The Role of the Measure of Functional Complexity in Effort Estimation

Luigi Lavazza
Dipartimento di Informatica e Comunicazione
Università degli Studi dell'Insubria
Via Mazzini, 5 – 21100 Varese, Italy
+39 0332 218930
luigi.lavazza@uninsubria.it

Gabriela Robiolo
Facultad de Ingeniería
Universidad Austral
Av. Juan de Garay 125, 1063 Bs.As., Argentina
+54 11 5921 8000
grobiolo@austral.edu.ar

ABSTRACT

Background. Currently there are several definitions of measures that should represent the size of software functional requirements. These measures have gained a quite relevant role, since they are one of the few basis upon which effort estimation can be based. However, traditional Functional Size Measures do not take into account the amount and complexity of the elaboration required, concentrating instead on the amount of data accessed or moved. This is a problem, when it comes to effort estimation, since the amount and complexity of the required data elaborations affect the implementation effort, but are not adequately represented by the current measures (including the standardized ones).

Objective. The paper evaluates different types of functional size measures as effort estimators. Moreover, the consequences of taking into consideration also the amount and complexity of required elaboration in the effort estimation models are evaluated.

Methods. In this paper we take into consideration a representative set of functional size measures (namely, function points, COSMIC function points and use case points) and a recently proposed elaboration complexity measure (Paths) and evaluate how well these measures are correlated with the development effort. To this end, we measured a set of 17 projects and analyzed the resulting data.

Results. We found that it is possible to build statistically valid models of the development effort that use the functional size and complexity measures as independent variables. In fact, we discovered that using the measure of elaboration complexity in addition to the functional size substantially improves the precision of the fitting.

Conclusions. The analysis reported here suggests that a measure of the amount and complexity of elaboration required from a software system should be used, in conjunction with traditional functional size measures, in the estimation of software development effort. Further investigations, involving a greater number of projects, are however needed to confirm these findings.

Categories and Subject Descriptors

D.2.8 [**Software Engineering**]: Metrics – *Complexity measures, product metrics, process metrics.*

General Terms

Measurement.

Keywords

Functional size measurement; Function Points; COSMIC function points; Use case points; Effort estimation; Use case-based measurement; Functional complexity measurement.

1. INTRODUCTION

Functional Size Measurement (FSM) is quite important in the current software development practice, because most effort estimation models rely on an evaluation of the size of the applications to be developed. While traditional models accepted mainly size indications expressed in lines of code (LOC), modern models tend to accept or even require measures of the functional size of the product to be developed.

Currently there are a few FSM methods that are widely used. Among these, in this paper we consider Function Points (FP) [2] [17], Use Case Points (UCP) [20][21], and COSMIC Function Points (CFP) [14].

It is well known that the effort to be dedicated to the development of a software application does not depend exclusively on the functional size of the application. In fact, the most widely used models (including COCOMO [9][10]) are of the type Effort=f(Size, <product characteristics parameters>, <process characteristics parameters>), where <product characteristics parameters> aim at describing the characteristics of the product (e.g., complexity, non functional requirements, etc.) that are known to affect the development effort. The definitions of these models properly take into account that the functional size alone is not able to support the estimation of the development effort. However, the measure of complexity that they adopt is of purely qualitative nature. For instance, COCOMO II [9][10] provides a table that allows the user to evaluate complexity on an ordinal scale, on the basis of qualitative and subjective criteria. It is quite clear that it would be greatly beneficial to replace such subjective and approximate evaluations of complexity with a real measure, based on objective and quantitative evaluations.

Concerning the functional size, there are some legitimate doubts about the ability of currently used measures to correctly represent the functionality of the software products. This issue is explicitly mentioned in the definition of COSMIC function points [14]: a 'functional process' is considered composed of data movements and data elaborations. Nevertheless, only data movements are measured. The exclusion of data elaboration from the measurement is due only to the difficulty of finding a quantitative way of characterizing data elaboration in an effective and reasonably easy to measure way.

In practice, the FSM methods proposed until now fail to capture the "amount of elaboration" required. The consequence is that two applications that differ only for 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 [24], 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.

Recently, a method to measure the amount of elaboration required by functional user requirements has been proposed [30][31]. The method (synthetically described in Section 2) applies the principles of McCabe complexity measurement [28] to requirements described via use cases.

In this paper we explore –on the basis of the measures of a set of projects– the possibility of using the measure of the required data elaboration in conjunction with the traditional functional size measures, for the purpose of effort estimation.

The paper is organized as follows: section 2 provides an overview of the measures used; section 3 describes the measurement activities and the resulting dataset; section 4 describes the data analysis and its results; section 5 discusses the results and lessons learnt; section 0 accounts for related work, while section 7 draws some conclusions.

2. THE INVOLVED MEASURES

In the 70's Allan Albrecht introduced Function Points as a way of measuring "the function value delivered to customers"[2]. Function points, being meant to capture the point of view of the customer in a technology independent manner, are a measure of the functional requirements that takes into account elements that can be identified by the user. In fact, the measurement of FPs is based on five Basic Functional Components (BFCs): External Input (EI, operations whose main intent is to input distinct application-oriented data), External Output (EO, operations whose main intent is to generate user-relevant outputs), External Inquiry (EQ, operations whose main intent is to provide user-relevant outputs by retrieving data), Internal Logical File (ILF, distinct application-oriented data managed within the system) and External Interface File (EIF, distinct application-oriented data managed out of the system, typically machine-readable interfaces to other systems). The complexity of each element is classified as Low, Average or High. This complexity classification depends on the number of different elements:

a. For ILF and EIF, the number of Data Element Types (DET) and Record Element Types (RET).

b. For EI, EO and EQ, the number of DET and File Types Referenced (FTR).

