Análise de Pontos por Função *Fuzzy*

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

Pedro Porfírio Muniz Farias (Orientador)
porfirio@unifor.br

Arnaldo Dias Belchior (Co-orientador)
belchior@unifor.br

Universidade de Fortaleza (UNIFOR)
Mestrado de Informática Aplicada (MIA / CCT)
Av. Washington Soares, 1321, Edson Queiroz
CEP 60.811-341 - Fortaleza-CE, Brasil

Resumo

A técnica de Análise de Pontos por Função (FPA) é largamente utilizada para se estimar tamanho de projetos de desenvolvimento, de manutenção ou aplicações de software já implantadas. Este trabalho propõe a utilização de conceitos e propriedades da teoria dos conjuntos fuzzy, para estender a FPA em FFPA (Análise de Pontos por Função Fuzzy). A teoria fuzzy busca construir uma estrutura formal quantitativa, capaz de capturar as imprecisões do conhecimento humano. Com os pontos por função produzidos através da FFPA, a funcionalidade do projeto ficou melhor representada. Conseqüentemente, valores derivados como prazo e custo de desenvolvimento podem ser obtidos com precisão maior do que os gerados através dos dados fornecidos pela FPA.

Palavras-chave: Conjuntos *fuzzy*, FPA, FFPA, métricas, projeto de software.

Abstract

Function Point Analysis (FPA) is a largely used technique to estimate the size of development projects, projects or applications already installed. This work proposes the use of concepts and properties from fuzzy set theory to extend FPA into 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, the functionality of the project was better represented than it was through FPA. Thus, derived values such as costs and terms of development can be determined with higher precision than the values generated from the data supplied by FPA.

Keywords: Fuzzy sets, FPA,FFPA, metrics, software design.

1. Introdução

A era Internet e o estabelecimento definitivo dos modelos de avaliação de processo e de produto fizeram surgir um novo patamar para a qualidade de software. A Internet disseminou a exigência por sistemas mais robustos e confiáveis, carecendo ainda de novas versões do produto em um curto espaço de tempo. Por sua vez, os modelos de avaliação, como CMM e a série ISO 9000, acirraram a competitividade entre as organizações [4], fazendo com que características vinculadas a nichos específicos do mercado de software passassem a compor os requisitos dos sistemas de computação, independentemente da plataforma e do público alvo a que se destinam.

Diante de tais circunstâncias, o gerenciamento de projeto adquiriu primordial importância, cuja efetividade está diretamente relacionada com a realização de medições quantitativas e qualitativas. A fim de se conhecer a dimensão do que se está gerenciando, muitos métodos foram desenvolvidos, tais como o Sistema Métrico de Halstead [11], COCOMO [6], PUTNAM-SLIM [25].

A FPA tem alcançado uma expressiva aplicação na gerência de projetos de software, sobretudo devido à sua independência tecnológica, simplicidade e concisão [1, 2]. Tem sido utilizada largamente para se estimar tamanho de projetos de desenvolvimento, de manutenção [24, 33] ou aplicações de software já implantadas.

1.1 Objetivos do Trabalho

Muitos estudos já propuseram estender a FPA, criando métodos como o FFP [9], visando, principalmente, obter uma maior precisão na pontuação de sistemas de maior complexidade algorítmica, como sistemas de tempo real, sistemas embutidos, sistemas de comunicação, entre outros. Este trabalho propõe a utilização de conceitos e propriedades da teoria dos conjuntos fuzzy, para estender a FPA em FFPA (*Fuzzy Function Point Analysis*).

A principal motivação da teoria dos conjuntos *fuzzy* é o desejo de construir uma estrutura formal quantitativa, capaz de capturar as imprecisões do conhecimento humano, isto é, como esse conhecimento é formulado na linguagem natural. Essa teoria visa ser uma ponte que une modelos matemáticos tradicionais precisos de sistemas físicos, e a representação mental, geralmente imprecisa, desses sistemas [8].

Desta forma, o modelo FFPA, além de estar em consonância com a forma como a mente humana trabalha termos lingüísticos, buscará contornar as situações adversas ocasionadas pela estrutura da FPA original. A proposta da FFPA busca dar um tratamento mais acurado ao processo de contagem dos pontos por função, estendendo a FPA em FFPA, garantindo, no entanto, a validade do cálculo final dos pontos por função tradicional.

Este trabalho está organizado da seguinte forma: na seção 2, apresenta-se a FPA na sua forma original; na seção 3, descreve-se como o modelo FFPA foi elaborado através de quatro etapas; na seção 4, estão os resultados obtidos pela aplicação do modelo FFPA sobre uma base de dados reais; finalmente, as principais conclusões da dissertação se encontram na seção 5.

