Errors

	NESMA ind. non norm.	NESMA ind. norm.	NESMA estim.	E&QFP unspec.	E&QFP generic	Tichenor ILF model	ISBSG distrib.	sFP	ISBSG average weights
Mean (RT only)	127%	63%	10%	9%	10%	20%	112%	17%	10%
Stdev (RT only)	64%	44%	7%	7%	7%	16%	59%	11%	7%
Mean (non RT)	79%	34%	8%	17%	14%	32%	58%	23%	14%
Stdev (non RT)	56%	33%	7%	8%	8%	17%	50%	4%	8%
Mean (all)	103%	49%	9%	13%	12%	26%	85%	20%	12%
Stdev (all)	63%	40%	7%	8%	8%	17%	60%	9%	8%

Table 35 shows that the NEMSA estimated, the two E&QFP methods and the ISBSG average weights methods provide essentially equivalent accuracy. This is not surprising,

given that these methods propose very similar weight values. The NESMA estimated method appears the best, but for Real-Time projects the E&QFP methods perform similarly, often even better.

For Real-Time projects, E&QFP (either in the unspecified or generic flavor) tends to provide the most accurate results, while the NESMA estimated method provides quite reasonable estimates.

It is worthwhile noticing that E&QFP is more accurate than NESMA for Real-Time applications because it uses bigger weights, which suite better Real-Time application, which are more complex than non Real-Time applications.

B. Analysis of results

As mentioned in Chapter 3 , simplified FSM methods are based on skipping one or more phases of the standards Function Point measurement process (see Table 3). It is reasonable to assume that the accuracy of the measure is inversely proportional to the number of measurement phases not performed, hence to the amount of data not retrieved from the functional user requirements of the software to be measured.

To confirm such hypothesis, we have enhanced the information reported in Table 3 with the data concerning mean errors and error standard deviations: the result is given in Table 36. The direct comparison of accuracy data with the information used for measurement makes the following observations possible.

Any simplified method that does not involve the weighting appears to be bound to a 10-15% mean absolute error.

It does not appear true that the more you measure, the best accuracy you get. For instance, E&QFP considering unspecified generic functions appear more accurate than sFP, even though the former method does not involve classifying function types.

Among methods that use the same type and amount of data, there are relatively large differences in accuracy: for instance, the Tichenor ILF model appears more precise than both the NESMA indicative (with normalized data) and the ISBSG distribution.

Table 36 Measurement Process: Required Data VS. Accuracy

	IFPUG	NESMA indic. Norm.	NESMA estim.	E&QFP Generic func.	E&QFP Unspec. generic func.	Tichenor ILF Model	ISBSG distrib.	sFP	ISBSG average weights
Identifying logic data	√	√	√	√	√	√	√	√	√
Identifying elementary processes	√		√	√	√	(*)[2]	(*)	√	√

[2] required to verify applicability.

Classifying logic data as ILF or EIF	√	√	√	v		√	√	√	√
Classifying elementary processes as EI, EO, or EQ	√		√	√		(*)	(*)	√	√
Weighting data functions	√								
Weighting transaction functions	√								
Mean error	-	49%	9%	13%	12%	26%	85%	20%	12%
Error stdev	-	40%	7%	8%	8%	17%	60%	9%	8%

The last two observations suggest that exploiting the knowledge provided by statistical analysis can be decisive for achieving accurate measures via simplified processes. For instance, the E&QFP method considering unspecified generic functions is quite accurate because the likely complexity of data and transactions assumed by the method (see Table 2) were derived via accurate statistical analysis. On the contrary, the complexity values assumed by the sFP method were chosen on the basis of expectations, not on rigorous statistical analysis.

The exploitation of statistical data is the base for the new methods described in the next section.

5.1.5 Model-based simplified FSM models

In this section, we derive a simplified FSM model in a way similar to those described in Chapter 3 , but based on the measures of our own applications (as reported in Table 23 and Table 24).

Table 37 Average Function Type Weighs for Out Dataset

Function type	EQFP generic	NESMA Estim.	ISBSG average	sFP	Our dataset (all proj.)
ILF	7.7	7	7.4	7	7.4
EIF	5.4	5	5.5	5	5.3
EI	4.2	4	4.3	3	3.7
EO	5.2	5	5.4	4	4.6
EQ	3.9	4	3.8	3	4

In the rightmost column of Table 37 we give the average weights of the BFC computed over all the measured applications. Note that the given averages are computed as the mean –at the dataset level– of the mean values computed for each application. In the table, the mean weights derived from our dataset are shown together with the weights proposed by other simplified FSM methods, for comparison. The fact that our EI and EO means are smaller than the values proposed by other methods, while the ILF and EIF means are very close to those proposed by other methods, probably means that our applications were simpler than those considered in the definition of other methods.

Table 38 Mean and Median Weights for the Projects in Our Dataset

Dataset	Mean (median) weight							
	ILF	EIF	EI	EO	EQ	TF	DF	UFP/#ILF
All non RT projects	6.5	5.5	3.5	4.4	3.4	7.0	3.7	22.7
All RT projects	8.2	5.0	4.0	4.8	5.1	8.1	4.4	17.0
All projects	7.4	5.3	3.7	4.6	4.0	7.6	4.1	19.9

In Table 38 we give the average values of weights derived from our dataset, distinguishing Real-Time and non Real-Time applications. We also give the average value of the ratio between the number of ILF and the size in UFP. It is possible to note that the average number of UFP per ILF we found is quite larger than that found by Tichenor. This suggests that models based just on ILF can be hardly generalized.

Note that we computed also the weights for transaction functions (TF) and data functions (DF). These weights can be used in simplified measurement processes like the E&QFP unspecified generic method.

The values in Table 38 suggest that transactions were generally more complex in Real-Time applications than in non Real-Time applications. The latter are probably responsible for relatively smaller weights of transaction (EI, EO, and EQ) in Table 1.

Using the values in Table 38 it was possible to derive models that are similar to those described in Subsection 5.1.3: they are described in Table 39 and Table 40.

Table 39 Models for NON Real-Time Projects

Average weights (all BFC)	UFP = 6.6 #ILF+ 5.5 #EIF + 3.5 #EI + 4.4 #EO + 3.4 #EQ
Average weights (DF and TF)	UFP = 7.0 #DF + 3.7 #TF
ILF based model	UFP = 22.7 #ILF

Table 40 Models for Real-Time Projects

Average weights (all BFC)	UFP = 8.2 #ILF+ 5 #EIF + 4 #EI + 4.8 #EO + 5.1 #EQ
Average weights (DF and TF)	UFP = 8.1 #DF + 4.4 #TF
ILF based model	UFP = 17 #ILF

5.1.6 Evaluate our new model

We used such models to estimate the size of the projects in our dataset. The results of the estimations are reported in Table 41 and Table 42 for Real-Time and non Real-Time projects, respectively.

Table 41 Estimates of RT Projects based on Models using the our new models

Proj. ID	Actual size	Average weights (all BFC)		Average weights (DF and TF)		ILF based model	
		Est. size	% err	Est. size	% err	Est. size	% err
1	289	273	-6%	277	-4%	306	6%
2	101	110	9%	109	8%	136	35%

3	136	109	-20%	105	-23%	119	-13%
4	195	187	-4%	198	2%	255	31%
5	106	104	-2%	103	-3%	68	-36%
6	223	223	0%	213	-4%	153	-31%
7	15	17	13%	17	13%	17	13%
8	33	39	18%	37	12%	51	55%
9	44	52	18%	51	16%	51	16%

Table 42 Estimates of NON RT Projects based on Models using the our new models

Proj. ID	Actual size	Average weights (all BFC)		Average weights (DF and TF)		ILF based model	
		Est. size	% err	Est. size	% err	Est. size	% err
1	92	85	-8%	90	-2%	136	48%
2	94	87	-7%	97	3%	91	-3%
3	79	81	3%	84	6%	68	-14%
4	96	98	2%	102	6%	91	-5%
5	90	91	1%	92	2%	68	-24%
6	91	85	-7%	90	-1%	159	75%
7	80	79	-1%	79	-1%	68	-15%
8	109	98	-10%	101	-7%	136	25%
9	82	88	7%	88	7%	68	-17%

Table 41 and Table 42 show a rather poor accuracy of the estimation based on ILF, with error greater than 20% for several projects.

On the contrary, the estimations based on average weights are reasonably accurate; the obtained results are particularly good for non Real-Time projects, with all the estimates featuring errors not greater than 10%.

Table 43 Mean and Stdev of Absolute Relative Errors

	Average weights, all BFC	**Average weights, DF & TF**	**Average UFP / #ILF**	**NESMA estim.**	**E&QFP unspec.**	**E&QFP generic**	**ISBSG average weights**
Mean (RT only)	10%	9%	26%	10%	9%	10%	10%
Stdev (RT only)	8%	10%	29%	7%	7%	7%	7%
Mean (non RT)	5%	4%	25%	8%	17%	14%	14%
Stdev (non RT)	3%	4%	22%	7%	8%	8%	8%
Mean (all)	8%	10%	31%	9%	13%	12%	12%
Stdev (all)	6%	6%	19%	7%	8%	8%	8%

The average values of the absolute relative errors are reported in Table 43 together with the average values of the absolute relative errors obtained with the best among the other methods, for comparison.

It is easy to see that the estimates obtained using the average weights of the projects being estimated feature practically the same accuracy as the other methods.

It is a bit surprising that in the literature a few models of type UFP = k × #ILF were proposed, while model of type UFP = k × #EP (where #EP is the number of elementary

processes) received hardly any attention. We computed the ratio UFP/#EP for each application, and used the average value k in models UFP = k × #EP, to estimate the size of the applications in our dataset. The obtained estimates were characterized by estimation errors quite similar to those of ILF-based models (the average absolute error was 25% for Real-Time projects and 27% for non Real-Time projects). Accordingly, it seems that models of type UFP = k × #EP are not likely to provide good estimates.

5.1.7 Conclusion

In this work, we applied simplified functional size measurement processes to both traditional software applications and Real-Time applications. The obtained results make it possible to draw a few relevant conclusions:

- Some of the simplified FSM methods we experimented with seem to provide fairly good accuracy. In particular, NESMA estimated, E&QFP, and ISBSG average weights yielded average absolute relative errors close to 10%. This level of error is a very good trade-off, if you consider that it can be achieved without going through the expensive phase of weighting data and transactions.
- Organizations that have historical data concerning previous projects can build their own models. We showed that with a relatively small number of projects it is possible to build models that provide a level of accuracy very close to that of methods like NESMA estimated and E&QFP.
- The simplified FSM methods are generally based on average values of ratios among the elements of FP measurement. Accordingly, projects that have unusual characteristics tend to be ill suited for simplified size estimation. For instance, project 3 in our set of Real-Time projects is more complex than the other projects in the set, having most EI and EO characterized by high complexity. This causes most method to underestimate the size of the project by over 20%. Therefore, before applying a simplified FSM method to a given application, it is a good idea to verify that this application is not too much (or too less) complex with respect to “average” applications. Our Real-Time project 3 was characterized by the need to store or communicate many data at a time: this situation could have suggested that using average values for an early measurement leads to a rather large underestimation.

E&QFP methods proved more accurate in estimating the size of Real-Time applications, while the NESMA estimated method proved fairly good in estimating both Real-Time and non Real-Time applications. However, the relatively small number of projects involved in the analysis does not allow generalizing these results.

Even considering the relatively small dataset, it is however probably not casual that the NESMA estimated method happened to underestimate all projects. Probably NESMA should consider reviewing the weights used in the estimated method, in the sense of increasing them.

When considering the results of our analysis from a practical viewpoint, a very interesting question is “What simplified method is the best one for my application(s)?”. Table 33 and Table 34 show that the methods that are better on average are not necessarily the best ones for a given project. To answer the question above it would be useful to characterize the projects according to properties not considered in FSM and look for correlations with the measures provided by different simplified methods. This

would allow selecting the simplified measurement method that provided the best accuracy for applications of the same type as the one to be sized. Unfortunately, it was not possible to analyze the possibly relevant features of the dataset described in sub-section 5.1.2 (we had no access to the code of Real-Time projects), thus this analysis is among our future objectives.

As already mentioned, the results presented here are based on datasets in which the largest project has size of 289 FP: further work for verifying the accuracy of simplified measurement methods when dealing with larger project is needed.

Among the future work is also the experimentation of simplified measurement processes in conjunction with measurement-oriented UML modelling [57], as described in [58].

The models used in Subsection 5.1.3 are generally derived in a rather naive way, i.e., simply computing averages of some elements that are involved in the measurement: e.g., the average ration between the measure of BFC and their number. Simplified measurement models should be better derived via regression analysis. Unfortunately, the relatively little number of applications in our datasets does not support this type of analysis, especially if multiple independent variables are involved, as in models of type UFP = f(EI, EO, EQ, ILF, EIF) or UFP = f(TF, DF). Performing this type of analysis is among our goal for future activities, provided that we can get enough data points.

5.2 Empirical evaluation of Model-based Simplified COSMIC Measurement

Most simplified FSM methods address the simplification of FPA, since its process of measuring function points involves activities –such as the classification of transactions and data and the evaluation of the complexity of every transaction and logic data file– that require a relevant measurement effort, and can be carried out only when the specification of user requirements is fairly complete and detailed. However, also the process of measuring CFP (which is generally faster and less expensive than FP measuring) may need to be carried out faster and at a smaller cost than required by the official counting manual [6]. This may happen because the size estimates are needed within a given deadline (e.g., for cost estimation and bidding) or because detailed requirements specifications are not available (and will not be available for a while). So the simplified measurement processes for CFP have been proposed: see for instance the section on “early or rapid approximate sizing” in [59].

The process of applying the COSMIC FSM method is relatively long and effort-consuming. In particular, the need to describe every functional process in terms of data movements – which implies identifying the possible data movement types for every data group type – can easily require a relevant amount of work. Therefore, the COSMIC measurement process involves:

- The identification of functional processes (FPr).
- The identification of data groups.
- For each functional process, the identification of unique data movements involving the identified data group types.

The COSMIC measurement process is schematically represented in Figure 37. When looking at the graphical representation in Figure 37, it must be remembered that the first two phases are carried out once each, at the application level, while the third is carried out for each functional process.

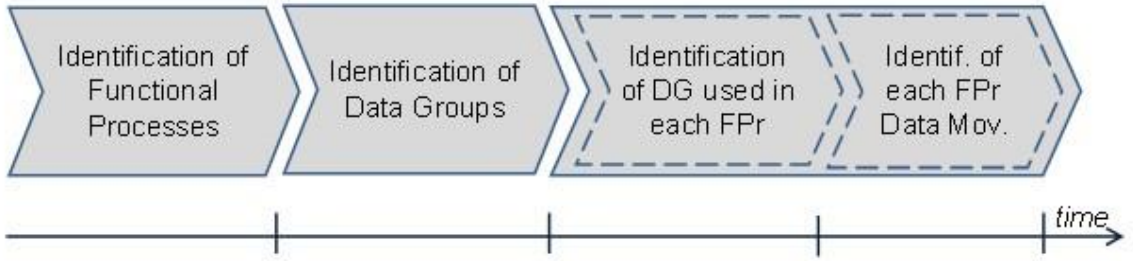

Figure 37 COSMIC measurement process

Model-based method process

As we described in Figure 26 in Section 4.1, the model-based object-oriented measurement process consists of the following activities: building Use case diagram, building class diagram, building component diagram, and building sequence diagrams. At the end of each above steps, as described in Figure 38, respectively the following artefacts can be obtained:

- In the first step, Use case diagram or component diagram with user interface;
- In the second step, class diagram component diagram with classes;
- In the third step, component diagram with operation-class dependencies;
- In the last step, sequence diagrams.

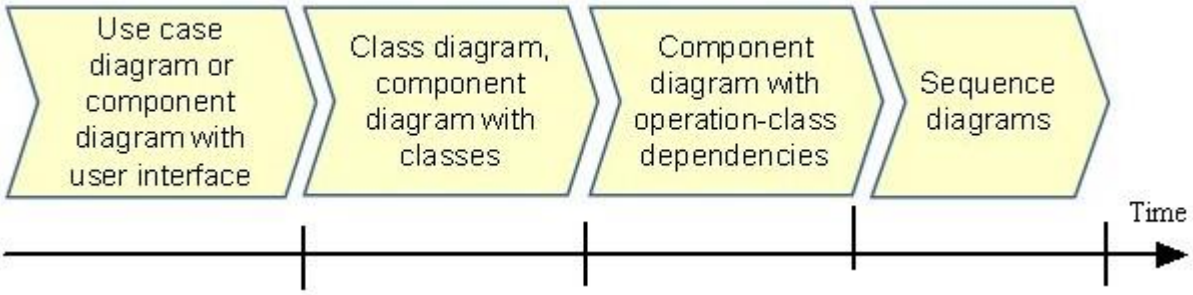

Figure 38 UML modelling process

It is easy to see that while progressing in the development, namely construct Use case diagram, construct class diagram, construct component diagram, and construct sequence diagrams, UML models become more and more complete and detailed and in general include an increasing number of diagrams. This means that UML models convey an increasing amount of information, which can be used for FSM [61].

Comparing the UML modeling and COSMIC measurement processes, it is easy to see that while progressing in the development, UML models become more and more complete and detailed and in general include an increasing number of diagrams; while proceeding in the execution of the process we get more and more information, which allows for the application of increasingly sophisticated measure estimation processes.

When we examine these two processes at every stage, we will interestingly find, according to the definition of element mapping between UML and COSMIC in Table 22 , that the information provided by the various UML models matches the information required by the various stages of the COSMIC process, as schematically described in Figure 39.

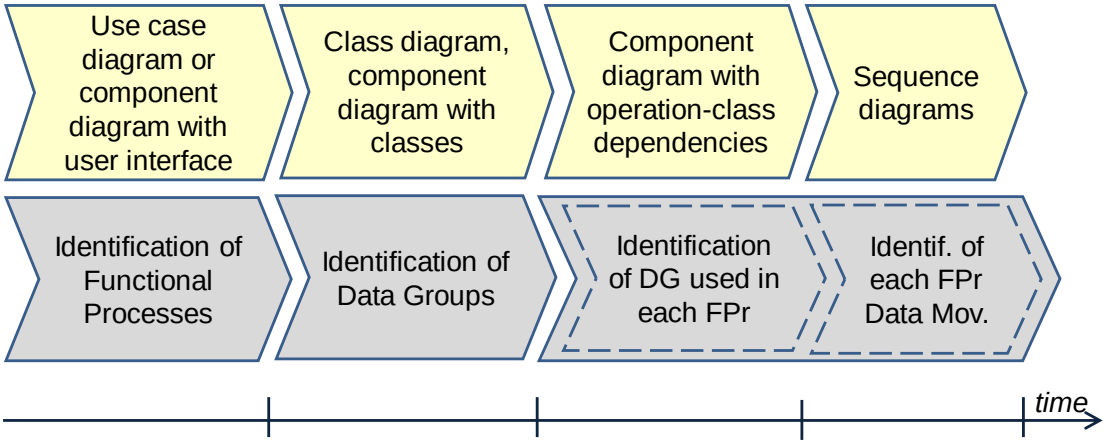

Figure 39 UML modeling process and COSMIC measurement process phases

In view of the above observations, for the entire process, we propose the following questions:

- Q1. During the requirements elicitation and specification phase, is it possible to write progressively more complete and detailed UML models that support progressively more accurate simplified CFP measurement methods?
- Q2. What is the accuracy of the estimates provided by different simplified CFP measurement methods?
- Q3. Do simplified CFP measurement methods provide a level of accuracy that is proportional to the amount of information required?

Here the term “accuracy” is used to indicate how close the estimated size and the actual size are.

5.2.1 Simplified measurement processes for COSMIC function point

Different simplified processes are possible, depending on the stage of requirements collection (as described in Figure 39). In what follows, we describe the type of processes that can be applied at the various stages.

A. Size estimation based on the number of functional processes and the number of data groups

A first very rough simplification of the measurement process was proposed in the COSMIC manual itself [59]. This simplification is perfectly coherent with the COSMIC model of software requirements: user requirements are composed of a set of functional processes, each involving a set of data movements. If data movements cannot be counted, the measurer has to count the elements at the abstraction level immediately above: i.e., functional processes must be sized directly.

So, the simplified process requires that only the first one of the activities required for CFP measurement be performed. Of course, in this way we get the number of functional processes involved in the software application being measured, but not their size. To transform the number of functional process into the application size, the simplified process requires that the mean number of data movements per functional process is used.

Let AvDMperFPr be the mean number of data movements per functional process, computed on the basis of historical data. If we assume that the software application to be measured is similar to those previously measured, then it is reasonable to assume that the mean number of data movements per functional process of the new application will be close to MDM. Thus,

$$CFP = AvDMperFPr \times \#FPr \quad (25)$$

where #FPr is the number of Functional processes.

If the historical data required to compute AvDMperFPr are available, it is also possible to use Ordinary Least Squares (OLS) regression to derive a model of type

$$\mathbf{CFP = a \times \#FPr + b} \quad \mathbf{(26)}$$

This type of model is not mentioned in [59]; however in this paper we test the ability of UML models to support also this type of models and the corresponding simplified measurement process.

As in common practice, log-log transformation can be applied to data, thus yielding a model of type:

$$CFP = b \times (\#FPr)^a \quad (27)$$

B. Size estimation based on the number of data movements.

The method described in the previous part (Part A) assumes that the size in CFP is proportional to the number of functional processes. It is also reasonable to assume that the size in CFP is proportional to the number of data groups: the more data groups, the more opportunities for data movements.

A simplified computation of CFP can thus be achieved via a model that computes the estimated size by means of some formula to be defined applied to #FPr and #DG (the number of data groups in the application). This procedure is more complex than the one described in the previous part (Part A), as it requires the identification of data groups, but it is still simpler than the "full" COSMIC counting process, as data movements do not need to be identified and classified. Besides, a conceptual model of the data involved in the application is usually built very early in the requirements modeling process. Thus, its availability is generally an easily satisfied assumption.

Data groups do not contribute directly to the measure of size in CFP: as we mentioned above, the size of an application in CFP is the sum of the sizes of its functional processes. Therefore, the COSMIC method does not suggest how to use #DG in the estimation of size. However, the model can be derived via regression analysis, provided that historical data reporting both #FPr and #DG are available. The resulting equations can be of the form

$$CFP = \#FPr \times a + \#DG \times b + c \quad (28)$$

or

$$CFP = c \times \#FPr^{a} \times \#DG^{b} \quad (29)$$

C. Size estimation based on the number of data groups involved in each functional process

The two methods described above are based on the total number of functional processes and data groups. Accordingly, such measures characterize the whole application. It is reasonable to expect that a more accurate estimate can be derived if information that characterizes each functional process individually is available. The number of data groups involved in each functional process provides such information, thus allowing for potentially more precise measures of size.

If the historical dataset includes the suitable information, statistical analysis can yield models of the following type:

$$CFP = f(\#FPr, AvDGperFPr) \quad (30)$$

where AvDGperFPr is the mean number of data groups involved in functional processes in the application to be measured.

5.2.2 UML model supporting the simplified measurement approaches

In this subsection, we describe the UML models that are needed to support the simplified approaches to CFP measurement described in previous subsection 5.2.1. The corresponding FUR of these models comes from the case study quoted in Section 4.2. We also present the model supporting the measure of CFP performed as described in the manual [33].

A. UML model supporting the size estimation based on the number of data movements

Figure 40 illustrates a UML diagram that can effectively support the first simplified measurement method, described in sub-section 5.2.1. It is a use case diagram. The information that is needed to use equation (25), i.e., #FPr, can be obtained by counting the use cases.