Therefore, the Unadjusted Function Points (UFP) are defined as the sum of the products of the factors of complexity (CF_{ij}) by the number of classified elements (E_{ij}) in an application. This can be expressed as follows:

$$UFP = \sum_{i=1}^{5} \sum_{j=1}^{3} CF_{ij}\, E_{ij}$$

Albrecht introduced also the possibility of "adjusting" function points to make the more suitable for effort estimation. However, the adjustment has been widely criticized (see for instance [22]), and has been excluded from the standard definition of FP [17]; accordingly here we use only unadjusted FPs (even when we omit the adjective "unadjusted").

COSMIC function points [14] were introduced to solve some theoretical problems with function points, and also for making the measurement process better defined and simpler to apply. The measurement method considers the Functional User Requirements as a set of functional processes, each characterized by a set of data movements. The latter are classified as Entries (data that enters the system from outside), Exits (data that exits from the system), Reads (data read from the storage), Writes (data written to the storage). The size of the system is the sum of the sizes of functional processes; the size of the latter is given by the non-repeated number of data movements it involves:

$$CFP = \sum_{i}^{Proc} \sum_{j}^{Mov_i} Entries_{ij}\ Exits_{ij}\ Writes_{ij}\ Reads_{ij}$$

Where Proc is the set of functional processes and Mov_i is the set of data movements of the i-th functional process.

In 1993, Karner introduced use case points, a redefinition of FP in the context of use case [20][21]. The measurement of UCP is defined on only two basic elements: Actors and Use cases. Like in the computation of FPs, these factors are weighted according to three complexity levels: the complexity of Actors is evaluated on the basis of actors' characteristics, while use cases are weighted according to the number of transactions. So UCP is computed –for each use case– as follows:

$$UCP = \sum_{i=1}^{2} \sum_{j=1}^{3} CF_{ij}\, E_{ij}$$

FP, UCP and CFP all aim at measuring the functional size of applications. However, none of these measures takes into account the amount and complexity of elaboration required, concentrating instead on the amount of data accessed or moved. This is a problem, when it comes to effort estimation, because the amount and complexity of the required data elaborations affect the implementation effort.

In order to improve this situation, Paths were defined as a simple measure of complexity, based on information typically available from use case descriptions [30].

The measure of the complexity of use cases is based on the application of the principles of McCabe's complexity measure [28] 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' behavior of the user and system, and a set of alternative scenarios, which account for all the possible deviations from the normal behavior that have to be supported by the system. Robiolo et al. [31] 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. For example, in the use case reported in Figure 1 the main path has an alternative path identified by the "if the" expression. So, for each use case the number of paths is the number of scenarios (main and alternative).

A detailed description of the measure and its applicability to use cases described in UML can be found in [25].

Note that throughout the paper we use the terms "elaboration amount" and "elaboration complexity" as synonyms. This is due to the fact that at the requirements level it is not quite clear what is the amount of elaboration and what is its complexity. Moreover, the term "complexity" fits well when considering use case scenarios and their articulation; on the contrary, the "amount of elaboration" is a more general concept, which does not depend on the way the functional requirements are modeled. In any case, we always make reference to the characteristics of the software system that are measured via Paths.

1. The system displays a screen with the list of categories and subcategories.
2. The user must choose the category to which the new product belongs. The list of properties of the selected categories and subcategories will appear below. ***If the*** categories and subcategories are new, a detailed description is also displayed.
3. By pressing enter, the user will go to a new screen in which he should fill in gaps with the definition of the properties of the product.
4. The use case finishes once the user enters a product, saves the changes, and leaves the system or cancels the operation.

Figure 1. Description of a use case.

3. DATA COLLECTION

In order to evaluate the measures mentioned in Section 2 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– because they do not report the size of each project according to different FSM methods; moreover, the Paths measure is very recent, and no historical data exist for it.

We measured 17 small business projects, which were developed in three different contexts: an advance 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 these requisites:

a. Use cases-based definitions of requirements were available.

b. They were all new developments.

c. The use cases had been completely implemented, and the actual development effort in PersonHours was also known.

So, the projects were sufficiently similar to avoid major issues due to heterogeneity. The unavoidable differences do not hinder the purpose of the study, i.e., to evaluate the contribution of functional complexity to the development effort. The evaluation of other factors that affect the development effort is out of the scope of the paper.

The collected data, together with a minimal description of the projects, are reported in Table 1.

Table 1. The dataset

ID	Application description	Implementation environment	Use Cases	Effort [PH]	Path	UFP	UCP	CFP
P1	Controls trips using mobile phones	Java, Tomcat, J2ME Mobile Phone Emulator	39	410	71	144	201	129
P2	Manages a centralized web purchase system	Java, Tomcat, Hibernate	28	473.5	73	266	149	115
P3	Manages a hotel information system	Java, Struts, Hibernate	7	382.4	60	171	84	108
P4	Tracks projects information	Jave, Hibernate, Tomcat	6	285	49	142	72	74
P5	Manages information about music bands	Java, Apache, Tomcat	12	328	34	89	72	48
P6	Sells with an authorized card	MS (Visual Studio, Office Sharepoint Portal Server)	8	198	35	75	62	66
P7	Sells cinema tickets using mobile phones	MS Windows XP Professional, Apache, Tomcat	6	442.02	50	57	71	81
P8	Tracks files	JSP, J2EE, Hibernate, Tomcat	27	722.65	97	210	175	116
P9	Manages an accounting process	JSP, J2EE, Hibernate	15	392	83	311	111	119
P10	Manages university graduate alumni information	Power Builder, Javascript	19	272	42	155	119	73
P11	Controls clients' debts	Power Builder	13	131	18	117	68	51
P12	Manages travel information	Web application$^{(\circ)}$	20	1042	118	130	169	85
P13	Manages trips information	Web application$^{(\circ)}$	12	348	32	93	71	43
P14	Transport and distribution of fuel	Java , Tomcat, Spring MVC, MySQL, JSTL	12	242.5	68	84	99	113
P15	Bus ticket purchases on line	Java, Apache, Tomcat	4	299.76	33	28	57	53
P16	Company intranet	Java, Apache Wicket, Maven,	10	147	20	47	53	53
P17	Turn-based strategy game	Java	5	169	17	43	28	29

