A Fuzzy Model for Function Point Analysis to Development and Enhancement Project Assessments

Osias de Souza Lima Júnior
osias.lima@serpro.gov.br

Pedro Porfirio Muniz Farias
porfirio@unifor.br

Arnaldo Dias Belchior
belchior@unifor.br

Department of Computer Science, University of Fortaleza (UNIFOR)
Fortaleza, CE, Brazil

Abstract

Function Point Analysis (FPA) is among the most commonly used techniques to estimate the size of software system projects or software systems. During the point counting process that represents the dimension of a project or an application, each function is classified according to its relative functional complexity. Various studies already propose to extend FPA, mainly aimed at achieving greater precision in the point assessment of systems of greater algorithmic complexity. This work proposes the use of concepts and properties from fuzzy set theory to extend FPA to FFPA (Fuzzy Function Point Analysis). Fuzzy theory seeks to build a formal quantitative structure capable of emulating the imprecision of human knowledge. With the function points generated by FFPA, derived values such as costs and terms of development can be more precisely determined.

Keywords : *software engineering, fuzzy sets, FPA, FFPA, metrics, evaluation model*

1. Introduction

The project management is of crucial importance within software engineering. It encompasses the entire process of software development, making it comprehensible and organized, and assuming the function of coordinating the activity of personnel in the execution of processes in a disciplined and consequential manner.

Upon determining the scope of a project, the manager must stipulate, for instance, the development period and the cost of the product. Such activities are critical since unrealistic estimates here may lead to the failure of the project. The development process may suffer shifts in focus what can sacrifice quality or reduce final product functionality. All too frequently, software projects fall into worst case scenarios characterized by the bloating of schedules and costs.

In order to calculate such estimates, various methods have been developed, such as the Metric System of Halstead (Halstead, 1977), COCOMO (Boehm, 1981), PUTNAM-SLIM (Putnam, 1978) and FPA (Albrecht, 1979). FPA has attained expressive application in software project management; above all, due to its technological independence, simplicity and conciseness, being among the software metrics best practices along the year of 2001 (Kulik, 2002).

FPA begins with the decomposition of a project or application into its data and transactional functions. The data functions represent the functionality provided to the user by attending to their internal and external requirements in relation to the data, whereas the transactional functions describe the functionality provided to the user in relation to the processing this data by the application. After identifying the functions, each needs to be classified as either simple, average, or complex in accordance with its relative functional complexity, which is expressed by a certain value in points, depending on the function. Upon completing a point assessment to all functions, the application is then adjusted in accordance with the general characteristics of the system, which evaluates the general functionality of the application.

As originally conceived, FPA uses an abrupt and disjoint manner of classifying its functions. For example, an *external input* that references two files and five data items is classified as “average”, receiving four points. Another *external input* that references two files and fifteen data items is also

classified as “average”, receiving four points. In another case, when the function references two files and sixteen data items, this then is classified as “complex”, receiving six points. Two serious problems are immediately evident in such a system of classification: (i) functions of different sizes receive the same point values, and (ii) similar functions are abruptly classified into different groups. This inconsistency can become even worse when there is a large number of functions in the system that lie within the border areas of the specified intervals.

Governmental systems for instance, especially those that control tax payments or that investigate the good faith of legal entities in terms of their financial transactions, normally possess a large number of transactional functions that cross the limit of high complexity by referencing a very large number of files and data items in a single elementary process. This fact puts functions with “intuitively” different sizes into the same category. Therefore, the estimates of FPA indicate that these functions should be built into similar developmental terms and costs, which is not verified in reality.

Another key question is the software enhancement. It is impossible to develop systems, regardless of their sizes, without facing the necessity of future changes. During the software life cycle, its original requirements are modified to reflect user changes and customer needs.

Since the end of the eighties, statistics reveal that many organizations allocate at least 50% of their financial resources in software maintenance (Zitouni and Abran, 1996; Bourque and *et al.*, 1996 e April and Abran, 1995). This shows the importance of software maintenance researches. Throughout them, new methods are created or existing methodologies are adapted to adjust the activities needed to this specific process.

Many studies have already proposed extending FPA, such as FFP (FFP, 1997), mainly seeking to obtain higher precision in the point assessment of systems of greater algorithmic complexity, such as real-time systems, embedded systems, and communications systems, among others. Other extensions are specific to enhancement projects. Maya (Maya, 1996), for instance, suggests altering the structure of FPA for a purer evaluation of small functional increments.

In contrast to the conventional estimates, it appears that an increasing concern in explore techniques that may be an alternative or a complement to the existing ones, using intelligent resources, such as neural networks, case-based reasoning, regression trees, analogies and fuzzy logic (Schofield, 1995; Shepperd, 1996; Mair, 1999; Brian, 1999; Kadoda, 2000; Idri, 2000). This work proposes the utilization of concepts and properties taken from fuzzy set theory in order to extend FPA into FFPA (Fuzzy Function Point Analysis).

