ANÁLISE DE PONTOS POR FUNÇÃO FUZZY: MODELO E RESULTADOS

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

Pedro Porfírio Muniz Farias
porfirio@unifor.br

Arnaldo Dias Belchior
belchior@unifor.br

Resumo

A técnica de Análise de Pontos por Função (FPA) está entre as mais utilizadas para se estimar tamanho de projetos ou sistemas de software. Durante o processo de contagem dos pontos, que representam a dimensão do projeto ou da aplicação, cada função é classificada de acordo com sua complexidade funcional relativa. Muitos estudos já propuseram estender a FPA, objetivando, principalmente, obter uma maior precisão na pontuação de sistemas de maior complexidade algorítmica. 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, valores derivados como prazo e custo de desenvolvimento podem ser obtidos com maior precisão.

***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 project, enhancement project or applications already installed. 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. Several studies resulted in FPA extensions, and most of them are 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 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 is better represented than it was through FPA. Thus, derived values such as costs and terms of development can be more precisely determined.

***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, 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 (1977), COCOMO (BOEHM, 1981), PUTNAM-SLIM (PUTNAM, 1978) e FPA (ALBRECHT, 1979). A FPA tem alcançado uma expressiva aplicação na gerência de projetos de *software*, sobretudo devido à sua independência tecnológica, simplicidade e concisão.

A FPA inicia-se pela decomposição de um projeto ou de uma aplicação em funções de dados ou funções transacionais. As funções de dados representam a funcionalidade provida ao usuário, para atender seus requisitos internos e externos em relação aos dados. As funções transacionais descrevem a funcionalidade entregue ao usuário em relação ao processamento dos dados pela aplicação. Após a identificação das funções, a complexidade funcional relativa de cada função deve ser classificada em *baixa*, *média* ou *alta*, o que significa um determinado valor em pontos, dependendo da função. Com o término da pontuação de todas as funções, a aplicação é ajustada de acordo com as características gerais do sistema, que avaliam a funcionalidade geral da aplicação.

Originalmente, a FPA utiliza uma forma abrupta e disjunta de classificar suas funções. Por exemplo, uma entrada externa, que referencia 2 arquivos e 5 itens de dados, é classificada como *média*, recebendo 4 pontos por função. Uma outra entrada externa, que referencia 2 arquivos e 15 itens de dados, também é classificada como *média*, recebendo 4 pontos por função. Em outro caso, se a função referenciar 2 arquivos e 16 itens de dados, esta então é classificada como sendo de complexidade *alta*, recebendo 6 pontos. Nesta forma de classificação, pode-se detectar dois problemas: (*i*) funções com diferentes tamanhos receberam a mesma pontuação, e (*ii*) funções semelhantes foram classificadas em grupos diferentes abruptamente. Isto pode vir a se agravar, quando há muitas funções no sistema, que se encontram em regiões limítrofes dos intervalos utilizados.

Sistemas governamentais, por exemplo, especialmente os que controlam pagamentos de impostos e investigam a lisura de pessoas jurídicas em relação às suas transações financeiras, normalmente possuem uma grande quantidade de funções transacionais, que ultrapassam o limite da complexidade *alta*, referenciando um número muito elevado de itens de dados e arquivos em um mesmo processo elementar. Este fato dispõe, em uma mesma categoria, funções que possuem intuitivamente tamanhos diferentes. Portanto, as estimativas apontam que essas funções deveriam ser construídas dentro de prazos e custos semelhantes, o que não se verifica na realidade.

Outra questão importante é a fase de manutenção do *software*. Sabe-se que é impossível desenvolver sistemas, qualquer que seja o tamanho, sem a necessidade de efetuar mudanças no futuro. Ao longo do tempo de vida do sistema, seus requisitos originais poderão ser modificados, para refletirem mudanças de natureza evolutiva, corretiva ou legal, atendendo às necessidades dos clientes.