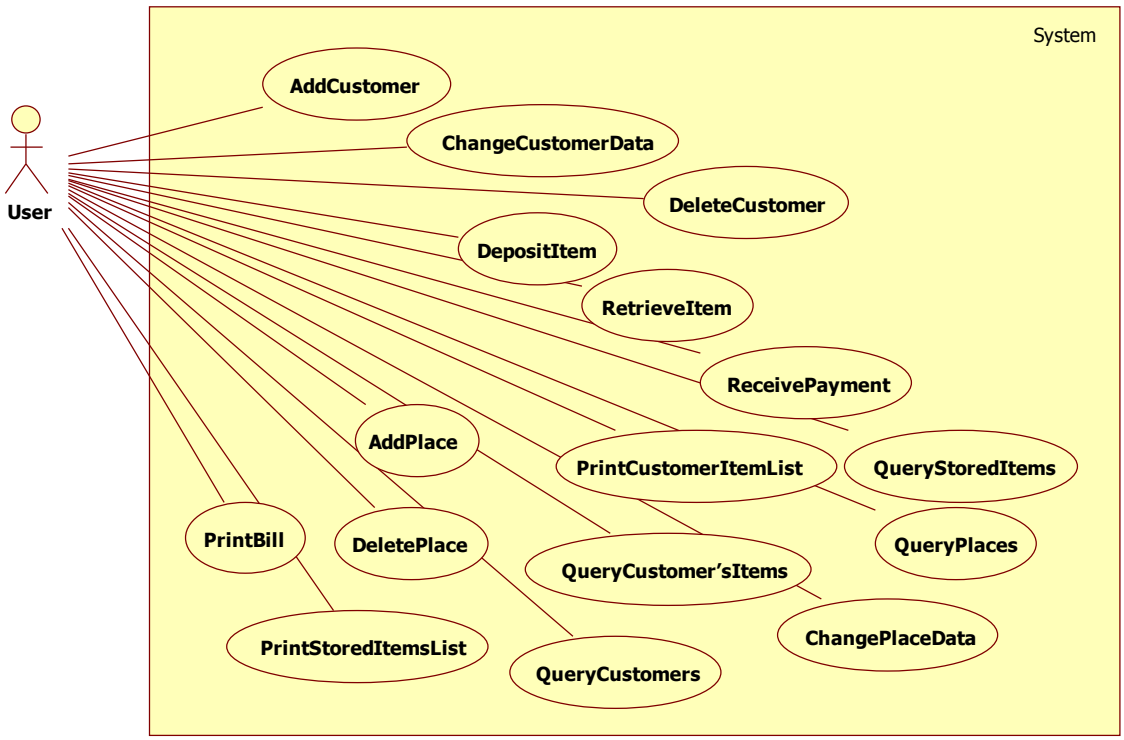

Figure 40 UML use case diagram showing the functional processes

Using a use case diagram can be interesting mainly for the organizations that employ this type of diagram for specifying requirements and do not use other UML diagrams. However, it should be paid attention to the fact that use cases have a meaning (or can be given a meaning) that does not always match with the concept of functional process; it is therefore recommended that each use case is evaluated against the properties required for functional processes.

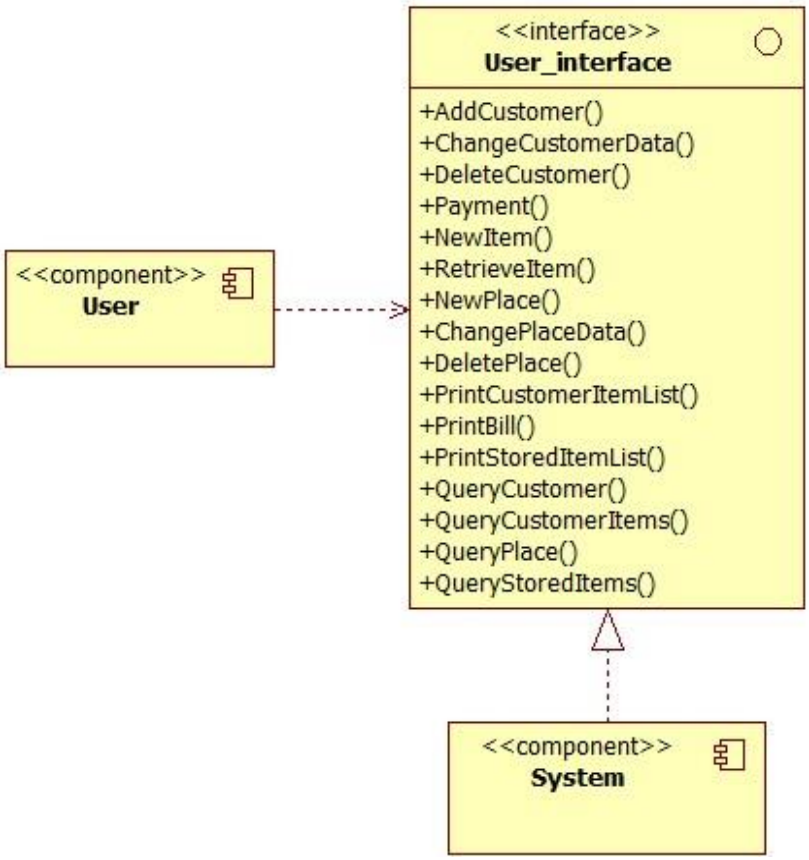

Figure 41 UML component diagram showing the functional processes

In place of a use case diagram like the one depicted in Figure 40, it is also possible to write a component diagram, like the one depicted in Figure 41, where the relevant information is provided by the interface realized by the system. The interface lists the functional processes that can be triggered by the user. So, the information that is needed to use equation, i.e., #FPr, can be obtained by counting the operations listed in the User_interface.

Note that while components corresponded to software artifacts in previous versions of UML, in more recent versions of the language, components can be also used to describe the specifications of software artifacts. Therefore, our usage of component diagrams complies with the definition of UML [52].

B. UML model supporting Size estimation based on the number of functional processes and the number of data groups

To get the number of data groups #DG, required for using equation (28), we can exploit the closeness of UML classes to the concept of data group. So, the class diagram described in Figure 42 describes the data groups involved in the Warehouse Software Portfolio.

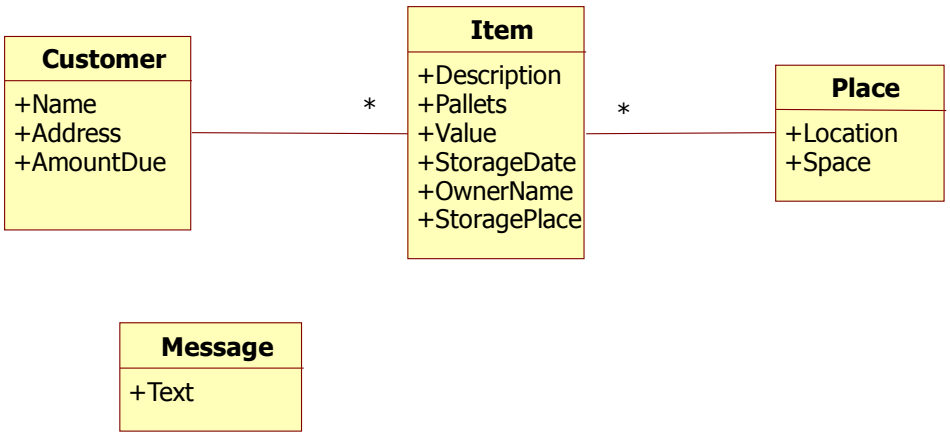

Figure 42 UML class diagram, showing the data groups

It can be noted that the entities described in Figure 27 and Figure 28 appear in Figure 42 as well. However, Figure 42 includes also a class that describes the Message transient data group. In the COSMIC method, transient data groups are data groups that are not persistent, but are needed to capture user requirements. In our case, the Message data group is needed to represent the data that –according to the FUR– have to be communicated to the user.

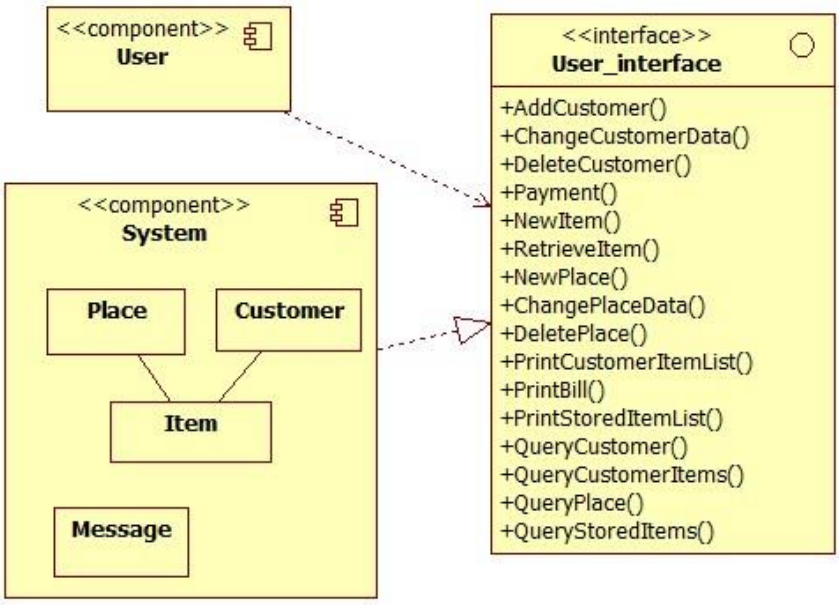

Figure 43 UML component diagram showing the functional processes

Figure 43 illustrates the same diagram as Figure 41, where the system component has been refined with the description of the classes that represent the data managed by the system. In practice, the information from the class diagram is reported in the system component. It is easy to see that the diagram in Figure 44 provides all the data needed to use equations (26) and (27), i.e. #FPr and #DG.

C: UML model supporting Size estimation based on the number of data groups involved in each functional process

Figure 44 illustrates a diagram providing the information needed to use equation (30). In the diagram, UML ports are used to precisely indicate which classes (i.e., data groups)

are used in each functional process. To this end, sets of functional processes that use the same set of classes are grouped into a single interface: both ChangeCustomerData and DeleteCustomer use Customer, Item and Message.

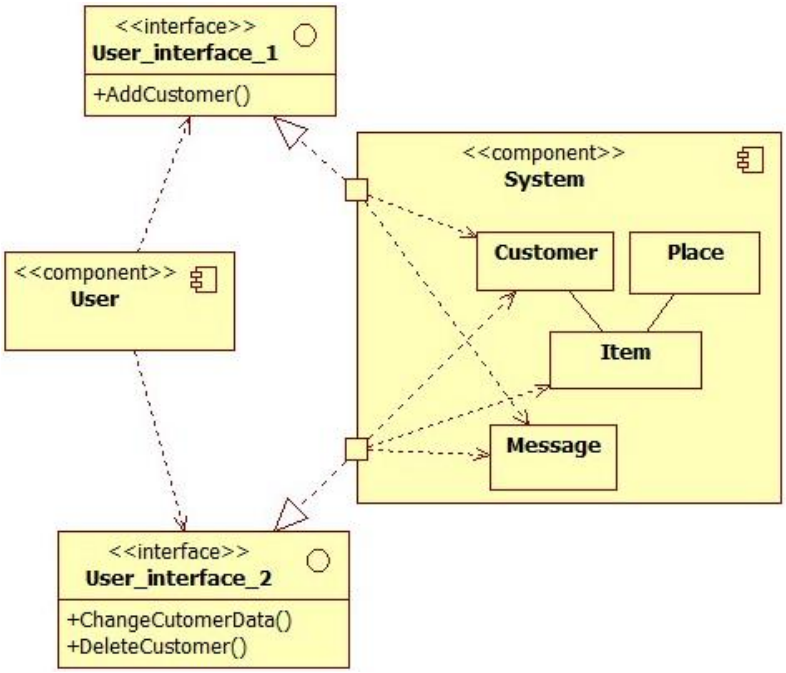

Figure 44 UML component diagram showing the functional processes and the data groups

In Figure 44 only the interfaces needed to add, change, and delete clients are shown. It can be noticed that grouping functional processes according to the used classes may lead to a rather large number of interfaces, which could decrease the readability of the diagram. However, interfaces that are homogeneous with respect to the used classes not only allow for a quite precise estimation of size (as shown in next Subsection 5.2.3), but explicitly represent the logical relationship between interface elements and system data: this poses the basis for the identification of important traceability information when the design model is built.

An alternative to the model shown in Figure 44 is a sequence diagram that shows only the classes involved in the functional process, as in Figure 45. In fact, the diagram represents a specific functional process (AddCustomer) and the involved class instances. Excluding the User and the System, which represent the functional user and the application being measured, respectively we have that AddCustomer uses two data groups: Customer and Message. This type of diagram is convenient because it can be refined into the diagram described in Figure 46, which provides a detailed specification of the AddCustomer operation and supports full fledged COSMIC measurement.

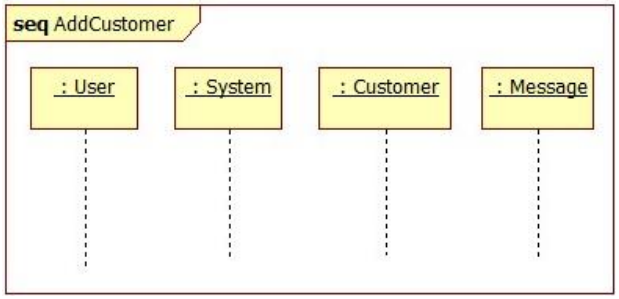

Figure 45 UML component diagram showing the class (data group) instances participating in the AddCustomer functional process

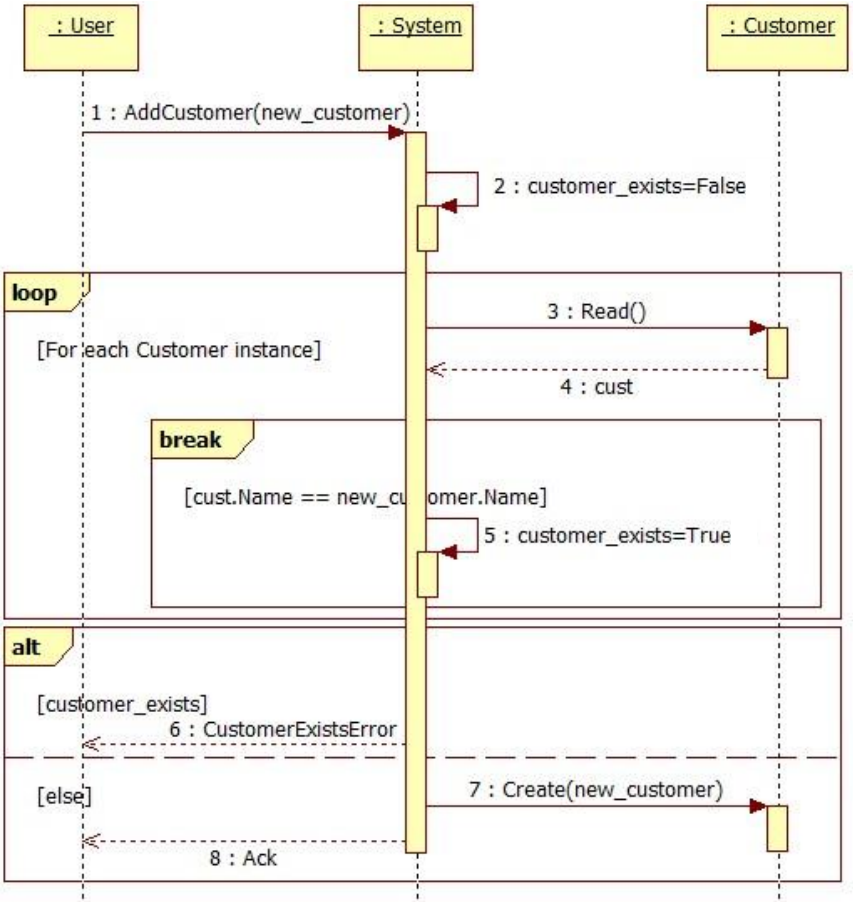

Figure 46 UML sequence diagram showing the data movements involved in a given functional process

Figure 46 illustrates a sequence diagram that contains all the information needed to measure the size of the functional process according to the official manual [33]. Messages that cross the application boundary (in our case, messages from or to the user) are entries and exits, while messages directed to class instances representing data groups are reads or writes.

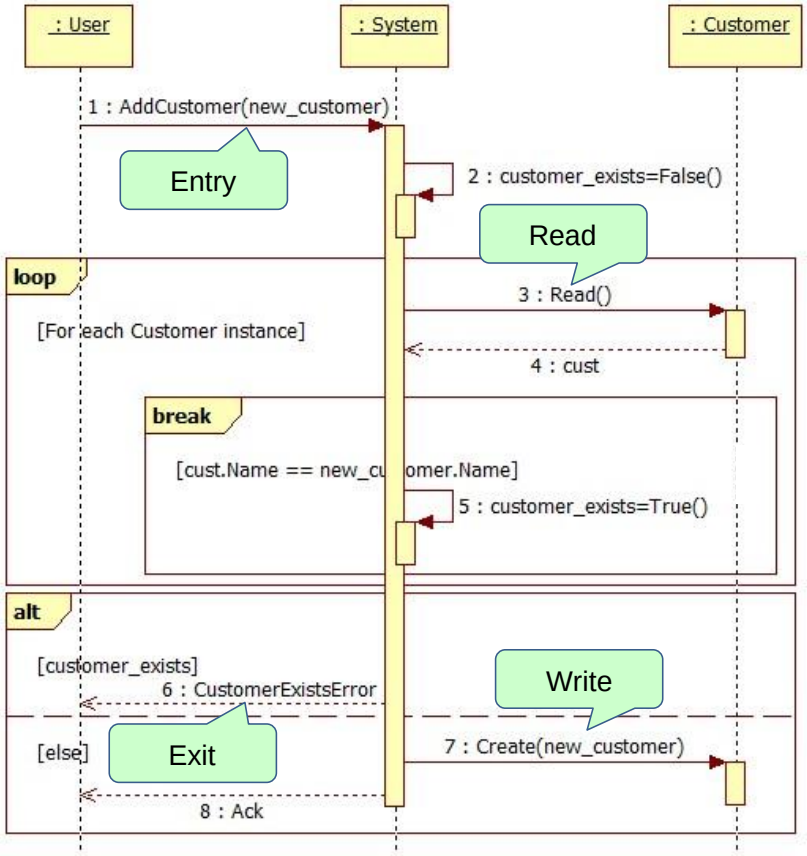

Figure 47 UML sequence diagram with the data movements highlighted

Figure 47 shows the same sequence diagram as Figure 46, with data movements highlighted. Note that message 8 is not a data movement, since it is –like message 6– an exit involving a Message, thus it is not unique. Figure 47 shows that –once the model is available– performing the measurement is quite straightforward; so, the main effort required by the COSMIC method is in modeling, rather than in measuring.

Brief summary

The UML models presented in this section provide an increasingly larger amount of information: the use case diagram in Figure 40, the diagram in Figure 44 provides more information than the diagram Figure 43, finally, the diagram in Figure 46, together with the other sequence diagram representing the functional processes, provides the most detailed representation of user functional requirements.

5.2.3 Empirical analysis

To answer the research questions defined above, we modeled a set of software applications and measured them, with the goal of obtaining the measures needed to support the simplified methods described in 5.2.1. Then, we applied the simplified measurement methods in two ways:

– a) By computing the means of the relevant measures and using the means as parameters in equations (25) and (28).
– b) Deriving size models via regression analysis and applying them.

In both cases, the obtained estimates were compared with the measures obtained via the standard COSMIC method [33], without any simplification.

A. The dataset

The projects considered belonged to different types: sample projects provided by COSMIC to illustrate the counting process (5 projects); academic examples used in teaching (7 projects); Web based Management Information Systems (MIS) (10 projects), project management tools (1 project).

Part of the dataset containing the measures of the models of the applications described above is given in Table 44. The only missing part is the number of involved data groups for each functional process, which is not reported because of space reasons (the dataset includes over 700 functional processes).

Table 44 The dataset

Pid	CFP	#FPr	#DG	AvDG perFPr	AvDM perDG	AvSize ofFPr_others	AvDG perFpr_others	AvCFP perDG_others
1	86	16	6	2.88	1.90	7.58	2.88	1.80
2	56	11	11	3.55	1.60	7.57	3.55	1.80
3	91	15	10	4.00	1.57	7.57	4.00	1.80
4	69	19	12	2.32	1.72	7.64	2.32	1.80
5	103	19	16	3.06	1.93	7.58	3.06	1.80
6	64	14	7	2.64	1.71	7.59	2.64	1.80
7	116	20	14	3.60	1.65	7.60	3.60	1.80
8	124	20	10	2.80	2.38	7.57	2.80	1.78
9	66	14	7	3.79	1.28	7.59	3.79	1.81
10	117	19	9	3.47	1.78	7.57	3.47	1.80
11	90	13	14	3.92	1.99	7.55	3.92	1.80
12	31	7	16	4.71	1.18	7.56	4.71	1.81
13	252	60	24	2.07	2.40	7.83	2.07	1.75
14	360	23	18	8.17	1.76	7.27	8.17	1.80
15	514	74	29	4.50	1.74	7.60	4.50	1.81
16	186	27	13	4.56	1.61	7.56	4.56	1.81
17	948	152	22	4.76	1.37	7.87	4.76	1.91
18	189	30	13	4.20	1.50	7.59	4.20	1.81
19	107	9	20	5.60	2.12	7.48	5.60	1.80
20	273	22	63	5.95	2.00	7.38	5.95	1.79
21	502	45	15	6.98	1.60	7.30	6.98	1.81
22	260	34	7	2.85	3.36	7.53	2.85	1.72
23	895	68	24	7.28	1.84	6.96	7.28	1.80

The meaning of the columns in Table 44 is as follows.
• CFP is the size in COSMIC Function Points, measured according to the manual;
• #FPr is the number of Functional Processes;
• #DG is the number of data groups;
• AvDGperFPr is the mean number of data groups involved in the project's Functional Processes;
• AvDMperDG is the mean number of data movements per Functional Process, i.e., the mean size of the applications' functional processes;

• AvSizeofFPr_others is the mean number of data movements per FP, computed on all the other applications;
• AvDGperFpr_others is the mean number of data groups per FP, computed on all the other applications;
• AvCFPperDG_others is the mean number of data movements (i.e., size) per data group, computed on all other applications.

B. Models based on the number of data movements

AvDMperFPr (the mean size of functional processes, i.e., the mean of CFP/#FPr) is 7.3. We can use this value in equation (25), thus obtaining the model

$$CFP = \#FPr \times 7.3 \quad (31)$$

Of course, model (Equation (31)) is as good as the CFP/#FPr ratio of the considered applications is close to the mean. Actually, the standard deviation of CFP/#FPr for the applications in Table 44 is 3.25, i.e., 44.8% of the mean; therefore we do not expect a very good accuracy.

To evaluate the accuracy of this model we estimated the size of each application using the data of the others as a historical dataset. So, for instance, to estimate the size of project 23 we computed the mean CFP/#FPr of projects 1 to 22, and multiplied that value (6.99) for the number of #FPr of project 23 (68), which results in an estimated size of 475 CFP.

Through this process we got the estimates reported in Table 45. As expected, the accuracy of the model is far from good: the estimates are characterized by MMRE = 36.6%, Pred(25) = 39.1%, error range = [-56%,104%].

Table 45 Estimates obtained using equation (31)

P.Id	Estimated Size [CFP]	Error	% Error
1	117	31	36.0%
2	81	25	44.6%
3	110	19	20.9%
4	141	72	104.3%
5	139	36	35.0%
6	103	39	60.9%
7	146	30	25.9%
8	146	22	17.7%
9	103	37	56.1%
10	139	22	18.8%
11	95	5	5.6%
12	52	21	67.7%
13	444	192	76.2%
14	158	-202	-56.1%
15	538	24	4.7%
16	196	10	5.4%
17	1110	162	17.1%
18	219	30	15.9%
19	63	-44	-41.1%
20	154	-119	-43.6%
21	319	-183	-36.5%

22	246	-14	-5.4%
23	475	-420	-46.9%

Via OLS regression we obtained a first statistically significant model:

$$CFP = -16.5 + \#FPr \times 6.698 \quad (32)$$

The adjusted R^2 is 0.882, the p-value is < 0.001

By using this model, we obtained the estimates reported in Table 46 and characterized by MMRE = 22.7%, Pred(25) = 69.6%, Error range is [-62%, 61%].

Table 46 Estimates obtained using equation (32)

P.Id	Estimated Size [CFP]	Error	% Error
1	91	5	5.8%
2	57	1	1.8%
3	84	-7	-7.7%
4	111	42	60.9%
5	111	8	7.8%
6	77	13	20.3%
7	117	1	0.9%
8	117	-7	-5.6%
9	77	11	16.7%
10	111	-6	-5.1%
11	71	-19	-21.1%
12	30	-1	-3.2%
13	385	133	52.8%
14	138	-222	-61.7%
15	479	-35	-6.8%
16	164	-22	-11.8%
17	1002	54	5.7%
18	184	-5	-2.6%
19	44	-63	-58.9%
20	131	-142	-52.0%
21	285	-217	-43.2%
22	211	-49	-18.8%
23	439	-456	-50.9%

C. Models based on the number of functional processes and the number of data groups.

Via OLS regression, no significant model of type $CFP = k \times \#FPr + m \times \#DG$ was found. Also the log-log transformation of data did not help.