The primary motivation of fuzzy set theory is the desire to build a formal quantitative structure capable of capturing the imprecision of human knowledge, that is, the manner in which knowledge is expressed in natural language. This theory seeks to bridge the gap that separates traditional mathematical models needed for physical systems, and the mental representation, generally imprecise, of such systems (Dubois and Prade, 1991).

This work is organized in the following manner: section 2 focuses on fuzzy set theory, the underlying basis of the proposed model; section 3 succinctly describes FPA; section 4 describes the proposed model, FFPA, which is an extension of standard FPA that incorporates fuzzy set theory; section 5 shows the results of a case study as part of the validation process of this model; section 6 offers the fundamental conclusions of the work.

2. Fuzzy Set Theory Approaches

A fuzzy set is characterized by a membership function, which maps the elements of a domain, space or discourse universe X for a real number in $[0,1]$. Formally, $\tilde{A} : X \rightarrow [0,1]$. Thus, a fuzzy set is presented as a set of ordered pairs in which the first element is $x \in X$, and the second, $\mu_{\tilde{A}}(x)$, is the degree of membership or the membership function of x in $\tilde{A}$, which maps x in the interval $[0,1]$, or, $\tilde{A} = \{(x, \mu_{\tilde{A}}(x)) \mid x \in X\}$ [12] (Zadeh, 1965).

The membership of an element within a certain set becomes a question of degree, substituting the actual dichotomic process imposed by set theory (Pedrycz and Gomide, 1998), when this treatment is not suitable. In extreme cases, the degree of membership is 0, in which case the element is not a

member of the set, or the degree of membership is 1, if the element is a 100% member of the set (Turksen, 1991; Zimmermann, 1991).

Therefore, a fuzzy set emerges from the “enlargement” of a crisp set that begins to incorporate aspects of uncertainty. This process is called fuzzification. Defuzzification is the inverse process, that is, it is the conversion of a fuzzy set into a crisp value (or a vector of values) (Zimmermann, 1991).

Theoretically, any function in the form of $\tilde{A} : X \rightarrow [0,1]$ can be associated with a fuzzy set depending on the concepts and properties that need to be represented along with the context in which the set is inserted. Nevertheless, the literature already contains families of parameterized membership functions such as triangular, exponential, and Gauss functions, etc. Each one of these functions is characterized by a fuzzy number that is a convex and normalized fuzzy set defined in the set of the real numbers R , such that its membership function has the form [15, 24]: $\mu_{\tilde{A}} : R \rightarrow [0, 1]$. (Klir and Yuan, 1995 e Zimmermann, 1991).

Fuzzy logic is also another extension realized in boolean logic that may be considered a generalization of multi-valued logic. By modeling the uncertainties of natural language through concepts of partial truth—truth-values falling somewhere between completely true and completely false (Kantrowitz, 1997) – fuzzy logic deals with such values through fuzzy sets in the interval $[0,1]$. These characteristics allow fuzzy logic to manipulate real-world objects that possess imprecise limits. Utilizing fuzzy predicates (old, new, high etc.), fuzzy quantifiers (many, few, almost all etc.), fuzzy truth-values (completely true, more or less true) (Dubois and Prade, 1991) and generalizing the meaning of connectors and logical operators, fuzzy logic is seen as a means of approximate reasoning (Grauel, 1999).

In the next section, are the main concepts of FPA and the manner in which these concepts have been extended by fuzzy set theory.

3. Function Point Analysis

At the beginning of the 1970s, researchers at IBM initiated studies aimed at determining what critical variables were involved in programming productivity. Instead of considering code volume or complexity, they discovered that a system would be better evaluated by analyzing the functions executed by programs and mapping pertinent questions to estimating and evaluating software development productivity in heterogeneous environments (Albrecht, 1979).

According to Smith (Smith, 1997), the first definitions of FPA were refined and extended in *IBM CIS Guideline 313, AD/M Productivity Measurement and Estimate Validation*, in 1984. In 1986, a group of FPA users formed the *International Function Point User Group* (IFPUG), which is responsible for keeping their associates informed regarding any updates in technique.

FPA can be applied to calculating the size of applications, project development, or enhancement projects. Thus, the procedures to be executed are: (i) determine counting type; (ii) identify the counting scope and boundary of the application to be measured; and (iii) execute the actual calculation process itself.

The counting types may be:

- *Calculating a development project*: measure the size of a new application development project considering the requested functions and delivery to the user including those relative to the data conversion process.
- *Calculating an enhancement project*: measure a maintenance project of an existing application, considering all inclusions, changes, and exclusions of user functions and the data conversion process.
- *Calculation of an application*: measure the size of existing applications, that is, the functionality of the application available to the user from their point of view, excluding data conversion functions.