A partir do final dos anos 80, surgiram estatísticas revelando que muitas organizações alocam no mínimo 50% de seus recursos financeiros em manutenção de software (ZITOUNI & ABRAN, 1996; BOURQUE, MAYA & ABRAM 1995; APRIL & ABRAN, 1995). Isto revela a importância dos estudos sobre manutenção, através dos quais novos métodos são criados ou metodologias existentes são adaptadas, para contemplar as atividades necessárias a este processo.

Muitos estudos já propuseram estender a FPA, criando métodos como o FFP (UNIVERSITY OF QUEBEC, 1997), 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 (DUBOIS & PRADE, 1991).

Este trabalho está organizado da seguinte forma: a seção 2 apresenta enfoques sobre a teoria dos conjuntos *fuzzy*, que embasam o modelo proposto; a seção 3 mostra sucintamente a FPA padrão; a seção 4 descreve o modelo proposto, FFPA, que é a extensão da FPA, utilizando a teoria dos conjuntos *fuzzy*; a seção 5 fornece os resultados de casos estudados, como parte do processo de validação desse modelo; a seção 6 apresenta as principais conclusões do trabalho.

2 Aspectos sobre a Teoria dos Conjuntos *Fuzzy*

A teoria *fuzzy* inspira-se no modo de como o cérebro humano adquire e processa informação com baixo custo e alta eficiência (WANG & TAN, 1997), 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 (PEDRYCZ & GOMIDE, 1998; TEODORESCU, KANDEL & SCHNEIDER, 1999).

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, $m_{\tilde{A}}(x)$, é o grau de pertinência ou a função de pertinência de x em $\tilde{A}$, que mapeia x no intervalo [0,1], ou seja, $\tilde{A} = \{(x, \mu_{\tilde{A}}(x)) \mid x \in X\}$ (ZADEH, 1965).

A pertinência de um elemento a um determinado conjunto passa a ser uma questão de gradação, substituindo o atual processo dicotômico imposto pela teoria dos conjuntos (PEDRYCZ & GOMIDE, 1998), quando tal abordagem não for conveniente. Nos casos extremos, o grau de pertinência é 0, ocasião em que o elemento não pertence ao conjunto, ou o grau de pertinência é 1, se o elemento pertence 100% ao conjunto (TURKSEN, 1991; ZIMMERMANN, 1991).

Um conjunto *fuzzy* surge a partir do "alargamento" de um conjunto nítido, isto é, um conjunto nítido é convertido em um conjunto *fuzzy* apropriado, passando a incorporar medidas de incerteza. A este processo chama-se "fuzificação". A "defuzificação" é a operação inversa, isto é, a conversão de um conjunto *fuzzy* em um valor nítido (ou um vetor de valores) (BELCHIOR, 1997; ZIMMERMANN, 1991).

Teoricamente, qualquer função na forma $\tilde{A} : X \rightarrow [0,1]$ pode ser associada a um conjunto *fuzzy*, dependendo dos conceitos e das propriedades que deseja representar, considerando-se, ainda, o contexto no qual o conjunto está inserido. No entanto, a literatura já dispõe de famílias de funções de pertinência parametrizadas, como funções triangulares, exponenciais, de Gauss etc. Cada uma dessas funções é caracterizada por um número *fuzzy*, que é um conjunto *fuzzy* convexo e normalizado definido no conjunto dos números reais R, tal que sua função de pertinência tem a forma (KLIR & YUAN, 1995; ZIMMERMANN, 1991): $\mu_{\tilde{A}} : R \rightarrow [0, 1]$.

A lógica *fuzzy* pode ser considerada uma generalização da lógica multivalorada. Modelando as incertezas da linguagem natural, através de conceitos de verdade parcial – valores verdade entre *completamente verdade* e *completamente falso* (KANTROWITZ et al, 1997) – a lógica *fuzzy* lida com tais valores através de conjuntos *fuzzy* no intervalo [0,1]. Estas características permitem que objetos de limites imprecisos do mundo real sejam manipulados. Através de predicados *fuzzy* (velho, novo, alto etc.), quantificadores *fuzzy* (muito, pouco, etc.), valores-verdade *fuzzy* (totalmente verdadeiro, mais ou menos verdadeiro) (DUBOIS & PRADE, 1991) e generalizando o significado de conectivos e operadores lógicos, essa lógica revela-se como um modo de raciocínio aproximado (GRAUEL, 1999).