$^{(\circ)}$ No more data is available due to confidential restrictions.

4. DATA ANALYSIS

4.1 Effort vs. Functional Size

4.1.1 Effort vs. UFP

The effort and Unadjusted Function Points (UFP) data are plotted in Figure 2. Spearman's test indicates a mild correlation: ρ = 0.549 (p-value = 0.02445).

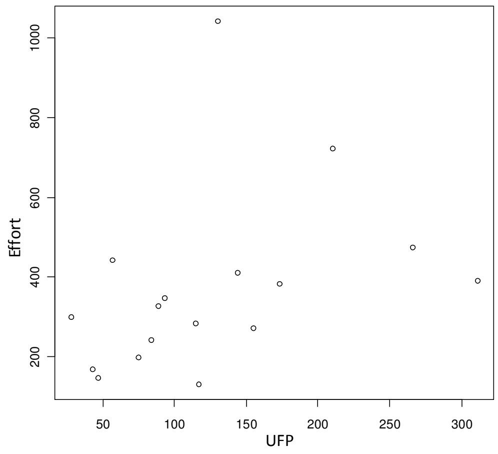

Figure 2. Plot of Effort vs. UFP.

We proceed now to evaluate the correlation of Effort with unadjusted FP using linear regression. Figure 2 suggests the existence of outliers. In order to identify outliers, we use Cook's distance [13]. Following the indications from the literature, we consider outliers the data points having Cook's distance > 4/n, where n is the size of the dataset. Using this technique leads to dropping projects P9, P12, P8, P7, and P15 from the dataset. In this way we get a model featuring an adjusted R^2 = 0.5424 and significant at the 0.01 threshold (p-value = 0.0038). The linear regression line has equation:

$$\text{Effort} = 1.3538\ \text{UFP} + 124$$

The residuals are normally distributed (i.e., Shapiro-Wilk W test [32] does not allow rejecting the normality hypothesis), as are the dependent and independent variables.

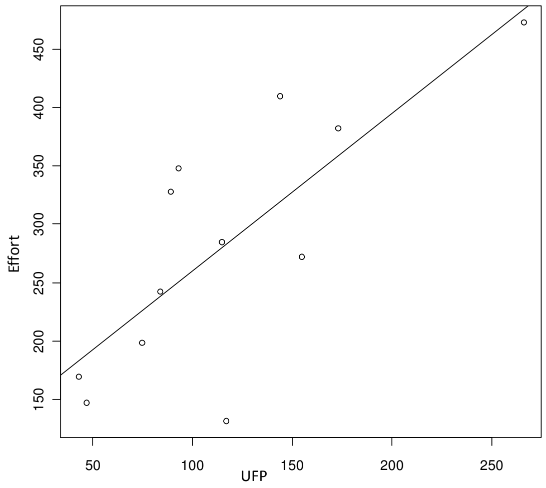

Figure 3. Effort vs. UFP linear regression line.

The precision of the fitting is characterized by: MMRE= 23%, Pred(25) = 66.7%, error range: -28.1%..115.7%. The distributions of residuals and relative residuals are displayed in Figure 4. Note that here we consider residuals with respect to the dataset without the outliers.

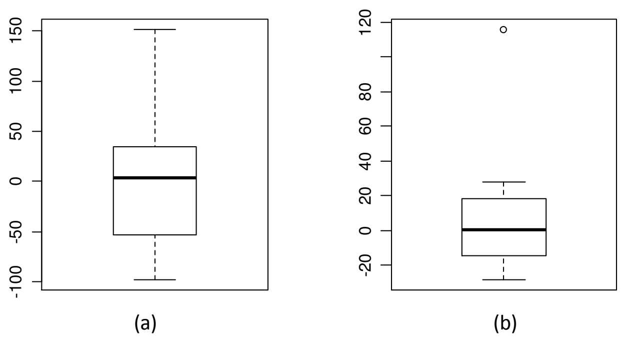

Figure 4. Effort vs. UFP: distribution of residuals (a) and relative residuals (b).

4.1.2 Effort vs. UCP

The effort and Unadjusted Use Case Points (UCP) data are plotted in Figure 5.

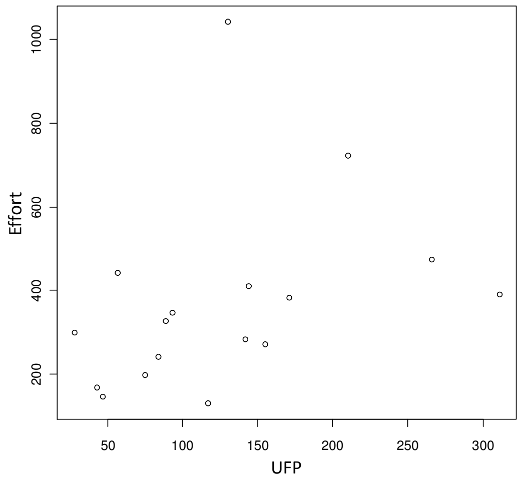

Figure 5. Plot of Effort vs. UCP.