The counting scope determines what functionality will be considered as part of the point calculation. Normally, the calculation is applied to all application or project functions. The boundary

of the application is a key question in order to correctly recover the data properties that belong to the application, as well as to correlate the relationship of the application under study to external systems.

The actual calculation process itself is accomplished in three stages: (i) determine the unadjusted function points (UFP); (ii) calculate the value adjustment factor (VAF); (iii) calculate the final adjusted function points.

The first stage, determining the unadjusted function points (UFP), reflects the functionality of modules delivered to the user that they have requested and defined. The unadjusted function points include data and transactional functions. The data functions are:

- *Internal Logical File* (ILF): are groups of logically related data or control information maintained by the application itself. For instance, the file that stores the customer data of a company is an ILF for the customer database system since such a system is responsible for customer maintenance.
- *External Interface File* (EIF): are groups of logically related data or control information whose maintenance is under the responsibility of another application. For instance, the file that stores employee data of a company is an EIF for the customer database system, assuming that it accesses employee data whose actual maintenance is accomplished by the employee database system.

The transactional functions are classified in the following manner:

- *External Input* (EI): are elementary processes that involve data or control information that are input at the boundary of the application with the main objective of doing ILF maintenance. For instance, updating the personal data of a customer of an organization.
- *External Output* (EO): are elementary processes that send control information or calculated data to the end user or to other applications. For instance, a report that summarizes sales volume per quarter for a particular customer of a company.
- *External Inquiry* (EQ): are elementary processes that send control information or uncalculated data to the end user or to other applications. For instance, a personal data query operation for an employee of a company.

Once identified, each function presented must be classified according to its relative functional complexity as *low*, *average*, or *high*. The data functions' relative functional complexity is based on the number of data element types (DETs) and the number of record element types (RETs). A RET may be defined as a subset of logically related data within an ILF or an EIF. A subset can be classified as optional or mandatory in terms of its use during the process that creates the instances of data. A DET is a singular, non-repeated field recognized by the user as having its own meaning. Thus, it represents a subdivision of an ILF or an EIF. The transactional functions are classified according to the number of file types referenced (FTRs) and the number of DETs. The number of FTRs is the sum of the number of ILFs and the number of EIFs updated or queried during an elementary process.

The second stage, calculating the value adjustment factor (VAF), is an earmark of the general functionality provided to the user. The VAF is derived from the sum of the degree of influence (DI) of the 14 general system characteristics (GSCs). The DI of each one of these characteristics ranges from 0 to 5 as follows: (i) 0 – no influence; (ii) 1 – incidental influence; (iii) 2 – moderate influence; (iv) 3 – average influence; (v) 4 – significant influence; and (vi) 5 – strong influence.

The general characteristics of a system are: (i) data communications; (ii) distributed data processing; (iii) performance; (iv) heavily used configuration; (v) transaction rate; (vi) online data entry; (vii) end-user efficiency; (viii) on-line update; (ix) complex processing; (x) reusability; (xi) installation ease; (xii) operational ease; (xiii) multiple sites; and (xiv) facilitate change.

The third and last stage is the final calculation of the function points. The calculation formula varies with the type of counting. According to Eq. (1), the total points of an application, for instance, is obtained as follows:

$$AFP = UFP * VAF \quad (1)$$

where AFP = adjusted function points; UFP = unadjusted function points; and VAF = value adjustment factor.

The formula to determine the function points of an enhancement project is described in Eq. (2):

$$EFP = [(ADD + CHGA + CFP) * VAFA] + (DEL * VAFB) \quad (2)$$

where, EFP = enhancement project function points; ADD = unadjusted function points of those functions added by the project; CHGA = new unadjusted function points of those functions modified by the project, CFP = function points of the conversion functions; VAFA = value adjustment factor after project is finished; DEL = unadjusted function points of those functions deleted by the project; and VAFB = actual value adjustment factor.

After executing the enhancement, the application function points must reflect the results of the enhancement project. To do this, the Eq. (3):

$$AFP = [(UFPB + ADD + CHGA) - (CHGB + DEL)] * VAFA \quad (3)$$

where, AFP = adjusted function points of the application; UFPB = unadjusted function points of the application before the enhancement project; and CHGB = unadjusted function points of those functions modified by the enhancement project before the changes.

All definitions, rules for counting and classifying, exception handling, and practical examples that illustrate this process can be found in (FPCPM, 1999).

3.1 The Measurement of Function Points

During the measurement process, a function (data or transactional) goes through various implicit transformations (Abran and Robillard, 1996), with the final objective of obtaining a representation of its relative functional complexity expressed in function points .