Na próxima seção, serão abordados alguns conceitos clássicos de FPA e a utilização da teoria dos conjuntos *fuzzy* para a extensão desses conceitos.

3 Análise de Pontos por Função

No início da década de 70, pesquisadores da IBM iniciaram estudos para determinar que variáveis críticas poderiam determinar a produtividade na programação. Em vez de considerar o volume ou a complexidade do código, descobriram que um sistema seria melhor avaliado através das funções executadas pelos programas, mapeando questões pertinentes à estimativa e à avaliação de produtividade no desenvolvimento de *software* em ambientes heterogêneos (ALBRECHT, 1979).

Segundo SMITH (1997), as primeiras definições da FPA foram refinadas e estendidas no *IBM CIS Guideline 313, AD/M Productivity Measurement and Estimate Validation*, em 1984. Em 1986, um grupo de usuários da FPA formou o *International Function Point User Group* (IFPUG), o qual é responsável por manter seus associados informados a respeito das novas atualizações da técnica.

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; (*ii*) identificação do escopo de contagem e da fronteira da aplicação a ser medida; (*iii*) execução do processo de cálculo propriamente dito.

Os tipos de contagem podem ser:

- *Cálculo de um projeto de desenvolvimento*: mensura o tamanho de um projeto de desenvolvimento de uma nova aplicação, considerando as funções solicitadas e entregues ao usuário, incluindo aquelas relativas ao processo de conversão de dados.
- *Cálculo de um projeto de manutenção*: dimensiona um projeto de manutenção de uma aplicação já existente, contabilizando todas as inclusões, alterações e exclusões de funções do usuário e do processo de conversão de dados.
- *Cálculo de uma aplicação*: medida do tamanho de aplicações já existentes, isto é, a funcionalidade da aplicação que está disponível para o usuário, sob o seu ponto de vista, não sendo incluídas as funções de conversão de dados.

O escopo da contagem determina que 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. Por exemplo, o arquivo que armazena os clientes de uma empresa é um ILF para o sistema de cadastro de clientes, considerando que tal sistema é o responsável pela manutenção dos clientes.
- *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. Por exemplo, o arquivo que armazena os funcionários de uma empresa é um EIF para o sistema de cadastro de clientes, supondo que este acesse os dados dos funcionários, mas que a manutenção dos mesmos é realizada pelo sistema de cadastro de funcionários.

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. Por exemplo, a atualização dos dados pessoais de um cliente de uma organização.
- *Saída Externa* (EO): são processos elementares que enviam informações de controle ou dados calculados para o usuário final ou outras aplicações. Por exemplo, um relatório que condense o volume de compras, por trimestre, de um determinado cliente da empresa.
- *Consulta Externa* (EQ): são processos elementares que enviam informações de controle ou dados não calculados para o usuário final ou outras aplicações. Por exemplo, uma operação de consulta aos dados pessoais de um funcionário da empresa.

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 subgrupo pode ser categorizado como opcional ou obrigatório em relação a seu uso durante o processo, que cria as instâncias de dados. 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 da quantidade de EIFs atualizados ou consultados durante um processo elementar.

A segunda etapa, 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). Cada uma dessas características deve ser avaliada de acordo com seu nível de influência (DI), o qual varia de 0 a 5, como se segue: (*i*) 0 – sem influência; (*ii*) 1 – influência incidental; (*iii*) 2 – influência moderada; (*iv*) 3 – influência média; (*v*) 4 – influência significativa; e (*vi*) 5 – influência forte.

As características gerais de um sistema são: (*i*) comunicação de dados; (*ii*) processamento distribuído; (*iii*) desempenho; (*iv*) utilização do equipamento; (*v*) volume de transações; (*vi*) entrada de dados on-line; (*vii*) eficiência do usuário final; (*viii*) atualização on-line; (*ix*) processamento complexo; (*x*) reutilização de código; (*xi*) facilidade de implantação; (*xii*) facilidade operacional; (*xiii*) múltiplos locais; e (*xiv*) facilidade de mudanças.