Spearman's test indicates the existence of a correlation: ρ = 0.726 (p-value = 0.0009593).

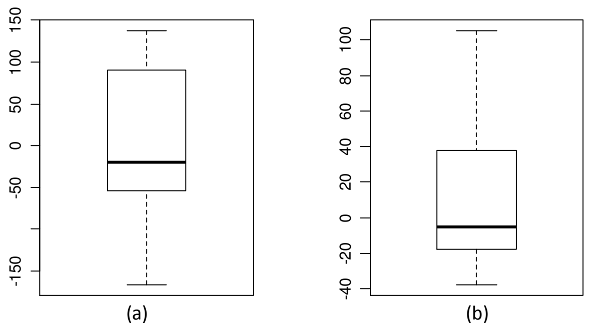

Figure 6. Effort vs. UCP: distribution of residuals (a) and relative residuals (b).

Linear regression analysis yields a model (with p-value: 0.02353) featuring a rather low adjusted R^2 (0.3057). Coherently, the precision of the model is rather low. In fact, it is characterized by MMRE= 29.5%, Pred(25) = 57.1%, error range: -37.7%..105.3%. The distributions of residuals and relative residuals are displayed in Figure 6.

4.1.3 *Effort vs. CFP*

The effort and CFP data are plotted in Figure 7.

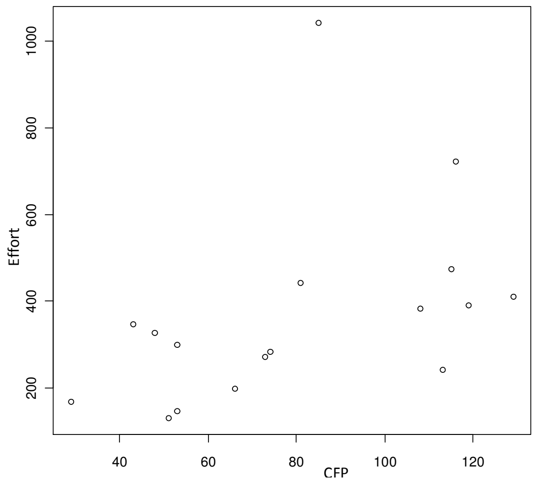

Figure 7. Plot of Effort vs. CFP.

Spearman's test indicates the existence of a correlation: ρ = 0.6327 (p-value = 0.006411).

Linear regression yields a significant (p-value = 0. 01296) model featuring R^2 0.3891 (Adjusted R^2 = 0. 3421). The variables and the residuals are normally distributed.

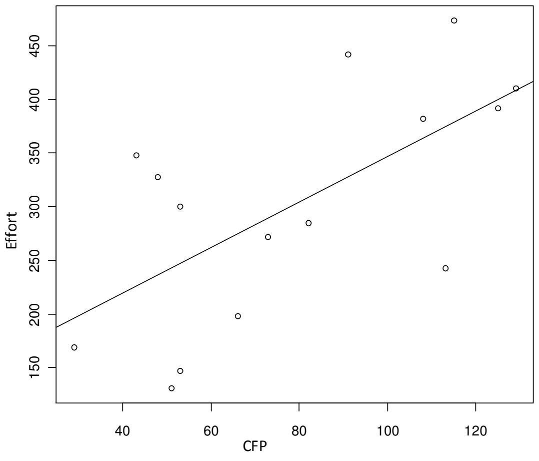

Figure 8. Effort vs. CFP linear regression line.

The linear regression line has equation:

$$\text{Effort} = 2.1\ \text{CFP} + 139$$

Also in this case, the precision of the model is not very good. In fact, it is characterized by MMRE= 27.7%, Pred(25) = 53.3%, error range: -34%..88.3%. The distributions of residuals and relative residuals are displayed in Figure 9.

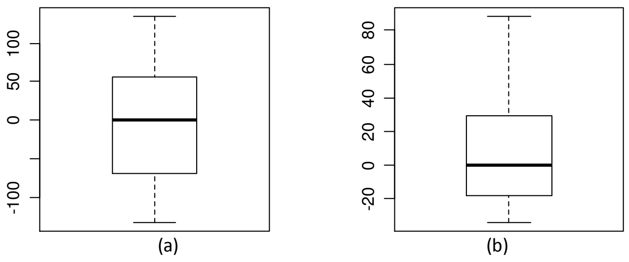

Figure 9. Effort vs. CFP: distribution of residuals (a) and relative residuals (b).

4.2 Effort vs. Functional Complexity

The relation between Effort and Paths is illustrated by the plot reported in Figure 10.

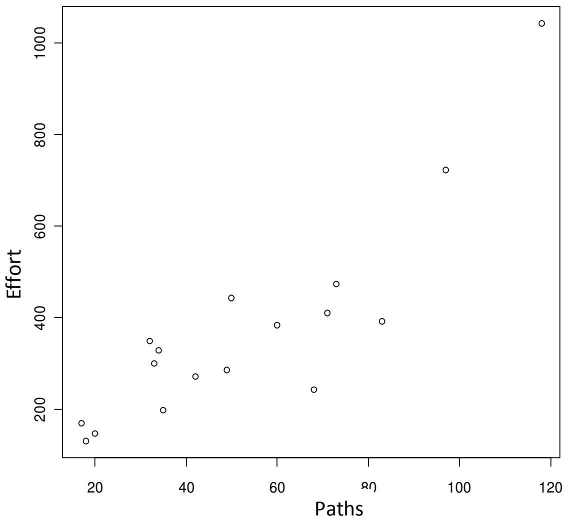

Figure 10. Plot of Effort vs. Paths.

