Using Functional Complexity Measures in Software Development Effort Estimation

Luigi Lavazza
Dipartimento di Scienze Teoriche e Applicate
Università degli Studi dell'Insubria
Varese, Italy
luigi.lavazza@uninsubria.it

Gabriela Robiolo
Departamento de Informática
Universidad Austral
Buenos Aires, Argentina
grobiolo@austral.edu.ar

Abstract — **Several definitions of measures that aim at representing the size of software requirements are currently available. These measures have gained a quite relevant role, since they are one of the few types of objective measures upon which effort estimation can be based. However, traditional Functional Size Measures do not take into account the amount and complexity of elaboration required, concentrating instead on the amount of data accessed or moved. This is a problem since the amount and complexity of the required data elaboration affect the implementation effort, but are not adequately represented by the current size measures, including the standardized ones. Recently, a few approaches to measuring aspects of user requirements that are supposed to be related with functional complexity and/or data elaboration have been proposed by researchers. In this paper, we take into consideration some of these proposed measures and compare them with respect to their ability to predict the development effort, especially when used in combination with measures of functional size. A few methods for estimating software development effort –both based on model building and on analogy– are experimented with, using different types of functional size and elaboration complexity measures. All the most significant models obtained were based on a notion of computation density that is based on the number of computation flows in functional processes. When using estimation by analogy, considering functional complexity in the selection of analogue projects improved accuracy in all the evaluated cases. In conclusion, it appears that functional complexity is a factor that affects development effort; accordingly, whatever method is used for effort estimation, it is advisable to take functional complexity into due consideration.**

Keywords – Functional size measurement; Function Points; COSMIC function points; effort estimation; functional complexity measurement.

I. INTRODUCTION

Several definitions of measures intended to represent the functional size of software are being used. The popularity of these measures is due to the fact that functional size measures are typically used to drive the estimation of the development effort. To this end, effort models require several inputs in addition to the functional size, including the complexity of the software to be developed [11][47]. In fact, problem complexity is recognized as one of the elements that contribute to the comprehensive notion of software size [17].

The need to account for software complexity when estimating the development effort does not depend on the functional size measurement (FSM) method used.

Before proceeding, it is useful to spend some words on the fact that throughout the paper we treat the terms "complexity" and "amount of data elaboration" as synonyms. This is due to the facts that complexity is an inherently elusive concept, and at the functional requirements level it is not clear what should be the difference between the amount and the complexity of data elaboration: for instance, in many cases, complexity is considered proportional to the number of alternatives in a process execution, but this number is also clearly related also to the size of the process.

When dealing with effort estimation, the most popular methods require an evaluation of the complexity of the application. Currently such evaluation is of a purely qualitative nature. For instance, COCOMO II [11] provides a table that allows the user to evaluate complexity on an ordinal scale (from "very low" to "extra high") according to five aspects (control operations, computational operations, device-dependent operations, data management operations, user interface management operations) that have to be evaluated in a qualitative and subjective way: e.g., the characterization of computational operations corresponding to the "Nominal" complexity is "Use of standard math and statistical routines. Basic matrix/vector operations" [15].

It is quite clear that it would be greatly beneficial to replace such subjective and approximate assessment of complexity with a real measure, based on objective and quantitative evaluations, since this would enable the construction of more reliable and accurate models of effort.

Previous work showed that effort models that take into consideration complexity measures are more precise than those based on the functional size only. In particular, the authors of this paper showed that development effort correlates well with COSMIC function points (CFP) [16] and Path [43], and that the inclusion of a Path-based complexity measure improves the models based on size, whatever size measure is used (IFPUG Function Points [24], CFP [23], or even Use Case Points [28]) [34].

In [1], the work reported in [34] was extended, by taking into consideration some measures that represent potential complexity dimensions, by building effort estimation models that exploit these measures, and by discussing the precision of fit of these models.

In this paper, we further enhance the work reported in [1] by using an extended dataset, and by refining it (the largest project was removed, being an evident outlier). More important, here we test the importance of functional complexity measures in effort estimation, by experimenting with a wider range of estimation methods. In particular, we use not only model-based estimation, but also Estimation by Analogy (EbA), as this is a very popular technique: model-based estimation and EbA are definitely the most relevant techniques for cost estimation [25].

The results of the measurements and analyses reported in the paper contribute to enhancing the knowledge of how to measure functional complexity at the requirements level, and what is the contribution of such measure to effort estimation.

The paper is organized as follows: Section II accounts for related work; Section III is dedicated to illustrating the measures of functional size and functional complexity used in this study; Section IV describes the dataset and the types of analysis performed; Section V and VI illustrate the results of the analyses via regression and analogy, respectively; in Section VII the outcomes of the research are discussed; in Section VIII the threats to the validity of the study are discussed. Finally, Section IX draws some conclusions and outlines future work.

II. Related Work

A few attempts to account for data elaboration in FSM have been done. Feature points by Capers Jones [26] aim at capturing the algorithmic complexity of the elaboration. However, according to Capers Jones, "the feature point metric was created to deal with the psychological problem that members of the real-time and systems software world viewed function point metrics as being suitable only for management information systems" [27]. Therefore, feature points simply moved part of the 'size' from data to algorithms, leaving the measure substantially unaltered with respect to FPA. In fact, currently Capers Jones recommends "the use of the standard IFPUG methodology combined with a rating of 'Project Complexity' to properly scale effort".

3D Function Points [50] consider three dimensions of the application to be measured: Data, Function, and Control. The Function measurement considers the complexity of algorithms; and the Control portion measures the number of major state transitions within the application.

Gencel and Demirors [19] point out that we still need a new Base Functional Component (BFC) Types for the Boolean operations of Functional User Requirements, which are often not considered to be algorithmic operations, but which are related to complexity. This point of view highlights the necessity of considering the complexity of elaboration required in FSM, and they suggested introducing as a new BFC type that differs from authors' proposal.

Bernárdez et al. [10] measured the cyclomatic complexity of a use case in order to validate the use case definition, while Levesque et al. [35] measured the conditions of inputs in a sequential diagram in order to add the concept of complexity to the COSMIC method.

Yavari et al. [51] evaluated the weak points of Use Case complexity measures, in particular, those of transaction identification, and introduced other measures to determine Use Case complexity. They focused on Use Case specification and flow of events. Also, the authors considered main and alternative scenarios. However, this is only an early definition of the new measures, as they did not use them in a case study.

Aggarwal et al. [4] defined an estimation model that can be used to estimate the effort required for designing and developing hypermedia content management systems (CMS). The model is designed to help project manager to estimate effort at the very early stage of requirement analysis. Questionnaires are used to estimate the complexity of the project. The final effort is estimated using the project size and various adjustment factors. The size of the project is evaluated by using a modified object point analysis approach. The proposed model shows a great improvement as compared to the earlier models used in effort estimation of CMS projects.

Visaggio [48] proposes a metric for expressing the entropy of a software system and for assessing the quality of its organization from the perspective of impact analysis. The metric is called "structural information" and is based on a model dependency descriptor. The metric is characterized by its independence from the techniques used to build the system and the architectural styles used to represent the system at the various levels of abstraction. The metric is sensitive to and reflects both internal and external complexity, but is independent of and ignores intrinsic complexity, which is our interest focus.

Briand and Wust [14] used structural design properties of an object-oriented development project, such as coupling, cohesion, and complexity (of late design) as additional cost factors. They empirically conclude that the measures of such properties did not play a significant role in improving system effort predictions.

Mendes et al. [38] compared length, functionality and complexity metrics as effort predictors by generating corresponding prediction models and comparing their accuracy using boxplots of the residuals for web applications. Their results suggest that in general the various considered measures provide similar prediction accuracy.

Baresi and Morasca [8] analyzed the impact of attributes like the size and complexity of W2000 (a special-purpose design notation for the design of Web applications [7]) design artifacts on the total effort needed to design web applications. They identified for Information, Navigation, and Presentation models a set of size and complexity metrics. The complexity metrics are based on associations and links identified in the models. The three studies performed correlated different size measures with the actual effort: no general hypotheses could be supported by the analyses that were conducted, probably because the designer's background impacted the perception of complexity.

Lind and Heldal [36] conducted four experiments in the automotive industry, which showed a strong correlation between COSMIC functional size measures and implemented code size in Bytes of real-time applications. They reported that it was possible to obtain accurate Code Size estimates even for software components containing complex calculations –which are not captured by COSMIC– as the factors affecting the relationship are functionality type, quality constraints, development methods and tools, and information regarding hardware interfaces missing in the requirement specification.