2. Análise de Pontos por Função (FPA)

A FPA pode ser aplicada para calcular o tamanho de aplicações, de projetos de desenvolvimento ou de projetos de manutenção. Desta forma os procedimentos a serem executados são: (*i*) determinar o tipo de contagem (desenvolvimento ou manutenção); (*ii*) identificação do escopo de contagem e da fronteira da aplicação a ser medida; (*iii*) execução do processo de cálculo propriamente dito.

O escopo da contagem determina quais funcionalidades serão consideradas para efeito do cálculo dos pontos. Normalmente, o cálculo é aplicado para todas as funções da aplicação ou do projeto. A fronteira da aplicação é uma questão chave a fim de se recuperar corretamente a propriedade dos dados que pertencem à aplicação, bem como para se perceber o relacionamento da aplicação em estudo com sistemas externos.

O processo de cálculo propriamente dito é executado em três etapas: (*i*) a determinação dos pontos por função não-ajustados (UFP); (*ii*) cálculo do valor de ajuste (VAF); (*iii*) cálculo final dos pontos por função.

A primeira etapa, a determinação dos pontos por função não-ajustados (UFP), reflete a funcionalidade entregue ao usuário a partir dos módulos por ele requisitados e definidos. Os pontos por função não-ajustados contemplam as funções de dados e as funções transacionais. As funções de dados são:

- *Arquivo Lógico Interno* (ILF): são grupos de dados logicamente relacionados ou informações de controle que sofrem manutenção pela própria aplicação.
- *Arquivo de Interface Externa* (EIF): são grupos de dados logicamente relacionados ou informações de controle cujo processo de manutenção está sob a responsabilidade de outra aplicação.

As funções transacionais estão agrupadas da seguinte forma:

- *Entrada Externa* (EI): são processos elementares que tratam dados ou informações de controle, que entram pela fronteira da aplicação com o objetivo principal de efetuar manutenção nos ILF.
- *Saída Externa* (EO): são processos elementares que enviam informações de controle ou dados calculados para o usuário final ou outras aplicações.
- *Consulta Externa* (EQ): são processos elementares que enviam informações de controle ou dados não calculados para o usuário final ou outras aplicações.

Cada uma das funções apresentadas, após ser identificada, deve ser classificada de acordo com sua complexidade funcional relativa em *baixa*, *média* ou *alta*. As funções de dados têm sua complexidade funcional relativa baseada no número de itens de dados (DETs) e no número de tipos de registros lógicos (RETs). Um RET pode ser definido como um subgrupo de dados relacionados logicamente dentro de um ILF ou EIF.

Um DET é um campo não repetitivo, único, reconhecido pelo usuário, e que possui significado próprio. Representa, desta forma, uma subdivisão do ILF ou EIF. As funções transacionais são classificadas de acordo com o número de arquivos referenciados (FTRs) e o número de itens de dados (DETs). O número de FTRs é a soma da quantidade de ILFs e EIFs atualizados ou consultados durante um processo elementar.

A *Tabela 1*, por exemplo, apresenta a matriz de complexidade de um ILF, cujos termos *baixa*, *média* e *alta* complexidade correspondem, respectivamente, a 7, 10 e 15 pontos por função:

Tabela 1: Matriz de complexidade de um ILF

Número de registros lógicos (RET)	**Item de Dados Referenciados (DET)**		
	1 a 19	**20 a 50**	**51 ou mais**
1	BAIXA	BAIXA	MÉDIA
2 a 5	BAIXA	MÉDIA	ALTA
6 ou mais	MÉDIA	ALTA	ALTA

Através da *Tabela 1*, observa-se que FPA possui uma forma abrupta e disjunta de classificar suas funções, o que provoca pelo menos duas situações indesejáveis no processo de medição de pontos por função:

- *Situação* 1 (S_1): um ILF que possui 1 RET e 19 DETs (função f_1) é classificado como de complexidade *baixa*, o que é traduzido para 7 pontos por função. Pelo

mesmo critério, um ILF que possui 1 RET e 50 DETs (f_2) também é classificado como de complexidade *baixa* (7 pontos por função). Porém, se houver um acréscimo de apenas um DET a este último caso, passando para 51 DETs, o ILF (f_3) passa a ser considerado de complexidade *média*, contribuindo com 10 pontos por função no processo de contagem. Assim sendo, a FPA considera que f_1 e f_2 possuem funcionalidades idênticas e que f_2 e f_3 têm funcionalidades distintas. Caso isto se configure em um mesmo projeto, o resultado final de sua mensuração não corresponderá a um valor suficientemente acurado em pontos por função.