Spearman's test indicates the existence of a good correlation: ρ = 0.8137 (p-value = 9.092e-05).

Linear regression yields a significant model featuring Adjusted R^2 = 0.7142 and p-value: = 0.0001687.

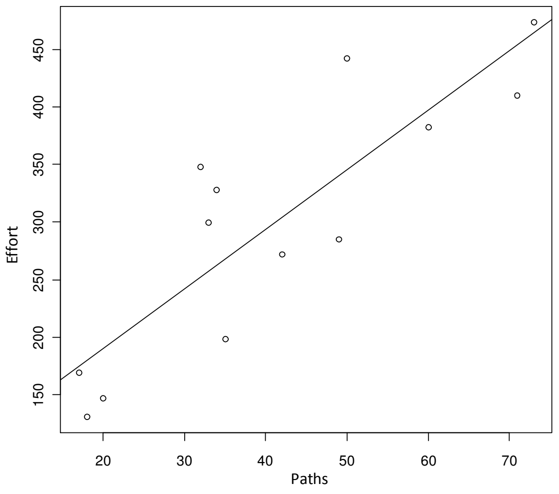

Figure 11. Effort vs. Paths: linear regression line.

Quite noticeably, the model is more precise than the ones using functional size measures as independent variables. In fact, the precision is characterized by MMRE= 18.1%, Pred(25) = 69.2%, error range: -27.6%..37%. The distributions of residuals and relative residuals are displayed in Figure 12.

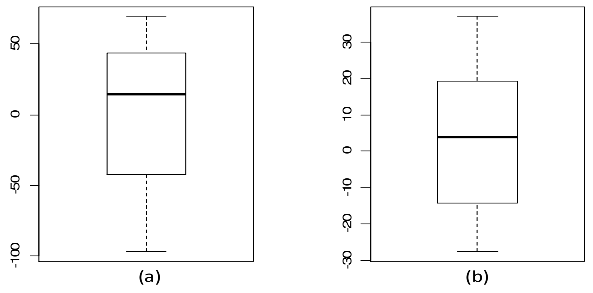

Figure 12. Effort vs. Paths: distribution of residuals (a) and relative residuals (b).

4.3 Effort vs. Functional Size and Complexity

In this section we experiment with multiple linear regression, using both size and complexity metrics as independent variables.

As a complexity metric, we do not use the number of paths, since a big number of paths may indicate a complex application as well as a large one; in fact, even for simple applications, the number of paths increases with the number of involved use cases. Instead, we use the number of paths divided by the measure of size: the relative amount of paths indicates how 'complex' is every 'unit of functionality' released to the user. This practice is not new: for instance, the complexity of object-oriented systems is often measured by computing the McCabe number of every method, and then considering the mean value. Here we are making something similar, at the requirements specification level, instead than at the code level.

4.3.1 Effort vs. UFP and Complexity

When using FP to measure the functional size, the analysis of the relationship that links effort and functional size and complexity yielded the following model (after eliminating as outliers projects P12, P9, P8, P15 and P14):

Effort = 1.6837 UFP + 516.0687 Paths/UFP +103

The model has Adjusted R^2 = 0.7499 and p-value = 0.0007926.

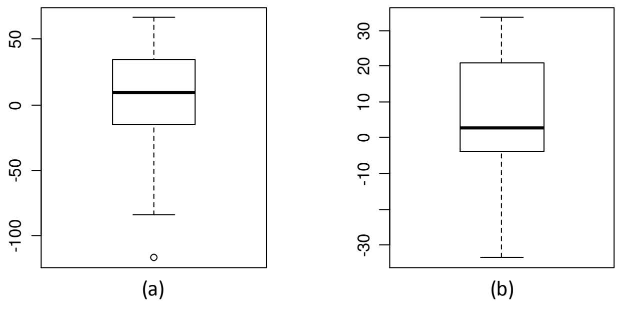

Figure 13. Effort vs. UFP and Paths/UFP: distribution of residuals (a) and relative residuals (b).

The residuals of the model (not considering outliers) are characterized by MMRE = 15.9%, range -33.5%.. 33.5%, Pred(25) = 58.3%. The distributions of residuals and relative residuals are displayed in Figure 13.

4.3.2 Effort vs. UCP and Complexity

When using use case points to measure the size, the model obtained via multiple linear regression (after eliminating project P12 and P8, considered outliers) is:

Effort = 1.7802 UCP + 340.7816 Paths/UCP – 38

Adjusted R^2 = 0.5114, p-value = 0.005398.

The residuals of the model (not considering outliers) are characterized by MMRE = 21.7%, range -30.4%..53.6%, Pred(25) = 53.3%. The distributions of residuals and relative residuals are displayed in Figure 14.

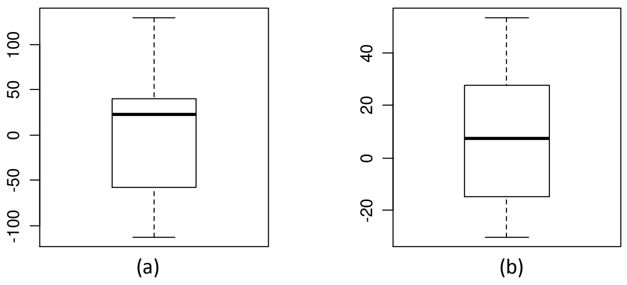

Figure 14. Effort vs. UCP and Paths/UCP: distribution of residuals (a) and relative residuals (b).