Bashir and Thomson [9] used traditional regression analysis to derive two types of parametric models: a single variable model based on product complexity and a multivariable model based on product complexity and requirements severity. Generally, the models performed well according to a number of accuracy tests. In particular, product complexity explained more than 80% of variation in estimating effort. They concluded that product complexity as an indicator for project size is the dominant parameter in estimating design effort. Our results agree with those by Bashir and Thomson, as the results they obtained using functional complexity measures ($0.64<R^2<0.81$) are quite similar to ours.

Quite interestingly, in the parametric models that are most used in practice –like COCOMO II [11] or SEER/SEM [18]– the functional complexity is taken into account as part of the product characteristics in formulas of the type Effort=f(Size, <product characteristics>, <process characteristics>).

Hastings and Sajeev [21] proposed a Vector Size Measure (VSM) that incorporates both functionality and problem complexity in a balanced and orthogonal manner. VSM is used as the input to a Vector Prediction Model (VPM), which can be used to estimate development effort early in the software life cycle. The results indicate that the proposed technique allows for estimating the development effort early in the software life cycle with errors not greater than 20% across a range of application types.

AlSharif et al. [5] introduced a measure for assessing the overall complexity of software architecture. To accomplish this, they chose to use the Full Function Points (FFP) methodology –a former version of COSMIC Function Points– as a building block to measure complexity. The new measure was inspired by the fact that, in general, the components of an architecture comprise collections of services (functionality) that each component provides for other components. The allocation of these functionalities affects the required interface (external dependency) and the internal work performed by each component. Therefore, measuring the functionality of the components can serve as an indicator of the internal and external complexity of the components and, consequently, the complexity of the architecture. Also, Sengupta et al. [44] proposed the Component Architecture Complexity Measurement Metrics (CACMM), based on Component Architecture Graph (CAG), a graphical model used for representing a UML component diagram. An analysis of the graph was performed to measure complexity at different levels – the individual component level, the component-to-component level and the overall architecture. However, neither in [5] nor in [44] the relationship between architecture complexity and effort was analyzed.

Misra [40] proposed a modified cognitive complexity measure (MCCM), which is a modification of the Cognitive Information Complexity Measure (CICM). In the cognitive functional size measure, the functional size depends upon the internal architecture of the software and its inputs and outputs. For the new measure, the occurrence of operators and operands is taken into account, instead of the number of inputs and outputs. The author compared the values obtained by calculating the complexity of eight C programs; however, a relation with Effort was not reported.

Wijayasiriwardhane and Lai [49] described a Function Point-like measure named Component Point (CP), which was used to measure the system-level size of a Component-Based Software System (CBSS), specified in the Unified Modeling Language. In the CP counting process, the complexity of the component was assessed, which depended not only on the number, but also on the complexity of its interfaces and interactions. The complexity level of each interface was specified using the Number of Operations (NO) and the Number of Parameters (NP), which were derived from the operation signatures for each interface. They provided an empirical analysis of seven projects in order to verify the validity and usefulness of the CP measure with regard to its correlation to the effort of component-based development. They reported that the R^2 obtained was greater than 0.9.

Our results are in accordance with the consideration expressed by Morasca on the definition of measures [42], as it appears that the notion of complexity may be represented by taking into account several basic indicators (size, control flow, data, etc.) that can be used individually (i.e., without the need to build a derived measure defined as a weighted sum) in estimation models.

Mittas and Angelis [41] introduced the use of a semi-parametric model that managed to incorporate some parametric information into a non-parametric model, combining in this way regression and analogy. They demonstrated the procedure used to build such a model from two well-known datasets. The MMRE reported for EbA were 35.57% and 33.45% and the improvement using the combination model was about 50%. The results using EbA fell within the range of our results, but the improvement obtained was higher. However, the method proposed by Mittas and Angelis has some limits in practical applicability, because the models are more difficult to build, as more variables and several estimation techniques have to be used.

Shepperd and Schofield [45] described an approach to estimation based upon the use of analogies. The underlying principle was to characterize projects in terms of features (for example, the number of interfaces, the development method or the size of the functional requirements document). Similarity was defined as the Euclidean distance in an n-dimensional space, where n is the number of project features. Each dimension is standardized, so all dimensions have equal weight. The known effort values of the closest neighbors to the new project are then used as the basis for prediction. The method was validated on nine different industrial datasets (a total of 275 projects) and in all cases analogy outperformed algorithmic models based upon stepwise regression. Although we had a different research objective, it was useful to see that the results they obtained were in a range of values similar to ours: the MMRE of analogy based method of homogeneous data set were in the 26%-60% range.

Finally, Gupta et al. [20] studied analogy based estimation methods and compared estimation results reported by nine authors using Fuzzy logic, Grey System Theory, Machine Learning techniques such as Genetic Algorithms, Support vector Machines. The reported results are characterized by quite large variations: MMRE varies from a minimum 12% to a maximum 111%, while Pred(25) is in the 15%–83.75% range. Our results fall in these ranges of values.

III. MEASURES INVOLVED IN THE STUDY

In this study, we used the size measures and functional complexity measures described in the following sections.

A. *Function Points*

The Function Point method was originally introduced by Albrecht to measure the size of a data-processing system from the end-user's point of view, with the goal of estimating the development effort [2][3]. IFPUG FPA is now an ISO standard [24] in its "unadjusted" version. So, throughout the paper, unless otherwise explicitly stated, we refer exclusively to Unadjusted Function Points, which are generally referred to as "UFP."

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. Both data and transactions are evaluated at the conceptual level, i.e., they represent data and operations that are relevant to the user. Therefore, Function Points (FP) are counted on the basis of the user requirements specification. The boundary indicates the border between the application being measured and the external applications and user domain.

The core of the counting procedure consists in identifying and weighting so-called data function types and transactional function types. Data functions represent data that are relevant to the user and are required to perform some function. Data functions (DF) are classified into internal logical files (ILF), and external interface files (EIF). An ILF is a user identifiable group of logically related information maintained (i.e., managed –in FPA terminology, "maintaining" data means creating, modifying, deleting data–) within the boundary of the application. 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 a read-only file.

Transactional functions represent operations that are relevant to the user and cause input and/or output data to cross the application boundary. Transactional 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 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.

Every function, either data or transaction, contributes a number of FP that depends on its weight. The weight of ILF and EIF is evaluated based on Data Element Types (DET) and Record Element Types (RET). A DET is a unique, non-repeated field recognizable by the user. A RET is a subgroup of the information units contained in a file. To give a rough idea of what RET and DET are, if the specifications are written in an object-oriented language (like UML), the concept of RET maps (with some exceptions) onto the concept of class, while DET are the class attributes.

For transactions, the weight is based on the number of DET and File Type Referenced (FTR). An FTR can be an ILF referenced or maintained by the transaction or an EIF read by the transaction. The DET considered are those that cross the application boundary when the transaction is performed.

Each function is weighted according to given tables (weight tables can be found here: http://www.eng-it.it/qg-ifpug-fpa-v42en.pdf).

Finally, the number of so-called Unadjusted Function Points (UFP) is obtained by summing the contribution of the function types UFP = EI + EO + EQ + ILF + EIF.

According to the definition of UFP, it is clear that the amount and complexity of the elaboration required in the transaction functions is not taken into consideration. For instance two External Output transactions that involve 2 FTR and 10 DET both have the same weight, even if one just performs sums and the other performs very complex operations according to a very sophisticated algorithm.

B. *COSMIC Function Points*

COSMIC (Common Software Measurement International Consortium) [16][23] function points are growingly used for measuring the functional size of applications, i.e., to measure the size of functional user requirements.

COSMIC measurement is applied to the "Functional User Requirements" of a software application (actually, there is no difference between the user requirements used to count FP and CFP). The result is a number representing the functional size of the application in COSMIC Function Points.

In the COSMIC model of software (illustrated in Fig. 1), the Functional User Requirements can be mapped into unique functional processes, initiated by functional users. Each functional process consists of sub-processes that involve data movements. A data movement concerns a single data group, i.e., a unique set of data attributes that describe a single object of interest. In practice, the COSMIC data groups correspond to FPA logical data files (or to RET), but do not contribute directly to the size in CFP: they are relevant only because they are the objects of data movements. There are four types of data movements:

- An Entry moves a data group into the software from a functional user.
- An Exit moves a data group out of the software to a functional user.
- A Read moves a data group from persistent storage to the software.
- A Write moves a data group from the software to persistent storage.

In the COSMIC approach, the term "persistent storage" denotes data (including variables stored in central memory) whose value is preserved between two activations of a functional process.