However, before being expressed in points, the complexity of a function is characterized by one of the following linguistic terms: *low*, *average*, or *high*. The attribution of these terms to the functions follows [what determines] the functional complexity matrix of each function. For example, Table 1 represents the complexity matrix of an ILF, whose terms *low*, *average*, and *high* complexity correspond to 7, 10, and 15 function points, respectively:

Table 1: Complexity Matrix of an ILF

RETs	DETs		
	1 to 19	20 to 50	51 or more
1	Low	Low	Average
2 a 5	Low	Average	High
6 or more	Average	High	High

The Table 2 presents the value in function points for the terms *low*, *average* and *high* to each FPA function:

Table 2: Translation table for the terms *low*, *average* and *high*

Term	ILF	EIF	EI	EO	EQ
Low	7	5	3	4	3
Average	10	7	4	5	4
High	15	10	6	7	6

There are at least two clear situations in FPA that do not accurately translate the function points measurement process as can be observed in the data of Table 1:

- *Situation 1* (S_1): an ILF with 1 RET and 19 DETs (function f_1) is classified as *low* complexity, which translates to 7 function points. By the same criteria, an ILF with 1 RET and 50 DETs (f_2) is also classified as *low* complexity (7 function points). However, with an increment of only one more DET to the latter case, thereby increasing it to 51 DETs, the ILF (f_3) would be considered

of *average* complexity, contributing with 10 function points in the counting process. Thus, FPA considers f_1 and f_2 as identical functionalities and f_2 and f_3 as different functionalities. In the case where they are configured into the same project, the final resulting measurement will not correspond to a sufficiently accurate function points value.

- *Situation 2 (S_2)*: an ILF with 6 RETs and 20 DETs has the same number of function points as an ILF with 6 RETs and 70 DETs; that is, they have the same functionality. In such a case, the number of items referenced, which determines the lower limits of the *high* complexity range, can lead to the same measurement precision difficulties observed in the above situation, especially in systems that reference a large number of DETs.

Although function points represent the functionality of a system, many empirical studies point to an existing relationship between these points and the amount of work-effort necessary to develop them (Albrecht and Gaffney, 1983; Desharnais, 1990 e Kemerer, 1997). Measurements derived from function points, such as cost and term, may produce non-factual values as a result of the current methods of specifying the functionality of the functions that constitute the system.

Additionally, the abrupt and disjoint manner of classifying the functions avoids the application to reflect in function points the functionality that was added to it after the enhancement. For example, if an enhancement project is constituted only by one modification on an existing function, which will not change its level of complexity after the enhancement is complete, thus CHGB has the value of CHGA. According with the Equation 3, the value of AFP will remain unchanged in relation to what is already installed.

The fuzzy model, as proposed below, aims to provide a more precise approach to the function points counting process by extending FPA to FFPA while at the same time guaranteeing the validity of the final calculation of traditional function points.

4. Fuzzy Function Point Analysis (FFPA)

The central idea of extending FPA to FFPA through fuzzy set theory is to expand traditional FPA semantics by making use of the concepts and mathematical formalism of this already well-established theory.

The types of data functions (ILF and EIF) and transactional functions (EI, EO and EQ), within their respective functional complexity matrixes, can be mapped to discourse universe X , which corresponds to referenced DETs. These matrixes all use the same linguistic terms *low*, *average*, and *high*, to express their complexity. For each line of these matrixes, fuzzy numbers were generated for each of their linguistic terms.

Through experimentation, it has been shown that trapezoid-shaped fuzzy numbers best preserve FPA complexity matrix values in addition to circumventing the difficulties presented in S_1 (item 3.1).

A trapezoid-shaped fuzzy number can be represented by $\tilde{N}(a, m, n, b)$, whose membership functions are presented in Eq. 4 below. The values a and b identify the lower and upper limits respectively of the larger base of the trapezoid, where $\mu_{\tilde{N}}(x) = 0$. The values m and n are the lower and upper limits respectively of the smaller base of the trapezoid, where $\mu_{\tilde{N}}(x) = 1$, as shown in Figure 1.

$$\mu_{\tilde{N}}(x) = \begin{cases} 0, & \text{se } x < a \\ (x - a)/(m - a), & \text{se } x \in [a, m] \\ 1, & \text{se } x \in [m, n] \\ (b - x)/(b - n), & \text{se } x \in [n, b] \\ 0, & \text{se } x > b \end{cases} \quad (4)$$

Figure 1: Trapezoid-shaped fuzzy number

FFPA consists of the four following stages (Lima *et al.*, 2001a and Lima *et al.*, 2001b):

- i. Fuzzification of the linguistic terms of FPA complexity matrixes by generating fuzzy (trapezoidal) numbers;
- ii. Extend FPA complexity matrixes by generating new linguistic terms;
- iii. Determine function points values for these new linguistic terms;
- iv. Defuzzification of the linguistic terms of FFPA function points values.

First Stage

In this stage, trapezoid-shaped fuzzy numbers $\tilde{N}(a, m, n, b)$ are generated for each linguistic term T_i (*low*, *average*, *high*) belonging to the complexity matrix of the data and transactional functions. The value m_i assumes the lower limit of the linguistic term i of the complexity matrix being considered. The value n_i is calculated from the mathematical average of the values for m_i and m_{i+1} , whose result must be a rounded whole number. The values for n_{i-1} and m_{i+1} are attributed to a_i and b_i , respectively. Necessary adjustments are made when dealing with the first or last linguistic term in accordance with (4).