4.3.3 Effort vs. CFP and Complexity

When using CFP for measuring the functional size, the multiple linear regression analysis yields the following model (after removing projects P12, P8, P14, P9, which are considered outliers):

Effort = 2. 5967 CFP + 606.8266 Paths/CFP -236

Adjusted R^2 = 0.8278, p-value: 6.923 10^{-5}

The residuals of the model (not considering outliers) are characterized by MMRE = 12.1%, range -21.1%..29.7%, Pred(25) = 84.6%. The distributions of residuals and relative residuals are displayed in Figure 15.

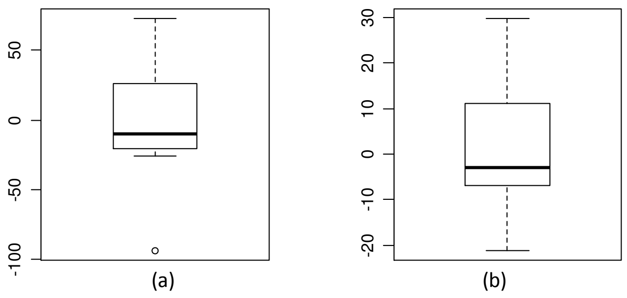

Figure 15. Effort vs. CFP and Paths/CFP: distribution of residuals (a) and relative residuals (b).

5. DISCUSSION OF RESULTS

The results of our analyses are summarized in Table 3. Some interesting indications can be derived.

A first observation deriving from the analysis of Table 3 is that the measures of functional size alone are not satisfactory predictors of the development effort. In fact, the adjusted R^2 indicates that only a fraction of the change in effort can be explained on the basis of changes in functional size. This is not really a new result: when defining COCOMO, Barry Boehm had already realized that the development effort depends on size, but also on a relatively large set of additional factors that account for the characteristics of the product and the development process.

An interesting result –already reported in [30]– is that Paths are a better than size at predicting the development effort.

However, the really new result and the main contribution if this study is that the complexity of the application –or the amount of computation required– can be used in conjunction with functional size to derive models that are more precise than those using the functional size alone. In fact, Table 3 shows that all the models of the type Effort=f(FS, Paths/FS), where FS is a measure of the functional size, are better than the corresponding models of type Effort=f(FS). Specifically, the decreased error range gives greater confidence into the estimations that take into consideration the amount of computations required.

Moreover, the model that uses CFP and Paths/CFP as independent variables outperforms the model that uses Paths alone as the independent variable. This is a quite interesting result: it indicates that software developers that aim at developing models of their productivity should collect and use historical data involving not only functional size measures, but also functional complexity measures. The good news is that both measures can be derived easily from well designed UML models, including detailed specifications of use case by means of sequence diagrams, as described in [23] and [25].

The value of R^2 suggests that there are other factors (e.g., process-related factors) that affect the estimation of effort. Of course, it is desirable that these other factors are taken into account in full-fledged effort estimation models. To this end, it would be much preferable that the relevant factors are represented as objective measures of the reality, as is the case for Paths, rather than subjective evaluations, as is the case with traditional FPA adjustment techniques.

Paths are the equivalent of McCabe's complexity for requirements. Under this point view they are definitely a measure of complexity. Nevertheless, as mentioned in Section 2, the difference between functional complexity and amount of elaboration is not at all clear, for requirements. Someone may even wonder what is the relation between Paths and functional size (since code size id often correlated with McCabe complexity [28], this relationship could hold for requirements as well).

In order to clarify the role and relationship of Paths with respect to other measures, we analyzed the correlation of various measures of size and Paths and we found that they are highly correlated (see Table 2). This seems to confirm that the relationship between size and complexity already observed at the code level [27] holds at the requirements level as well.

The COSMIC method considers acceptable not to measure explicitly the data elaboration because "the functionality of any data manipulation is assumed to be accounted for by the data movement with which it is associated" [14]. However, it is reasonable to assume that applications that involve more data manipulation require bigger development effort than applications that involve the same amount of data movements, but less data manipulation (i.e., in order to implement data manipulations, you have to write code, and writing code requires effort). Since Paths contribute to improve the effort estimation, it is reasonable to guess that they take into account the amount of data manipulation. Therefore, in COSMIC context, Paths could be seen as the ideal complement of the measure of data movements to provide a complete measure of functional processes. Of course, this hypothesis needs to be confirmed by further research.

In general, more evidence –hopefully based on a larger set of projects– is needed to clarify the relationship of Paths with the measures of size and elaboration amounts.

Table 2. Correlations of measures of Size with Paths.

	Pearson	Spearman	Kendall
CFP vs. Paths	0.78	0.90	0.77
CFP vs. Paths/CFP	N.A.	N.V.	N.V.
UFP vs. Paths	0.62	0.71	0.51
UFP vs. Paths/UFP	N.A.	N.V.	N.V.
UUCP vs. Paths	0.82	0.87	0.71
UUCP vs. Paths/UUCP	N.V.	N.V.	N.V.

As it is not advisable to build models that have inter-correlated independent variables, we did not use CFP (or UFP, or UUCP) and Paths as the independent variables of the model of Effort. Rather, we used Paths/FS, where FS is a measure of the functional size, as the second independent variable to be used together with the measure of functional size FS. As shown in Table 2, Paths/FS is not correlated with FS.