The size in CFP is given by equation CFP = Entries + Exits + Reads + Writes, where each term in the formula denotes the number of corresponding data movements. So, there is no concept of "weighting" of a data movement in COSMIC, or, equivalently, all data movement have the same unit weight.

COSMIC function points do not represent the amount and complexity of data elaboration required. COSMIC function points concentrate on the measure of data movements, neglecting data elaboration. More precisely, the model of software used by the COSMIC method –illustrated in Fig. 1– includes data elaboration, but no indication on how to measure it is provided. The COSMIC measurement manual [16] simply assumes that every data movement accounts for some amount of data elaboration, and that such amount is proportional to the number of data movements, so that by measuring data movements, one measures also data manipulation.

As size units, we adopted both CFP and the number of functional processes. In fact, the number of functional processes is suggested as a reasonable approximation of the size in CFP in [16]. Moreover, being a sort of "by product" of CFP measurement, computing the number of Functional Processes does not actually require additional effort.

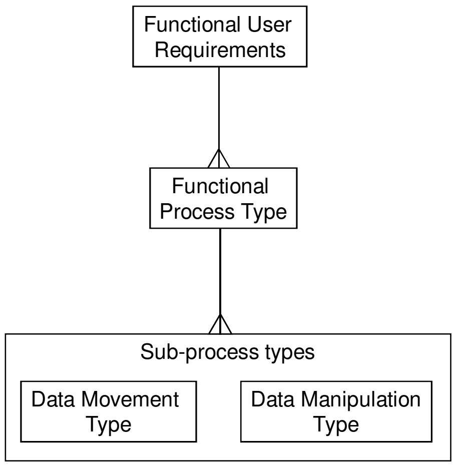

Figure 1. The COSMIC generic software model.

C. *Use Case-based Measures*

In 1993, Karner introduced Use Case Points (UCP), a redefinition of the Function Point method in the context of the use case requirements specification [28][29].

Use cases yield an observable result that is meaningful for the actors [12]. Karner based the conception of use case functional size on two elements: the use cases themselves and the actors, which are the external entities that interact with the system.

Every element, i.e., every use case and actor, contributes to the software product size with a number of UCP that depends on the "complexity" of each use case and actor. The complexity of use cases is defined in terms of the number of transactions and analysis objects (which are conceptually similar to ILF). The complexity of actors is associated to the characteristics of the interface and protocol used in the interaction with the system. Each element is weighted on the basis of its complexity according to the values specified in [28][29].

Since the concept of transaction is not clearly defined in Karners's method –as already pointed out by [51]– we decided to interpret it as the stimulus triggered by an actor: each stimulus that an actor triggers defines a transaction, as described in [32]. Also, the analysis objects were not taken into account to measure the complexity of the use cases; only the number of transactions was used. This was done in order to make our results comparable to those obtained by [6].

We consider UCP in their "unadjusted" version. So, throughout the paper, unless otherwise explicitly stated, we will exclusively refer to Unadjusted Use Case Points, which are generally referred to as "UUCP."

Finally, it is important to note that the number of UUCP, i.e., the "amount of functionality" of a use case, is obtained by adding up the contribution of the elements: UUCP = Use Case + Actor.

D. *Functional Complexity Measures*

Several different possible measures of functional complexity were proposed. For instance, in [46] the number of inputs and outputs, the number of decision nodes, the sum of predicates of all decision nodes, the depth of decision tree and the length of paths are considered as possible indicators of complexity.

In [47], Tran Cao et al. propose the usage of the number of data groups (NOD), the number of conditions (NOC) and entropy of system (EOS). They also study how these measures (also in combination with COSMIC FP) are correlated with the development effort.

Another measure of complexity, the Paths, was defined on the basis of the information typically available from use case descriptions [43]. The measure of the complexity of use cases is based on the application of the principles of McCabe's complexity measure [37] to the descriptions of use cases in terms of scenarios. In fact, use cases are usually described giving a main scenario, which accounts for the 'usual' behaviour of the user and system, and a set of alternative scenarios, which account for all the possible deviations from the normal behaviour that have to be supported by the system. Robiolo and Orosco [43] apply to the use case textual descriptions the same measure applied by McCabe to code. Every different path in a given use case scenario contributes to the measure of the use case's complexity. The definition of Paths conforms to several concepts enounced by Briand et al. [13]: Paths represent "an intrinsic attribute of an object and not its perceived psychological complexity as perceived by an external observer", and they represent complexity as "a system property that depends on the relationship between elements and is not an isolated element's property". A detailed description of the Paths measure and its applicability to use cases described in UML can be found in [33].

In the research work reported here, we used measures that are conceptually very close to those proposed in previous studies [46][47]. However, we did not stick exactly to the previous proposals, essentially for practical reasons. We used Paths instead of NOC because both measures capture essentially the same meaning, and the measures of Paths were already available. Similarly, we used the number of COSMIC data groups and the FPA number of logic data files instead of NOD because –having measured the size of the applications in FP and CFP, the documentation on data groups and logic files was already available, thus the measurement could be performed very easily.

TABLE I. THE DATASET

Project ID	Type	Actual Effort	Path	Use Cases	UUCP	UFP	FPA transactions	CFP	Functional Processes	Data Groups	Pers. DG
P1	Academic	410	71	39	201	185	39	143	39	21	7
P2	Academic	474	73	28	149	269	58	118	28	15	9
P3	Academic	382	60	7	84	171	19	109	24	15	12
P4	Academic	285	49	6	72	113	15	74	25	14	8
P5	Academic	328	34	12	72	110	14	48	12	17	7
P6	Academic	198	35	8	62	86	9	67	10	15	7
P7	Academic	442	50	6	71	75	10	81	16	12	6
P8	Industrial	723	97	27	175	214	33	115	27	19	10
P9	Industrial	392	83	15	111	340	47	105	24	35	24
P10	Industrial	272	42	19	119	179	27	73	21	9	9
P11	Industrial	131	18	13	68	115	17	51	13	5	5
P12	Industrial	348	32	12	71	107	16	46	12	13	7
P13	Academic	243	68	12	99	111	12	96	26	18	9
P14	Academic	300	33	4	57	40	4	54	12	12	4
P15	Academic	147	20	10	53	59	10	53	14	15	4
P16	Academic	169	17	5	28	61	5	30	5	10	6
P17	Academic	121	21	13	52	72	13	47	13	15	5
P18	Academic	342	24	9	48	11	27	40	9	12	2
P19	Academic	268	16	9	49	12	27	30	9	10	3

IV. THE EXPERIMENTAL EVALUATION

A. *The Dataset*

In order to evaluate the measures mentioned above with respect to their usability as effort predictors, we collected all such measures for a set of projects. We could not use data from the best known repositories –such as the PROMISE or ISBSG datasets– because they do not report the size of each project according to different FSM methods; moreover, the Paths measure is quite recent, and no historical data exist for it.

We measured 19 small business projects, which were developed in three different contexts: an advanced undergraduate academic environment at Austral University, the System and Technology (S&T) Department at Austral University and a CMM level 4 Company. The involved human resources shared a similar profile: advanced undergraduate students, who had been similarly trained, worked both at the S&T Department and at the CMM level 4 Company. All the selected projects met the following requisites:

a) Use cases describing requirements were available.
b) All projects were new developments.
c) The use cases had been completely implemented, and the actual development effort in PersonHours was known.

The dataset is reported in Table I. Note that we distinguished the number of persistent data groups (column *Pers. DG*) from the total number of data groups, which includes also transient data groups. Our hypothesis is that persistent data groups are more representative of the amount of data being handled by the application.

B. The Estimation Methods Used

We first checked if statistically significant models could be built using ordinary least squares (OLS) linear regression.

We used also linear regression after log-log transformation, as is usually done in studies concerning effort [11][47].

In model building, a 0.05 statistical significance threshold was used throughout the paper, as is customary in Empirical Software Engineering studies. All the results reported in the paper are characterized by p-value < 0.05. All the validity requirements for the proposed models (e.g., the normal distribution of residuals of OLS regressions) were duly checked.

Moreover, in order to avoid overfitting, we retained only models based on datasets containing –after the elimination of outliers– at least seven data points for each independent variables. In this way we get datasets containing enough data points to support statistically significant analyses.

Then, we applied Estimation by Analogy (EbA). That is, we estimated the development effort required by each project on the basis of the actual development effort required by similar projects (i.e., projects having similar size and/or complexity characteristics).

C. The Measures

The measures actually employed in the analysis are described in Table II. They are a superset of those used in [1].

Among the measures listed in Table II, we have not only size and complexity measures, but also several density measures. These density measures introduce the concept of functional complexity per size unit: for instance, we consider the Path/UFP ratio, which indicates how many Paths per UFP are required in the software being measured. The reason for using functional complexity per size unit is that functional complexity measures are often correlated to size itself, thus the density appears more relevant to indicate how complex and difficult the development is.