We have not found any relationships existing between classes and sizes of CFP. This means that (at least with respect to the data available) there is no relationship between the number of classes and CFP.

D. Models based on the number of data groups involved in each functional process

When trying to use the knowledge of the number of data groups involved in each functional process, we discovered that the number of data movement per data group

involved in a functional process, computed for each application, was fairly constant throughout the applications of our dataset: the mean is 1.8 and the standard deviation 0.03 (i.e., 1.7% of the mean). We exploit this fact to define the following model:

$$CFP = AvDGperFPr \times \#FPr \times 1.8 \quad (33)$$

Term (1.8 × AvDGperFPr) is an estimate of the number of data movements per functional process: multiplied by the number of functional processes it yields an estimate of the number of data movements, i.e., the size of the application.

By using this model, and computing the AvDGperFPr of each application on the basis of the other applications' data, we obtained the estimates reported in Table 47 and characterized by MMRE=19.3%, Pred(25)= 82.6%, error range [-36%,93%].

Table 47 Estimates obtained using equation (33)

P.Id	Estimated Size [CFP]	Error	% Error
1	83	-3	-3.5%
2	70	14	25.0%
3	108	17	18.7%
4	79	10	14.5%
5	104	1	1.0%
6	67	3	4.7%
7	130	14	12.1%
8	100	-24	-19.4%
9	96	30	45.5%
10	119	2	1.7%
11	92	2	2.2%
12	60	29	93.5%
13	216	-36	-14.3%
14	339	-21	-5.8%
15	601	87	16.9%
16	222	36	19.4%
17	1384	436	46.0%
18	228	39	20.6%
19	90	-17	-15.9%
20	235	-38	-13.9%
21	569	67	13.3%
22	167	-93	-35.8%
23	889	-6	-0.7%

Via OLS regression we found a statistically significant model involving the number of Functional Processes and the mean number of data groups involved in each functional process:

$$CFP = -64.6 + \#FPr \times 7.63 + AvDGperFPr \times 9.71 \quad (34)$$

$Pr(>|t|) < 0.05$ for each independent variable; the adjusted $R^2 = 0.952$.

By using this model, we obtained the estimates reported in Table 48 and characterized by MMRE = 19.8%, Pred(25) = 69.6%, error range = [-47, 64%].

Table 48 Estimates obtained using equation (34)

P.Id	Estimated Size [CFP]	Error	% Error

1	85	-1	-1.2%
2	54	-2	-3.6%
3	89	-2	-2.2%
4	103	34	49.3%
5	110	7	6.8%
6	68	4	6.3%
7	123	7	6.0%
8	115	-9	-7.3%
9	79	13	19.7%
10	114	-3	-2.6%
11	73	-17	-18.9%
12	35	4	12.9%
13	413	161	63.9%
14	190	-170	-47.2%
15	544	30	5.8%
16	186	0	0.0%
17	1141	193	20.4%
18	205	16	8.5%
19	58	-49	-45.8%
20	161	-112	-41.0%
21	347	-155	-30.9%
22	223	-37	-14.2%
23	525	-370	-41.3%

By applying a log-log transformation on data, it was possible to get another statistically significant model:

$$CFP = \#FPr^{1.00357} \times 1.588 \times AvDGperFPr^{1.0312} \quad (35)$$

$Pr(>|t|) < 0.001$ for each independent variable; the adjusted $R^2 = 0.968$.

This model is characterized by MMRE = 16.8%, Pred(25) = 78.3%, error range = [-38, 79%]. Estimates and errors are reported in Table 49.

Table 49 Estimates obtained using equation (35)

P.Id	Estimated Size [CFP]	Error	% Error
1	-10	-11.6%	11.6%
2	9	16.1%	16.1%
3	9	9.9%	9.9%
4	3	4.3%	4.3%
5	-7	-6.8%	6.8%
6	-3	-4.7%	4.7%
7	4	3.4%	3.4%
8	-31	-25.0%	25.0%
9	23	34.8%	34.8%
10	-7	-6.0%	6.0%
11	-5	-5.6%	5.6%
12	24	77.4%	77.4%
13	-48	-19.0%	19.0%
14	-38	-10.6%	10.6%
15	49	9.5%	9.5%
16	21	11.3%	11.3%
17	281	29.6%	29.6%

18	23	12.2%	12.2%
19	-22	-20.6%	20.6%
20	-51	-18.7%	18.7%
21	35	7.0%	7.0%
22	-99	-38.1%	38.1%
23	-46	-5.1%	5.1%

5.2.4 Results and observations

Answer to Q1

The first relevant result of the work described in this paper is that we can answer positively to the first research question Q1 (During the requirements elicitation and specification phase, is it possible to write progressively more complete and detailed UML models that support progressively more accurate simplified CFP measurement methods?).

In subsection 5.2.2 we described how to write UML models of user requirements that support simplified methods for measuring CFP. In particular, we described UML models that correspond to four completeness and detail levels (as depicted also in Figure 39):

- a) At the most abstract level, the model represents just the list of functional processes that are provided by the application being measured to functional users.
- b) At a slightly more detailed level, the model represents the data groups managed by the application.
- c) At a further detailed level, the model specifies the data groups involved in each functional process.
- d) At the most detailed level, the model includes all the details required to identify the data movements involved in each functional process, i.e., the information required to compute the size in CFP.

Answer to Q2

To answer question Q2 (What is the accuracy of the estimates provided by different simplified CFP measurement methods?) we first built the models, then we evaluated their accuracy.

Using the dataset described in Subsection 5.2.3 we checked the possibility of deriving simplified measurement methods corresponding to each of the abstraction level listed above. The most detailed models (level d) were used to measure the size in CFP according to [33]. No methods corresponding to level b) could be found (more precisely, no statistically significant model could be derived via OLS linear regression). For levels a) and c) two types of methods were defined: those based on mean values of the elements of the COSMIC method (like the mean size of functional processes or the mean number of data groups per functional process) and those derived from linear regression analysis. These estimation models and the accuracy of the obtained estimates are given in Table 50.

Table 50 Simplified size estimation models and their accuracy

Model		MMRE	Pred(25)
name	**formula**		
avg1	CFP = 7.3 × #FPr	36.6%	39.1%
reg1	CFP = -16.5 + 6.698 #FPr	22.7%	69.6%
avg2	CFP = AvDGperFPr × 1.8 × #FPr	19.3%	82.6%
reg2	CFP = -64.6 + 7.63 #FPr + 9.71 AvDGperFPr	19.8%	69.6%
log2	$CFP = 1.588 \times \#FPr^{1.00357} \times AvDGperFPr^{1.0312}$	16.8%	78.3%

The data reported in Table 50 were computed under the hypothesis that the measures obtained via the official COSMIC method [33] are correct. COSMIC measures were obtained by analyzing UML models that describe user requirements and are measurement-oriented, i.e., they are built so as to contain all the information required by FSM methods, as described in [3] and [2]. So, a measurer that bases his/her counting on a well written UML model has very little chances of making mistakes.

It can be seen that the first method (the one suggested in [59], and named "avg1" in Table 50) does not yield very accurate estimates. This clearly appears also by looking at the boxplots that represent the relative errors (Figure 48) and the absolute relative errors (Figure 49). Since it could be argued that the heterogeneity of the dataset affected the accuracy of this method, we computed the mean size of functional processes on a homogeneous subset of ten applications, including all the real-life applications from the same company: the accuracy obtained was only marginally better than the accuracy reported in Table 50.

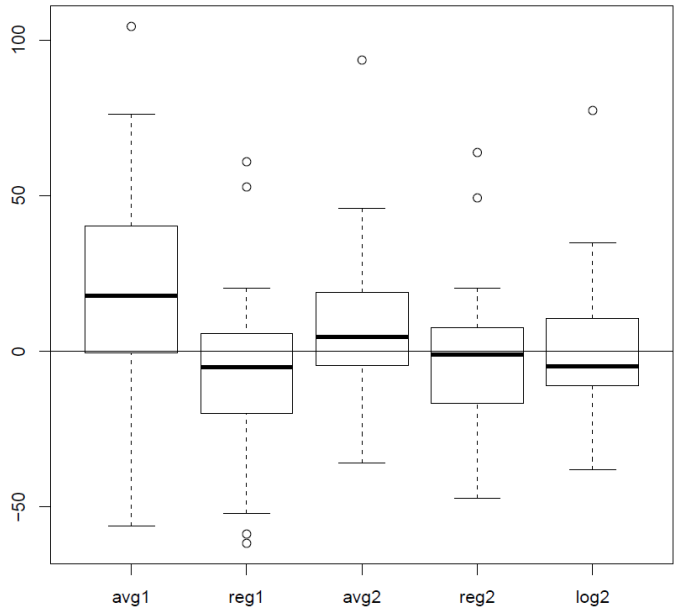

Figure 48 Boxplot of relative size estimation errors

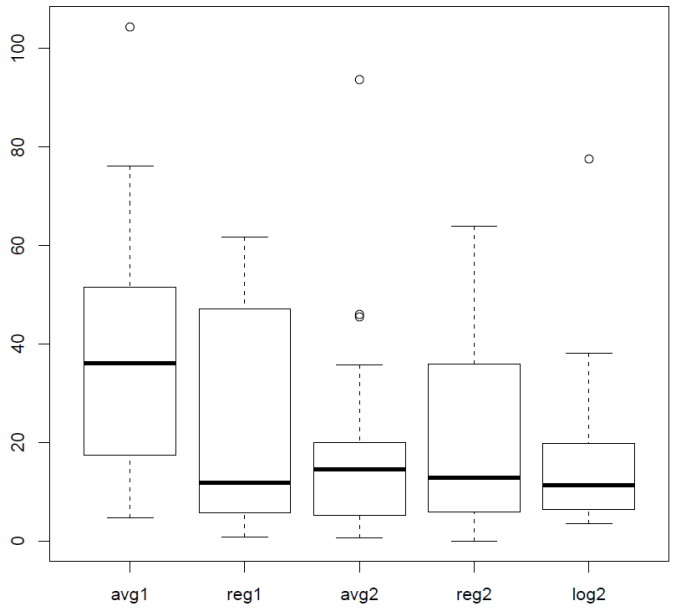

Figure 49 Boxplot of absolute relative size estimation errors

By means of linear regression we obtained a definitely more accurate estimation model that uses only the number of functional processes as independent variable (named "reg1" in Table 50).

When looking for models that take into account both the number of functional processes and the number of data groups involved in each functional process, we are able to define:

- A model (named avg2 in Table 50) that exploits the quasi-constancy of the number of data movement per data group involved in a functional process.
- A model (named reg2 in Table 50) obtained via OLS linear regression.
- A model (named log2 in Table 50) obtained via OLS linear regression after log-log transformation.

While model avg2 is clearly more accurate than model avg1, model reg2 appears preferable to model reg1 in that both its mean and median magnitudes of errors are closer to zero, and because it is less prone to give negative errors. According to the latter observation, using reg2 it is less likely that an application size (hence, its development effort) is grossly underestimated, with consequent potentially disastrous consequences. Model log2 is –like reg2– less prone to give negative errors; moreover, it features a quite little variance of absolute relative errors (see Figure 49).

Answer to Q3

Finally, concerning question Q3 (Do simplified CFP measurement methods provide a level of accuracy that is proportional to the amount of information required?) we have to address it on a qualitative basis, since we are not really interested in quantifying the amount of information needed by each simplified measurement process. In fact, aims at assessing if the effort needed to produce UML models that provide more and more information is worth the improvement in accuracy of the size estimations that can be obtained.

A first answer to Q3 is that progressing from a model like the one represented in Figure 40 to a model like the one in Figure 45 requires some effort, but allows to get better

distributions of estimation errors (as discussed above), if not a great increase in mean accuracy. In practice, the risks connected with wrong estimates are decreased.

However, additional considerations depend on the development process being adopted. If an organization uses UML for requirements modeling and the COSMIC method as described in the manual [33] for size measurement, then models like those illustrated in Figure 40 and Figure 45 will be produced anyway, during the modeling process. Thus, using them to get early estimates of the application's size is just an opportunity that comes for free. It is also interesting to note that in order to be able to compute the parameter required by equation (25) and (28), or to be able to perform regression analysis, full-fledged COSMIC measures (including #FPr, #DG, etc.) have to be collected and stored in a historical data repository.

On the contrary, organizations that do not perform full-fledged COSMIC measurement will not be able to collect historical data, including those data (#FPr, #DG, etc.) that are needed to compute measure estimates. These organizations can still use UML-based models for simplified measurement processes, but will have to use parameters derived from measures from other organizations.

5.2.5 Threats to validity

A possible threat to internal validity is the limited number of projects in our sample. Despite the relatively small numbers of data points, we still filtered out outliers (using Cook's distance as an indicator), to make sure that the results are not unduly influenced by a very small number of high-leverage points, even though this further reduced the cardinality of the samples.

The main threat to the external validity of the study may come from the projects chosen, which are a limited sample of a much larger population. However, this kind of threat is typical in most empirical software engineering studies. Also, the sample of projects is a "convenience" sample, i.e., it is made of projects that were selected because the data that we needed for our study were available. Note that, however, we are not interested here in specific models (e.g., we are not interested in the coefficients of the models), but, rather, in the performance of the techniques we propose. At any rate, it is not easy to assess the extent to which our results may apply in general.

There may be a threat to construct validity due to the use of MMRE, which has been criticized in the past as an accuracy indicator [60]. To mitigate this risk, we used MMRE along with other accuracy indicators. Our results show that they provide concordant results as for the accuracy of the models we built, so the indications provided by the set of our accuracy indicators can be deemed reliable.

5.2.6 Conclusions

It is not uncommon that a project manager needs an estimate of the functional size of the software application to be built even before the requirement specification phase is completed. Alternatively, project managers could simply want to limit the cost or time needed to measure the functional size of the application to be built. In these cases, simplified FSM methods are often used.

When UML is used in the early phases of development, it would be very convenient to apply simplified FSM methods to UML models. In particular, during the requirements

specification phase, UML models grow in detail, thus providing the information required by progressively more accurate size estimation methods. Actually, in this paper we showed that it is possible to build UML models that support adequately the application of two simplified measurement methods and the standard COSMIC method.

Based on the UML models, and using a dataset composed in large part of real-life project data, we were able to define quantitative size estimation models based on only the number of functional processes, or the number of functional processes and the number of data groups used in each functional process. We showed that size estimation methods' accuracy grows with the amount of information used. The models and their accuracy are summarized in Table 50.

It is also important to observe that the information contained in the UML models illustrated in sub-section 5.2.2 is just the information required to document applications' requirements properly. Therefore, size estimates obtained via simplified measurement processes can be seen as 'by products' of the progressive refinement of UML requirements models.

Future work includes extending the dataset, to increase the reliability and to guarantee the general validity of the results presented here.

This page intentionally left blank.

Chapter 6 Conversion between FPA and CFP

Several software development organizations are considering to change functional size measurement method from Function Point Analysis (FPA) [67] [10] to COSMIC [33], mainly because the latter is more easily and generally applicable than FPA. Such phenomenon is witnessed –for instance– by the growing number of COSMIC measures in the ISBSG database [68].

However, moving from FPA to COSMIC implies that the experience bases funded on Function Points (FP) become unusable. Since most organizations are not willing to make their historical data no longer usable, converting functional size measures – especially FP into CFP– is necessary and is a growingly interesting problem, in order to continually use them. This problem (see for instance the discussion in [69]) leads to the need of a conversion procedure that transforms FP measures into CFP measures. For the sake of precision, it is correct to remember that the COSMIC method allows measuring software that is structured in layers and peers, while FPA only addresses the measurement of an entire software application. Accordingly, the problem of size measure conversion applies only in the latter case.

The problem of converting functional size measures expressed in Function Points into measures expressed in CFP has received much attention from researchers. The work concerning convertibility among Functional size measures can be classified into three main streams.

- Manual conversion.
- Theoretical conversion within an empirical range
- Statistically based conversion.

6.1 The analytical convertibility of FSM

In [7] we proposed to exploit the knowledge produced by the FPA counting process and the similarity of FPA and COSMIC concepts in a procedure that guides the measurer in deriving COSMIC BFC from FPA BFC, thus greatly simplifying the COSMIC sizing. The procedure can be supported by a tool, which incorporates the knowledge of how FPA concepts map onto COSMIC concepts, thus easing conversions.

6.1.1 The conceptual basis

Our approach to convertibility between FP and CFP is based on the observation that the software models used by COSMIC and FPA have several elements in common, as already recognized in [46]. Figure 50 shows the possible conversion procedures.

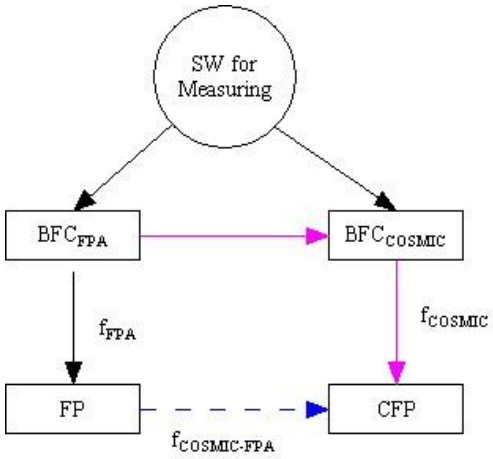

Figure 50 Roadmap to resolve the problem

FPA and COSMIC are characterized by a first phase, in which BFC and their characteristics are identified, and a second phase in which the size is computed as follows:

$$Size = f(BFC) \quad (36)$$

When dealing with a conversion from FP to CFP, $FP = f_{FPA}(BFC_{FPA})$ has already been computed, thus FP (the size in IFPUG unadjusted function points) is known. The conversion problem can be expressed as

$$CFP = f_{COSMIC\text{-}FPA}(FP) \quad (37)$$

However, defining function $f_{COSMIC\text{-}FPA}$ proved to be quite difficult. Therefore, we try a different way. We star by observing that the problem of measuring the size in CFP is articulated in the usual two stes:

- 1) Identifying BFC_{COSMIC}
- 2) Computing $CFP = f_{COSMIC}(BFC_{COSMIC})$

Now, it is clear that computing $CFP = f_{COSMIC}(BFC_{COSMIC})$ would be straightforward, if BFCCOSMIC were know, since it would simply require applying the COSMIC counting procedure as described in [3]. The real problem is thus identifying the $BFC_{COSMIC,}$ but this could be also quite easy, if we were able to convert BFC_{FPA} into BFC_{COSMIC}. The feasible conversion road map is from BFC_{FPA} to BFC_{COSMIC}, then to CFP.

To convert BFC_{FPA} to BFC_{COSMIC}, we can exploit the similarities among the two methods' elements, as described in Table 51.

Table 51 FPA to COSMIC element mapping

FPA	COSMIC
DET (in data file)	Data attribute
RET	Data Group
Transaction	Functional process
FTR	Data Group(s) read or written in the execution of a functional process

DET (in transaction)	Attribute of a Data Group that is subject of an Entry or Exit when executing the corresponding functional process
Set of DET crossing the boundary of the application as part of a transaction	Data movement of type Entry or Exit
FTR access within a transaction	Data movement of type Read or Write

Some of the mappings given in Table 51 are mentioned in the COSMIC literature. Namely, the fact that DET correspond to data attributes is mentioned in [33]; the fact that FPA transaction correspond quite closely to COSMIC functional processes is acknowledged in [33], as well as in [70] and [71]. The fact that FTR correspond to data groups read or written in a functional process is implicitly acknowledged in [33] and explicitly stated in [70].

The less obvious correspondence between FPA and COSMIC elements concerns data groups.
It is clear that the concept of a COSMIC data group matches quite closely the concept of FPA logic data file (as recognized in [70], for instance). Actually, most logic data files contain just one RET: in those cases we have that one logic data file corresponds to one RET and one data group. However, logic data files can contain multiple data subgroups (the RET), while the concept of data subgroup is absent in the COSMIC method (more precisely, such concept is mentioned when dealing with data exchanges between software layers, that are not considered here, since we deal only with the measurement of the application at the FUR level, as in FPA). According to the indications on data groups given in [33] and the definitions of logic data files and RET given in [10], it seems reasonable the COSMIC data group is mapped onto FPA RET.

Note that a consequence of mapping data groups onto RET is that when a FTR corresponds to a multi-RET logic data file, multiple data groups are involved.

It is important to stress that the correspondences illustrated in Table 51 do not hold *always*; however they hold *in most cases*. This is a very important point in practice. In fact, we do not need that the mappings are always valid (in that case, a totally automatic "translation" from FP to BFC would be possible, but at the cost of ensuring that each enforced correspondence is valid, which requires some intelligence). Instead, since we are looking for an efficient "manual" conversion process, it is sufficient that the described mappings hold on most occasions. So, the user performing a conversion can just check for cases when the mappings do not hold and deal with them. As long as the exceptions to the mappings described in Table 51 occur quite seldom, the conversion is very fast.

6.1.2 Proposed procedure of our approach

So, the approach we propose is organized as follows:

- 1) Convert FP software model elements into COSMIC software model elements. To this end, make reference to Table 51. In most cases the conversion is straightforward. In some particular cases, the mapping could be not applicable, and the converter has to use his/her knowledge and judgment.

– 2) The derived COSMIC BFC and related information are used to size the software application.

One could observe that this procedure configures a sort of "double measurement". In fact, at the end of the process we have both IFPUG and COSMIC measures, and both are documented by the detailed description of the respective BFC.

This observation suggests that –especially if a suitable tool is available– it is more efficient to perform the double measurement straight away, when the knowledge of FUR is fresh in the mind of the measurer, rather than a real conversion a posteriori. That is, when one measures an application, he/she can measure it according to both FP and CFP. The effort required for this double measurement is expected to be just a little bit greater than the effort required for applying one FSM method, thanks to the mapping defined in Table 51.

A suitable tool could also contribute to make the additional effort required to perform the second measure as little as possible. Ideally, we would like to get two measures at one measure's cost.

6.2 Tool support

In this section, we describe the tool that was developed to support our conversion approach. The tool is described via the use case "Warehouse Software Portfolio (WSP)" quoted in Section 4.2.

6.2.1 Initiation

The initial view of the tool is illustrated in Figure 51. The user can provide basic information concerning the project and the measurer, or switch directly to FPA or COSMIC specific views.

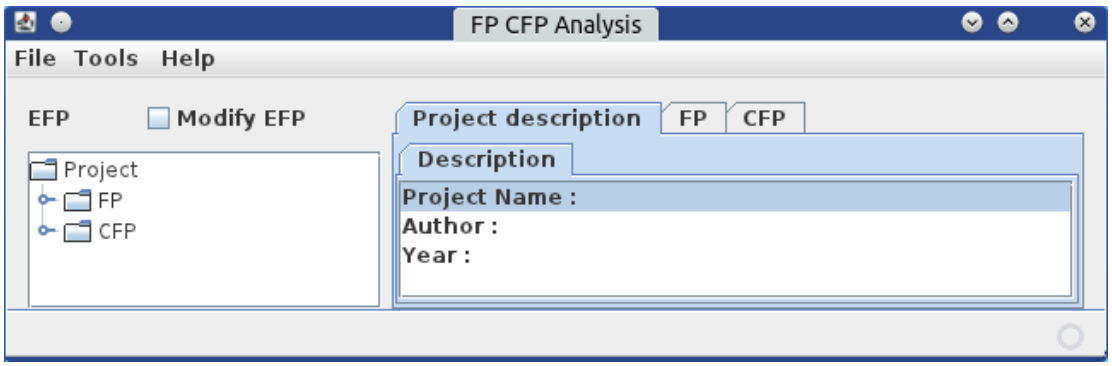

Figure 51 Initial view

6.2.2 Counting FPA

The tool supports measurement according to FPA. The user is required to provide the list of both data and transaction functions, and for each function the list of BFC (i.e., RET, DET and FTR).

In the FPA view (Tab "*FP*"), the tool provides two sub-views, one (Tab "*Data*") for describing the software model and the other ("*Analysis*") for computing the size.

FPA view - Software model – Data sub-view
In the software model view (Tab "*Data*" in Figure 52) it is possible to give the list of ILF, EIF, EI, EO and EQ, and for each function the relevant characteristics can be specified.

Consider for instance Figure 52: the user has already entered ILF Place (see Figure 28) and has specified that such ILF contains a single RET (also named Place) and is entering a new DET, whose name is Space.