The "elaboration density" expressed as Paths/FS is adding to the model the idea of complexity per size unit. The complexity of a system is a property that depends on the relationships among the system's elements [11]. So, the measure of Paths/FS represents the density of relationships among elements per unit size.

Table 3. Summary of the models of effort found.

Model	Outliers	Adj. R^2	p-value	MMRE	Pred(25)	Error range
Effort = 1.3538 UFP + 124	5	0.5424	0.0038	23%	66.7%	-28.1% .. 115.7%
Effort = 2.1024 UCP + 126	3	0.3057	0.02353	29.5%	57.1%	-37.7%..105.3%
Effort = 2.1273 CFP + 134	2	0.372	0.009326	27.4%	53.3%	-35.2%..85.2%
Effort = 5.1769 Paths + 86	4	0.7142	0.0001687	18.1%	69.2%	-27.6%..37%
Effort = 1.6837 UFP + 516.0687 Paths/UFP - 103	5	0.7499	0.0007926	15.9%	58.3%	-33.5%.. 33.5%
Effort = 1.7802 UCP + 340.7816 Paths/UCP - 38	2	0.5114	0.005398	21.7%	53.3%	-30.4%..53.6%
Effort = 2. 5967 CFP + 606.8266 Paths/CFP -236	4	0.8278	$6.923\ 10^{-5}$	12.1%	84.6%	-21.1%..29.7%

A number of threats may exist to the validity of a correlational study like ours. Concerning the internal validity of the study, we have to notice that the models involving UFP and UCP reported in Section 4.3 have some problems, from a statistical point of view: the distribution of Paths/UFP is hardly normal, and the independent variables are mildly correlated (Spearman's ρ = -0.5); the distribution of use case points is also hardly normal. These problems are nevertheless not very worrying, since we are interested in the fact that considering the functional complexity tends to improve the models, rather than defining valid, usable models. In any case, the best model (i.e., Effort=f(CFP, Paths/CFP)) is perfectly valid from a statistical point of view.

6. RELATED WORK

A few attempts to account for data elaboration in FSM have been done. Feature points by Capers Jones [19] 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" [18]. 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 [34] 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.

Anda et al. [3] reported that the application of UCP is affected by the different aspects of the structure of the use case model, which are: the use of generalizations among actors, the use of included and extended use cases, the level of detail in the use case descriptions. Also, Anda et al. presented an experience that involved the effort estimation of a specific system based on use case points [4]. Four companies were asked to develop an application fulfilling given requirements. Each company used a different development process. The actual efforts were very different from the UCP-based effort estimates; this leads to the conclusion that the development process greatly affects costs, therefore using the historical productivity calculated in each development process seems essentially to achieve accurate estimates.

Bernárdez et al. [8] measured the cyclomatic complexity of a use case in order to validate the use case definition, while Levesque [26] measured the conditions of inputs in a sequential diagram in order to add the concept of complexity to the COSMIC method.

Mendes et al. [29] 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.

Aggarwal et al. [1] defined an estimation model which 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 [33] 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 [12] 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.

Bashir and Thomson [7] 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 are in agreement with those by Bashir and Thomson, in fact Paths variation explains more than 70% of the variation of effort; however, only the size (in CFP) and the complexity of elaboration (in Paths/CFP) appear really convincing in estimating the development effort.

Baresi and Morasca [6] analyzed the impact of attributes like the size and complexity of W2000 (a special-purpose design notation for the design of Web applications [5]) 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.

Hastings and Sajeev [16] 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.

Quite interestingly, in the parametric models that are most used in practice –like like COCOMO II [10] or SEER/SEM [15]– 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>).

7. CONCLUSIONS

In this paper we considered the problem of measuring the amount of data elaboration required from an application according to its functional user requirements. In particular, we considered the measure of Paths proposed by Robiolo et al. [30] and the possibility to use such measure in conjunction with traditional FSM methods for the purpose of development effort estimation. To this end, we measured the functional size of 17 projects according to the most popular FSM methods (function points, use case points and COSMIC function points), the Paths, and collected the development effort data. The analysis of these data yielded these main results:

– The functional size alone is not very good at explaining the amount of development effort required.
– The Paths are a much better predictor of the development effort.
– Considering the measure of Paths in conjunction with the functional size it is possible to get models that are substantially better than those using the functional size measure alone.
– The best model obtained is the one that relates the size in CFP and the “elaboration density” in Paths/CFP with the development effort. Such model outperforms both the models based on the functional size alone and the model based on Paths alone.

In conclusion, we may state that the measure of Paths appears useful to explain the amount of effort that is dedicated by developers to implementing data elaboration. Of course this indication needs to be confirmed by further evidence, which we plan to look for in future work.

Future work also includes experimenting with different regression models (e.g., using log-log transformations) in order to deal with the non-normality of distributions involving functional complexity.

Among the planned activities is also the comparison of the effort estimations yielded by the model based on CFP and Paths/CFP with the estimations obtained via traditional tools, like those implementing COCOMO II [10] or SEER/SEM [15].

Finally, we remind that the COSMIC method, although recognizing that a functional process is composed of both data movements and data elaborations, does to provide a measure of the latter. As a consequence, COSMIC FP do not represent correctly the functional size of software that is characterized by complex mathematical algorithms or by other specialized or complex rules or elaborations. Therefore, we shall explore the possibility of enhancing the definition of COSMIC FP by introducing a Paths-based measure of data elaboration in the notion of functional size.