As an example, observe the complexity matrix of an ILF, Table 1, with a number of 2 to 5 RETs, and the linguistic term of *average*, as depicted in Fig. 2. In this case, $m = 20$ and $n = (20+51)/2 = 36$. The values for a and b are 11 and 51, respectively.

Figure 2: Trapezoid-shaped fuzzy number to the linguistic term *average* of an ILF

Second Stage

This stage is seen to minimize the problems mentioned in S_2 (subsection 3.1), which becomes all the more critical as the number of RETs and the number of FTRs increase in the data functions and in the transactional functions, respectively.

In this context, a new interval of *high* complexity was added to data and transactional functions that called for at most an interval of *average* complexity. For the same reason, a new interval of *very high* complexity was added for the remaining functions, applying the modifier *very* to the linguistic term *high*.

Due to the semantics of the new fuzzy number generated in FFPA (*very high*), the transformation operations indicated in the literature for the modifier *very* were not applied. The critical question concerning this new fuzzy number $\tilde{N}_i$ is the calculation of the most appropriate value for m_i and n_{i-1} . The value of n_{i-1} , for which the value of m_i is calculated, is the point at which the membership function will begin to lose characteristics of *high* complexity and consequently begin to acquire characteristics of *very high* complexity.

The last line of the complexity matrix of each function was the starting point for the creation of the new fuzzy number. In accordance with what has been established in (FPCPM, 1999), the functions that fall into the last two cells of the matrix are of *high* complexity. In order to maintain the use of

values used by the (FPCPM, 1999), it was decided that the number that indicates the lower limit of the third column of the matrix represents the value n of the fuzzy set of *high* complexity functions.

In an ILF, for example, the value of n_{i-1} would be 51 DETs, according to Table 1. Since the value of n_{i-1} of any given fuzzy number corresponds to the value of a_i , it follows that the value of a_i for the fuzzy number of a function of *very high* complexity would also be 51. Since the value of a_i is calculated from the mathematical average of m_i and m_{i-1} , then for an ILF, the value of $m_i = 82$, as follows:

$$(m + 20) / 2 = 51 \rightarrow m = 82$$

Table 3 presents the extension made to the values of the linguistic term *very high* to each FPA function:

Table 3: Values of m_i to the linguistic term *very high*

Function Type	T_{i-1}	n_{i-1}	m_{i-1}	T_i	m_i
ILF	High	51	20	Very High	82
EIF	High	51	20	Very High	82
EI	High	16	5	Very High	27
EO	High	20	6	Very High	34
EQ	High	20	6	Very High	34

From this point forward, the value of m_i for a fuzzy number of *very high* complexity will be referred to as k , generalizing this new linguistic term in relation to the others that already exist. The value corresponding to k must be calculated for each of the five function types belonging to FPA. Upon obtaining the value of k , for example, Table 1 (subsection 3.1) was extended (shaded area) to Table 4 below.

Table 4: An extended ILF complexity matrix

RETs	DETs			
	1 to 19	20 to 50	51 to $k-1$	k or more
1	Low	Low	Average	High
2 to 5	Low	Average	High	Very High
6 or more	Average	High	High	Very High

Taking a value of $k = 82$ as calculated above and based on the data from Table 4, the membership functions of the trapezoid-shaped fuzzy numbers are presented in the following three figures (Fig. 3, Fig. 4, and Fig. 5) for each of the three lines of the complexity matrix of an ILF, in accordance with the model depicted.

Figure 3: Membership functions to the fuzzy numbers defined for ILFs with 1 RET

Figure 4: Membership functions to the fuzzy numbers defined for ILFs from 2 to 5 RETs

Figure 5: Membership functions to the fuzzy numbers defined for ILFs with 6 or more RETs

From a historical basis of function points estimated systems, a more appropriate value for k can be arrived at, which conforms to the organization of software development in focus. This will become clearer in the case study described in section 5.

Third Stage

In FPA, p_i function points are attributed to each linguistic term T_i (*low*, *average*, and *high*) in accordance with the complexity matrix under consideration. In FFPA, these points are directly associated with the fuzzy number of the linguistic term, where $\mu_N(x) = 1$.

The value of the function points for the new linguistic term of *very high* complexity is calculated by the extrapolation of the values already defined for the terms *low*, *medium*, and *high* of each function.