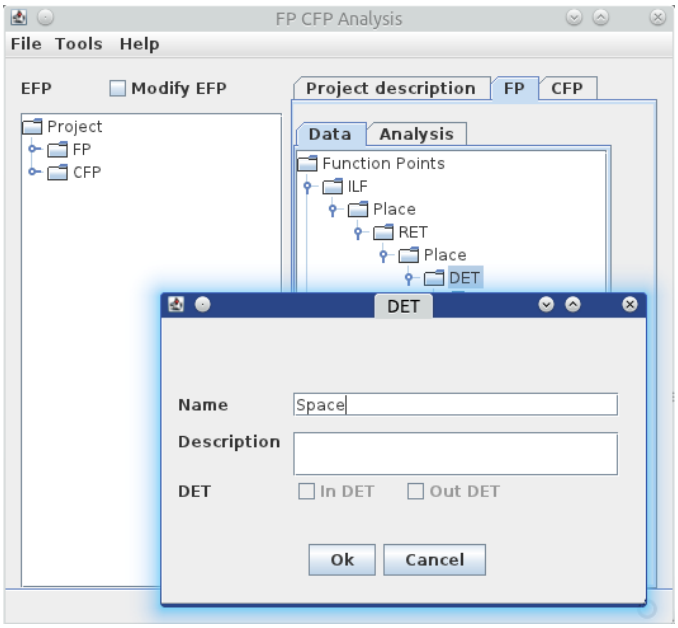

Figure 52 DET input form

At the end of the data description process, the tool will include a complete description of the data maintained by the WSP process, as shown in Figure 53.

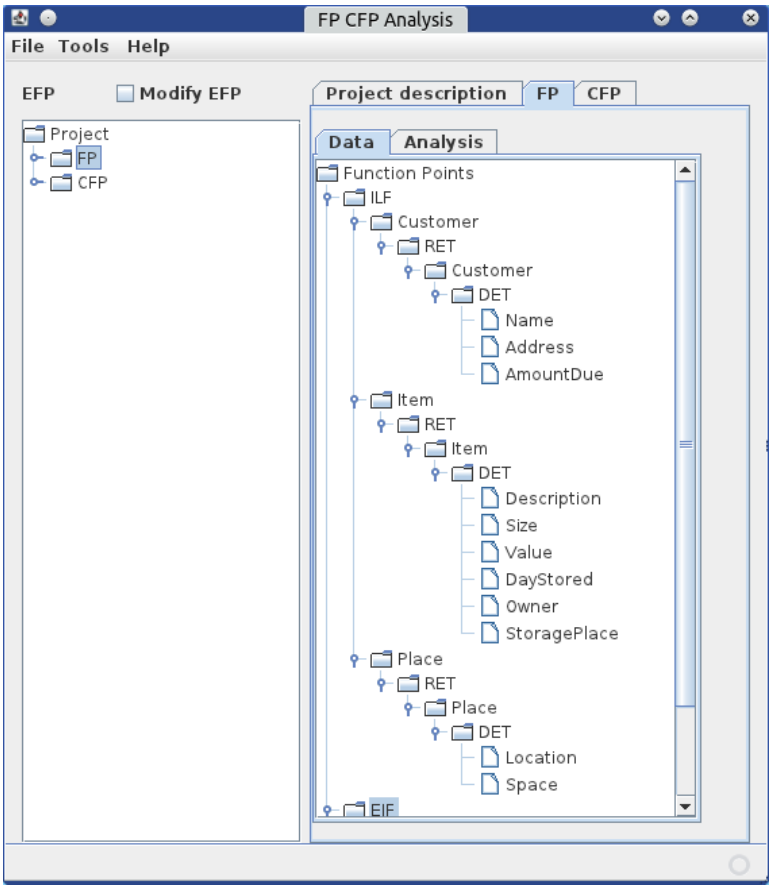

Figure 53 WSP data in the FP-software model specific views

FPA view - Software model – Transaction subview

The procedure for describing transactions is similar. The tool eases the identification of software model elements as far as possible: for instance, when describing a transaction, the user is presented the list ILF and EIF from which he/she can choose the FTR. Consider for instance Figure 54: the user is specifying the FTR of the external query QueryCustomers: the list of ILF and EIF is given, so that he/she can choose one.

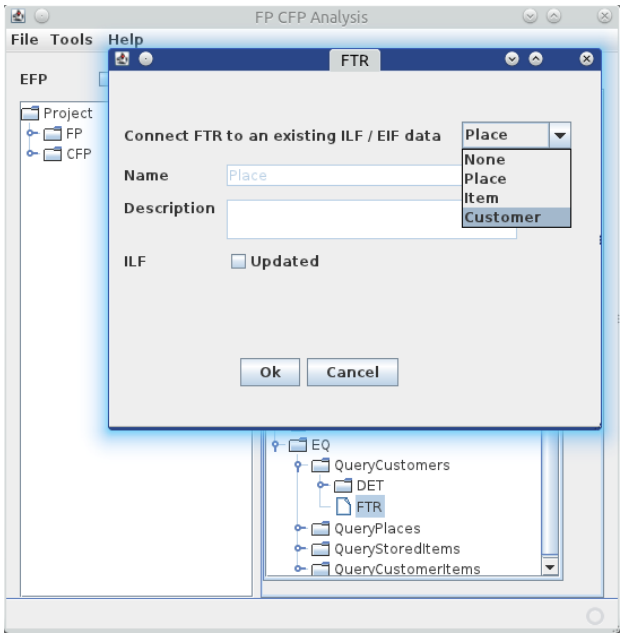

Figure 54 FTR choice

An interesting feature of the tool is that when specifying the DET of a transaction, it is possible to tell if they are entering or exiting the application (or both), as illustrated in Figure 55, where the AmountDue is specified as an outbound DET in the QueryCustomers. This piece of information is not relevant for FPA, but is useful when converting to CFP, since an entering DET can suggest the existence of an Entry data movement, while an exiting DET suggests an Exit data movement.

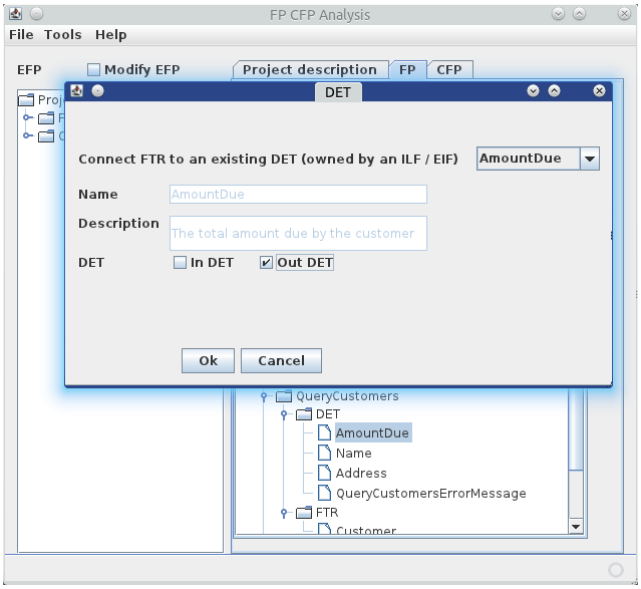

Figure 55 Specifying a function's DET

FPA view - Computes the size in FP

Once the BFCs have been properly identified, the tool automatically computes the size in FP: the results of the computation are shown in Figure 56. The total size of the WSP application is 77 FP (more precisely, 77 IFPUG Unadjusted FP).

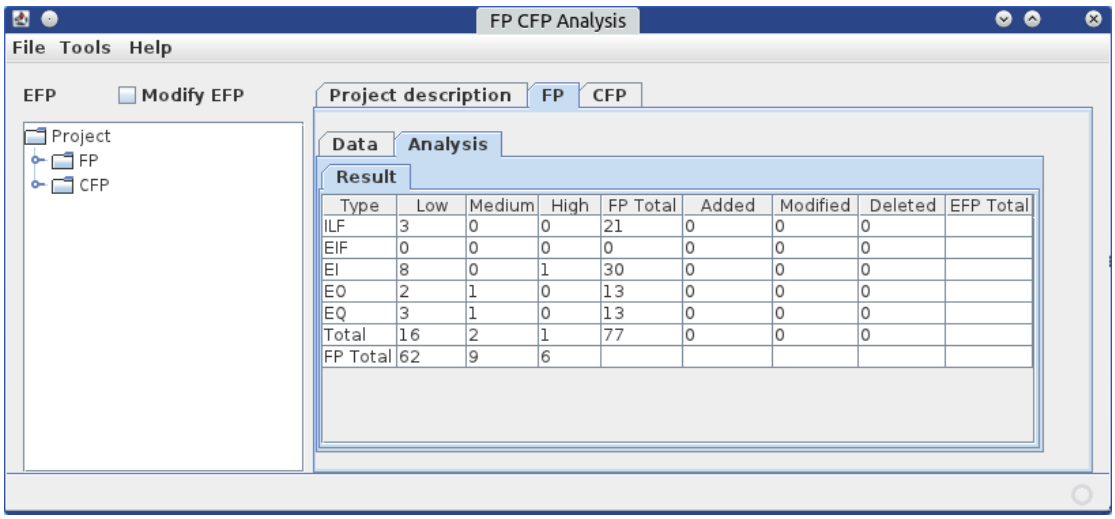

Figure 56 Function Point count of FP

Other function

Of course, the tool supports saving and loading measurement data. It is also able to export project data in csv (comma separated values), so that data can then be imported

into spreadsheets, databases, and other tools for permanent storing or for further elaboration.

6.2.3 Counting COSMIC

Similarly, the tool supports the counting of COSMIC FP. If the size of the given software application has not yet been measured in FP, the measurer can proceed with a "native" COSMIC measurement. In such case, the user is required to list the data groups, to identify the functional processes, and –for each functional processes– to tell which data movement are required. When describing a data movement, the list of data groups is made available, so that the user can pick the involved one.

If the size of the given application has already been measured in FP and the measure in CFP has to be computed, it is possible to exploit the convertibility concepts described in subsection 6.1.1.

Suppose that we have measured the WSP application in FP and we want to size it in CFP as well, we switch to the COSMIC view and find it empty, as expected (Figure 57)

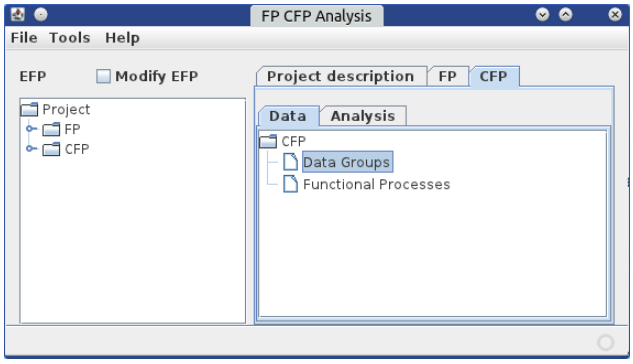

Figure 57 Empty COSMIC view

CFP view - Software model

When entering data groups, the tool suggests picking one of the logical data files identified during the FPA, as shown in Figure 58.

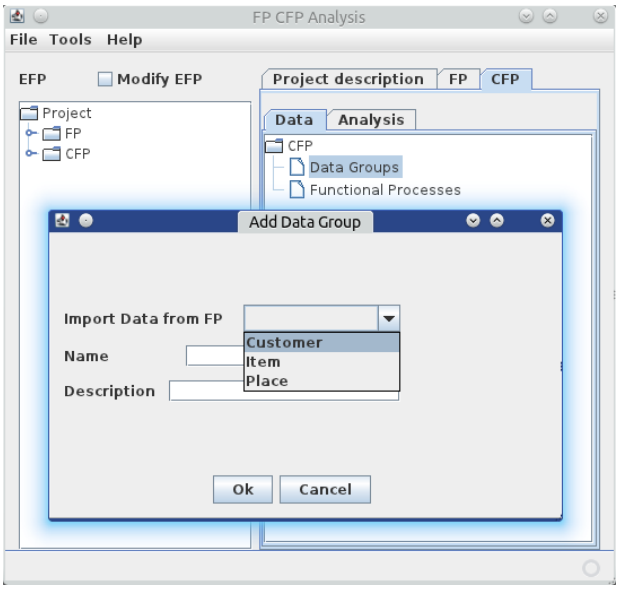

Figure 58 Specifying a data group after a FP logical data file

Of course, it is always possible to ignore the suggestions of the tool and insert different data. This is usually the case for transient data groups, which do not correspond to any FPA logical data file. The situation for the SWP after data group specification –not considering transient data groups– is illustrated in Figure 59.

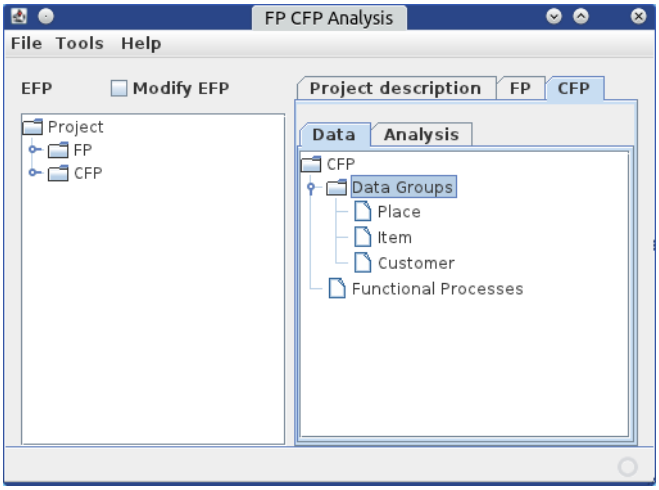

Figure 59 Data group

CFP view - Software model-Functional process

When inserting functional processes, the tool suggests picking them from the list of FPA transactions. It is thus easy to create the list of functional processes from the list of FPA transactions. The result is shown in Figure 60.

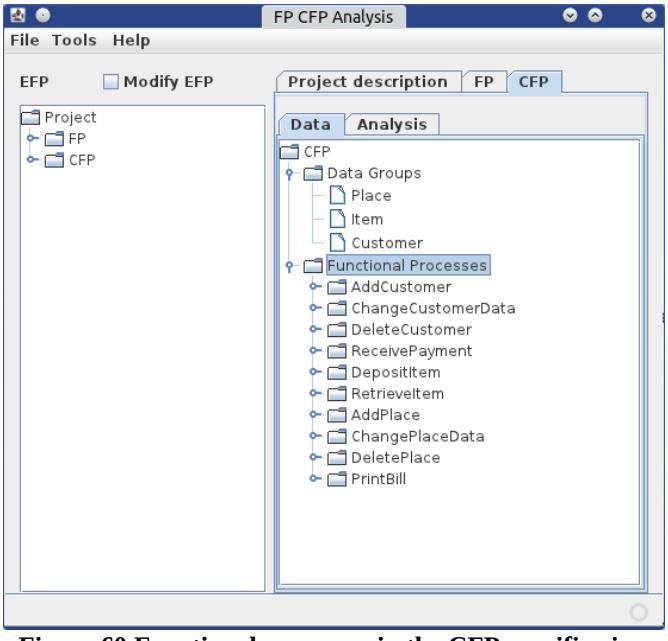

Figure 60 Functional processes in the CFP specific view

When entering data movements, the tool presents the list of data groups that have already been defined, since every data movement has to involve a data group. If the involved data group is not in the list, then the user has to add it. For instance, specifying data movements of the AddCustomer functional process is illustrated in Figure 61.

When considering the exits of the process, it is likely that the user realizes that a diagnostic is issued when the customer to be added is already in the database: transient data group CustomerAlreadyPresentErr is thus added to the list of data groups.

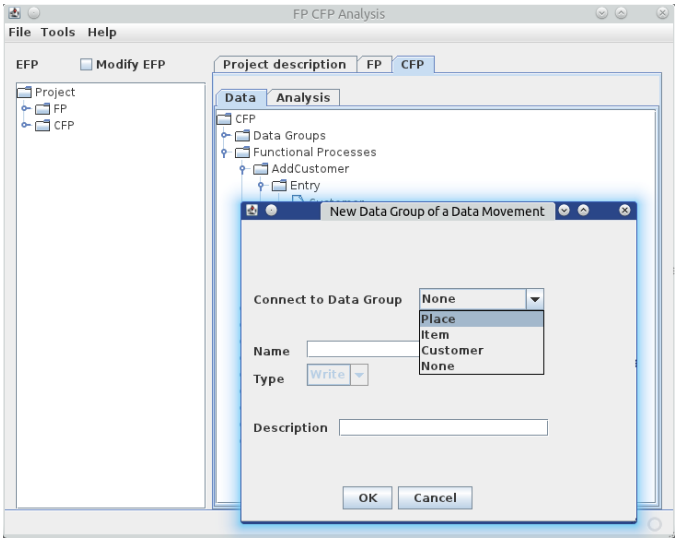

Figure 61 Data movement specification

COSMIC view - Computes the size in CFP

When all the data movements have been specified, the tool computes the total size in CFP, as illustrated in Figure 62. The resulting size is 83 CFP, slightly different from the size (81 CFP) obtained via a direct application of the COSMIC measurement process as reported in [33]: this difference is due to the fact that we based our counting on FUR that are slightly different from those reported in [33]; besides, we followed the indications of [39], which were not yet available when the WSP was described and measured in [33].

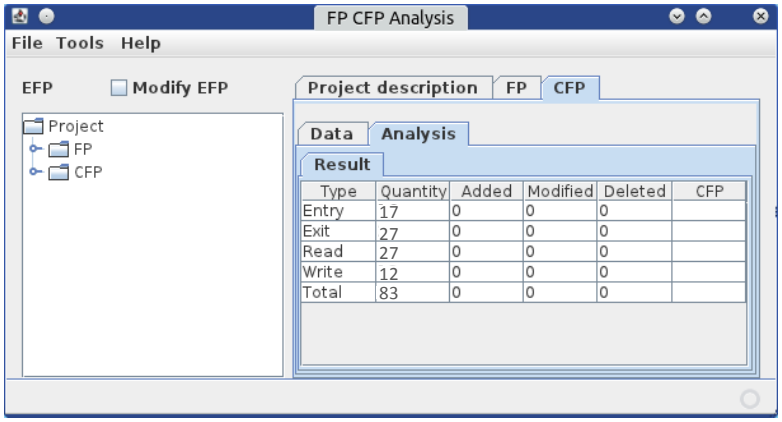

Figure 62 CFP count

6.3 Tool validation

In order to further test the proposed approach, we used the tool to size the tool itself. Although this tool is not a very large piece of software, it is larger than about 40% of the new development projects measured according to IFPUG Function Points appearing in the ISBSG dataset [68].

Table 52 Results of FPA for the tool in section 6.2

Function Type	Complexity			
	Low	Average	High	Size
ILF	14	0	0	98
EIF	0	0	0	0
EI	40	4	0	136
EO	2	0	0	8
EQ	11	1	3	55
Total				297

The first author of [7] used the tool to measure the tool itself according to FPA only. The results of the measurement are reported in Table 52.

The third author of [7] was given the FPA measures and model produced by the first author and used the tool to derive the COSMIC measure, using the conversion capability of the tool. In the meantime, the first author used the tool to measure it according to the COSMIC methods in "native" mode, that is, without using the conversion capabilities of the tool. The results of the measures are given in Table 53.

Table 53 Results of COSMIC measurement of the tool presented in Section 6.2

	Author 1	Author 3
Functional processes	60	60
Entries	60	69
Exits	57	82
Reads	76	97
Writes	70	70
COSMIC size	263	318

We gathered the data of the two cases in Table 54 to facilitate the analysis.

Table 54 Data gathered from the two cases study

Type of Values		Initial	Converted (Results)		Standard
Type of results		FPA	CFP	CFP	CFP
Case	1^	77	83	82	81
	2^	297	263	318	/
Measurement mode		Using the tool	Using the tool	Using the tool	via direct application of COSMIC
Performer		No.1 author	No.1 author	No.3 author	CASE
Ability of performer		quite experienced	quite experienced	little experienced	quite experienced

6.4 Lessons learned and conclusions

6.4.1 Lessons learned from the first case study

In conclusion, the tool effectively supports not only the sizing of the WSP application in terms of Function Points, but also the conversion of the measure into COSMIC Function Points.

It must be noted that –besides the availability of the BFC_{FPA} and their details– the conversion process needs that some knowledge about the process is available. Such knowledge is partly made available by the tool: for instance, the information that in the PrintCustomerItemList process the Value of each item and the total value of all items are output can be retrieved from the data associated with FPA transaction function (there are corresponding outbound DET). However, getting a clear idea of the role of each FTR and DET can require some effort, if the process details are not known.

Because of this reason, the most efficient way of performing the conversion is to do it on the fly, while measuring the size in FP. In fact, in this case, the procedure would be organized as follows:

- Step 1: Processes and data are analyzed and modeled according to FPA rules;
- Step 2: The corresponding COSMIC functional process and data groups are derived; since correspondences are suggested by the tool according to Table 51, this step is usually quite straightforward.
- Step 3: Data movements are specified. This step is eased by the fact that the processes have just been analyzed and data groups have already been defined.

If the conversion is made some time after the FPA measurement, step 3 is less straightforward, since the knowledge of the processes is no longer fresh, and has to be "reconstructed" with some effort.

6.4.2 Lessons learned from the second case study

The results of the measures are given in Table 53. It is easy to see that in this case the differences are larger than for the WSP system. We analyzed in details the differences and found two main types of reasons for differences in the measured sizes:

- Different interpretations of requirements. For instance, the two authors had different ideas about the feedback that the system has to provide to the user after performing (or not performing) some operations.
- Different interpretations of COSMIC counting rules in very specific cases. For instance, when some data is deleted, it is not clear whether the fact that the system shows nothing in place of the original data should involve an exit (i.e., writing "nothing" is an exit?).

In conclusion, we noted, from Table 54, that the tool's performance is only as good as its user's. In fact, during the modeling phase the tool acts as a simple editor, letting the user create and define FPA and/or COSMIC elements as he/she considers correct. Also in the "conversion" phase, the tool just highlights possible correspondences among FPA and COSMIC elements, but the responsibility of accepting the suggestions is ultimately with the user.

Actually, as described in Section 6.1.1, the "intelligence" of the tool is limited to the mappings among FPA and COSMIC concepts (see Table 51). Accordingly, our tool is not comparable with those that aim at automatically measuring the given FUR. In order to get a smarter support from the tool, it should be necessary to provide the tool with additional information. For instance, in principle one could think of deriving information concerning a FSM method form a model specifically built to support FSM, as described in [26]. However, this procedure is out of the scope of the work presented here. It could be the objective of future work.

6.4.3 Conclusion

In conclusion, the tool effectively supports not only the sizing of the WSP application in terms of Function Points, but also the conversion of the measure into COSMIC Function Points.

The case study showed that the approach is effective. However, we found out that even though the data groups and functional processes can be identified very easily on the basis of the FPA software model, identifying the type of data movements in which each data group is involved is not so immediate. To overcome this problem, two strategies are possible.

A first strategy consists in double measurement. When measuring the given application, each process is measured according to both FPA and COSMIC methods. This procedure costs very little more than applying a single measurement method. In fact, once a transaction has been measured according to FPA, the corresponding COSMIC functional process and data groups involved are immediately known, and identifying the data movements is very easy, thanks to the fresh analysis of the process.

A second strategy –which can be applied at any moment after FPA– requires exploiting all the details of transactions that were recorded. For instance, knowing that in a given transaction a given DET crosses the application boundary inbound suggests that the data group corresponding to the logic data file to which the DET belongs is subject to an Entry data movement, in the functional process corresponding to the given transaction.

This second strategy could also greatly benefit from automated support. The tool could look for the DET involved in transactions, identify the logical data file they belong to, identify the corresponding data group, and suggest the proper data movement according to the direction of the DET movement. We plan to extend the tool to implement this type of functionality.

Finally, it is worth stressing that the proposed technique is applicable also to convert COSMIC measures into FP measures, though this is less often required. Experimenting with this type of conversion is also among future objectives.

This page intentionally left blank.

Chapter 7 Investigation of statistical correlations between FSM and Object-Oriented Measures of Requirements models

It has been shown that functional size measures can be derived from UML models of requirements. In particular, if UML models are measurement-oriented, i.e., if they were written with the goal of clearly representing the elements upon which functional size emasurment is based, it is easy to identify BFC and all those elements that contribute to size measures. However, the analysis of UML diagrams to identify BFC and the elements that have to be taken into consideration to compute functional size measures is still a manual process. On the contrary, it is very easy to automatically derive object-oriented measures –like those proposed by Chidamber and Kemerer [111]– from UML models. So, if we were able to find an association (and the corresponding quantitative model) between the object-oriented measures of a measurement-oriented UML model and the functional size measures derived from the same UML model, we could exploit this knowledge to simplify the measurement.