8. ACKNOWLEDGMENTS

The research presented in this paper has been partially funded by the IST project Qualipso (http://www.qualipso.eu/), sponsored by the EU in the 6th FP (IST-034763); and the project “Elementi metodologici per la specifica, la misura e lo sviluppo di sistemi software basati su modelli” funded by the Università degli Studi dell’Insubria and Research Fund of School of Engineering of Austral University.
The authors wish to thank the anonymous reviewers who expressed constructive critics and useful suggestions.

9. REFERENCES

[1] Aggarwal, N., Prakash, N., Sofat, S. 2009. Web Hypermedia Content Management System Effort Estimation Model. *SIGSOFT Software Engineering Notes*, Volume 34, Number 2.

[2] Albrecht, A.J. 1979. *Measuring Application Development Productivity*. Joint SHARE/GUIDE/IBM Application Development Symp.

[3] Anda, B.C.D., Benestad, H.C. and Hove S.E. 2005. A multiple-Case study of effort estimation based on use case point. *Fourth International Symposium on Empirical Software Engineering, IEEE Computer Society*, 407–416.

[4] Anda, B. C. D., Dreiem, H. Sjøberg D. I. K. and Jørgensen M. 2001. Estimating Software Development Effort Based on Use Cases - Experiences from Industry. *In 4th International Conference on the Unified Modeling Language (UML2001)*, (Toronto, Canada, Springer-Verlag) *Lecture Notes in Computer Science*, 487-502.

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

[6] Baresi, L. and Morasca S. 2007. Three Empirical Studies on Estimating the Design Effort of Web Applications. *ACM Transactions on Software Engineering and Methodology*, Vol. 16, No. 4, Article 15.

[7] Bashir, H., Thomson,V. 2001. Models for estimating design effort and time. *Elsevier. Design Studies*, Vol 22, Issue, 2, 141-155.

[8] Bernárdez B., Durán A. and Genero M. 2004. Empirical Evaluation and Review of a Metrics–Based Approach for Use Case Verification. *Journal of Research and Practice in Information Technology*, Vol. 36, No. 4, 247-258.

[9] Boehm, B. W. 1981. *Software Engineering Economics*. Prentice-Hall.

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

[11] Briand L.C., Morasca S., Basili V.R. 1996. Property-Based Software Engineering Measurement. *IEEE Transactions on Software Engineering*, Vol. 22, 68 – 86.

[12] Briand,L. , Wust, J. 2001. Modeling Development Effort in Object-Oriented Systems Using Design Properties. *IEEE Transactions on Software Engineering*, vol. 27, n. 11.

[13] Cook, R. D. and Weisberg, S. 1982. *Residuals and Influence in Regression*. Chapman and Hall.

[14] COSMIC – Common Software Measurement International Consortium, 2007. *The COSMIC Functional Size Measurement Method - version 3.0 Measurement Manual* (The COSMIC Implementation Guide for ISO/IEC 19761: 2003).

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

[16] Hastings, T., Sajeev, A. 2001. A Vector-Based Approach to Software Size Measurement and Effort Estimation. *IEEE Transactions on Software Engineering*, vol. 27, n. 4.

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

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

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

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

[21] Karner, G. 1993. *Metrics for Objectory*. Diploma thesis, University of Linköping.

[22] Kitchenham, B. 1997. Counterpoint: the problem with Function Points. *IEEE Software*, Vol. 14, No. 2.

[23] Lavazza, L., del Bianco, V. and Garavaglia, C. 2008. Model-based Functional Size Measurement. *In Proceedings of 2nd International Symposium on Empirical Software Engineering and Measurement* (Incorporating ISESE and Metrics, Kaiserslautern, Germany*). ESEM 2008.*

[24] Lavazza, L. and Garavaglia, C. 2008. Using Function Point in the Estimation of Real-Time Software: an Experience. *Software Measurement European Forum (Milano). SMEF. 2008*

[25] Lavazza, L. and Robiolo, G. Introducing the Evaluation of Complexity in Functional Size Measurement: a UML-based Approach, *ESEM 2010*.

[26] Levesque G., Bevo V. and Cao, D. 2008. Estimating Software size with UML Models. *In Proceedings of the 2008 C3S2E conference, ACM International Conference Proceeding Series*, Vol. 290, 81-87.

[27] Li, F., Cheung, W.K. 1987. An Empirical Study of Software Metrics. *IEEE Transactions on Software Engineering*, 697-708.

[28] McCabe, T.J. 2005. A complexity measure. *IEEE Transactions on Software Engineering*, vol.2, n.4.

[29] Mendes, E., Mosley, N., Counsell, S., 2001. A Comparison of Length, Complexity & Functionality as Size Measures for Predicting Web Design & Authoring Effort. *Proc. of the 2001 EASE Conference*.

[30] Robiolo, G., Badano, C., Orosco, R. 2009. Transactions and Paths: two use case based metrics which improve the early effort estimation. *In Proceedings of 3rd Int. Symp. on Empirical SW Engineering and Measurement ESEM 2009* (Lake Buena Vista, Florida), 15-16.

[31] Robiolo, G. and Orosco, R. 2008. Employing use cases to early estimate effort with simpler metrics. *Innovations Syst. Softw. Eng*, vol.4.

[32] Shapiro, S. S. and Wilk, M. B. 1965. An analysis of variance test for normality (complete samples). *Biometrika 52* (3-4).

[33] Visagio, G. 1997. Structural information as a quality metric in software systems organization. *Proceedings of the International Conference on Software Maintenance*, 92-99.

[34] Whitmire, A., 1995. An Introduction to 3D Function Points, *Software Development*, Vol. 3 No.4.