Since the interpolation routines and functions may be used to extrapolation (NUMREC, 1992) and considering that the groups which represent the linguistic terms are equally spaced, it became viable to apply interpolation with finite differences, a special case of binomial interpolation in which the abscissas of the points are equidistant. Specifically, the approximant function of Newton's Formula is thusly defined (Santos, 1972):

$$p_n(u) = f_0 + \binom{u}{1} \Delta f_0 + \binom{u}{2} \Delta^2 f_0 + \dots + \binom{u}{n} \Delta^n f_0, \quad (5)$$

where u is the value corresponding to the abscissa x in the interpolator and Δ is the operator of the progressive differences. In this case, the values for the abscissas 1, 2, 3 and 4 were attributed to the terms *low*, *medium*, *high* and *very high*, respectively. The value of u is obtained by the equation below:

$$u = \frac{x - x_0}{h}, \quad (6)$$

where h is the step, in other words, the difference between the value of the two subsequent values of x . The value of u is 3 for all the data and transactional functions, since $h = 1$ (step), $x = 4$ (very high) e $x_0 = 1$ (low).

The terms Δf_i e $\Delta^n[f(x)]$ of the approximant function are calculated in the following way:

$$\Delta f_i = f_{i+1} - f_i \text{ and} \quad (7)$$

$$\Delta^n[f(x)] = \Delta^{n-1}[f(x+h)] - \Delta^{n-1}[f(x)], \quad (8)$$

where $n = 2, 3, \dots$

Applying the definitions above, the values obtained for the *fuzzy* numbers of *very high* complexity functions were estimated (Table 5). Below, follow the calculations for the data functions ILF:

Table 5: Table of the Progressive Differences - ILF

Complexity	i	f_i	Δf_i	$\Delta^2 f_i$
Low	0	7	3	2
Average	1	10	5	-
High	2	15	-	-

The value of the function points for the term *very high* of an ILF was obtained by substituting the above terms in the approximant function:

- $p_2(u) = f_0 + \binom{u}{1} \Delta f_0 + \binom{u}{2} \Delta^2 f_0$
- $p_2(3) = 7 + \binom{3}{1} \cdot 3 + \binom{3}{2} \cdot 2$
- $p_2(3) = 7 + 3 \cdot 3 + 3 \cdot 2 = 22$

In this way, an ILF of *very high* complexity equates to 22 function points. Repeating the above definitions, the values of 14, 9, 10 and 9 function points were obtained for the *fuzzy* numbers of the *very high* complexity functions for function types EIFs, EEs, EOs and EQs, respectively.

Fourth Stage

Following (LAEKWIJCK, 1999), individual criteria for defuzzification, which are not directly related to theoretical concepts and fundamentals, can be formulated when practical results of greater importance exist. In FFPA, to obtain the number of function points p_d from *fuzzy* trapezoidal numbers, where $\mu_{\tilde{N}}(x) < 1$, execute the following defuzzification process:

$$p_d = \mu_{\tilde{N}}(x) \cdot p_i + \bar{\mu}_{\tilde{N}}(x) \cdot p_{i+1} \quad (9)$$

Applying the definition above to an ILF with 1 RET and 35 DETs, we have:

$$\mu_{\tilde{N}}(35) = (51 - 35)/(51 - 26) = 0.64, \text{ and the complement } \bar{\mu}_{\tilde{N}}(35) = 0.36$$

$$p_d = 0.64 (7) + 0.36 (10) = 8.08 \text{ function points}$$

In the following section, we present some cases study based on work done at a governmental organization as part of the validation process of the proposed FFPA model.

5. Cases Study

The FFPA model is being validated on the basis of real data, which contains information on governmental systems, including development and enhancement projects. This database is mostly constituted of legacy systems developed mainly in Natural 2. Table 6 presents estimates in FPA and FFPA of some systems. The term estimates (in days) to program these systems were calculated in accordance with data supplied by Jones (Jones, 1996), considering both the level of the language used as well as the experience of the team utilizing it. D1 and D2 are development projects, while M1, M2, M3 and M4 are enhancement projects.

Table 6: FPA and FFPA estimates

1 Project	2 Standard FPA points	3 Programming term estimate (standard FPA)	4 Actual Programming term	5 Error 1 (%)	6 FFPA points	7 Programming term estimate (FFPA)	8 Error 2 (%)
D ₁	1347,80	456	510	11,84	1398,19	473	7,82
D ₂	1334,64	293	330	12,62	1442,69	317	4,10
M ₁	10,20	6	9	50,00	10,20	6	50,00
M ₂	169,65	24	30	25,00	200,06	28	7,14
M ₃	96,05	60	65	8,33	97,75	61	6,55
M ₄	96,30	42	50	19,04	115,94	51	-1,96
M ₅	12,20	13	17	30,77	12,76	14	21,43
M ₆	13,16	7	10	42,86	13,29	7	42,86
M ₇	101,65	56	70	25,00	117,70	65	7,70

In order to obtain Error 1, the column 3 is subtracted from column 4. After this, the result is divided by column 3. In the same way, to calculate column 2, the column 7 is subtracted from column 4. Then, the result is divided by column 7.