- *Situação* 2 (S_2): um ILF com 6 RETs e 20 DETs possui o mesmo número de pontos por função de um ILF com 6 RETs e 70 DETs, isto é, têm a mesma funcionalidade. Neste caso, a quantidade de itens referenciados, que determina os limites inferiores da faixa de *alta* complexidade, pode conduzir às mesmas dificuldades na acurácia da medição, observadas na situação anterior, especialmente para sistemas, que referenciam um grande número de itens de dados.

Embora pontos por função representem a funcionalidade de um sistema, muitos estudos empíricos apontam para uma relação existente entre esses pontos e o esforço de trabalho necessário para desenvolvê-los [3, 14]. Medidas derivadas a partir de pontos por função, como custo e prazo, podem estimar valores não factíveis em conseqüência deste modo atual de classificar a funcionalidade das funções que constituem o sistema.

Em sua forma original, a FPA também pode comprometer as estimativas para os projetos de manutenção. De acordo com a Tabela 1, seja a seguinte situação de manutenção:

- *Situação* 3 (S_3): um ILF que possui 1 RET e 19 DETs (função f_1) é classificado como de complexidade *baixa*, o que é traduzido para 7 pontos por função. Caso esta função sofra manutenção e passe a referenciar 50 DETs ($f_{1'}$), tal função continuará a ser classificada como de complexidade *baixa* (7 pontos por função). Este fato transmite a idéia de que não houve incremento de funcionalidade da função f_1 para a função $f_{1'}$.

A segunda etapa deste processo, o cálculo do fator de ajuste (VAF), é um assinalamento da funcionalidade geral provida ao usuário. O VAF é determinado a partir da soma dos fatores de influência de quatorze características gerais dos sistemas (GSCs), tais como comunicação de dados, desempenho, volume de transações etc. Cada uma dessas características deve ser avaliada de acordo com seu nível de influência (DI), o qual varia de 0 (sem influência) a 5 (influência forte).

A terceira e última etapa é o cálculo final dos pontos por função. A fórmula de cálculo varia com o tipo da contagem. O total de pontos de uma aplicação, por exemplo, é obtido pela multiplicação do UFP (valor encontrado na primeira etapa) pelo VAF. Todas as definições, regras de contagem e classificação, tratamento de exceções e exemplos práticos, que ilustram esse processo, podem ser encontrados em [12].

3. O Modelo FFPA

Devido a resultados insatisfatórios de técnicas de estimativas convencionais, percebe-se um interesse crescente em explorar técnicas alternativas ou complementares às existentes, como redes neurais, raciocínio baseado em casos, árvores de regressão, analogias e lógica

fuzzy [7, 21, 27, 28]. A busca por modelos alternativos também é incentivada por algumas características negativas dos modelos de regressão linear, a saber [10, 20]:

- Dificuldade em determinar os valores exatos das métricas que serão usadas como entrada;
- A utilização de valores nítidos provoca uma expectativa de que os resultados são precisos, e
- O volume de dados é pequeno ou os dados são muito heterogêneos para o desenvolvimento destes modelos.

A idéia central de se estender a Análise de Pontos por Função (FPA) para Análise de Pontos por Função *Fuzzy* (FFPA) é expandir a semântica da FPA tradicional, fazendo-se uso de conceitos e do formalismo matemático da teoria *fuzzy* já bem estabelecidos.

A teoria *fuzzy* inspira-se no modo de como o cérebro humano adquire e processa informação com baixo custo e alta eficiência [30], isto é, a forma de como a mente humana opera com conceitos subjetivos tais como *alto*, *baixo*, *velho* e *novo* (termos lingüísticos), considerando sua inclinação natural em organizar, classificar e agrupar em conjuntos, objetos que compartilham características comuns [23, 32].

Um conjunto *fuzzy* pode ser caracterizado por uma função de pertinência que mapeia todos os elementos de um domínio, espaço ou universo de discurso X para um número real em [0,1], isto é, $\tilde{A} : X \rightarrow [0,1]$. Um conjunto *fuzzy* apresenta-se como um conjunto de pares ordenados, em que o primeiro elemento é $x \in X$, e o segundo, $\mu_{\tilde{A}}(x)$, é o grau de pertinência ou a função de pertinência de x em $\tilde{A}$, que mapeia x no intervalo [0,1], ou seja, $\tilde{A} = \{(x, \mu_{\tilde{A}}(x)) \mid x \in X\}$ [29, 31].