The complexity of a system is a property that depends on the relationships among system's elements [13]. So, the measures discussed above represent the density of relationships among elements per unit size.

V. Using Functional Complexity Measures in Effort Estimation Models

In this section, we report about the construction of development effort models. We systematically tried to build models that have the development effort as dependent variable and one or two independent variables belonging to the set described in Table II. In particular, when building models having two independent variables, all the possible pairs of measures from Table II were tried out. Models with more than two independent variables were not sought, because the available dataset does not contain enough data points to support such analysis while avoiding data overfitting.

TABLE II. Measures Used

Name	Description
Path	Path
CFP	COSMIC Function Points
FPr	Functional Processes
DG	Data Groups
PDG	Persistent Data Groups
UC	Number of Use Cases
UUCP	Unadjusted Use Case Points
UFP	Unadjusted Function Points
FPAtrans	Number of unweighted FPA transactions
NumFiles	Number of unweighted FPA logic data files
Path/FPr	Path per Functional Process
Path/CFP	Path per CFP
DG/FPr	Data Groups per Functional Process
DG/CFP	Data Groups per CFP
PDG/FPr	Persistent Data Groups per Functional Process
PDG/CFP	Persistent Data Groups per CFP
Path/UFP	Paths per unadjusted Function Point
Path/FPAtrans	Paths per unweighted FPA transactions
NumFiles/UFP	Unweighted FPA logic data files per unadjusted Function Point
NumFiles/FPAtrans	Unweighted FPA logic data files per unweighted FPA transactions
Path/UC	Paths per use case
Path/UUCP	Paths per use case point

A. Analysis of the dataset using linear regression

Linear regression did not provide any statistically significant model when FPA or UC measures were used. On the contrary, when using COSMIC-based measures, a single statistically significant model was found, having equation

$$\text{Effort} = -29.9 + 139.1 \times \text{Path/FPr}$$

The model –obtained after eliminating 4 outliers (P1, P6, P8, P17)– has adjusted R^2=0.64. The regression line is illustrated in Fig. 2. This model is quite interesting, as a functional size density measure is the independent variable: the model seems to suggest that the actual complexity of software, rather than its size, determines development effort.

The accuracy of the model is characterized by MMRE = 34%, while Pred(25) –i.e., the percentage of project whose absolute relative estimation error is in the ±25% range– is 63%, and Error range = -46%–162%. The distribution of relative residuals is illustrated in Fig. 3.

Throughout the paper, MMRE and Pred(25) are used as indicators of the accuracy of models, because they are often quoted as the *de facto* current accuracy indicators used in

Empirical Software Engineering, even though criticisms have been cast on the usefulness and meaning of MMRE and Pred(25) [31]. Boxplots representing the distributions of relative residuals are always given, to provide meaningful and unbiased information on estimation accuracy.

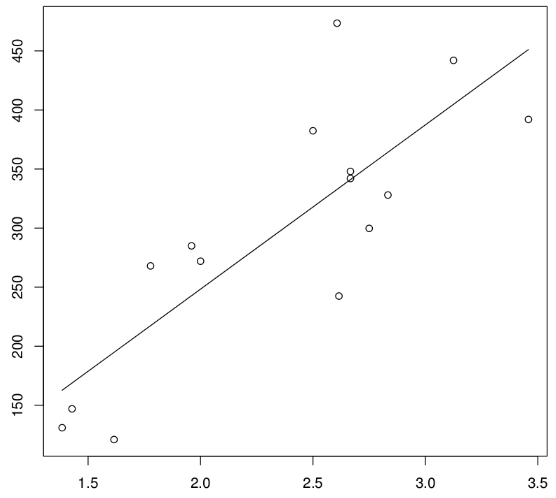

Figure 2. Effort vs. Path/FPr: OLS regression line.

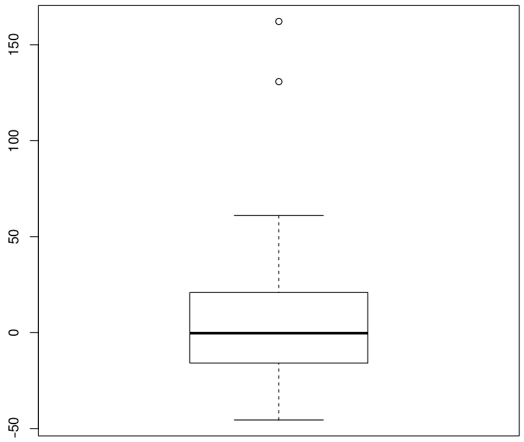

Figure 3. Effort vs. Path/FPr OLS model: relative residuals.

B. *Analysis of the dataset using log-log transformations*

We used linear regression after log-log transformation, as is usually done in studies concerning effort [11][47].

The models found by checking the correlation between effort and FPA measures are summarized in Table III, while their accuracy is illustrated in Table IV. Both the models found involve a size measure (the number of FPA transactions) and a complexity density measure (Paths/UFP or Path/FPA transactions).

TABLE III. OLS EFFORT MODELS INVOLVING FPA-BASED MEASURES

Model	Adj. R^2	Outl.
Effort = 120.6 × $FPAtrans^{0.654}$ × $(Path/UFP)^{0.958}$	0.642	3 (P21, P22, P17)
Effort = 12.2 × $FPAtrans^{0.816}$ × $(Path/FPAtrans)^{0.976}$	0.812	3 (P22, P21, P14)

TABLE IV. ACCURACY OF OLS FPA-BASED EFFORT MODELS

Model	MMRE	Pred(25)	Error range
Effort = 120.6 × $FPAtrans^{0.654}$ × $(Path/UFP)^{0.958}$	71%	53%	-40%−543%
Effort = 12.2 × $FPAtrans^{0.816}$ × $(Path/FPAtrans)^{0.976}$	25%	63%	-60%−108%

The models found by checking the correlation between effort and COSMIC measures are summarized in Table V. It is interesting to note that both the models found involve the usage of a size measure (the Function Points or the number of functional processes) and a complexity density measure (Paths/CFP). The accuracy of the models found is illustrated in Table VI.

TABLE V. OLS EFFORT MODELS INVOLVING COSMIC-BASED MEASURES

Model	Adj. R^2	Outliers
Effort = 112.2 × $CFP^{0.391}$ × $(Path/CFP)^{1.298}$	0.658	0
Effort = 231.8 × $FPr^{0.377}$ × $(Path/CFP)^{1.468}$	0.744	1 (P14)

TABLE VI. ACCURACY OF OLS COSMIC-BASED EFFORT MODELS

Model	MMRE	Pred(25)	Error range
Effort = 112.2 × $CFP^{0.391}$ × $(Path/CFP)^{1.298}$	22%	68%	-30%−77%
Effort = 231.8 × $FPr^{0.377}$ × $(Path/CFP)^{1.468}$	23%	68%	-27%−97%

Finally, we checked the correlation between effort and Use Case. The models found are summarized in Table VII. It is noticeable that both the models found involve the usage of a size measure (either the number of use cases or the use case points) and a complexity density measure.

TABLE VII. OLS EFFORT MODELS INVOLVING USE CASE-BASED MEASURES

Model	Adj. R^2	Outliers
Effort = 21.9 × $UC^{0.68}$ × $(Path/UC)^{0.728}$	0.609	1 (P22)
Effort = 24.9 × $UUCP^{0.679}$ × $(Path/UUCP)^{0.775}$	0.608	1 (P22)

The accuracy of the models found is illustrated in Table VIII. Again, we got significant models only based on variables involving Paths.

It is quite interesting to observe that all the models found have two independent variables, and that one is a measure of size, while the other is a measure of complexity density.