The situation is depicted in Figure 63. It is noticeable that activity "Object-oriented measurement" is fully automatic, while activity "Identification of factors characterizing the SW application" is performed manually. In principle, the automation of the latter activity would be possible, but it would imply either rather sophisticated expert reasoning, or a decoration of the input model with information that could help the counting program in difficult decisions (e.g., what is the main purpose of a process). In practice, till now no tool implementing fully automated FSM has been recognized compliant with the standards.

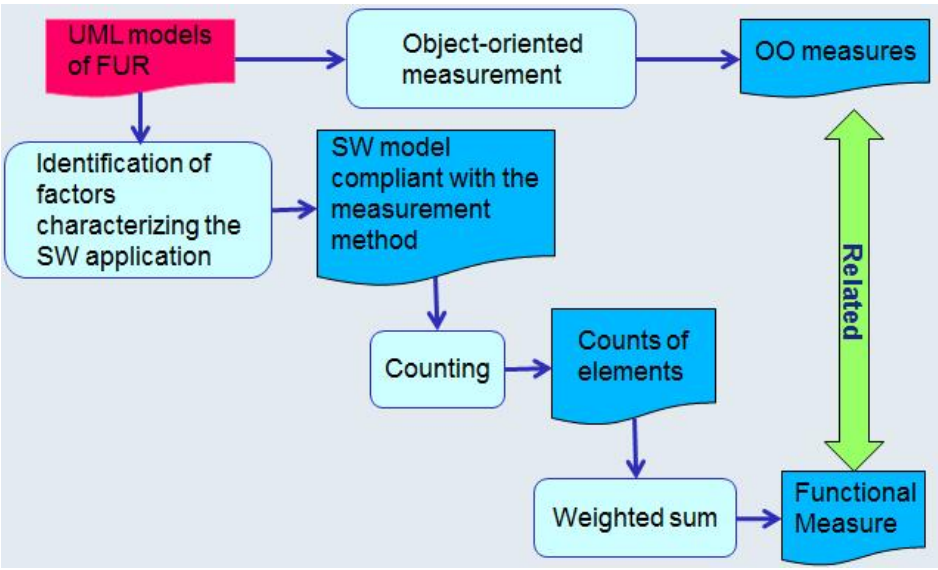

Figure 63 FSM Vs. OO measure

If we were able to find and model the relationship between OO measures and functional size measures, we could automatically measure the UML models and

- Estimate the functional size based on the OO measures, or

- Use the OO measures in place of the functional size measures, which would be no longer needed.

7.1 Object-oriented measurement

Before describing the work done, let's have a look at the measures that can be obtained from UML diagrams. After a brief survey of the available tools, we selected SDMetrics [54] as the most complete, mature, usable, and easily available tool.

SDMetrics accepts as input XMI and is able to measure nine UML diagrams, including sequence diagram, activity diagrams and use cases diagram. SDMetrics calculates about 120 measures and is also able to evaluate rules, covering all UML diagram types. After the initial configuration, measurement is performed automatically.

How to carry out the OO measurement using SDMetrics
In order to correctly carry out the OO measurement using SDMtrics, a set of project files, such as XML Source File, XMI Transformation File, Metamodel Definition File, and Metrics Definition File, are needed to be prepared and specified (only the first one is obligatory). Figure 64 [112] illustrates the role of the project files.

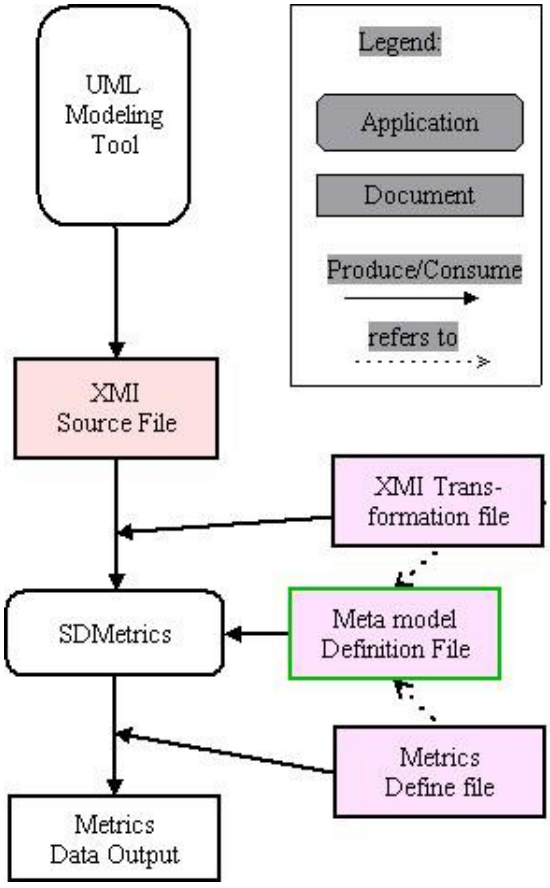

Figure 64 SDMetrics Project files

XMI Source File
SDMetrics works on XMI file generated from a UML case tool for extracting the needed information (such as attributes of a class, methods and their parameters, etc.).

This is the input of the SDMtrics and it includes all the information about the elements in the model being measured. We use the StarUml modelling tool to construct the models and export relative XMI files.

The SDMetrics metamodel definition file
The *SDMetrics* metamodel defines which UML model elements (e.g., classes, packages, associations, and so on) *SDMetrics* knows about, and what information is stored with each UML model element. This information is used to define and calculate design metrics.

The SDMtrics XMI Transformation File
XMI transformation file specifies how to retrieve the information pertaining to each *SDMetrics* metamodel element and its attributes from the XMI file.

Metrics definition file
Metrics definition file defines the set of metrics to be calculated for your UML model. The file contains a list of definitions of metrics, as well as sets (sets of UML elements, sets of values), rules and word lists, relation matrices, literature reference and glossary terms.

7.2 Organization of the empirical investigation

The research question we addressed is the following:
Is there a quantitative relationship that links functional size measures (namely, Function Points and COSMIC Function Points) of an application to some object-oriented measure of the UML model that describes the requirements of the same application?

In order to answer this question, we took the requirement specifications of a set of 11 software applications; then
1. UML measurement-oriented models of requirements were built;
2. Functional size measures were derived from models;
3. OO measures of UML models were obtained using *SDmetrics*;
4. Possible correlations between the measures obtained at steps 2 and 3 were studied, using statistical methods.

The process described above was carried out for both Function Point Analysis and the COSMIC method.

7.3 Datasets

In this section we give the datasets resulting from the measurement activities described in above.

Table 55 Measures collected according to the FPA method

Proj ID	UFP	ILF	EIF	EI	EO	EQ	Num_ ILF	Num_ EIF	Num_ EI	Num_ EO	Num_ EQ	AvFTR perTr
1	160	98	0	49	4	9	14	0	12	1	3	2.00
2	140	56	25	46	4	9	8	5	11	1	3	2.93
3	84	35	5	30	4	10	5	1	9	1	3	1.46
4	163	84	0	66	4	9	12	0	18	1	3	2.18

5	128	49	0	66	4	9	7	0	19	1	3	2.26
6	130	63	0	54	4	9	9	0	14	1	3	2.06
7	78	21	0	32	12	13	3	0	9	3	4	1.69
8	107	35	10	48	14	0	5	2	12	3	0	2.13
9	102	42	0	31	8	21	6	0	9	2	5	2.13
10	79	42	5	28	4	0	6	1	7	1	0	3.13
11	105	49	0	56	0	0	7	0	17	0	0	1.29

Table 56 Measures collected according to the COSMIC method

ProjID	CFP	Num_ FPr	Num_ DG	Entry	eXit	Read	Write	AvDM perFPr	AvDG perFPr
1	93	15	15	29	25	16	23	6.20	2.47
2	83	14	12	18	18	26	21	5.93	2.86
3	66	13	7	20	19	15	12	5.08	2.00
4	146	22	16	50	29	35	32	6.64	2.41
5	154	24	7	37	42	36	39	6.16	2.21
6	102	18	12	28	27	23	24	5.67	2.28
7	86	16	3	19	30	27	10	5.38	1.69
8	92	15	7	20	28	30	14	6.13	2.40
9	86	16	7	22	24	23	17	5.38	2.31
10	65	8	9	20	16	11	18	8.13	2.88
11	99	17	7	31	30	21	17	5.82	1.29

The measures obtained from our FPA-oriented models are given in Table 57.

Table 57 OO measures obtained from FPA-oriented UML models

Proj ID	Num_ Class	Num_ Attr	Num_ Met	AvMet perClass	AvAtt perClass	Num_ UseCase	Num_ Msgs	Num_ Sent Msgs	Num_ Rec. Msgs	AvMsgs perClass	AvMsgs perSD
1	17	88	65	2.04	3.67	15	127	44	83	7.47	10.62
2	13	53	38	2.92	4.08	22	104	34	70	8.00	8.00
3	7	36	38	5.43	5.14	12	64	19	45	9.14	5.33
4	17	49	55	3.24	2.88	23	128	44	84	7.53	5.57
5	8	34	27	3.38	4.25	24	114	37	77	14.25	4.75
6	15	39	36	2.40	2.60	19	111	39	72	7.40	5.84
7	3	11	16	5.33	3.67	16	65	26	39	21.67	4.06
8	7	29	20	2.86	4.14	15	65	26	39	9.29	4.33
9	8	30	37	4.63	3.75	17	91	34	57	11.38	5.35
10	9	19	33	3.67	2.11	7	70	24	46	7.78	10.00
11	7	25	28	3.11	3.44	18	75	29	46	10.71	9.61

The measures obtained from our COSMIC-oriented models are given in Table 58.

Table 58 OO measures obtained from COSMIC-oriented UML models

Proj ID	Num_ Class	Num_ Attr	Num_ Met	AvMet perClass	AvAtt perClass	Num_ UseCase	Num_ Msgs	Num_ Sent Msgs	Num_ Recv Msgs	AvMsgs perClass	AvMsgs per UseCase
1	17	86	0	2.17	3.74	15	44	44	83	5.29	10.35
2	13	44	38	2.92	3.38	22	104	34	70	8.00	4.73
3	7	36	34	4.86	5.14	12	64	19	45	9.14	5.33
4	17	47	54	3.18	2.76	23	128	44	84	7.53	5.57
5	8	34	26	3.25	4.25	24	114	37	77	14.25	4.75
6	15	39	38	2.53	2.60	19	110	39	71	7.33	5.79
7	3	11	9	3.00	3.67	16	65	26	39	21.67	4.06
8	7	24	10	1.43	3.43	15	65	26	39	9.29	4.33
9	8	30	11	1.38	3.75	17	77	29	48	9.63	4.53
10	9	19	32	3.56	2.11	7	70	24	46	7.78	10.00
11	7	23	25	2.78	3.22	18	77	29	48	8.56	9.65

7.4 Analysis

Linear regression was applied to the data described in the previous section, to find statistically significant models.

As is usual in empirical software engineering studies, we considered models having p-value < 0.05. Other validity conditions –like the normal distribution of residuals– were taken in due account.

Outliers were identified by means of Cook's distance.

The multivariate models described below are characterized by non-correlated independent variables.
Given the relatively small number of data points, to avoid overfitting, intercepts were forced to be null; i.e., all regression lines pass through the origin. This is reasonable, since an application having a null object oriented measure cannot account for non-null functional size.

7.4.1 FP vs. OO measures

In the following paragraphs the statistically significant models found from our FPA-oriented models are described.

7.4.1.1 UFP vs. Number of classes

A first model associates the size in UFP to the number of classes (the regression line is shown in Figure 65):

$$UFP = \#Class \times 10.56 \quad (38)$$

The model is characterized by $R^2 = 0.942$. No outliers were found.

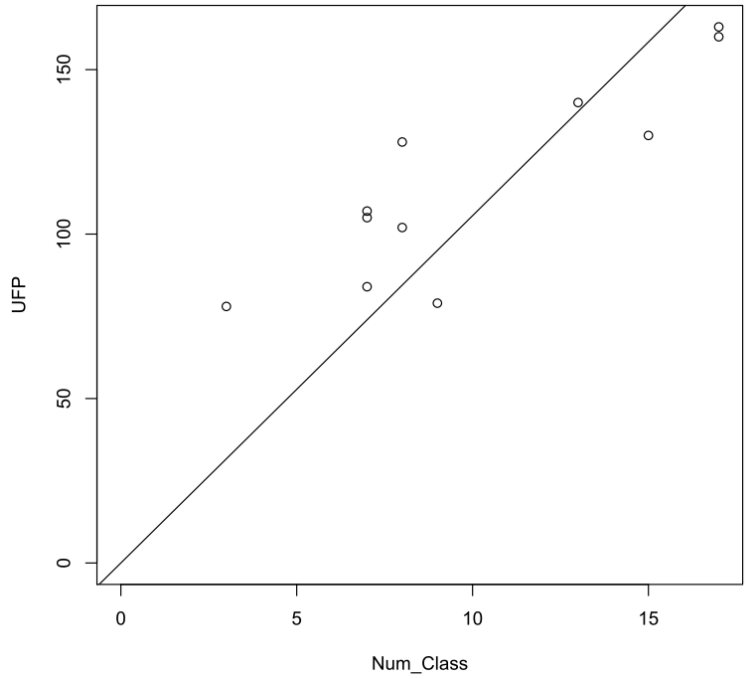

Figure 65 UFP vs. Num_Class regression line

The accuracy of the model is characterized by:

- MMRE = 22.7%
- Pred(25) = 63.6%
- Error range = (-59.4%, 21.8%)

The distribution of model's relative residuals is described in Figure 66.

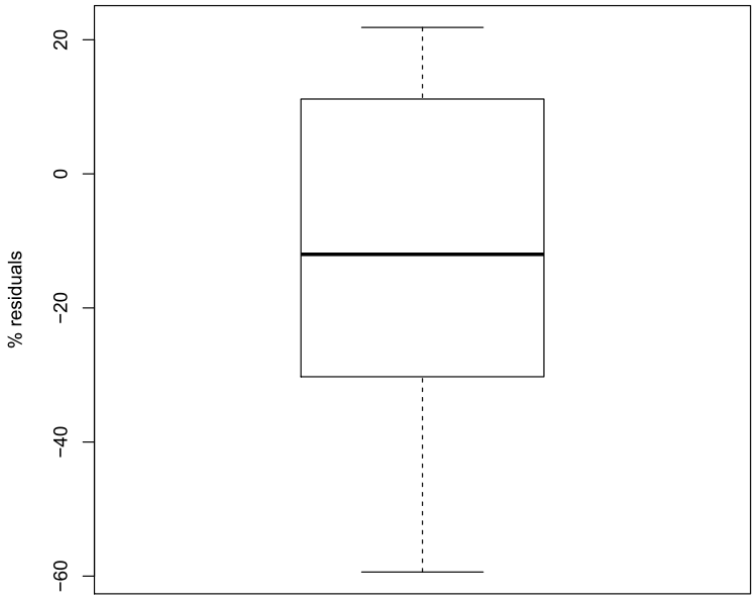

Figure 66 UFP vs. Num_Class residuals' distribution

7.4.1.2 UFP vs. Number of attributers

A second model associates the size in UFP to the number of attributes (the regression line is shown in Figure 67):

$$UFP = \#Attr \times 3.60 \quad (39)$$

The model is characterized by $R^2 = 0.976$. Three outliers were excluded in the computation of the regression.

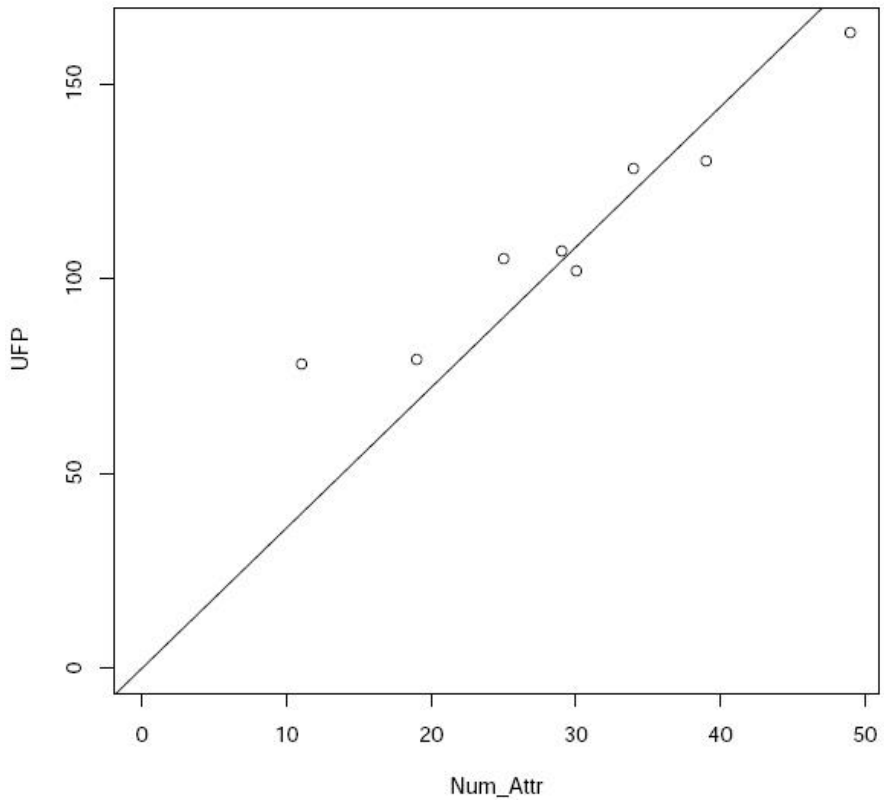

Figure 67 UFP vs. Num_Attr regression line

The accuracy of the model is characterized by:

- MMRE = 26.8%
- Pred(25) = 63.6%
- Error range = (-49.2%, 98.2%)

The distribution of model's relative residuals is described in Figure 68.

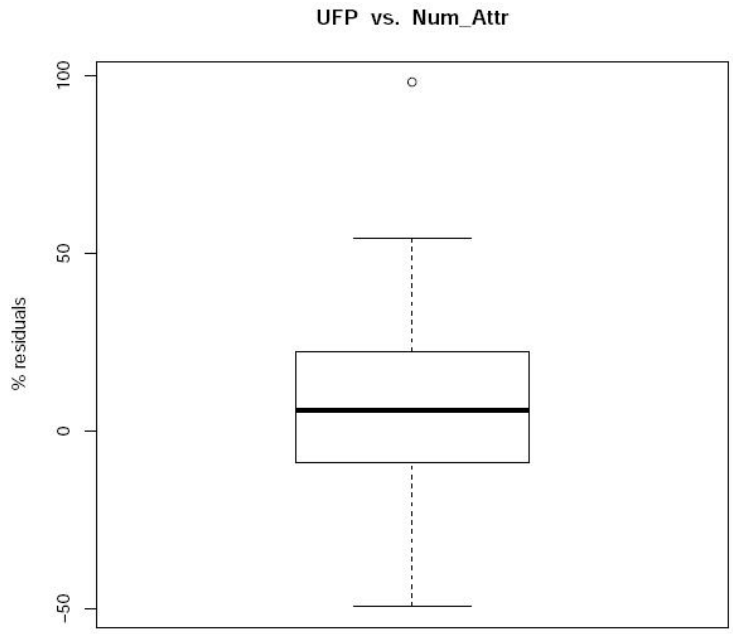

Figure 68 UFP vs. Num_Attr residuals' distribution

7.4.1.3 UFP vs. Numer of methods

A third model associates the size in UFP to the number of methods (the regression line is shown in Figure 69):

$$UFP = \#Met \times 3.28 \quad (40)$$

The model is characterized by $R^2 = 0.936$. One outlier was excluded in the computation of the regression.

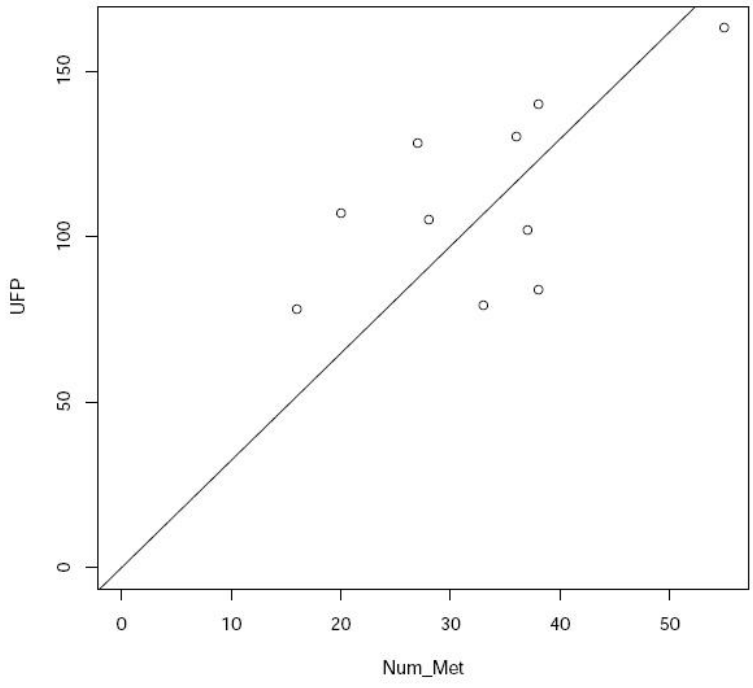

Figure 69 UFP vs. Num_Met regression line

The accuracy of the model is characterized by:

- MMRE = 25.3%
- Pred(25) = 45.5%

– Error range = (-39.5%, 46.5%)

The distribution of model's relative residuals is described in Figure 70.

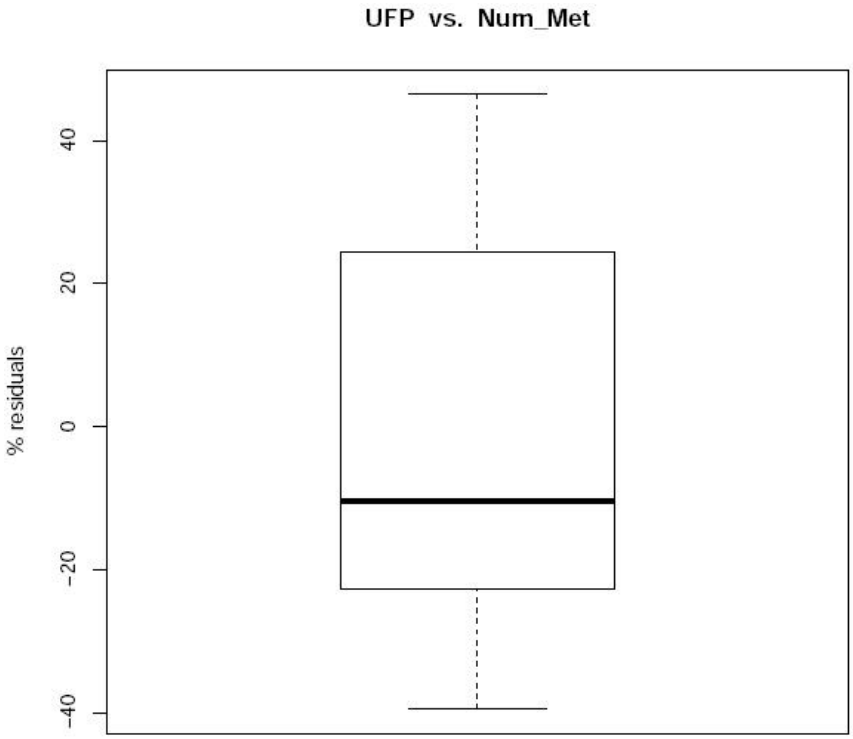

Figure 70 UFP vs. Num_Met residuals' distribution

7.4.1.4 UFP vs. Number of SentMessage

A fourth model associates the size in UFP to the number of SentMessage (the regression line is shown in Figure 71):

$$UFP = \#SentMessage \times 3.57 \quad (41)$$

The model is characterized by $R^2 = 0.989$. No outliers were found.