Os tipos das funções de dados, ILF (Arquivo Lógico Interno) e EIF (Arquivo de Interface Externa) e das funções transacionais, EI (Entrada Externa), EO (Saída Externa) e EQ (Consulta Externa), em suas respectivas matrizes de complexidade funcional, podem ser mapeados no universo de discurso X, que corresponde aos Tipos de Elementos de Dados referenciados (DETs).

Como todas as matrizes de complexidade em FPA utilizam os mesmos termos lingüísticos *baixa*, *média* e *alta*, para expressarem a sua complexidade, para cada uma das linhas dessas matrizes foram gerados números *fuzzy* para cada um de seus termos lingüísticos.

Os números *fuzzy* representados pelas funções do tipo S e Γ podem ser aplicados para faixas de valores. No entanto, estas funções passam a gerar o valor 1 quando o número de DETs ultrapassa uma determinada quantidade. Este resultado é desejável apenas nas funções que pertencem ao último intervalo de complexidade. Como se deseja manter a uniformidade ao longo do modelo, aplicando uma mesma classe de funções de pertinência para todos os intervalos das matrizes de complexidade da FPA, tais funções foram descartadas.

Seguindo estes critérios, os números *fuzzy* trapezoidais foram selecionados para serem aplicados no modelo FFPA, pois se revelam apropriados para utilizar faixas de valores. Através dos números fuzzy trapezoidais, foi possível preservar os valores utilizados nas matrizes de complexidade em FPA, além de contornarem as dificuldades apresentadas nas situações apresentadas acima (S_1, S_2 e S_3).

Neste caso, um número *fuzzy* trapezoidal pode ser representado por $\tilde{N}$ (a, m, n, b), conforme a *Figura 1*. Os valores a e b identificam, respectivamente, os limites inferior e superior da base maior do trapézio, onde $\mu_{\tilde{A}}(x) = 0$. Os valores m e n são, respectivamente, os limites inferior e superior da base menor do trapézio, onde $\mu_{\tilde{A}}(x) = 1$.

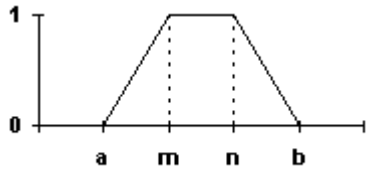

Figura 1: Número *fuzzy* trapezoidal

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

Equação 1: Número fuzzy trapezoidal

O desenvolvimento do modelo proposto para a extensão da FPA, utilizando a teoria dos conjuntos fuzzy, isto é, a Análise de Pontos por Função Fuzzy (FFPA), consistiu nas quatro etapas seguintes [16, 17, 18, 19]:

i. *Fuzificação dos termos lingüísticos das matrizes de complexidade da FPA, através da geração de números fuzzy (trapezoidais)*;
ii. *Extensão das matrizes de complexidade da FPA, gerando-se novos termos lingüísticos*;
iii. *Determinação do valor em pontos por função dos novos termos lingüísticos, gerados na etapa anterior*; e
iv. *Defuzificação dos valores dos termos lingüísticos da FFPA em pontos por função*.

Primeira Etapa

Nesta etapa, os números *fuzzy* trapezoidais $\tilde{N}$ (*a*, *m*, *n*, *b*) foram gerados para cada um dos termos lingüísticos T_i (*baixa*, *média* e *alta*) pertencente à matriz de complexidade das funções de dados e transacionais. O valor de m_i assume o limite inferior do termo lingüístico *i* da matriz de complexidade considerada. O valor de n_i é calculado a partir da média aritmética entre os valores de m_i e m_{i+1}, sendo que este resultado deve ser inteiro e arredondado. O valor de n_{i-1} e o valor de m_{i+1} foram atribuídos a a_i e b_i, respectivamente. Quando se trabalha com o primeiro ou o último termo lingüístico são feitos os devidos ajustes, segundo a *Equação 1*.

Seja o seguinte exemplo, a partir da matriz de complexidade de um ILF, Tabela 1, com 2 a 5 números de registros lógicos (RETs), considerando o termo lingüístico *média*, que pode ser observado na *Figura 2*. Neste caso, *m* = 20 e *n* = (20+51)/2 = 36. Os valores de *a* e *b* são 11 e 51, respectivamente.

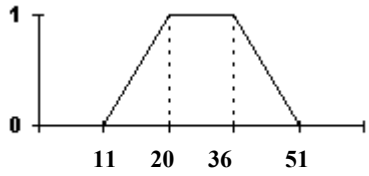

Figura 2: Número fuzzy trapezoidal para o termo lingüístico *média*

Segunda Etapa

Esta etapa veio minimizar os problemas mencionados em S_2 (seção 2), que se tornam mais críticos à medida que o número de registros lógicos (RETs) e o número de arquivos referenciados (FTRs) aumentam nas funções de dados e funções transacionais, respectivamente.