TABLE VIII. ACCURACY OF OLS USE CASE-BASED EFFORT MODELS

Model	MMRE	Pred(25)	Error range
Effort = 21.9 × $UC^{0.68}$ × $(Path/UC)^{0.728}$	24%	63%	-45%−73%
Effort = 24.9 × $UUCP^{0.679}$ × $(Path/UUCP)^{0.775}$	24%	63%	-45%−74%

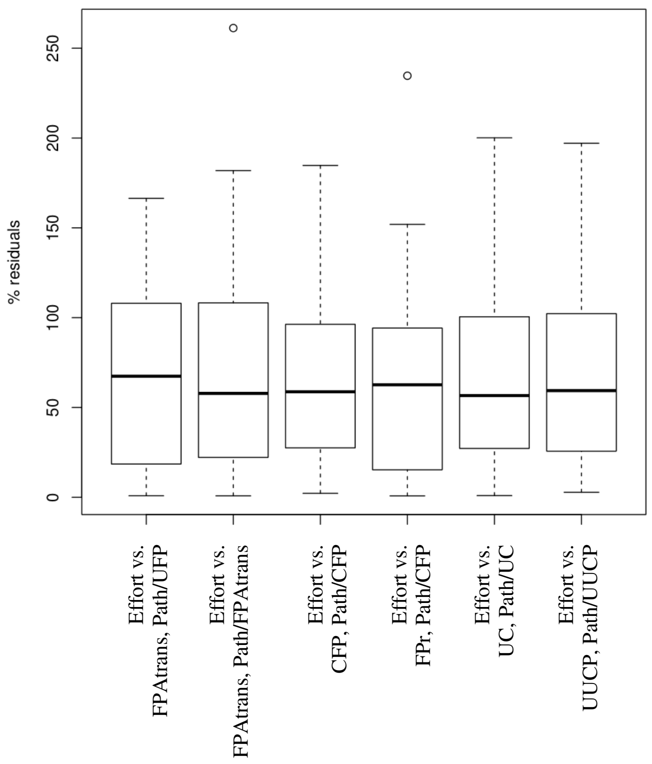

Figure 4. Model comparison: relative residuals.

It is also interesting to see that these models appear reasonably good both in terms of their ability to explain the variation of effort depending on the variation of the size and complexity measures (as indicated by the values of the adjusted R^2) and in terms of precision of the fit (as indicated by MMRE, Pred(25) and the relative error range).

The data illustrated in the tables above do not indicate if a model is definitely better than the others with respect to accuracy. A possible way for identifying the best model is by comparing the relative residuals (since we are considering the ability to predict effort, we have to look at *relative* residuals, since an error of, say, two PersonMonths can be irrelevant or very important, depending on the total effort).

The boxplots representing relative residuals of the models obtained via OLS regression after log-log transformation are reported in Fig. 4 (where two extreme outliers of the first model were omitted to preserve the readability of the figure). The boxplots indicate quite clearly that the values and distributions of residuals are very similar for all the models.

VI. Using Functional Complexity Measures in Estimation by Analogy

In this section, we test the effects of considering complexity density when selecting "analogous" projects upon which effort estimation is based. To this end, we compute effort estimates in two ways: identifying analogous projects only on the basis of size, and considering both size and complexity density. The hypothesis that we want to test is that considering both size and complexity density leads to better estimates.

There are several criteria that can be used to identify analogous projects. Some of these involve identifying the *k* closest projects, where closeness is generally evaluated as the distance between projects. Accordingly, we chose to use a criterion that ensures –as far as possible– that the analogous projects are all within a given distance from the project to be estimated.

For each project, we select among the remaining projects those having both size and complexity density in a ±20% range. The mean of these projects' efforts is assumed as the estimate. If no project is found in the ±20% range, the range is progressively increased until at least one project in the range is found. The estimation for a given project of size S is thus performed according to the following procedure:

```
R=0.2
P = {}
while |P|<1 do
    for all Pi in the set of projects
        if (1-R)×S ≤ Pi.size ≤ (1+R)×S
            then P = P ∪ {Pi.effort}
    R=R+0.1
done
EstimatedEffort = mean(P)
```

So, for instance, given P3 (whose size is 171 UFP), P1 and P10 (which have size 185 UFP and 179 UFP, respectively) are selected as analogue projects. In fact, these are the only projects that have size in the range 171 ±20%, i.e., in range 137–205. Since the developments of the selected projects required 410 and 272 PersonHours, we assume that P3 requires (410+272)/2 = 341 PersonHours.

When analogue projects are selected also on the basis of complexity density, the condition for inclusion into the set of analogous projects is modified in order to select projects whose size S_i and complexity density C_i satisfy the condition $((1\text{-}R)\times S \leq S_i \leq (1+R)\times S) \wedge ((1\text{-}R)\times C \leq C_i \leq (1+R)\times C)$, where C is the complexity density of the project being estimated.

A. *Estimation by Analogy using FPA Measures*

In this subsection, we present the results of estimation by analogy based on FPA measures: we used UFP as a size measure and Path/UFP as complexity density measure.

UFP were chosen because they are the most obvious size measures, when FPA is used. Similarly, we used CFP and UUCP in the following sections.

Effort estimates and the differences with respect to actual efforts are illustrated in Table IX, where columns labeled "CA est. (2 var)" report results concerning the estimation based on both size and complexity density.

It is easy to see that the estimation error is generally larger for the estimates based only on size similarity, than for the estimates based on the similarity of both size and complexity density.

TABLE IX. ANALOGY-BASED ESTIMATES OBTAINED USING FPA MEASURES

PID	Actual Effort	CA est.	CA est. (2 var)	Error CA		Error CA (2 var)	
1	410	459	553	49	(12%)	143	(35%)
2	474	557	392	84	(18%)	-82	(-17%)
3	382	341	410	-41	(-11%)	28	(7%)
4	285	262	263	-23	(-8%)	-22	(-8%)
5	328	252	348	-76	(-23%)	20	(6%)
6	198	282	266	84	(42%)	68	(34%)
7	442	163	198	-279	(-63%)	-244	(-55%)
8	723	341	410	-382	(-53%)	-313	(-43%)
9	392	474	474	82	(21%)	82	(21%)
10	272	505	230	233	(86%)	-43	(-16%)
11	131	301	272	170	(130%)	141	(108%)
12	348	237	328	-111	(-32%)	-20	(-6%)
13	243	273	285	31	(13%)	43	(18%)
14	300	147	147	-153	(-51%)	-153	(-51%)
15	147	169	169	22	(15%)	22	(15%)
16	169	134	121	-35	(-21%)	-48	(-28%)
17	121	239	158	118	(98%)	37	(31%)
18	342	268	268	-74	(-22%)	-74	(-22%)
19	268	342	342	74	(28%)	74	(28%)

The relative estimation errors reported in parentheses are computed as the estimation errors divided by the actual efforts, expressed as percentages. For instance, for project 1 ErrorCa = 49, ActualEffort = 410, thus the relative error is 49/410 = 12%.

For eleven projects, considering also the complexity density in the selection of analogous projects leads to smaller absolute relative errors. In five cases, considering also the complexity density does not cause any change in the absolute relative error. Only for three projects (P1, P13 and P16) the relative absolute error is smaller when the estimate is based only on size.

The mean and median absolute relative errors (i.e., MMRE and MdMRE) are reported in Table X, together with Pred(25). Table X shows that considering complexity density allows for better accuracy than considering size alone to identify analogous projects. This fact is confirmed by the distributions of relative errors (illustrated by the boxplot in Fig. 5) and absolute relative errors (illustrated by the boxplot in Fig. 6). Fig. 5 shows that when complexity density is considered, the median, the mean (represented as a diamond) and the errors in general are closer to zero. Fig. 6 shows that the median, the mean and the errors in general are smaller when complexity density is taken into consideration in the EbA.

TABLE X. ANALOGY-BASED ESTIMATES OBTAINED USING FPA MEASURES: MMRE, MDMRE AND PRED(25)

	CA	CA (2 var)
Mean	39.2%	28.8%
Median	23.3%	21.6%
Pred(25)	52.6%	52.6%

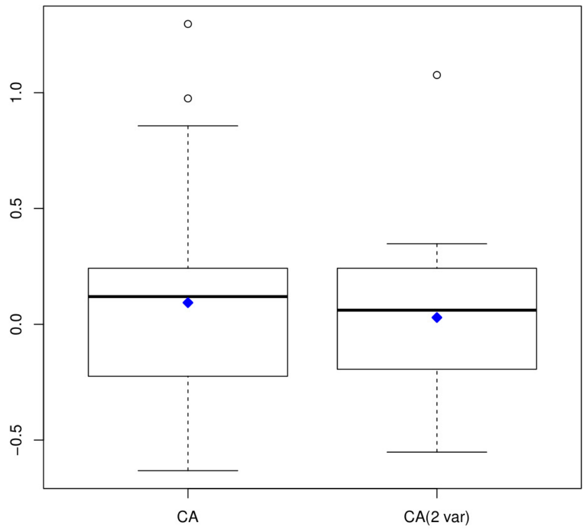

Figure 5. Estimation by analogy based on FPA measures: relative errors.