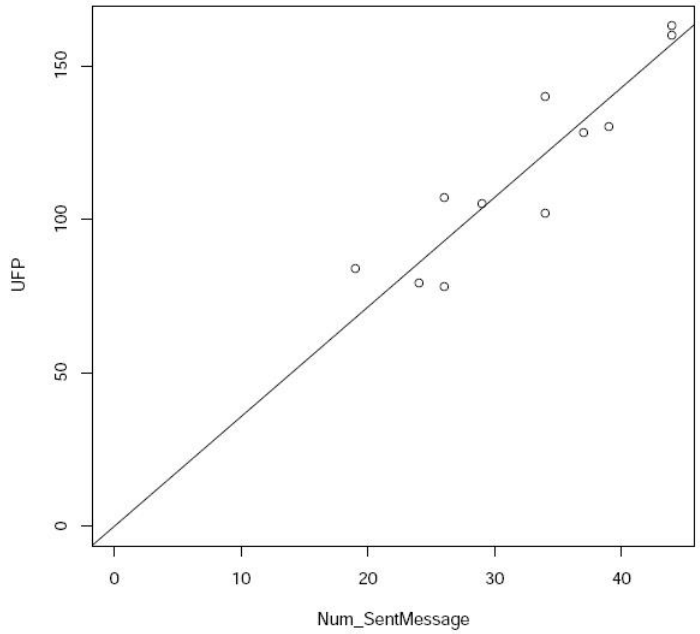

Figure 71 UFP vs. Num_SentMessage regression line

The accuracy of the model is characterized by:

- MMRE = 9.9%
- Pred(25) = 100.0%
- Error range = (-19.2%, 19.1%)

The distribution of model's relative residuals is described in Figure 72.

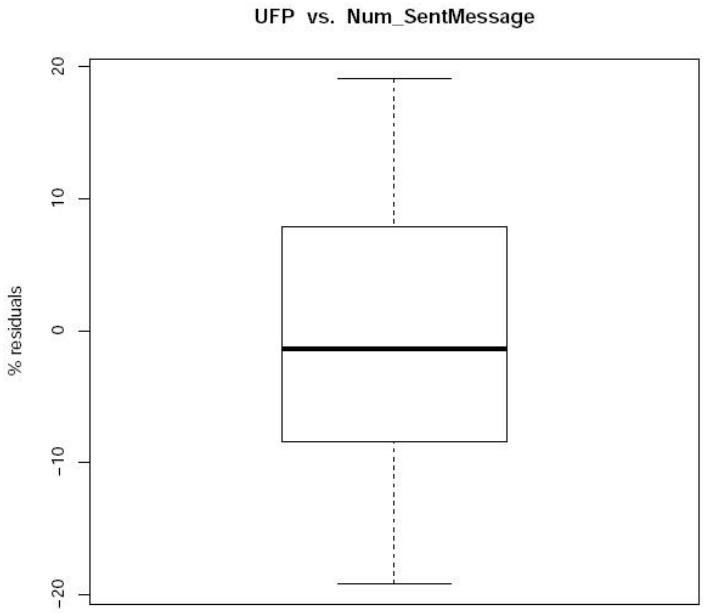

Figure 72 UFP vs. Num_SendMessage residuals' distribution

7.4.1.5 UFP vs. Number of class, Average number of methods per each class

A fifth model associates the size in UFP to the number of class and the average number of methods per each class:

$$UFP = \#Class \times 8.13 + \#AvMetperClass \times 9.10 \quad (42)$$

The model is characterized by R^2 =0.972. No outliers were found.

The accuracy of the model is characterized by:

- MMRE = 14.6%
- Pred(25) = 72.7%
- Error range = (-25.2%, 34.9%)

The distribution of model's relative residuals is described in Figure 73.

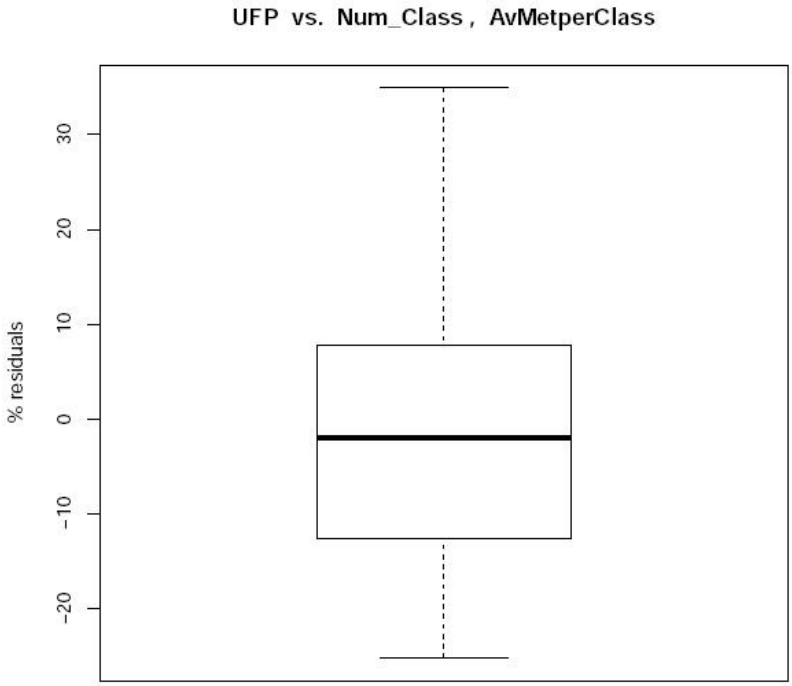

Figure 73 UFP vs. Num_Class and AvMetperClass residuals' distribution

7.4.1.6 UFP vs. Number of methods, Average number of attributes per class

A sixth model associates the size in UFP to the number of methods and the average number of attributes per class:

$$UFP = \#Met \times 2.07 + \#AvAttperClass \times 14.11 \quad (43)$$

The model is characterized by R^2 = 0.981. Two outliers were excluded in the computation of the regression.

The accuracy of the model is characterized by:

- MMRE = 17.8%
- Pred(25) = 81.8%
- Error range = (-14.6%, 79.9%)

The distribution of model's relative residuals is described in Figure 74.

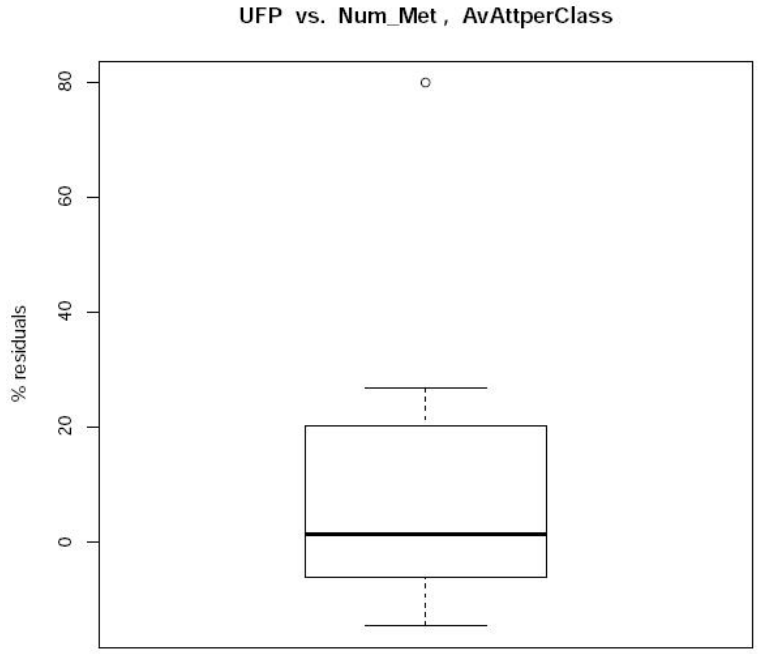

Figure 74 UFP vs. Num_ Met and AvAttperClass residuals' distribution

7.4.2 CFP vs. OO measures

In the following paragraphs the statistically significant models found from our COSMIC-oriented models are described.

7.4.2.1 CFP vs. Number of attributes

A first model associates the size in CFP to the number of attributes (the regression line is shown in Figure 75):

$$CFP = \#Attr \times 3.43 \quad (44)$$

The model is characterized by $R^2 = 0.929$. Three outliers were excluded in the computation of the regression.

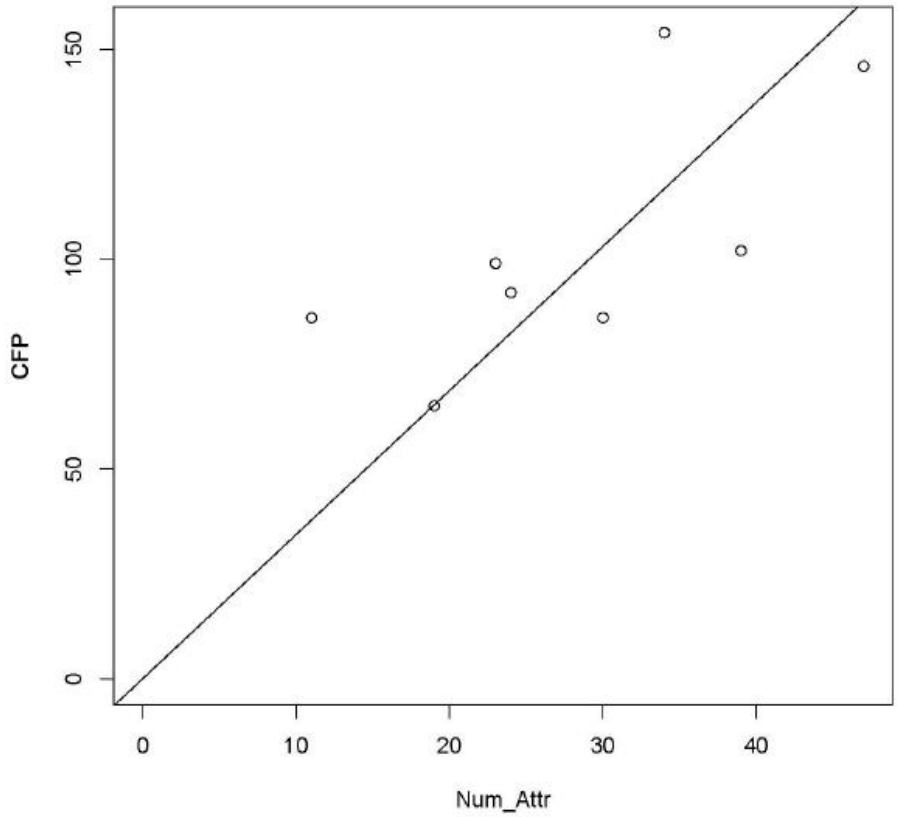

Figure 75 CFP vs. Num_Attr regression line

The accuracy of the model is characterized by:

- MMRE = 50.9%
- Pred(25) = 54.5%
- Error range = (-56.1%, 217.6%)

The distribution of model's relative residuals is described in Figure 76.

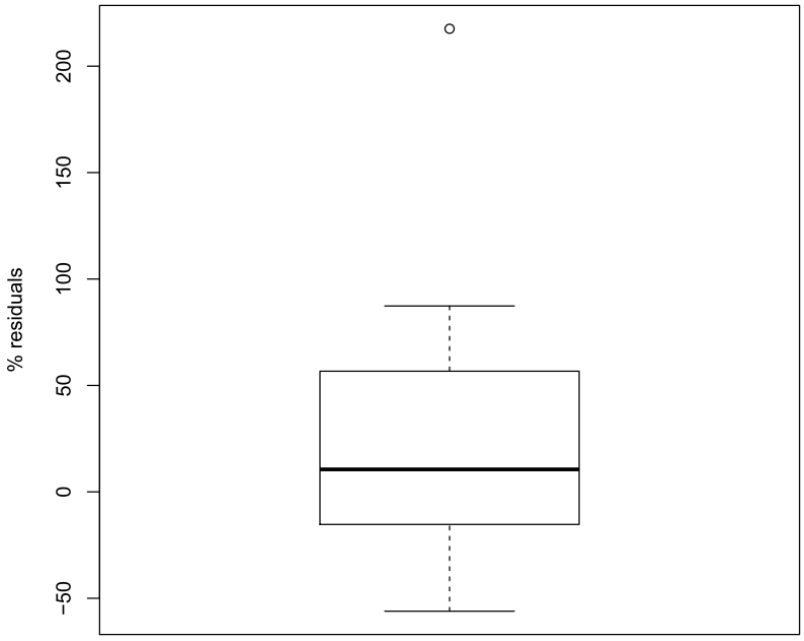

Figure 76 CFP vs. Num_Attr residuals' distribution

7.4.2.2 CFP vs. Number of sent messages

A second model associates the size in CFP to the number of SentMeaasges (the regression line is shown in Figure 77):

$$CFP = \#SentMessages \times 3.26 \quad (45)$$

The model is characterized by $R^2 = 0.994$. Four outliers were excluded in the computation of the regression.

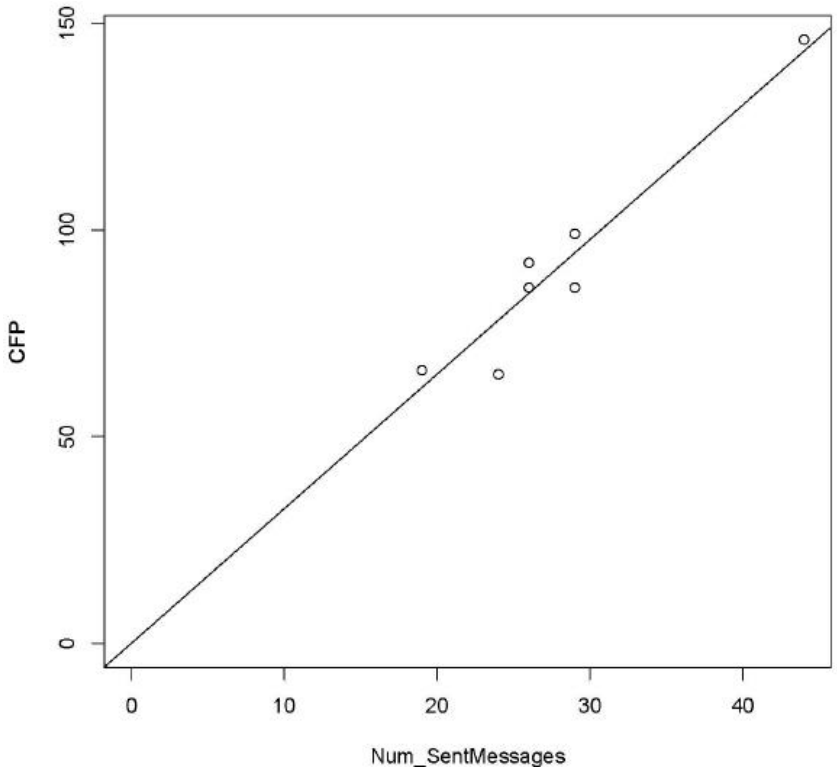

Figure 77 CFP vs. Num_SentMessages regression line

The accuracy of the model is characterized by:

- MMRE = 16.9%
- Pred(25) = 81.8%
- Error range = (-21.8%, 54.0%)

The distribution of model's relative residuals is described in Figure 78.

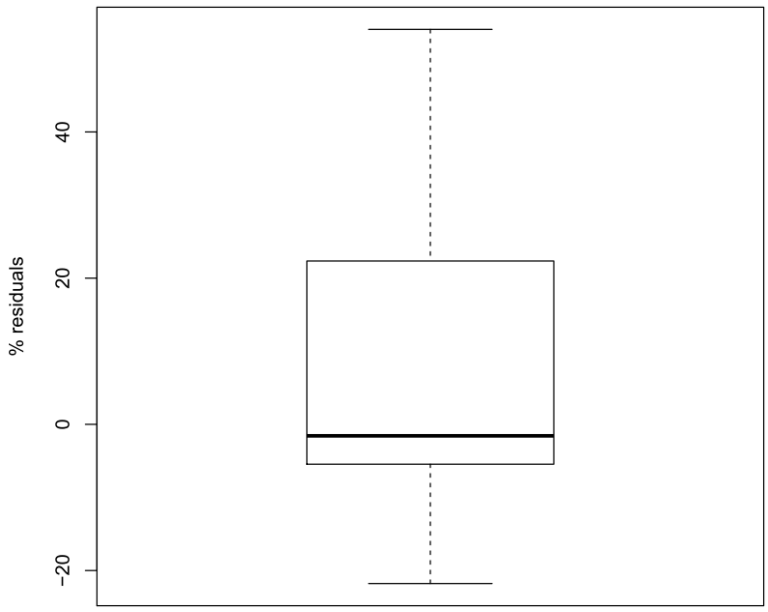

Figure 78 CFP vs. Num_Sent_messages residuals' distribution

7.4.2.3 CFP vs. Number of classes, Number of Use Case

A third model associates the size in CFP to the number of class and the number of Use Case:

$$CFP = \#Class \times 1.48 + \#UseCase \times 4.89 \quad (46)$$

The model is characterized by $R^2 = 0.988$. Two outliers were excluded in the computation of the regression.

The accuracy of the model is characterized by:

- MMRE = 13.5%
- Pred(25) = 81.8%
- Error range = (-27.0%, 53.6%)

The distribution of model's relative residuals is described in Figure 79.

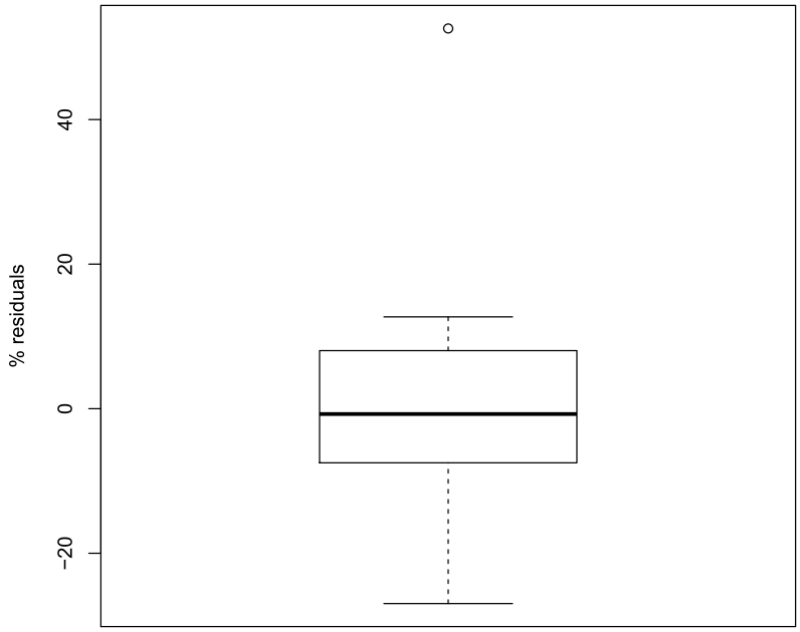

Figure 79 CFP vs. Num_Class and Num_UseCase residuals' distribution

7.5 Discussion of results

In Table 59, we summarized the results obtained in previous section.

Table 59 Model-based Measurement-oriented OO estimation models and their accuracy

FSM	ID	Factor(s)	Model	R^2	MMRE	Pred(25)
FPA	1	Class	UFP = #Class×10.56	0.942	22.7%	63.6%
	2	Attribute	UFP = #Attr×3.60	0.976	26.8%	63.6%
	3	Method	UFP = #Met×3.28	0.936	25.3%	45.5%
	4	SentMessage	UFP = #SentMessage×3.57	0.989	9.9%	100.0%
	5	Class, AvMetperClass	UFP = #Class×8.13 + #AvMetperClass×9.10	0.972	14.6%	72.7%
	6	Method, AvAttperClass	UFP = #Met × 2.07 + #AvAttperClass×14.11	0.981	17.8%	81.8%
CFP	7	Attribute	CFP = #Attr×3.43	0.929	50.9%	54.5%
	8	SentMessage	CFP = #SentMessages×3.26	0.994	16.9%	81.8%
	9	Class, UseCase	CFP = #Class × 1.48 + #UseCase×4.89	0.988	13.5%	81.8%

Statistical analysis showed that both FPA and COSMIC functional size measures appear correlated to object-oriented measures. In particular, associations with basic OO measures were found:

- FP appear associated with the number of classes, the number of attributes and the number of methods
- CFP appear associated with the number of attributes.

This result suggests that even a very basic UML model, like a class diagram, can support size measures that appear equivalent to functional size measures (which are much harder to obtain).

Both FPA and COSMIC functional size measures appear associated with the number of messages sent by a class in sequence diagrams (i.e., in elementary/functional processes). Moreover, the model that gives functional size as a function of the number of messages sent by a class is the most accurate for FPA and the second most accurate for COSMIC. This is noticeable, but not very surprising. In fact, the number of messages sent is derived from sequence diagrams, which convey quite detailed descriptions of elementary/functional processes. Noticeably, this result is coherent with the findings described in Section 5.2.

Therefore, practitioner can use our Model-based Measurement-Oriented object-oriented (MbMO-OO) method as a simplified method to measure OO applications.

However, analysis on the models listed in Table 59 shows that, although FSM rules are considered during the modeling process, several models consider only "plain" UML elements (as the number of classes or the number of attributes) and no elements contributing to functional size (like FTR or RET). Based on this observation, we could measure UML models built in the usual way: we do not need to build measurement-oriented model that incorporate FSM concepts. For example, the numbers of classes and use cases are the same in customary UML models and in COSMIC-oriented UML models; therefore the analyst does not have to consider FSM rules when he/she models the FURs. The model of CFP = #Class × 1.48 + #UseCase×4.89 (see Table 59) can be applied to the measures (#Class, #UseCase) obtained from a UML model built according to plain object-oriented analysis criteria. In this way, the modeling & measurement process can be much simpler and cost less.

7.6 Threats to validity

The dataset used for the empirical study described above is relatively small. Hence, it is possible that it is not representative of any possible application.

7.7 Conclusions

The results obtained tend to confirm that, having modeled an application's functional user requirements using UML and highlighting the typical elements of software models used by FSM methods (i.e., elementary/functional processes, data files/groups, etc.), the measures obtained *automatically* by means of measurement tools like SDMetrics are essentially equivalent to those obtained by certified functional size measurers.

The small cost, rapidity, accuracy, and repeatability of UML measurement suggests that manually performed FSM methods could be abandoned. However, the results reported here need further experimental evidence, before we can safely suggest practitioners to drop FSM methods.

This page intentionally left blank.

Chapter 8 Related work

This chapter consists of two parts: one is about simplified function points, the other part is about the conversion between FPA and COSMIC.

The generic concepts of Function Points Analysis were published in the late 1970s; later, more detailed measurement rules were developed to improve consistency of measurement. Due to lack of good software documentation, it is not always possible to apply all the detailed rules, and measurers must fall back on approximation techniques. [73]

In [73] M. Lelli and R. Meli pronounced this as a paradox: Size estimation is necessary when we do not have enough information (thus, early estimation methods must be used to obtain it). When we can measure with the greatest accuracy, we no longer need that information any more.

In order to research whether FPA in the early phases is a realistic option, the committee "FPA in the early phases" was established in September 1989. The committee investigates whether FPA can be used to perform an indicative size estimate before a complete logical (detailed) design is available.[41]

8.1 Terms

There are a large number of methods for function size measurement, and there are even more simplified methods. But there is no standard for the naming of simplified methods. Some key words often used include early, estimated, approximation, quick, fast, easy, predicting, simplified and so on.

8.1.1 Early measurement and the lifecycle of software development

"Early" is a relative concept. Its semantic context is the software development life cycle. In [73] M. Lelli and R. Meli described the various stages of a project life cycle and the type of estimation approach - ranging from approximate to accurate approach - applicable to the various project stages (i.e., requirements definition, analysis, design, implementation, test and setup), as described in Figure 80.

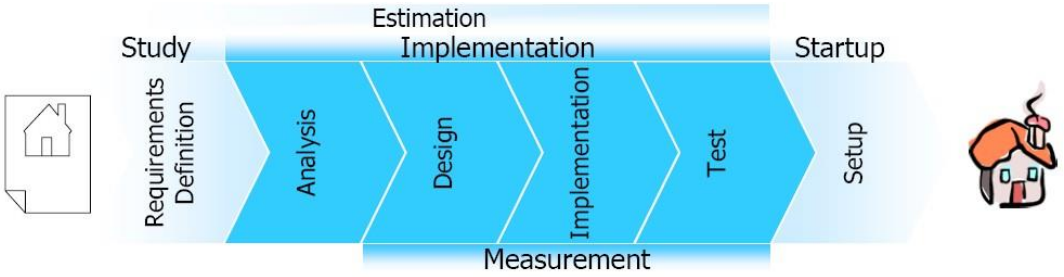

Figure 80 Approximate estimation and accurate measurement of the project life cycle

In [5] del Bianco et al. explored the relationships between the simplified measurement and the various FUR specification artefacts (such as use case diagram, class diagram, component diagram, and sequence diagram).