With the results obtained above, it can be noted that there was a reduction between the predicted and real time taken to develop or enhance a system when function points counting was done through FFPA. This corroborates the hypothesis that the fuzzy numbers generated better represents the functionality of an application when it possesses a large number of data or transactional functions with a large number of DETs.

From a prototype built in Java, the values attributed to k , for each function type, were successively refined as far as the model structure permits ($k = 53$ to ILF and EIF, $k = 18$ to EI and $k = 22$ to EO and EQ), to prevent fuzzy groups of high and very high complexity from agglutinating. This refinement concerns the adjustment of FFPA to the characteristics of the organization, especially for the complexity calculation of large systems, although initial values for k are suggested in the Table 3. The objective of the refinement process is to reduce the margin of error in the estimates, that is, to find a time estimate in FFPA as close as possible to the actual programming time.

The goal was to discover the combination of values for k whose *Mean Absolute Error* (MAE) for margin of error was as small as possible. In this case, the *Mean Absolute Error* corresponds to the average of the absolute values of the error percentiles, since negative values signify that the project was concluded before the predicted period, also indicating a forecasting error. Table 7 presents the MAE for different values of k .

According to the data of the Table 7, the combination of values for k equal to (53, 18 and 22) presented the smallest Mean Absolute Error (MAE). Therefore, these results indicate such values for k as indicated for use in maintenance project size estimates for this organization. However, it is worth pointing out that Project M6 strongly influenced these results since its small size gives it greater weight in percentile terms. The inclusion of new projects in the historical database may modify this scenario thereby identifying which would be the best combination for use in estimates for the organization.

Table 7: Estimates for different values of k

k			DM (%)
ILF and EIF	EI	EO and EQ	
82	27	34	16,62
74	25	31	16,28
67	25	31	16,87
67	23	31	16,55
67	23	28	16,52
61	23	28	16,32
61	21	28	16,15
61	21	25	16,10
55	21	25	16,47
55	19	25	16,36
55	19	23	16,28
53	19	23	16,62
53	18	23	14,61
53	18	22	14,60

The Figure 6 graphically presents the Values for MAE obtained from the variation in the values for k .

Figure 6: Mean Absolute Error of Estimates for different values for k

6. Conclusion

This work extended standard FPA (Function Point Analysis) to FFPA (Fuzzy Function Point Analysis), utilizing concepts and properties adopted from fuzzy set theory. By analyzing some of the results obtained through the use of FFPA, we note:

- Through the use of trapezoid-shaped fuzzy numbers for the linguistic terms *low*, *average*, and *high*, functions falling along the border areas of the intervals used receive values with a continuous graduation, without an abrupt change of those values;
- The creation of the linguistic term of *very high* complexity, pertaining to a parameterized interval through the value of k , which can be adjusted according to the characteristics of organization to better deal with larger systems;

- This model offers a more precise programming term estimate than standard FPA, especially when evaluating systems that cross the threshold of high complexity by referencing a large number of DETs and FTRs within the same elementary process;
- The model has become a more sensitive technique for modifying existing functionality and enabling the function points of an application to reflect the results of maintenance. This allows for better administration of the evolution of a system.
- Development of a prototype that automates the calculation of function points using FFPA.

The model proposed uses of a historical base of governmental systems data calculated in FPA to validate its results. This study will also be carried out using other historical bases within other application domains. It is important to point out that FFPA generates the same function points as traditional FPA for projects that have their functions with the complexity classified within non-extended intervals.