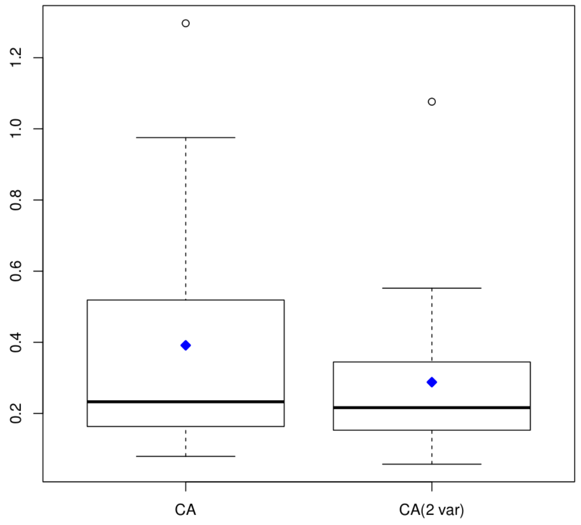

Figure 6. Estimation by analogy based on FPA measures: absolute relative errors.

TABLE XI. ANALOGY-BASED ESTIMATES USING COSMIC MEASURES

PID	Actual Effort	CA est.	CA est. (2 var)	Error CA		Error CA (2 var)	
1	410	598	428	188	(46%)	18	(4%)
2	474	435	312	-39	(-8%)	-161	(-34%)
3	382	458	474	75	(20%)	91	(24%)
4	285	304	357	19	(7%)	72	(25%)
5	328	231	330	-97	(-29%)	2	(1%)
6	198	286	286	88	(44%)	88	(44%)
7	442	249	249	-193	(-44%)	-193	(-44%)
8	723	373	317	-350	(-48%)	-405	(-56%)
9	392	455	483	63	(16%)	91	(23%)
10	272	308	308	36	(13%)	36	(13%)
11	131	249	147	118	(90%)	16	(12%)
12	348	228	323	-120	(-34%)	-25	(-7%)
13	243	485	519	242	(100%)	276	(114%)
14	300	215	338	-85	(-28%)	38	(13%)
15	147	246	126	99	(67%)	-21	(-14%)
16	169	268	268	99	(59%)	99	(59%)
17	121	266	147	145	(120%)	26	(21%)
18	342	266	338	-76	(-22%)	-4	(-1%)
19	268	169	169	-99	(-37%)	-99	(-37%)

B. *Estimation by using COSMIC Measures*

In this subsection, we present the results of estimation by analogy based on COSMIC measures: we used CFP as a size measure and Path/CFP as complexity density measure.

Effort estimates and the differences with respect to actual efforts are illustrated in Table XI.

For eight projects, considering also the complexity density in the selection of analogous projects leads to smaller absolute relative errors. For six projects the relative absolute error is smaller when the estimate is based only on size analogy. In five cases, considering also the complexity density does not cause any change in the absolute relative error.

MMRE, MdMRE and Pred(25) are given in Table XII. Table XII shows that considering complexity density allows for better accuracy than considering size alone to identify analogous projects. This fact is confirmed by the distributions of relative errors and absolute relative errors (illustrated by the boxplots in Fig. 7 and Fig. 8, respectively).

TABLE XII. ANALOGY-BASED ESTIMATES OBTAINED USING COSMIC MEASURES: MMRE, MDMRE AND PRED(25)

	CA	CA (2 var)
MMRE	43.8%	28.8%
MdMRE	36.9%	23.1%
Pred(25)	31.6%	57.9%

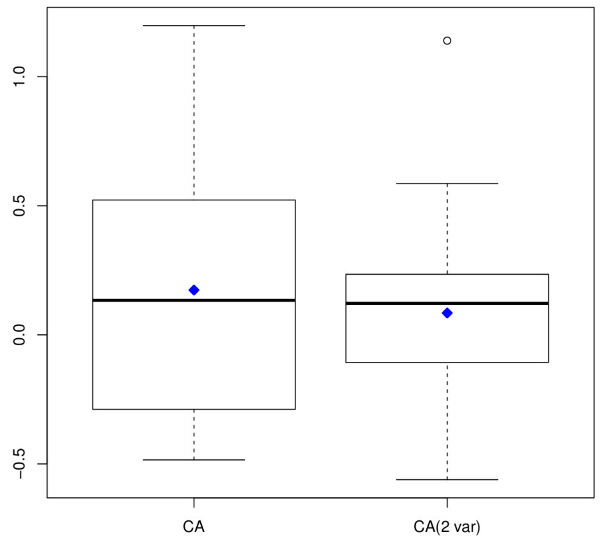

Figure 7. Estimation by analogy based on COSMIC measures: relative errors.

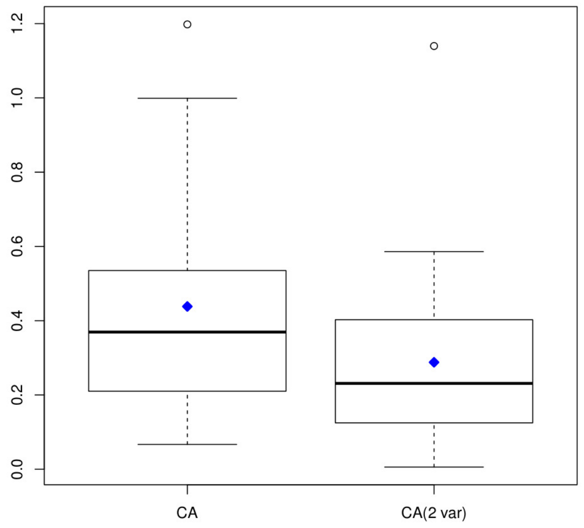

Figure 8. Estimation by analogy based on COSMIC measures: absolute relative errors.

Fig. 7 shows that when complexity density is considered, the median, the mean (represented as a diamond) and the errors in general are (slightly) closer to zero. Fig. 8 shows that the median, the mean and the errors in general are smaller when complexity density is taken into consideration in the EbA.

We can also observe that the results obtained when using COSMIC measures are very similar to those obtained using FPA measures.

C. *Estimation by Analogy using Use Case Measures*

In this subsection, we present the results of estimation by analogy based on Use Case measures: we used UUCP as a size measure and Path/UUCP as complexity density measure.

Effort estimates and the differences with respect to actual efforts are illustrated in Table XIII.

TABLE XIII. ANALOGY-BASED ESTIMATES OBTAINED USING USE CASE MEASURES

PID	Actual Effort	CA est.	CA est. (2 var)	Error CA		Error CA (2 var)	
1	410	723	474	313	(76%)	64	(15%)
2	474	723	723	249	(53%)	249	(53%)
3	382	296	323	-86	(-23%)	-59	(-15%)
4	285	305	341	20	(7%)	56	(20%)
5	328	298	273	-30	(-9%)	-55	(-17%)
6	198	263	314	65	(33%)	116	(59%)
7	442	282	291	-160	(-36%)	-151	(-34%)
8	723	442	474	-281	(-39%)	-249	(-34%)
9	392	257	243	-135	(-34%)	-150	(-38%)
10	272	317	401	45	(17%)	129	(47%)
11	131	317	268	186	(142%)	137	(105%)
12	348	295	328	-53	(-15%)	-20	(-6%)
13	243	387	387	145	(60%)	145	(60%)
14	300	201	270	-99	(-33%)	-30	(-10%)
15	147	246	195	99	(67%)	48	(32%)
16	169	305	305	136	(80%)	136	(80%)
17	121	251	208	130	(107%)	87	(71%)
18	342	209	210	-133	(-39%)	-132	(-38%)
19	268	227	147	-41	(-15%)	-121	(-45%)

For ten projects, considering also the complexity density in the selection of analogous projects leads to smaller absolute relative errors. For six projects the relative absolute error is smaller when the estimate is based only on size analogy.

The MMRE, MdMRE and Pred(25) are reported in Table XIV. Table XIV shows that considering complexity density allows for better accuracy than considering size alone to identify analogous projects only as far as is concerned.

The distributions of relative errors (illustrated by the boxplot inFig. 9) and absolute relative errors (illustrated by the boxplot in Fig. 10) show that taking into account complexity density when selecting analogue projects causes marginal improvements in estimation accuracy.