Neste contexto, acrescentou-se um novo intervalo de complexidade *alta* para as funções de dados e transacionais, que contemplavam no máximo o intervalo de complexidade *média*. Pelo mesmo motivo, adicionou-se um novo intervalo de complexidade *muito alta* para as funções restantes, aplicando-se o modificador *muito* ao termo lingüístico *alta*.

Devido à semântica do novo número *fuzzy* gerado na FPA (*muito alta*), não foram aplicadas as operações de transformação, indicadas pela literatura para o modificador *muito*. A questão crítica desse novo número *fuzzy* $\tilde{N}_i$ é o cálculo do valor mais adequado para m_i e n_{i-1}. O valor de n_{i-1}, pelo qual se calcula o valor de m_i, é o ponto a partir do qual a função de pertinência passaria a perder as características de complexidade *alta* e, conseqüentemente, começaria a adquirir as características de complexidade *muito alta*.

A última linha da matriz de complexidade de cada função foi o ponto de partida para a criação do novo número *fuzzy*. De acordo com o que está estabelecido em [12], as funções que se encaixam nas duas últimas células da matriz possuem complexidade *alta*. A fim de manter o uso dos valores utilizados pelo [12], deduziu-se a partir de experimentos realizados que o número que indica o limite inferior da terceira coluna da matriz representa o valor de *n* do conjunto *fuzzy* das funções de complexidade *alta*.

Para um ILF, por exemplo, o valor de n_{i-1} seria 51 DETs, conforme Tabela 1. Como o valor de n_{i-1} de um dado número *fuzzy* corresponde ao valor de a_i, tem-se que o valor de a_i para o número *fuzzy* da função de complexidade *muito alta* também é igual a 51. Como o valor de a_i é calculado pela média aritmética de m_i e m_{i-1}, tem-se que para um ILF, o valor de $m_i = 82$, como se segue:

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

Deste momento em diante, o valor de m_i para o número *fuzzy* de complexidade *muito alta* será referenciado como *k*, generalizando-se este novo termo lingüístico em função dos já existentes. Para cada um dos cinco tipos de função pertencentes à FPA foi o valor correspondente de *k*. Como exemplo, podem-se observar as funções de pertinência dos números fuzzy trapezoidais para a segunda linha da matriz de complexidade de um ILF na *Figura 3*.

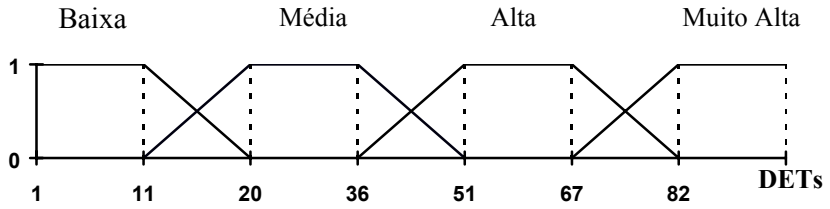

Figura 3: Funções de pertinência dos números *fuzzy* dos ILFs com 2 a 5 RETs

Partindo-se de uma base histórica de sistemas desenvolvidos através de pontos por função, pode-se chegar a um valor mais apropriado para *k*, que se amolde à instituição de desenvolvimento de software em foco. No estudo de caso realizado na seção 4, isto fica melhor evidenciado.

Terceira Etapa

Em FPA, a cada termo lingüístico T_i (*baixa*, *média* e *alta*), são atribuídos p_i pontos por função, de acordo com a matriz de complexidade considerada. Em FFPA, esses pontos são associados diretamente ao número *fuzzy* do termo lingüístico, onde $\mu_{\tilde{N}}(x) = 1$. O valor dos pontos por função do novo termo lingüístico de complexidade *muito alta* foi calculado pela extrapolação dos valores já definidos para os termos *baixa*, *média* e *alta* de cada função.