8.1.2 Level of accuracy, estimation, and measurement

In [110] [74] six accuracy levels for software sizing were defined and described (see Table 60). Each size estimation technique can be classified based on the following: detailed linked and flagged measure, detailed linked measure, detailed measure, default complexity measure, rough measure, and size approximation.

Table 60 Accuracy levels for software sizing and basic attributes of sizing levels

Lev.	Level Name	FSM result
1	Detailed linked and flagged measure	Most detailed
		Easily auditable
		Accurate (+/-5%)
		Very well documented
		Easily maintained
2	Detailed linked measure	More detailed
		Easily auditable
		Accurate (+/-5%)
		Very well documented
		Easily maintained
3	Detailed measure	Detailed
		Auditable
		Accurate (+/-10%)
		Well documented
		Easily maintained
4	Default complexity measure	Less detailed
		Auditable
		Reasonably accurate (+/-15%)
		Documented
		Maintained
5	Rough measure	Low detailed
		Less accurate (+/-20%)
		Documented(issues and assumptions)
		Skeleton(base for more refined measurement)
6	Size approximation	Very little detailed
		Accurate (+/-30%)
		Not documented
		Not maintained

There are a large number of software functional size estimation techniques available for levels 4 to 6. During a project, it is likely that you will start with an approximate technique close to Level 6, and move towards Level 1 as the project characteristics become better defined. In fact, measurement can be conducted to a number of "accuracy levels", based on the purpose of the measurement and desired accuracy of the result, the quality of project or application documentation available, and the time in which the measurement must be completed. It is important to choose an estimation technique based on the documentation, time available, and the measurement purpose. As the project progresses, the size estimate should be validated and refined (eventually moving from low-accuracy to high-accuracy techniques). This observation is suitable not only to select the measurement method, but also to analyze and establish the measurement method.

8.2 Methods adhering to IFPUG FA definition

In order to evaluate whether FPA in the early phase is a realistic option, the committee "FPA in the early phases" was established in September 1989. From then, many techniques for early size estimation have been proposed for FP, such as component sizing technique by Putnam and Myers [94] and the Early and Quick Function Point size estimation techniques by Conte et al. [95].

8.2.1 E&Q technique

The E&Q technique uses both analogical and analytical classification of functions; it permits the use of different levels of detail for different branches of the system. It was originally proposed in the approach of Early Functions Points (EFP) by R. Meli in 1997 for FPA. The EFP method provides a breake-down, hierarchical structure of the software functional items. In [109] L. Santillo proposed how to use the EFP in practice and evaluated the EFP through one year of actual use and more than 20 cases. This technique has proved to be very effective, providing a result within ± 10% of the actual size in most cases.

The general E&Q technique fully complies with the concepts, definitions and the structure of any functional size measurement method, as defined by ISO/IEC 14143: 1998. Thus, this technique can be extended to any Functional Size Measurement method that is found to be compliant with the ISO/IEC standard. Then, the E&Q technique has evolved and has been generalized, extending its applicability domain to the COSMIC measurement method [95]. In 2004, E&QCFFP 2.0 was proposed. The empirical evaluation of simplified estimation methods for FP indicates that some of these methods actually yield reasonably accurate estimates [6].

8.2.2 Average value

These methods – such as Estimated NESMA method [41], ISBSG average weights, simplified FP [18], prognosis of CNV AG [42] and so on - do not require the weighting of functions; instead each function is weighted with average values.

In [75], Vogelezang summarized the approximate technique and the refined approximate technique given in the COSMIC measurement manual. In the approximate technique, the average size of a functional process is multiplied with the number of functional processes the software should provide. The refined approximate technique uses the average sizes of small, medium, large and very large functional processes. The accuracy of the COSMIC-FFP approximate technique is good enough with less than 10% deviation on a project portfolio and less than 15% on a project within a specified environment [75].

Conversion between FP and COSMIC

An approach to simplified CFP measurement was obtained as a side effect of a work on convertibility between FP and CFP measures. In [72] Lavazza used the dataset published in [59] to analyze the relationships existing between FP and CFP in general, and between CFP and the non-weighted Base Functional Components of FP in particular. By means of linear OLS regression a statistically significant model was

found, which can be used to estimate the size in CFP, given the number of transactions identified via Function Point Analysis. This can be considered a sort of simplified CFP measurement method, since the identification of transaction functions is an activity much simpler and shorter than both the full fledged CFP and FP counting processes.

8.2.3 Size estimation based on a single component of FP

Some methods extrapolate the FP counts from the countable components (usually the Internal Logical Files) using statistical methods (mostly regression analysis). Some simplified methods – Mark II, NESMA's Indicative FP, Tichenor ILF Model, Prognosis by CNV AG, and ISBSG Benchmark – were constructed according to such technique.

8.2.4 Measure from models

The possibility of basing CFP measurement on UML models of user requirements has been widely studied [2], [76], [77], [78], [79], [80], [81], [82], [83], [84]. Some of the mentioned papers also proposed approaches to the automation of the measurement of UML models, and a few also prototyped such tools. All of the mentioned papers address the standard COSMIC method as described in [33], thus they consider the models that are available after the completion of the requirements elicitation and specification phase. On the contrary, hardly any works explore the relationship between the UML model process and the simplified FSM methods.

8.2.5 "Smart" technique

In [14], Santillo suggested probabilistic approaches, where the measurer can indicate the minimum, medium and maximum weight of each BFC, together with the expected probability that the weight is actually minimum, medium or maximum. This leads to estimate not only the size, but also the probability that the actual size is equal to the estimate.

8.2.6 Measurement in iterative process

Hericko and Zivkovic address size estimation in iterative development [61]. Their approach enables early size estimation using UML. However, they do not consider simplified measurement processes (hardly any work was devoted to defining simplified measurement processes for the COSMIC method). In fact, their method deals with the evolution of the functionality through iterations, rather than the level of detail that can be achieved in the requirements elicitation and specification phase, as we do.

8.3 Function Points like measures

Since the introduction of Object Oriented technologies and web technologies, a number metrics were proposed to evaluate the characteristics of object-oriented design, to estimate the development effortm and for other purposes. Among these are Use Case Points [62], Class Points (FP-like) [64], UML Points (UCP+Class Point) [85], Predictive Object Points (POPs) [86], Object-Oriented Function Points (OOFP) [87], Object Oriented Design Function Points [88], Web Points [89], Pattern Points (PP) [90], and TP method ("Transactions" and "Paths") proposed by Robiolo et al. [91]. The common characteristic of these proposals is that they, although inspired by FPs, do not

strive to adhere to the classical FP approach. So, in general the provided size measure is not given in FP.

8.4 Evaluated of the proposed methods

Meli and Santillo were among the first to recognize the need for comparing the various functional size methods proposed in the literature [92]. To this end, they also provided a benchmarking model. The E&Q technique has proved to be very effective, providing a result within ± 10% of the actual size in most cases, while the savings in measurement effort can be between 50% and 90% (depending on the aggregation level used, up to Macro Processes).[74]

In [21], van Heeringen et al. report the results of measuring 42 projects with the full-fledged, indicative and estimated NESMA methods. They found a 1.5% mean error of NESMA estimated method and a 16.5% mean error of NESMA indicative method.

Popović and Bojić compared different functional size measures –including NESMA indicative and estimated– by evaluating their accuracy in effort estimation in various phases of the development lifecycle [93]. Not surprisingly, they found that the NESMA indicative method provided the best accuracy at the beginning of the project.

Using a database of about 100 applications, NESMA did some research on the accuracy of the estimated and indicative function point counts. They got very good results (http://www.nesma.nl/section/fpa/earlyfpa.htm), although no statistics (e.g., mean relative error) are given.

8.5 Convertibility

8.5.1 Theoretical conversion within an empirical range

A first comprehensive discussion of the possible approaches to convertibility between different functional size measures is reported in [59]. Namely, the convertibility between unadjusted IFPUG function points [8] [10] and COSMIC function points [33] is considered. In [59] the impossibility of computing the conversion by means of a mathematical formula is discussed. However, the discussion in [59] makes reference exclusively to formulae of type CFP=f(FP). There is little doubt that no such formula can work, since FP and CFP are defined differently, and each of these size measures "hides" different details.

A very specific approach to convertibility concerns the cases when not only the size in FP of a given program is known, but also the number of File Type Referenced (FTR).

In the COSMIC method, a file type can be referenced to read it or to update it, and updates are typically performed with data received from outside the application, while reads are often performed to deliver some output. The data groups being moved within a functional process are similar to the concept FTP in FPA. Exploiting the knowledge that the notion of FTR is close to the notion of COSMIC data group being moved in a functional process, it is possible to compute a range in which the corresponding COSMIC size should lie.

Based on this observation, Cuadrado et al. have published a method based on a mapping of the IFPUG and COSMIC BFCs that requires knowledge only of the number of file type references made in each of the FPA transactions [70] [71]. This method associates a minimum and maximum number of data movements with each FTR. As a result, the conversion delivers an upper and lower bound for the COSMIC size corresponding to a given size in FP. However, no attempt is made to devise the actual number of data movements associated with a given FTR. This technique [70] [71] is interesting from a conceptual point of view, but in practice it is of little utility, since the confidence range is usually quite wide.

8.5.2 Statistically based conversion

The literature [59] also discusses the statistical convertibility of IFPUG FP and COSMIC FP. Statistical convertibility has been widely investigated, and numerous papers were published on that topic, e.g., [69][96][97][98][99][101][102]. It is quite noticeable that the regression models illustrated in these papers are usually of the type $CFP = a + b \times FP$ or $CFP = a \times FP^b$. Such models should not hold, according to the discussion on mathematical conversion formulae in [59]. The reason why these empirical models are statistically significant (and reasonably accurate) is probably that the considered projects have specific characteristics that make them comparable at the functional size level.

Most of the mentioned works were not unexceptionable from a statistical point of view. To improve the situation, a systematic analysis of the known datasets according to well established statistical techniques was performed and documented in [103].

Following the indications given in COSMIC [33], piecewise linear models and other types of non-linear models (including parabolic ones) have been investigated in [104]. Also in this case, statistically significant models were found for the available datasets.

The possibility of establishing a relationship between the size of transaction functions and CFP was illustrated in [97].

4) DF/FP ratio

In [97] Desharnais et al. suggested that applications featuring an anomalous value of the DF/FP ratio could be the ones that are affected by the largest relative errors of convertibility based on FP or transaction functions.

However, Desharnais et al. provided an explanation of the convertibility errors only in terms of both FPA and COSMIC BFC values; since COSMIC BFC values are not known at conversion time (otherwise there would be no reason for performing a statistical conversion) the explanations provided in [97] are of little practical utility, though conceptually interesting. With respect to previous work, Lavazza showed the increasing role of transactions over data for larger applications, both in single datasets and in the ISBSG dataset [100].

5) “Cut-off” effect

The cut–off effect is the phenomenon due to the fact that a process has a maximum size of 7 FP according to IFPUG measurement, while it has no size limit in COSMIC.

In [97], it was also observed that models based on transaction functions provide good results, although the "cut-off" effect can affect convertibility. However, Desharnais et al. did not investigate the 'cut-off' effect quantitatively. In [100] Lavazza discussed the convertibility based on the measure of transaction functions and the "cut-off" effect, as well as the role of data functions.

Lavazza derived a piecewise linear model based on TF which clearly shows the presence of the "cut-off" effect [101].

By analyzing the non weighted transaction numbers from the dataset by van Heeringen [89] Lavazza concluded that the "cut-off" effect disappears (i.e., does not affect the conversion) when unweighted data are used [101].

6) Confidence intervals
In [101], the evaluation of confidence intervals for the parameters of the FP-CFP convertibility models was introduced. In [100] confidence intervals for model parameters were evaluated systematically, thus providing an increased level of confidence on the results. For instance, it was possible to state with high confidence that the models found for the datasets available in the literature actually involve a change in slope of the model, i.e., the CFP/FP ratio is bigger for larger software applications.

In [100] it is shown that the sets of projects for which statistical convertibility holds are characterized by a small variance in the size of managed data with respect to the size of provided functions. Since we cannot be sure that all projects are characterized by a given ratio between data size and operation size, we must be very careful with adopting statistical convertibility. It must also be noted that different datasets yield different conversion formulae, thus it is quite difficult to generalize the results of such approach.

8.5.3 Manual conversion

A last type of convertibility discussed in [59] is called "manual". In this type of conversion, only the basic raw data of the Function Point counting are available, the rest of the required data must be provided by people who have the necessary knowledge of the software being measured, so that good judgments or intelligent guesses on the equivalence between the base functional components (BFC) of the two methods can be made (ISO/IEC 14143-1 defines the BFC as "*an elementary unit of Functional User Requirement defined by and used by an FSM Method for measurement purpose*" [106]).

8.5.4 Unified Model based conversion

Demirors and Gencel suggested the creation of a Unified Model (UM) of the information upon which functional size is measured [46]. The UM should contain all the information needed by FPA, COSMIC and possibly other FSM methods, like MKII Function Points [107] [108]. The approach by Demirors and Gencel [46] is made possible by the fact that different FSM methods are based on a common set of basic elementary concepts (data group, data item, process, etc.). Therefore, a model that represents this information at a quite low level is able to support the identification of method-specific BFC.

8.5.5 Conversion method using analytical criteria

In Chapter 6 we proposed a conversion method that uses analytical criteria. Our proposal has some points in common with [46]. However, there are a few fundamental differences between the two proposals.

According to Demirors and Gencel, their model "*has certain restrictions. For one thing, it doesn't handle the detailed rules each method suggests (for example, the IFPUG method specifies, 'don't count code tables,' whereas the other methods don't have this restriction).*" We do not have this type of problems, because we do not establish precise mapping rules; instead, we just indicate the most probable correspondence between FPA BFC and COSMIC BFC, then we leave the user free to exploit the correspondence to directly create a COSMIC BCF out of a FPA BFC, or not (in the latter case, it is the user that has the responsibility of defining a different correspondence).

A second relevant difference is that we provide a tool for helping the user perform the conversions (or the double counting).

Finally, we note that there are several software tools that aim at supporting function point or COSMIC counting; among these are *Scope*, *NH's Function Point Analyzer*, *Function Point Modeler*, *Function Point Workbench*, *SFERA* and many others. However –as far as we know– none of the available tools supports conversion.

Chapter 9 Conclusion

9.1 Summary of results

Functional size measurement plays an important role in the development effort estimation, project management, and quality control of software development. In fact, Function Point Analysis is widely used, especially to quantify the size of applications in the early stages of development, when effort estimates are needed. However, the measurement process is often too long or too expensive, or it requires more knowledge than available when development effort estimates are due. To overcome these problems, simplified methods are proposed to measure Function Points.

In Chapter 3, we presented the main simplified methods and classified them into four categories, namely, E&QFP, Average complexity (weight) values, single component based, and Smart approximation techniques. Then we compared the simplified methods with respect to factor(s) used, factor granularity, factor capture difficulty, factor value and measurement process difficulty.

Such simplified methods were used for sizing both “traditional” and Real-Time applications, for the purpose of evaluating the accuracy of the sizing with respect to full-fledged Function Point Analysis.

9.1.1 Model-based FSM

It has been shown that functional size measures can be derived from UML models of requirements. The process is easy if UML models are built in a measurement-oriented way, i.e., highlighting the information required for FSM. Based on this idea, Lavazza et al. proposed Model-based Measurement-Oriented method [3]. Throughout this work, we used measurement-oriented models to support FSM.

9.1.2 Evaluation of simplified FSM (FPA)

Functional Size Measurement methods are widely used but have two major shortcomings: they require a complete and detailed knowledge of user requirements, and they involve relatively expensive and lengthy processes. So many simplified methods emerged. We assessed several simplified methods in Section 5.1 to answer questions like the following: "What is the accuracy of simplified FSM methods?” and “Which simplified method is the best one for my application(s)?”

We collected 18 applications’ models and obtained the “correct” values using the standard method (FPA) at first. Then we measured the applications using simplified methods, including those proposed by NESMA, the Early&Quick Function Points, the ISBSG average weights, and others: the resulting size measures were then compared.

It was found that all the methods that use predefined weights for all the transaction and data types identified in Function Point Analysis yielded similar results, characterized by acceptable accuracy. On the contrary, methods that rely on just one of the elements that

contribute to functional size tend to be quite inaccurate. In general, different methods show different accuracy for Real-Time and non Real-Time applications.

We also derived simplified size models on the basis of the measures from the dataset used for experimentations, and yielded results that are similar to those obtained via the methods proposed in the literature.

Therefore, it was clear that model-based simplified method is feasible, and its accuracy is also reliable.

9.1.3 Model-based simplified COSMIC measurement

Also the COSMIC method requires a complete and detailed knowledge of user requirements. The COSMIC measurement process involves: Identification of functional processes; Identification of data groups; Identification of Data Groups used in Each Functional Process. Identify the data group movements involved in each functional Process. Therefore, measurement-oriented model building involving the following diagrams: Use case diagram, class diagram, component diagram, and sequence diagrams.

Since software requirements can be effectively described by means of UML models, which grow in detail and completeness through the requirements analysis phase, it is reasonable to expect that progressively more accurate measures can be derived from these UML models. This is particularly useful when COSMIC measures have to be obtained earlier than assumed by the official counting manual, because of reasons such as a tight deadline, or not sufficiently detailed requirements specifications.

Therefore, we formulated the following research questions: (1) During the requirements elicitation and specification phase, is it possible to write progressively more complete and detailed UML models that support progressively more accurate simplified CFP measurement methods? (2) What is the accuracy of the estimates provided by different simplified CFP measurement methods? (3) Do simplified CFP measurement methods provide a level of accuracy that is proportional to the amount of information required?

To explore the above mentioned issues, we modeled a set of 23 software applications and measured them, with the goal of obtaining the measures needed to support simplified measurement methods.

Our analysis shows that it is possible to write progressively more detailed and complete UML models of user requirements that provide the data required by the simplified COSMIC methods, which in turn yield progressively more accurate measures of the modeled software. Initial measures are based on simple models and are obtained quickly and with little effort. As models grow in completeness and detail, the measures increase their accuracy.

Developers that use UML for requirements modeling can obtain an early estimation of the application size at the beginning of the development process, when only a very simple UML model has been built for the application, and can obtain increasingly more accurate size estimates while the knowledge of the product increases and UML models are refined accordingly.

9.1.4 FSM vs. OO measures

It has been shown that functional size measure can be derived from UML models of requirements. In particular, if UML models are measurement-oriented, it is easy to identify BFC and all those elements that contribute to size measures. However, the analysis of UML diagrams to identify BFC and the elements that have to be taken into consideration to compute functional size measures is still a manual process. On the contrary, it is very easy to automatically derive object-oriented measures from UML models via tools like *SDMetrics*. So, an association (and the corresponding quantitative model) between the object-oriented measures of a measurement-oriented UML model and the functional size measures derived from the same UML model, would make it possible to estimate the functional size based on the (automatically obtained) OO measures, or use the OO measures in place of the functional size measures, which would be no longer needed.

We took the requirement specifications of a set of 11 software applications, and we built the UML measurement-oriented models of requirements; then we measured the functional size of UML models and measured UML models using *SDMetrics*; finally, we analyzed the relationship between functional size and OO measures. Statistical analysis showed that both FPA and COSMIC functional size measures appear correlated to object-oriented measures. In particular, associations with basic OO measures were found: FP appear associated with the number of classes, the number of attributes and the number of methods; CFP appears associated with the number of attributes. This result suggests that even a very basic UML model, like a class diagram, can support size measures that appear equivalent to functional size measures (which are much harder to obtain).

The results obtained tend to confirm that, having modeled an application's functional user requirements using UML and highlighting the typical elements of software models used by FSM methods (i.e., elementary/functional processes, data files/groups, etc.), the measures obtained automatically by means of measurement tools like *SDmetrics* are essentially equivalent to those obtained by certified functional size measurers. The small cost, rapidity, accuracy and repeatability of UML measurement suggest that manually performed FSM methods could be abandoned. However, these results need further experimental evidence, before we can safely suggest practitioners to drop FSM methods.

9.1.5 Conversion between FPA and COSMIC

The introduction of the COSMIC method as an alternative of Function Point Analysis originated the problem of converting Function Point measures into other units. To this end, several methods – ranging from statistical analysis to "manual" conversion– have been used. However, none of the proposed conversion methods guarantees the necessary accuracy.

We defined a seamless and cheap procedure that allows measurers to derive functional size measures expressed in COSMIC Function Points from size measures expressed in Function Points, and viceversa.

To get accurate conversions, we exploit all the available information provided by the measurement process, that is, not only the size in Function Points, but also the details of basic functional components. To make the procedure efficient, a mapping of Function Point Analysis concepts onto COSMIC concepts was defined.

A conversion procedure based on the aforementioned mapping was proposed. Such procedure is supported by a software tool that eases the conversion process. The usage of both the procedure and tool was tested via an example of realistic complexity.

The proposed procedure and tool can be effectively used to perform the required measurement with very good accuracy. The tool can also be used to perform a sort of "double" measurement, i.e., both Function Points and COSMIC Function Points are measured at the same time.

9.2 Guidelines for developers

The work has led to an increase in knowledge on how to simplify the methods for measuring the functional size of the software, referring to the functional user requirements. The presented analyses can be replicated by other researchers, to increase the reliability and generality of the results.

The results of the work done that are most relevant in practice concern the identification and evaluation of possible FSM processes. In fact, this knowledge is immediately usable by developers.

The possible FSM processes that have been identified are:
- 1) Standard methods. Standard measurement manuals (either IFPUG or COSMIC) are applied.
- 2) Simplified methods. See Chapter 3 .
- 3) Model-based measurement using measurement-oriented models. See Chapter 4 .
- 4) Simplified methods applied to FUR modeled as measurement-oriented models. See Section 5.1.
- 5) Object-oriented measurement applied to measurement-oriented models. See Chapter 7 .
- 6) Object-oriented measurement applied to object-oriented models. See Chapter 7 .

Table 61 schematically describes the characteristics of the modeling phases involved in each type of FSM process.

Table 61 FSM processes: the modelling phase

Ord.	Method	Model Type	Analyst competence
1	Standard methods	Traditional	N/A
2	Simplified	Traditional	N/A
3	MbMO	Measurement-Oriented	Analysts understand FSM and the MbMO
4	Simplified MbMo	Measurement-Oriented	Analysts understand FSM and the MbMO
5	MbMO-OO	Measurement-Oriented	Analysts understand FSM and the MbMO
6	OO	OO model	OO Analysis

Table 62 schematically describes the characteristics of the measurement phases involved in each type of FSM process.

Table 62 FSM processes: the measurement phase

Ord.	Method	Measurer	Factors	Granularity	Result
1	Standard methods	Certified measurer	BFC	Small	FP/CFP
2	Simplified	Anybody who knows basics of FSM and the simplified model	Subsets and/or generalizations of BFC	Large	FP
3	MbMO	Anybody who knows basics of FSM and UML	BFC	Small	FP/CFP
4	Simplified MbMO	Anybody who knows basics of FSM, the simplified model and UML	Subsets and/or generalizations of BFC	Large	FP/CFP
5	MbMO-OO	SDMetrics(Tool)	elements of OO	Large (low accuracy)	FP/CFP
6	OO	SDMetrics(Tool)	elements of OO	idem	FP/CFP

Table 63 gives the main properties of the FSM processes, in terms of cost and results. Here it is important to note that:

- The accuracy of standard processes depends on measurers. The process is manual, so it is affected by errors and subjective interpretations.
- In Model-based measurement, the measurement phase is much less error-prone and hardly affected by subjectivity. On the contrary, the model can affect the measurement.
- The cost of providing measurement-oriented models depends on how FUR is written. If FUR is already written using UML, making the models measurement-oriented is easy; if FUR is written using a mix of E/R diagrams, data flow diagrams, tables, text, formulas, etc., the modeling phase can be quite expensive and long.
- When simplified methods are used, the cost and accuracy depend on how much the process is simplified. In general, the more information is modeled, the higher the cost and the higher the accuracy.

Table 63 FSM process properties

Ord.	Methods	Modeling cost	Measurement cost	Standard measure(FP / CFP)	Accuracy
1	Standard methods	N/A	High	Yes	Depends on measurer
2	Simplified	N/A	Low-Medium	Yes	Error ~ 10%

3	MbMO	Low if FUR are written in UML	Low	Yes	Depends on model
4	Simplified MbMO	Low if FUR are written in UML	Low	Yes	10% or more (depends on simplification)
5	MbMO - OO	Low/Short if FUR are written in UML	Null	Yes	N/A
6	OO	Null (if FUR are written in UML)	Null	Yes	Error > 10%

By considering the tables reported above, practitioner can choose the FSM process that most suites their needs.