TABLE XIV. ANALOGY-BASED ESTIMATES OBTAINED USING USE CASE MEASURES: MMRE, MDMRE AND PRED(25)

	CA	CA (2 var)
MMRE	46.6%	41.1%
MdMRE	36.3%	38.1%
Pred(25)	31.6%	31.6%

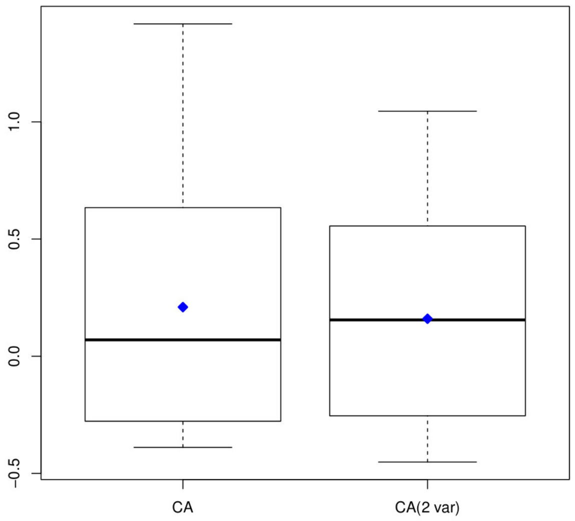

Figure 9. Estimation by analogy based on Use Case measures: relative errors.

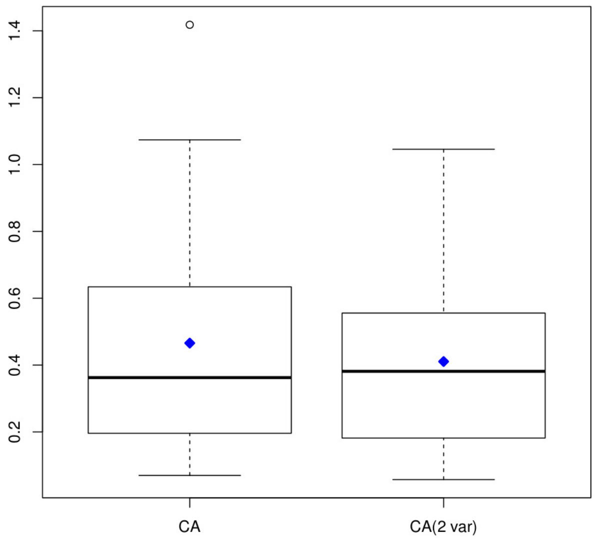

Figure 10. Estimation by analogy based on Use Case measures

As a final observation, we note that using both size and complexity density in the selection of analogue projects generally results in more accurate estimates. Only when the size measures are based on use cases, considering complexity density does not lead to a clear improvement in estimation accuracy.

We can conclude that complexity density appears to be a relevant factor to be considered when EbA is adopted.

VII. DISCUSSION

Table XV summarizes the accuracy of the effort models described in Section V. These models are all those statistically significant and featuring $R^2 > 0.6$.

TABLE XV. ACCURACY OF EFFORT MODELS

Model	MMRE	Pred(25)	Error range
Effort = -29.9 + 139.1× Path/FPr	34%	63%	-46%–162%
Effort = 120.6 × $FPAtrans^{0.654}$ × $(Path/UFP)^{0.958}$	71%	53%	-40%–543%
Effort = 12.2 × $FPAtrans^{0.816}$ × $(Path/FPAtrans)^{0.976}$	25%	63%	-60%–108%
Effort = 112.2 × $CFP^{0.391}$ × $(Path/CFP)^{1.298}$	22%	68%	-30%–77%
Effort = 231.8 × $FPr^{0.377}$ × $(Path/CFP)^{1.468}$	23%	68%	-27%–97%
Effort = 21.9 × $UC^{0.68}$ × $(Path/UC)^{0.728}$	24%	63%	-45%–73%
Effort = 24.9 × $UUCP^{0.679}$ × $(Path/UUCP)^{0.775}$	24%	63%	-45%–74%

It is easy to see that all models in Table XV have at least one parameter that accounts for functional complexity per size unit.

It should be noted that we found also a model based *exclusively* on complexity density. This model was rather unexpected, as it says that the size of the programs is not important at all. This result is probably due to the fact that the variation of size is relatively little in the set of projects that we analysed. Additional research is needed to explore this point.

Having shown that functional complexity per size unit is essential for regression models, we looked at the role that functional complexity per size unit can play in EbA. To this end, EbA was applied using two criteria for determining analogues projects:

– According to size only;
– According to both size and functional complexity per size unit.

Table XVI illustrates the results of EbA concerning the accuracy of estimates via MMRE, MdMRE and Pred(25).

TABLE XVI. ACCURACY OF ANALOGY-BASED ESTIMATES

Variables used to identify analogue projects	MMRE	MdMRE	Pred(25)
UFP	39.2%	23.3%	52.6%
UFP, Path/UFP	28.8%	21.6%	52.6%
CFP	43.8%	36.9%	31.6%
CFP, Path/CFP	28.8%	23.1%	57.9%
UUCP	46.6%	36.3%	31.6%
UUCP, Path/UUCP	41.1%	38.1%	31.6%

Table XVI confirms the relevance of functional complexity per size unit, as it helps increasing accuracy. Actually it can be noted that EbA's accuracy is generally worse than regression models'. However, using functional complexity per size unit for determining analogues projects tends to make estimation accuracy closer to regression models'.

VIII. THREATS TO VALIDITY

As in any software engineering empirical study, several issues threaten the validity of the results. Here we discuss such factors and the actions that have been undertaken to mitigate them.

The limited size of the available dataset can be regarded as a first threat to internal validity. The used dataset is sufficiently large to support statistically significant analysis; however, the fact that our dataset is representative of most software systems is doubtful. To this end, we note that the projects from which the data used in the paper were derived are all real development projects, as those in best known datasets, like the PROMISE [39] and ISBSG [22] datasets. Some of our projects are rather small, but the majority of our projects (namely 11 out of 19) have size in the 100–340 UFP, thus they can be considered medium-sized projects (40% of the projects in the ISBSG dataset are in the 100–340 UFP range, while more than half have size smaller than 340 UFP).

Another possible threat to the validity of the study derives from part of the projects being academic projects. However, the projects that were carried out at the Austral University were developed using techniques, tools and methodologies similar to those used in the industrial projects. Accordingly, we do not expect that these projects required substantially different effort than other projects.

IX. CONCLUSION

The work reported here moves from the consideration that development effort depends (also) on the complexity or the amount of computation required, but no suitable measure has emerged as a reliable way for capturing such complexity. In fact, very popular methods like COCOMO II [11][15] still use just an ordinal scale measure for complexity, based on the subjective evaluation performed by the user.

We approached the problem of measuring the required functional complexity by considering the most relevant approaches presented in the literature, and testing them on a set of projects that were measured according to FPA, COSMIC and Use Case-based functional size measurement methods.

The results of our analysis do not allow us to draw definite conclusions, since our observations are based on a specific set of data (see Table I). However, we observed that all the models obtained were based on a notion of computation density, which is based on the measure of Paths [43], i.e., the number of distinct computation flows in functional processes. Similarly, Estimation by Analogy appears to benefit from the possibility of using the notion of computation (or complexity) density in identifying analogue projects.

Since Paths are quite easy to measure [33] and appear as good effort predictors, we suggest that future research on effort estimation takes into consideration the possibility of involving a Path based measure of functional complexity.

An important results for practitioners is that functional complexity appears as a factor that affects development effort; accordingly, whatever method is used for effort estimation, it is advisable to take functional complexity into due consideration.

We plan to continue experimenting with measures of functional complexity. Since in this type of experimentations a critical point is the difficulty to get measures, we kindly invite all interested readers that are involved in effort estimations to perform functional complexity measurement and share the data with us and the research community.

ACKNOWLEDGMENT

The research presented in this paper has been partially funded by the project "Metodi, tecniche e strumenti per l'analisi, l'implementazione e la valutazione di sistemi software" funded by the Università degli Studi dell'Insubria, and by the Research Fund of the School of Engineering of Austral University.

REFERENCES

[1] L. Lavazza and G. Robiolo, "Functional Complexity Measurement: Proposals and Evaluations, " Proc. 6th Int. Conf. on Software Engineering Advances (ICSEA 2011).

[2] A. Albrecht, "Measuring Application Development Productivity," Proc. IBM Application Development Symp. I.B.M. Press, 1979.

[3] 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, November 1983.

[4] N.Aggarwal, N. Prakash, S. Sofat, "Web Hypermedia Content Management System Effort Estimation Model," SIGSOFT Software Engineering Notes, vol. 34, March 2009.

[5] M. AlSharif, W.P. Bond, and T. Al-Otaiby, "Assessing the complexity of software architecture," Proc 42nd annual Southeast regional conference (ACM-SE 42). ACM, New York, NY, USA, 2004, pp. 98-103.

[6] B.Anda, E. Angelvik, and K. Ribu, "Improving Estimation Practices by Applying Use Case Models," Lecture Notes In Computer Science, vol. 2559, Springer, 2002, pp. 383-397.