Considerando-se que as rotinas e funções de interpolação podem ser usadas para extrapolação [22], e que os conjuntos que representam os termos lingüísticos estão igualmente espaçados, tornou-se viável aplicar interpolação com diferenças finitas, caso especial da interpolação binomial em que as abscissas dos pontos são eqüidistantes. De forma particular, a função aproximante da Fórmula de Newton é assim definida [26]:

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

onde *u* é o valor correspondente à abscissa *x* na interpolar e Δ é o operador das diferenças progressivas. Neste caso, os valores das abscissas 1, 2, 3 e 4 foram atribuídos aos termos *baixa*, *média*, *alta e muito alta*, respectivamente. O valor de u é obtido através da equação abaixo:

$$\mathrm{u} = \frac{x - x_0}{h},$$

onde *h* é o passo, ou seja, a diferença de valor entre dois valores de *x* subseqüentes. O valor de *u* é 3 para todas as funções de dados e transacionais, pois *h* = 1 (passo) , *x* = 4 (*muito alta*) e x_0 = 1 (*baixa*).

Os termos Δf_i e $\Delta^n[f(x)]$ da função aproximante são calculados da seguinte forma:

$$\Delta f_i = f_{i+1} - f_i \quad \text{e} \quad \Delta^n[f(x)] = \Delta^{n-1}[f(x+h)] - \Delta^{n-1}[f(x)],$$

onde n =2, 3, ...

Aplicando-se as definições acima, os valores obtidos para os números *fuzzy* das funções de complexidade *muito alta* foram estimados. Como exemplo, segue o cálculo do valor dos pontos por função para o termo *muito alta* de um ILF:

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

Desta forma, um ILF de complexidade *muito alta* equivale a 22 pontos por função. Repetindo-se as definições acima, foram obtidos os valores 14, 9, 10 e 9 pontos por função para os números *fuzzy* das funções de complexidade *muito alta* dos tipo de função EIFs, EEs, EOs e EQs, respectivamente.

Quarta Etapa

Segundo [15], critérios próprios de defuzificação, os quais não estão diretamente relacionados com conceitos e fundamentos teóricos, podem ser formulados quando existirem resultados práticos de maior importância. Em FFPA, para se obter o número de pontos por função p_d a partir dos números *fuzzy* trapezoidais, onde $\mu_{\tilde{N}}(x) < 1$, realiza-se o seguinte processo de defuzificação:

$$p_d = \mu_{\tilde{N}}(x).p_i + \overline{\mu}_{\tilde{N}}(x).p_{i+1}$$

Aplicando a definição acima para um ILF com 1 RET e 35 DETs, obtém-se:

$\mu_{\tilde{N}}(35) = (51 - 35)/(51 - 26) = 0.64$, logo o complementar $\overline{\mu}_{\tilde{N}}(35) = 0.36$

$p_d = 0.64\ (7) + 0.36\ (10) = 8.08$ pontos por função

Na seção seguinte, serão apresentados os resultados de casos estudados em uma organização governamental, como parte do processo de validação do modelo FFPA proposto.

4. Estudos de Casos

O modelo FFPA foi validado através de uma base de dados reais, a qual contém informações de sistemas governamentais, incluindo projetos de desenvolvimento e de manutenção. Essa base é constituída, em sua maioria, por sistemas legados, implementados principalmente em *Natural 2*, *Microsoft Visual Basic*, *Microsoft Access* e *ORACLE Forms*.

Ao investigar as informações contidas da base, verificou-se que nem todos os dados eram confiáveis. A grande maioria dos projetos apresentou algum dos seguintes problemas:

- As datas de início e término do sistema não eram verídicas;
- O desenvolvimento foi interrompido durante um certo período;
- Houve modificação na equipe de desenvolvimento;
- Os requisitos foram bastante alterados ao longo do desenvolvimento etc.

Diante de tais fatos, considerou-se que apenas 9 projetos apresentavam dados que pudessem ser usados para validar o modelo FFPA, conforme o exposto na *Tabela 2*. As estimativas de prazo (em dias) para programar esses sistemas foram calculadas de acordo com os dados fornecidos por [13], considerando o nível da linguagem utilizada e a experiência da equipe no uso da mesma. Os projetos D_1 e D_2 se referem a desenvolvimento de novos sistemas, enquanto os projetos M_1, M_2, ..., M_7 são de manutenções.

Tabela 2: Estimativas em FPA e FFPA

1 Projetos	2 Pontos em FPA padrão	3 Estimativa de prazo de programação (FPA padrão)	4 Prazo real de programação	5 Erro 1 (%)	6 Pontos em FFPA	7 Estimativa de prazo de programação (FFPA)	8 Erro 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

Para se obter o Erro 1, subtrai-se a coluna 3 da coluna 4. Em seguida, divide-se o resultado pela coluna 3. De forma semelhante, para calcular o Erro 2, subtrai-se a coluna 7 da coluna 4. Em seguida, divide-se o resultado pela coluna 7.