9.3 Future research directions

The work described in this thesis can be continued in the following directions:

As already mentioned in the work on conversion between FP and CFP (see Chapter 6), the tool supports the mappings among FPA and COSMIC concepts (described in Table 51, but ultimately it is the user who has to choose if a given FPA element actually corresponds to a COSMIC element or not). A smarter support from the tool, involving less work by the user could be achieved by providing the tool with expert reasoning capabilities.

Simplified measurement models should be better derived via regression analysis, especially if multiple independent variables are involved. Unfortunately, our evaluation of simplified FSM methods (Section 5.1) and the analysis of FSM vs. OO measure (Chapter 7), were based on relatively small datasets. In order to increase the reliability and to guarantee the general validity of results, the dataset should be extended to include data points from additional applications.

As already mentioned, most results are based on relatively small applications: further work for verifying the accuracy of simplified measurement methods when dealing with larger project is needed.

Bibliography

[1] L. Lavazza, and G. Liu, "An Empirical Evaluation of Simplified Function Point Measurement Processes", Int. Journal on Advances in Software, vol 6, no 1&2, 2013.

[2] L. Lavazza, and V. del Bianco, "A Case Study in COSMIC Functional Size Measurement: the Rice Cooker Revisited", IWSM/Mensura 2009, 4 - 6 November 2009, Amsterdam.

[3] L. Lavazza, V. del Bianco, and C. Garavaglia, "Model-based Functional Size Measurement", ESEM 2008, 2nd International Symposium on Empirical Software Engineering and Measurement, Incorporating ISESE and Metrics, Kaiserslautern, Germany. October 9-10, 2008.

[4] L. Lavazza, S. Morasca, and G. Robiolo, "Towards a Simplified Definition of Function Points", Information and Software Technology, Volume 55, Issue 10, October 2013, Pages 1796–1809.

[5] V. del Bianco, L. Lavazza, G. Liu, S. Morasca, and A.Z. Abualkishik , "Model-based Simplified Functional Size Measurement – an Experimental Evaluation with COSMIC Function Points", In Proceedings of the 3rd International Workshop on Experiences and Empirical Studies in Software Modeling co-located with 16th International Conference on Model Driven Engineering Languages and Systems (MoDELS 2013) Miami, USA, October 1, 2013.

[6] L. Lavazza, and G. Liu, "A Report on Using Simplified Function Point Measurement Processes", The 7th Int. Conf. on Software Engineering Advances - ICSEA 2012, November 18-23, 2012 - Lisbon, Portugal.

[7] L. Lavazza, V. del Bianco, and G. Liu, "Analytical Convertibility of Functional Size Measures: a Tool-based Approach", The Joint Conf. of the 22nd Int. Workshop on Software Measurement (IWSM) and the 7th Int. Conf. on Software Process and Product Measurement (Mensura) IWSM-MENSURA 2012, October 17-19, 2012, Assisi.

[8] A.J. Albrecht, "Measuring Application Development Productivity", Joint SHARE/ GUIDE/IBM Application Development Symposium, 1979.

[9] A.J. Albrecht, and J.E. Gaffney, "Software function, lines of code and development effort prediction: a software science validation", IEEE Transactions on Software Engineering, vol. 9, 1983.

[10] International Function Point Users Group, "Function Point Counting Practices Manual - Release 4.3.1", 2010.

[11] ISO/IEC 20926: 2003, "Software engineering – IFPUG 4.1 Unadjusted functional size measurement method – Counting Practices Manual", ISO, Geneva, 2003.

[12] C. Jones, "A new business model for function point metrics", http://www.itmpi.org/assets/base/images/itmpi/privaterooms/capersjones/FunctPtBusModel2008.pdf, 2008.

[13] Total Metrics, "Methods for Software Sizing – How to Decide which Method to Use", www.totalmetrics.com/function-point-resources/downloads/R185_Why-use-Function-Points.pdf, August 2007.

[14] L. Santillo, "Easy Function Points – 'Smart' Approximation Technique for the IFPUG and COSMIC Methods", Joint Conf. of the 22nd Int. Workshop on Software Measurement and the 7th Int. Conf. on Software Process and Product Measurement, Oct. 2012.

[15] J. Geraci, and C. Tichenor, “The IRS Development and Application of the Internal Logical File Model to Estimate Function Point Counts”, IFPUG Fall Conference of Use (ESCOM-ENCRESS 1998), May 1998.
[16] “Early & Quick Function Points for IFPUG methods v. 3.1 Reference Manual 1.1”, April 2012.
[17] ISO/IEC 24570: 2004, “Software Engineering-NESMA Functional Size Measurement Method version 2.1 - Definitions and Counting Guidelines for the Application of Function Point Analysis”, International Organization for Standardization, Geneva, 2004.
[18] L. Bernstein, and C. M. Yuhas, “Trustworthy Systems Through Quantitative Software Engineering”, John Wiley & Sons, 2005.
[19] International Software Benchmarking Standards Group, “Worldwide Software Development: The Benchmark, release 11”, 2009.
[20] R. Meli, and L. Santillo, “Function point estimation methods: a comparative overview”, Software Measurement European Forum (FESMA 1999), Oct. 1999.
[21] H. van Heeringen, E. van Gorp, and T. Prins, “Functional size measurement - Accuracy versus costs - Is it really worth it?”, Software Measurement European Forum (SMEF 2009), May 2009.
[22] C. Jones, “A Short History of The Lines of Code (LOC) metric”, http://namcookanalytics.com/wp-content/uploads/2013/07/LinesofCode2013.pdf, 2013.
[23] G. Grau, and X. Franch, “Using the PRiM method to Evaluate Requirements Model with COSMIC-FFP”, In: Proceedings of the IWSM-MENSURA 2007, Mallorca, pp. 110–120 (2007)
[24] C. Dekkers, and M. Aguiar, “Applying Function Point Analysis to Requirements Completeness”, Pacific northwest software quality conference, Oregon (Portland), October 16-17, 2001.
[25] S. Furey, “Why we should use function points”, Software, IEEE (Volume: 14, Issue: 2), Mar./Apr. 1997.
[26] M. Jenner, “Automation of Counting of Functional Size Using COSMIC-FFP in UML”, 12th International Workshop on Software Measurement (IWSM 2002), Magdeburg (Germany), October 2002.
[27] A. Khelifi, and A. Abran, “Design Steps for developing Software Measurement Standard Etalons for ISO 19761 (COSMIC-FFP)”, Proceedings of the 11th WSEAS International Conference on COMPUTERS, Agios Nikolaos, Crete Island, Greece, July 26-28, 2007.
[28] C. Jones, “Software Measurement Programs and Industry Leadership”, http://www.crosstalkonline.org/storage/issue-archives/2001/200102/200102-Jones.pdf, 2001.
[29] F. Gramantieri, E. Lamma, P. Mello, and F. Riguzzi, “A system for measuring function points from specification,” Technical Report, Universitra di Bologna, 1997.
[30] E. Lamma, P. Mello, and F. Riguzzi, “A system for measuring function points from an ER-DFD specification,” The Computer Journal, vol. 47, no.3, pp. 358-372, 2004.
[31] C. Tichenor, “A New Software Metric to Complement Function Points: The Software Non-functional Assessment Process (SNAP)”, http://m.crosstalkonline.org/issues/8/66/, 2013

[32] B. Kitchenham, and K. Kansala, “Inter-item correlation among function points”, Proceedings of First International Software Metrics Symposium, Baltimore, USA, 1993.
[33] COSMIC – Common Software Measurement International Consortium, The COSMIC Functional Size Measurement Method - Version 3.0.1 Measurement Manual (The COSMIC Implementation Guide for ISO/IEC 19761: 2003), May 2009.
[34] C. Tichenor, “The IRS Development and Application of the Internal Logical File Model to Estimate Function Point Counts”, IFPUG Fall Conference of Use (ESCOM-ENCRESS 1998), May 1998.
[35] ISO, ISO/IEC 19761:2011 “Software engineering -- COSMIC: a functional size measurement method”, International Organization for Standardization, Geneva, April 2011.
[36] ISO, ISO/IEC 20968:2002, “Software engineering – Mk II Function Point Analysis – Counting Practices Manual”, International Organization for Standardization, Geneva, 2002.
[37] A. J. Albercht, “AD/M Productivity Measurement and Estimate Validation”, CIS & A Guidelines 313, IBM Corporate Information Systems and Administration, November 1, 1984.
[38] R. Dumke, and A. Abran, “COSMIC Function Points: Theory and Advanced Practices”, Germany: CRC Press, Taylor and Francis Group, Auerbach Publications, 2011.
[39] COSMIC – Common Software Measurement International Consortium, The COSMIC Functional Size Measurement Method Version 3.0: Guideline for Sizing Business Application Software, Version 1.1, May 2008.
[40] F. Vogelezang, “COSMIC full function points the next generation of functional sizing”, In: Software Measurement European Forum—SMEF 2005. 2005.
[41] NESMA, “The Application Of Function Point Analysis In The Early Phases Of The Application Life Cycle - A Practical Manual: Theory And Case Study”, V. 2.0, http://www.nesma.nl/download/boeken_NESMA/N20_FPA_in_Early_Phases_(v2.0).pdf
[42] M. Bundschuh, “Function Point Prognosis Revisited”, FESMA 99, Amsterdam, The Netherlands, October 4-8, 1999, pp. 287 – 297. http://www.academia.edu/1024603/FUNCTION_POINT_PROGNOSIS_REVISITED
[43] M. Bundschuh, “Function Point Approximation with the Five Logical Components”, http://www.academia.edu/1556168/FUNCTION_POINT_APPROXIMATION_WITH_THE_FIVE_LOGICAL_COMPONENTS, 2000.
[44] R. Asensio, F. Sanchis, F. Torre, V. Garcia, and G. Uria, “A preliminary study for the development of an early method for the measurement in function points of a software product”, ACM classes: D.4.8 [On Line], 2004 Available at: http://arxiv.org/abs/cs?papernum=0402015
[45] O. Demirors, and C. Gencel, “A Comparison of Size Estimation Techniques Applied Early in the Life Cycle”, European Software Process Improvement Conference (EurSPI 2004), Springer Verlag Springer Lecture Notes in Computer Science (LNCS), Vol.3281, (2004), 184.
[46] O. Demirors, and C. Gencel, “Conceptual Association of Functional Size Measurement Methods”, IEEE Software vol. 26 no. 3, May/June 2009, pp. 71-78.

[47] K.R. Jayakumar, **"**Why you must change to COSMIC for sizing and estimation", 2010, http://www.cosmicon.com/portal/public/Why_you_must_change_to_COSMIC_15Dec2010.pdf

[48] J.P. Jacquet, and A. Abran, "From software metrics to software measurement methods: A process model", In: Proceedings of the 3rd Int. Standard Symposium and Forum on Software Engineering Standards (ISESS'97). Walnut Creek, USA (1997).

[49] T. Fetcke, A. Abran, and R.A. Dumke, "Generalized Representation for Selected Functional Size Measurement Methods," Current Trends in Software Measurement, R.A. Dumke and A. Abran, eds., Shaker, 2001, pp. 1–25.

[50] S. Purao, and V. Vaishnavi, "Product metrics for object-oriented systems", ACM Computing Surveys (CSUR), v.35 n.2, p.191-221, June 2003 [doi>10.1145/857076.857090]

[51] S. Abrahao, P. Poels, and O. Pastor, "A Functional Size Measurement Method for Object-Oriented Conceptual Schemas: Design and Evaluation Issues", Software & System Modeling 5(1), 48–71 (2006).

[52] OMG – Object Management Group, Unified Modeling Language Superstructure, Version 2.4.1, August 2011.

[53] J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, and W. Lorensen, "Object-Oriented Modeling and Design", Prentice Hall, 1990.

[54] http://www.sdmetrics.com/

[55] T. Fetcke, "The warehouse software portfolio: a case study in functional size measurement", Technical report no. 1999-20, Département d'informatique, Université du Quebec à Montréal, Canada, 1999.

[56] L. Lavazza, and C. Garavaglia, "Using Function Points to Measure and Estimate Real-Time and Embedded Software: Experiences and Guidelines", 3rd Int. Symp. on Empirical Software Engineering and Measurement (ESEM 2009), Oct. 2009.

[57] L. Lavazza, and C. Garavaglia, "Using Function Point in the Estimation of Real-Time Software: an Experience", Software Measurement European Forum (SMEF 2008), May 2008.

[58] V. del Bianco, L. Lavazza, and S. Morasca, "A Proposal for Simplified Model-Based Cost Estimation Models", 13th Int. Conf. on Product-Focused Software Development and Process Improvement (PROFES 2012), June 2012.

[59] COSMIC, "The COSMIC Functional Size Measurement Method - Version 3.0 - Advanced and Related Topics", December 2007.

[60] B. Kitchenham, L. Pickard, S. MacDonell, and M. Shepperd, "What accuracy statistics really measure [software estimation]", Software, IEEE Proceedings, Vol. 148, IET, 2001, pp. 81–85.

[61] M. Hericko, and A. Zivkovic, "The size and effort estimates in iterative development", Information and Software Technology vol. 50 n. 7, 2008, pp. 772-781.

[62] G. Karner, "Metrics for Objectory", Diploma thesis, University of Linköping, Sweden. No. LiTH-IDA-Ex-9344:21. December 1993.

[63] B. Anda, D. Dreiem, D.I.K. Sjøberg, and M. Jørgensen, "Estimating software development effort based on use cases - Experiences from industry", In M. Gogolla, C. Kobryn (Eds.): UML 2001 - The Unified Modeling Language. Modeling Languages, Concepts, and Tools, 4th Int'l Conference. Springer-Verlag LNCS 2185, 487-502.

[64] G. Costagliola, F. Ferruci, G. Torttora, and G. Vitiello, "Class Point: An Approach for the Size Estimation of Object-Oriented Systems", IEEE Transactions on Software Engineering, Vol. 31, No. 1, pp. 52-74, Jan. 2005.
[65] J. Capers, "A new business model for function point metrics", Version 10.0, August, 2009, http://www.jfpug.gr.jp/information/Function%20Point%20Business%20Model%20CAPERS%20JONE%202009.pdf
[66] P. Morris, "Cosmic-FFP and IFPUG 4.1: Similarities and Differences", Presentation at 2003 IFPUG Fall Conference, Scottsdale, Arizona. http://www.totalmetrics.com/function-point-resources/downloads/IFPUG-COSMIC-and-IFPUG---Similarities-and-Differences_2009-.pdf
[67] A.J. Albrecht, "Measuring Application Development Productivity", Joint SHARE/ GUIDE/IBM Application Development Symposium, 1979.
[68] International Software Benchmarking Standards Group: Worldwide Software Development: The Benchmark, release 11, 2009.
[69] L. Lavazza, "Convertibility of Functional Size Measurements: New Insights and Methodological Issues", PROMISE 09 (PRedictOr Models In Software Engineering), Vancouver, May 18-19, 2009.
[70] J.J. Cuadrado-Gallego, F. Machado-Piriz, and J. Aroba-Páez, "On the conversion between IFPUG and COSMIC software functional size units: A theoretical and empirical study", Journal of Systems and Software, vol. 81, n. 5, Elsevier, 2008.
[71] J.J. Cuadrado-Gallego, D. Rodríguez, F. Machado, and A. Abran, "Convertibility between IFPUG and COSMIC functional size measurements", 8th Int. Conf. on Product-Focused Software Process Improvement, PROFES 2007, Riga, Latvia, July 2-4, 2007, Springer LNCS 4589, 2007.
[72] L. Lavazza, "An Evaluation of the Confidence in the Statistical Convertibility of Function Points into COSMIC Function Points", Empirical Software Engineering, 2013, 1-36.
[73] M. Lelli, and R. Meli, "From narrative user requirements to Function Point", IN: Proceedings of Software Measurement European Forum-SMEF 2005, Mar. 16-18, 2005, Rome, Italy.
[74] P. Hill, "Software early lifecycle- Function sizing", SoftwareTech, June 2006, Vol. 9, No.2.
[75] F.W. Vogelezang, "COSMIC Full Function Points, the Next Generation", in Measure! Knowledge! Action! – The NESMA anniversary book, NESMA, 2004.
[76] S. Azzouz, and A. Abran, "A proposed measurement role in the Rational Unified Process (RUP) and its implementation with ISO 19761: COSMIC-FFP", Presented in Software Measurement European Forum - SMEF 2004, Rome, Italy (2004).
[77] V. Bévo, G. Lévesque, and A. Abran, "Application de la méthode FFP à partir d'une spécification selon la notation UML: compte rendues premiers essais d'application et questions", Presented at International Workshop on Software Measurement (IWSM 1999), Lac Supérieur, Canada, September 8-10 (1999).
[78] M.S. Jenner, "COSMIC-FFP and UML: Estimation of the Size of a System Specified in UML – Problems of Granularity", Presented in the 4th European Conference on Software Measurement and ICT Control, Heidelberg, pp. 173–184 (2001).
[79] V. Luckson, G. Lévesque, "Une méthode efficace pour l'extraction des instances de nconcepts dans une spécification UML aux fins de mesure de la taille fonctionnelle de logiciels", In: The Seventeenth International Conference Software

& Systems Engineering & their Applications, ICSSEA 2004, Paris, Novembre 30-Décember 2 (2004).

[80] P. Habela, E. Glowacki, T. Serafinski, and K. Subieta, "Adapting Use Case Model for COSMIC-FFP Based Measurement", In: 15th International Workshop on Software Measurement – IWSM 2005, Montréal, pp. 195–207 (2005).

[81] K.G. Van den Berg, T. Dekkers, and R. Oudshoorn, "Functional Size Measurement applied to UML-based user requirements", In: Proceedings of the 2nd Software Measurement European Forum (SMEF 2005), Rome, Italy, March 16-18, pp. 69–80 (2005).

[82] G. Levesque, V. Bévo, and D.T. Cao, "Estimating software size with UML models", In: Proceedings of the 2008 C3S2E Conference, Montreal, pp. 81–87 (2008).

[83] A. Sellami, and H. Ben-Abdallah, "Functional Size of Use Case Diagrams: A Fine-Grain Measurement", In: Fourth International Conference on Software Engineering Advances, ICSEA 2009, pp. 282–288 (2009).

[84] S. Barkallah, A. Gherbi, and A. Abran, "COSMIC Functional Size Measurement Using UML Models", In proceeding of: Software Engineering, Business Continuity, and Education - International Conferences ASEA, DRBC and EL, pp. 137-146, 2011.

[85] S.E. Kim, W. Lively, and D. Simmons, "An effort estimation by UML points in the early stage of software development," in Proceedings of the International Conference on Software Engineering Research and Practice & Conference on Programming Languages and Compilers, SERP 2006, Las Vegas, Nevada, USA, Volume 1, June 26-29, 2006.

[86] A. Minkiewicz, "Measuring object-oriented software with the predictive object points," in Proceeding of 8th European Software Control and Metrics Conference, Atlanta, 1997.

[87] G. Caldiera, C. Lokan, G. Antoniol, R. Fiutem, S. Curtis, G. La Commare, and E. Mambella, "Estimating size and effort for object oriented systems", In Proceeding of 4th Australian Conference on Software Metrics, 1997.

[88] D.J. Ram, and S.V.G.K. Raju, "Object oriented design function points", In Proceedings of First Asia-Pacific Conference on Quality Software, October 30-31, 2000, pp121-126.

[89] D. Reifer, "Web-Development: "Estimating quick-time-to-market software", IEEE software, vol. 17, no. 8, pp.57-64, November/December 2000.

[90] O. Adekile, D.B. Simmons, and W.M. Lively, "Object oriented software development effort prediction using design patterns from object interaction analysis.", In Proceeding of 2010 Fifth International Conference on Systems, Menuires, April, 2010, pp.47-53.

[91] G. Robiolo, C. Badano, and R. Orosco, "Transactions and Paths: two use case based metrics which improve the early effort estimation", In Proceedings of the 2009 3rd International Symposium on Empirical Software Engineering and Measurement (ESEM 09), IEEE Computer Society, Washington, DC, USE, pp.422-435.

[92] R. Meli, and L. Santillo, "Function point estimation methods: a comparative overview", Software Measurement European Forum (FESMA 1999), Oct. 1999.

[93] J. Popović, and D. Bojić, "A Comparative Evaluation of Effort Estimation Methods in the Software Life Cycle", Computer Science and Information Systems, vol. 9, Jan. 2012.

[94] L.H. Putnam, and W. Myers, "Measures for excellence: reliable software on time within budget", Prentice Hall, UpperSaddle River, 1992.
[95] M. Conte, T. Iorio, R. Meli, and L. Santillo, "E&Q: An Early&Quick approach to function size measurement methods", In: Proceedings of Software Measurement European Forum-SMEF 2004, January 28-30, 2004, Rome, Italy.
[96] H. van Heeringen, "Changing from FPA to COSMIC - A transition framework", Software Measurement European Forum-SMEF 2007 (Rome, May 9-11, 2007).
[97] J.M. Desharnais, A. Abran, and J.J. Cuadrado-Gallego, "Convertibility of Function Points to COSMIC: Identification and analysis of functional outliers", Int. Conference on Software Process and Product Measurement – MENSURA 2006, (Madrid 2006).
[98] A. Abran, J.M. Desharnais, and F. Aziz, "Measurement Convertibility: From Function Points to COSMIC-FFP" 15th International Workshop on Software Measurement – IWSM 2005, Montréal (Canada), Shaker Verlag, Sept. 12-14, 2005, pp. 227-240.
[99] F. Vogelezang, and A. Lesterhuis, "Applicability of COSMIC Full Function Points in an administrative environment - Experiences of an early adopter", 13th International Workshop on Software Measurement – IWSM2003, Shaker Verlag, Montréal, 2003.
[100] L. Lavazza, "An evaluation of the statistical convertibility of Function Points into COSMIC Function Points", Empirical Software Engineering Journal, DOI 10.1007/s10664-013-9246-zz, Springer, March 2013.
[101] J.J. Cuadrado-Gallego, L. Buglione, M. Domínguez-Alda, M. Sevilla, Antonio Gutierrez de Mesa J, O. Demirors, O, "An experimental study on the conversion between IFPUG and COSMIC functional size measurement units", Information and Software Technology 52(3):347–357, Elsevier, 2010.
[102] V. Ho, A. Abran, and T. Fetcke, "A comparative study case of COSMIC-FFP, full function point and IFPUG methods", Department of Informatics, University of Quebec at Montreal, Canada, 1999.
[103] L. Lavazza, "Convertibility of functional size measurements: new insights and methodological issues", In Proceedings of the 5th international conference on predictor models in software engineering, May 18-19, 2009, Vancouver, British Columbia, Canada [doi>10.1145/1540438.1540451]
[104] L. Lavazza, "A systematic approach to the analysis of function point-COSMIC convertibility", In: 20th International Workshop on Software Measurement. ICSM/Mensura, Stuttgart, 2010.
[105] C. Gencel, "Do we really need to choose one functional size measurement method?", In: UKSMA/COSMIC international conference on software metrics and estimating, London, 27–28 October, 2011.
[106] ISO/IEC 14143-1: Information Technology—Software Measurement—Functional Size Measurement, Part 1: Definition of Concepts, Int. Org. for Standardization/Int Electrotechnical Commission, 1998.
[107] UKSMK-United Kingdom Software Metrics Association, "Mk II Function Point Analysis Counting Practices Manual", v. 1.3.1, September 1998.
[108] C.R. Symons, "Function point analysis: difficulties and improvements", IEEE Transactions on Software Engineering, 14(1), 1988.
[109] L. Santillo, and R. Meli, "Early Function Points: some practical experiences of use", ESCOM-ENCRESS 98, May 27-29, 1998, Rome.

[110] Total Metrics, “Levels of Function Points, Version 1.3”, January 2004, http://www.totalmetrics.com/total-metrics-articles/levels-of-function-point-counting, Total Metrics, 2004.
[111] S.R. Chidamber, and C. F. Kemerer, “Towards a metrics suite for object-oriented design”, in Proc. 6th OOPSLA Conference, ACM 1991, pp. 197-211.
[112] SDMetrics User Manual (V2.32), July 3, 2013, http://www.sdmetrics.com/down/SDMetricsManual.pdf
[113] ISO&IEC 29881:2010 “Information technology- Systems and software engineering- FiSMA 1.1 functional size measurement method”.