References

- Abran, A. and Robillard, P. N., 1996, "Function Points Analysis: An Empirical Study of Its Measurement Processes", IEEE Transactions on Software Engineering, vol. 22, n. 12, December.
- Albrecht, A.J., 1979, "Measuring Applications Development Productivity", Proceedings of IBM Application. Dev. Joint SHARE/GUIDE Symposium, Monterey, CA.
- Albrecht, A. J. and Gaffney, J. E., 1983, "Software Functions, Source Lines of Code, and Development Effort Prediction: A Software Science Validation", IEEE Transactions on Software Engineering, vol. 9, n. 6, November.
- April, A., Abran, A., 1995, "Industrial Research in Software Maintenance: Development of Productivity Models", Guide Summer '95 Conference and Solutions Fair, Boston.
- Belchior, A. D., 1997, "Um Modelo para Avaliação da Qualidade de Software", Doctoral Thesis, UFRJ / COPPE, May.
- Boehm, B. W., 1981, "Software Engineering Economics", Prentice-Hall.
- Bourque P., Maya M., Abram A., 1996, "A Sizing Measure for Adaptive Maintenance Work Products", IFPUG Spring Conference, Atlanta, April.
- Briand, L. Langley, T., Wiecezarek, I., 1999, "A replicated Assessment and Comparison of Common Software Cost Modeling Techniques", Technical Report.
- Desharnais, J. M., 1990, "Adjustment Model for Function Point Scope Factors – A Statistical Study", Proceedings of IFPUG Spring Conference, Florida, April.
- Dubois, D., Prade, H., 1991, "Fuzzy sets in approximate reasoning, Part 1: Inference with possibility distributions", Fuzzy Sets and Systems, IFSA, Special Memorial Volume: 25 years of fuzzy sets, North-Holland – Amsterdam.
- FFP, 1997, "Full Function Points: Counting Practices Manual", Technical Report, University of Quebec, Montreal.
- FPCPM, 1999, "Function Point Counting Practices Manual", Version 4.1, January.
- Grauel, A., 1999, "Analytical and Structural Considerations in Fuzzy Modeling", Fuzzy Sets and Systems, vol. 101 (205-206).
- Halstead, M.H., 1977, "Elements of Software Science".
- Idri, A., Abran, A., 2000, "COCOMO Cost Model Using Fuzzy Logic", presented on 7th International Conference on Fuzzy Theory & Technology, Atlantic City, New Jersey, February.
- Jones, C., 1996, "Programming Languages Table", Release 8.2, March.
- Jones, C., 2001, "Software Project Management in 2001: A Survey of the State of the Art",
- Kadoda, G. et al., 2000, "Experiences Using Case-Based Reasoning to Predict Software Project Effort", Bournemouth University, UK, March.
- Kantrowitz, M. et al., 1997, "Answers to Questions about Fuzzy Logic and Fuzzy Expert Systems", <http://www-cgi.cs.cmu.edu/afs/cs/project/ai-repository/ai/areas/fuzzy/faq/fuzzy.faq>, March.

- Kemerer, C. F., 1997, "An Empirical Validation of Software Estimation Models", *Comm. of ACM*, vol. 30, nº 5, May.
- Klir, G. J., Yuan, B., 1995, "Fuzzy Sets and Fuzzy Logic: Theory and Applications", Prentice Hall, New Jersey.
- Kulik, P., Weber, C., 2002, "Software Metrics Best Practices – 2001", ASM conference, Anaheim, USA, February.
- Laekwijck, W. V. e Kerre, E. E., 1999, "Defuzzification: criteria and classification", *Fuzzy Sets and Systems*, vol. 108 (159-178).
- Lima, O. S. J., Farias, P. P. M., Belchior, A. D., 2001, "Fuzzy Functions Points Analysis", *Fesma - Dasma*, Germany, May.
- Lima, O. S. J., Farias, P. P. M., Belchior, A. D., 2001, "Maintenance Project Assessments Using Fuzzy Function Point Analysis", *WESS*, Florence, Italy.
- Mair, C. et al., 1999, "An Investigation of Machine Learning Based Prediction Systems", Bournemouth University, UK, July.
- Maya, M., Abran A., Bourque P., 1996, "Measuring the Size of Small Functional Enhancements to Software", 6th International Workshop on Software Metrics, University of Regensburg, Germany, September.
- NUMREC, Numerical Recipes in C: The Art of Scientific Computing, 1992, Cambridge University Press.
- Pedrycz, W. and Gomide, F., 1998, "An Introduction to Fuzzy Sets – Analysis and Design", The MIT, Press.
- Putnam, L.H., 1978, "A General Empirical Solution to the Marco Software Sizing and Estimation Problem", *IEEE Transactions on Software Engineering*, Vol. SE-4, n. 4, July.
- Santos, V. R. B., 1972, "Curso de Cálculo Numérico", *Livros Técnicos e Científicos*, 3^a edition.
- Schofield, C., Shepperd, M., 1995, "Software Support for Cost Estimation by Analogy", presented at ESCOM 6, Rolduc.
- Shepperd, M., Schofield, C., 1996, "Effort Estimation by Analogy: A Case Study", presented at ESCOM 7, Wilmslow.
- Smith, L., 1997, "Function Point Analysis and Its Uses", Predicate Logic, INC.
- Teodorescu, H. N., Kandel, A. and Schneider, M., 1999, "Fuzzy Modeling and Dynamics", *Fuzzy Sets and Systems*, vol. 106 (1-2).
- Turksen, I. B., 1991, "Measurement of Membership Functions and Their Acquisition", *Fuzzy Sets and Systems*, IFSA, Special Memorial Volume: 25 Years of Fussy Sets, North Holland – Amsterdam.
- Wang P. and Tan S., 1997, "Soft Computing and Fuzzy Logic", *Soft Computing*, vol. 1 (35-41).
- Zadeh, L. A., 1965, "Fuzzy Sets, Information and Control", vol. 8 (338-353).
- Zimmermann, H. J., 1991, "Fuzzy Set Theory and Its Applications", Kluwer Boston, 2nd revised edition.
- Zitouni, M. and Abran A., 1996, "A Model to Evaluate an Improve the Quality of Software Maintenance Process", *Proceedings of Th 6th International Conference on Software Quality*, October, Ottawa, Canada.