Através dos resultados obtidos acima, percebe-se que houve uma redução entre a duração prevista e a real para desenvolver ou realizar manutenção num sistema, quando a contagem dos pontos por função foi realizado através da FFPA. Isto corrobora a hipótese de que os números *fuzzy* gerados representam melhor a funcionalidade de uma aplicação, quando esta possui um grande número de funções de dados ou transacionais com uma grande quantidade de itens de dados referenciados (DET).

A partir de um protótipo construído em Java, os valores atribuídos a k, para cada tipo de função, foram sucessivamente refinados até onde a estrutura do modelo permite (k = 53 para ILF e EIF, k = 18 para EI e k = 22 para EO e EQ), para evitar que os conjuntos fuzzy de complexidade alta e muito alta se aglutinem. Este refinamento diz respeito ao ajuste da FFPA

às características da organização, especialmente para o cálculo da complexidade de sistemas de grande porte, embora os valores iniciais para *k* possam ser calculados de acordo com a segunda etapa da seção 3.

O objetivo do processo de refinamento é reduzir a margem de erro nas estimativas, isto é, buscar um tempo estimado em FFPA o mais próximo possível do tempo real de programação da manutenção. A meta é detectar a combinação dos valores de *k*, cujo *desvio médio* (DM) das margens de erro seja o menor possível. Neste caso, o desvio médio corresponde à média dos valores absolutos dos erros percentuais, pois valores negativos significam que o projeto foi concluído antes do prazo previsto, também indicando um erro de previsão. A *Tabela 3* apresenta o desvio médio para os diversos valores de *k*.

Tabela 3: Estimativas para os diferentes valores de k

k			DM (%)
ILF e EIF	**EI**	**EO e 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

Conforme os dados da Tabela 3, a combinação de valores de k iguais a (53, 18 e 22) apresentou o menor desvio médio (DM). Portanto, estes resultados indicam tais valores de k como indicados para uso nas estimativas de tamanho dos projetos de desenvolvimento e manutenção, nesta organização. Vale salientar, porém, que o projeto M_6 influenciou fortemente este resultado, pois o seu pequeno porte lhe confere um alto peso em termos percentuais. A inclusão de novos projetos na base histórica poderá modificar este cenário, identificando qual a melhor combinação a ser usada para as estimativas da organização.

5. Conclusão

Este trabalho estendeu a FPA padrão em FFPA (Análise de Pontos por Função *Fuzzy*), utilizando conceitos e propriedades da teoria dos conjuntos *fuzzy*. Baseando-se em alguns dos resultados obtidos através da utilização da FFPA, pode-se destacar:

- através da utilização de números fuzzy trapezoidais para os termos lingüísticos *baixa*, *média* e *alta*, funções que se encontram nas regiões limítrofes dos intervalos utilizados passaram a receber valores com uma graduação contínua, sem uma mudança abrupta desses valores;

- criação do termo lingüístico de complexidade *muito alta*, pertencente a um intervalo parametrizado, através do valor de *k*, que pode ser ajustado, segundo as características da organização, para melhor lidar com os sistemas de grande porte;
- esse modelo forneceu uma estimativa de prazo de programação mais precisa que a FPA padrão, especialmente, quando se avaliou sistemas, que ultrapassaram o limite de alta complexidade, referenciando uma grande quantidade de itens de dados e arquivos em um mesmo processo elementar;
- o modelo torna a técnica mais sensível à modificação da funcionalidade existente, fazendo com que os novos pontos por função da aplicação reflitam os resultados da manutenção. Isto permite um melhor gerenciamento da evolução do sistema;
- construção de um protótipo em Java, que automatiza o cálculo de pontos por função usando FFPA.

O modelo fuzzy proposto utiliza uma base histórica de dados de sistemas governamentais calculados em FPA para a validação de seus resultados. É importante salientar que a FFPA gera os mesmos pontos por função que a FPA tradicional para os projetos que possuem suas funções com a complexidade classificada dentro dos intervalos não estendidos.

Referências Bibliográficas

[1] Abran, A., 1996, *Function Point Analysis: An Empirical Study of Its Measurement Process*, IEEE Transactions on Software Engineering, Vol. 22, nº 12.

[2] Albrecht, A.J., 1979, *Measuring Applications Development Productivity,* Proceedings of IBM Applications. Dev. Joint SHARE/GUIDE Symposium, Monterey, CA.

[3] Albrecht, A. J. e 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.

[4] April, A., Abran, A, 1995, *Industrial Research in Software Maintenance: Development of Productivity Models*, Guide Summer ’95 Conference and Solutions Fair, Boston.