[7] L. Baresi, S. Colazzo, L. Mainetti, and S. Morasca, "W2000: A modeling notation for complex Web applications," In Web Engineering, E. Mendes and N. Mosley. Eds., Springer-Verlag, 2006.

[8] L. Baresi, and S. Morasca, "Three Empirical Studies on Estimating the Design Effort of Web Applications," ACM Transactions on Software Engineering and Methodology, vol. 16, September 2007.

[9] H. Bashir, and V. Thomson, "Models for Estimating Design Effort and Time," Design Studies, vol. 22, March, Elsevier, 2001.

[10] B. Bernárdez, A. Durán, and M. Genero, "Empirical Evaluation and Review of a Metrics–Based Approach for Use Case Verification," Journal of Research and Practice in Information Technology, vol. 36, November 2004.

[11] B.W. Boehm, E. Horowitz, R. Madachy, D. Reifer, B.K. Clark, B. Steece, A. Winsor Brown, S. Chulani and C. Abts, Software Cost Estimation with Cocomo II. Prentice Hall, 2000.

[12] G. Booch, J. Rumbaugh, and I. Jacobson, The Unified Modeling Language User Guide, Addison Weley, 1998.

[13] L. Briand, S. Morasca, and V.R. Basili, "Property-Based Software Engineering Measurement," IEEE Transactions on Software Engineering, vol. 22, Month, 1996.

[14] L. Briand and J. Wust, "Modeling Development Effort in Object-Oriented Systems Using Design Properties," IEEE Transactions on Software Engineering, vol. 27, November 2001.

[15] COCOMO II Model Definition Manual. http:// csse.usc.edu/csse/research/COCOMOII/cocomo_downloads.h tm, last access Dec. 20, 2012.

[16] COSMIC – Common Software Measurement International Consortium, 2009. The COSMIC Functional Size Measurement Method - version 3.0.1 Measurement Manual (The COSMIC Implementation Guide for ISO/IEC 19761: 2003), May 2009.

[17] N.E. Fenton, Software Metrics: A Rigorous Approach. Chapman and Hall, London, 1991.

[18] D.D. Galorath and M.W. Evans, Software Sizing, Estimation, and Risk Management, Auerbach Publications, 2006.

[19] C. Gencel and O. Demirors, "Functional Size Measurement Revisited," ACM Transactions on Software Engineering and Methodology, vol. 17, June 2008.

[20] S. Gupta, G. Sikka, and H. Verma, "Recent methods for software effort estimation by analogy," SIGSOFT Softw. Eng. Notes , vol. 36, August 2011, pp 1-5.

[21] T. Hastings and A. Sajeev, "A Vector-Based Approach to Software Size Measurement and Effort Estimation," IEEE Transactions on Software Engineering, vol. 27 April 2001.

[22] International Software Benchmarking Standards Group: Worldwide Software Development: The Benchmark, release 11, 2009.

[23] ISO/IEC19761:2003, Software Engineering – COSMIC-FFP – A Functional Size Measurement Method, International Organization for Standardization, Geneve, 2003.

[24] ISO/IEC 20926: 2003, Software engineering – IFPUG 4.1 Unadjusted functional size measurement method – Counting Practices Manual, International Organization for Standardization, Geneve, 2003.

[25] M. Jørgensen, M. Boehm, S. Rifkin, "Software Development Effort Estimation: Formal Models or Expert Judgment?," IEEE Software, vol. 26, March-April 2009.

[26] C. Jones, A Short History of Function Points and Feature Points. Software Productivity Research, Inc., Burlington, Mass., 1986.

[27] C. Jones, Strengths and Weaknesses of Software Metrics. Version 5, Software Productivity Research, 2006.

[28] G. Karner, Resource Estimation for Objectory Projects. Objectory Systems, 1993.

[29] G. Karner, "Metrics for Objectory," Diploma thesis, University of Linköping, 1993.

[30] B. Kitchenham, S.L. Pfleeger, B. McColl, and S. Eagan, "An Empirical Study of Maintenance and Development Accuracy," Journal of Systems and Software, vol. 64, October 2002.

[31] B. Kitchenham, L.M. Pickard, S.G. MacDonell, M.J. Shepperd, "What accuracy statistics really measure [software estimation]," IEE Proceedings - Software vol. 148 , June 2001, pp. 81 – 85.

[32] S. Kusumoto, F. Matukawa, K. Inoue, S .Hanabusa, and Y. Maegawa, "Estimating effort by Use Case Points: Method, Tool and Case Study," Proc. 10th International Symposium on Software Metrics, 2004.

[33] L. Lavazza and G. Robiolo, "Introducing the Evaluation of Complexity in Functional Size Measurement: a UML-based Approach," Proc. 4th Int. Symposium on Empirical Software Engineering and Measurement (ESEM 2010).

[34] L. Lavazza and G. Robiolo, "The Role of the Measure of Functional Complexity in Effort Estimation," Proc. 6th Int. Conf. on Predictive Models in Software Engineering (PROMISE 2010).

[35] G. Levesque V. Bevo, and Tran Cao, D., “Estimating Software size with UML Models,” Proc. 2008 C3S2E conference, ACM International Conference Proceeding Series, vol. 290, 2008.

[36] K. Lind and R. Heldal, “Categorization of Real-time Software Components for Code Size Estimation,” Proc. ACM-IEEE Int. Symp. on Empirical Software Engineering and Measurement (ESEM 2010). ACM, New York, NY, USA, 2010.

[37] T.J. McCabe, “A Complexity Measure,” IEEE Transactions on Software Engineering, vol.2, December 1976.

[38] E. Mendes, N. Mosley, and S. Counsell, “A Comparison of Length, Complexity and Functionality as Size Measures for Predicting Web Design and Authoring Effort,” Proc. Evaluation and Assessment in Software Engineering Conference (EASE 2001).

[39] T. Menzies, B. Caglayan, E. Kocaguneli, J. Krall, F. Peters, and B. Turhan, “The PROMISE Repository of Empirical Software Engineering Data http://promisedata.googlecode.com,” West Virginia University, Department of Computer Science, 2012, last access Dec. 20, 2012.

[40] S. Misra, “Modified Cognitive Complexity Measure,” Proc. 21st international conference on Computer and Information Sciences (ISCIS'06), A. Levi, E. Savaş, H. Yenigün, S. Balcisoy and Y. Saygin Eds., Springer-Verlag, Berlin, Heidelberg, 2006, pp. 1050-1059.

[41] N. Mittas and L. Angelis, “Combining Regression and Estimation by Analogy in a Semi-parametric Model for Software Cost Estimation,” Proc. 2nd ACM-IEEE Int. Symp. on Empirical Software Engineering and Measurement (ESEM 2008). ACM, New York, NY, USA, 2008, pp. 70-79.

[42] S. Morasca, “On the Use of Weighted Sums in the Definition of Measures,” ICSE Workshop on Emerging Trends in Software Metrics (WETSoM 2010).

[43] G. Robiolo, G. and R. Orosco, “Employing Use Cases to Early Estimate Effort with Simpler Metrics,” Innovations Syst. Softw. Eng, vol.4, April 2008.

[44] S. Sengupta, A. Kanjilal, and S. Bhattacharya, “Measuring Complexity of Component Based Architecture: a Graph Based Approach,” SIGSOFT Softw. Eng. Notes vol. 36, January 2011, pp. 1-10.

[45] M. Shepperd and C. Schofield, “Estimating Software Project Effort Using Analogies”, IEEE Transactions on Software Engineering, vol. 23, month11, 1997, pp. 736-774.

[46] D. Tran Cao, G. Lévesque, and A. Abran, “From Measurement of Software Functional Size to Measurement of Complexity”, Int. Conf. on Software Maintenance (ICSM 2002).

[47] D. Tran Cao, G. Lévesque, and J-G. Meunier, “A Field Study of Software Functional Complexity Measurement”, Proc. 14th Int. Workshop on Software Measurement (IWSM/ METRIKON 2004).

[48] G. Visaggio, “Structural Information as a Quality Metric in Software Systems Organization,” Proc. Int. Conf. on Software Maintenance, 1997, pp. 92-99.

[49] T. Wijayasiriwardhane and R. Lai, “Component Point: A System-level Size Measure for Component-Based Software Systems,” Journal of Systems and Software vol. 83, month 2010, pp. 2456-2470.

[50] A. Whitmire, “An Introduction to 3D Function Points,” Software Development, vol. 3, April 1995.

[51] Y. Yavari, M. Afsharchi, and M. Karami, “Software Complexity Level Determination Using Software Effort Estimation Use Case Points Metrics,” 5th Malaysian Conference in Software Engineering (MySEC 2011), pp. 257-262.