[5] Belchior, A. D., 1997, *Um Modelo Fuzzy para Avaliação de Qualidade de Software*, Tese de Doutorado, COPPE/UFRJ.

[6] Boehm, B., 1981*Software Engineering Economics*, Prentice Hall.

[7] Briand, L. Langley, T., Wieczarek, I., 1999, *A replicated Assessment and Comparison of Common Software Cost Modeling Techniques*, Technical Report.

[8] 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.

[9] *Full Function Points: Counting Practices Manual*, Technical Report, University of Quebec, Montreal, 1997.

[10] Gray, A., MacDonell, S., 1997, *Applications of Fuzzy Logic to Software Metric Models for Development Effort Estimation*, Proceedings of the 1997 Annual Meeting of The North American *Fuzzy* Information Processing Society – NAFIPS, Syracuse, NY, USA.

[11] Halstead, M.H., 1977, *Elements of Software Science*, North Holland.

[12] *Function Point Counting Practices Manual*, Version 4.1, January, 1999.

[13] Jones, C., 1996, *Programming Languages Table*, Release 8.2, visitado em 14/11/2001: http://www.ciudadfutura.com/sap/sap/faq/ programming languajes_table.htm.

[14] Kemerer, C. F., 1997, *An Empirical Validation of Software Estimation Models*, Comm. of ACM, vol. 30, nº 5, May.
[15] Laekwijck, W. V. e Kerre, E. E., 1999, *Defuzzification: criteria and classification*, *Fuzzy* Sets and Systems, vol. 108 (159-178).
[16] Lima, O. S. J., Farias, P. P.M., Belchior, A. D., 2001, *Fuzzy Functions Points Analysis*, FESMA-DASMA, Germany.
[17] Lima, O. S. J., Farias, P. P.M., Belchior, A. D., 2001, *Modelo Fuzzy para Análise de Pontos por Função*, CITS, Curitiba, Brasil.
[18] Lima, O. S. J., Farias, P. P.M., Belchior, A. D., 2001, *Um Modelo Fuzzy da Análise de Pontos por Função para Estimativas de Projeto de Desenvolvimento e Manutenção de Software*, XXVII Conferência Latino-Americana em Informática, Ciudad de Mérida, Venezuela.
[19] Lima, O. S. J., Farias, P. P.M., Belchior, A. D., 2001, *Maintenance Project Assessments Using Fuzzy Function Point Analysis*, Proceedings of the 7th IEEE Workshop on Empirical Studies of Software Maintenance, Florence, Italy.
[20] MacDonell, S., Gray, A., Calvert, J., 1999, *FULSOME: Fuzzy Logic for Software Metrics Practioners and Researchers*, The Information Science Discussion Paper Series, June.
[21] Mair, C. et al., 1999, *An Investigation of Machine Learning Based Prediction Systems*, Journal of Systems and Software, July.
[22] Numrec, 1992, *Numerical Recipes in C: The Art of Scientific Computing*, Cambridge University Press.
[23] Pedrycz, W. e Gomide, F., 1998, *An Introduction to Fuzzy Sets – Analysis and Design*, The MIT, Press.
[24] Pereira, E. R., 2001, A model of factors affecting an information system's change in state, Journal of Software Maintenance end Evolution, Wiley Interscience, vol. 13 (245-261), nº 4, July-August.
[25] 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.
[26] Santos, V. R. B., 1972, *Curso de Cálculo Numérico*, Livros Técnicos e Científicos, 3ª edição.
[27] Schofield, C., Shepperd, M., 1995, *Software Support for Cost Estimation by Analogy*, Proceedings of ESCOM 6, Rolduc.
[28] Shepperd, M., Schofield, C., 1996, *Effort Estimation by Analogy: A Case Study*, Proceedings of ESCOM 7, Wilmslow.
[29] Turksen, I. B., 1991, Measurement of Membership Functions and Their Acquisition, *Fuzzy* Sets and Systems, IFSA, Special Memorial Volume: 25 Years of Fussy Sets, Amsterdam.
[30] Wang P. e Tan S., 1997, *Soft Computing and Fuzzy Logic*, Soft Computing, vol. 1 (35-41).
[31] Zadeh, L. A., 1965, *Fuzzy Sets*, Information and Control, vol. 8 (338-353).
[32] Zimmermann, H. J., 1991, *Fuzzy Set Theory and Its Applications*, Kluwer Boston, 2ª edition.
[33] Zitouni, M., Abran A., 1996, *A Model to Evaluate an Improve the Quality of Software Maintenance Process*, Proceedings of the Sixth International Conference of Software Quality, Ottawa.