A terceira e última etapa é o cálculo final dos pontos por função. A fórmula de cálculo varia com o tipo da contagem, conforme Eq. (1). O total de pontos de uma aplicação, por exemplo, é:

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

onde UFP é o total de pontos não-ajustados (valor encontrado na primeira etapa) e VAF é o fator de ajuste.

A fórmula para determinar os pontos por função de um projeto de manutenção está descrita na Eq. (2) (ALBRECHT & GAFFNEY, 1983):

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

Assim sendo, EFP = pontos por função de um projeto de manutenção; ADD = pontos por função não-ajustados das funções adicionadas pelo projeto; CHGA = novos pontos por função não-ajustados das funções modificadas pelo projeto; CFP = pontos por função das funções de conversão; VAFA = novo fator de ajuste; DEL = pontos por função das funções eliminadas da aplicação; e VAFB = fator de ajuste atual.

Após a execução da manutenção, os pontos por função da aplicação devem refletir os resultados do projeto de manutenção. Para isso, utiliza-se a Eq. (3):

AFP = [(UFPB + ADD + CHGA) – (CHGB + DEL)] * VAFA (3)

onde, AFP = novos pontos por função ajustados de uma aplicação; UFPB = pontos por função não ajustados da aplicação antes da manutenção; e CHGB = pontos por função não-ajustados das funções modificadas pelo projeto, antes da manutenção.

Todas as definições, regras de contagem e classificação, tratamento de exceções e exemplos práticos, que ilustram esse processo, podem ser encontrados em (IFPUG, 1999).

3.1 A Medição de Pontos por Função

Durante o processo de medição, uma função (de dados ou transacional) passa por várias transformações implícitas (ABRAN & ROBILLARD, 1996), com o objetivo final de se obter sua complexidade funcional relativa representada através de pontos.

Porém, antes de ser expressa através de pontos, a complexidade de uma função é caracterizada por um dos seguintes termos lingüísticos: *baixa*, *média* ou *alta*. A atribuição destes termos às funções segue o que determina a matriz de complexidade funcional de cada função. Como exemplo, a Tabela (1) apresenta a matriz de complexidade de um ILF, cujos termos *baixa*, *média* e *alta* complexidade correspondem, respectivamente, aos valores 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

Há pelo menos duas situações visíveis em FPA, que não traduzem de forma mais acurada, o processo de medição de pontos por função, que se pode observar através dos dados da Tabela 3.1:

- *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 (ALBRECHT & GAFFNEY, 1983; DESHARNAIS, 1990; KEMMERER, 1987). Medidas derivadas a partir de pontos por função, como custo e prazo, podem estimar valores não factíveis em conseqüência deste modo atual de especificar a funcionalidade das funções que constituem o sistema.

Adicionalmente, o modo corrente de classificar as funções impede que a aplicação reflita em pontos por função a funcionalidade que lhe foi adicionada após a manutenção. Por exemplo, se o projeto de manutenção for constituído apenas

de uma modificação de uma função já existente, a qual não terá seu nível de complexidade alterado após a manutenção, tem-se que CHGB é igual a CHGA. Pela Equação 3.3., o valor AFP permanece inalterado em relação ao que já está instalado.

O modelo *fuzzy*, proposto a seguir, 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.

4 Análise de Ponto por Função *Fuzzy* (FFPA)

A idéia central de se estender a FPA para FFPA através da teoria dos conjuntos *fuzzy* foi expandir a semântica da FPA tradicional, fazendo-se uso de conceitos e do formalismo matemático dessa teoria já bem estabelecidos.

Os tipos das funções de dados (ILF e EIF) e das funções transacionais (EI, EO e EQ), em suas respectivas matrizes de complexidade funcional, podem ser mapeados no universo de discurso X, que corresponde aos itens de dados referenciados (DETs). Todas essas matrizes utilizam os mesmos termos lingüísticos *baixa*, *média* e *alta*, para expressarem a sua complexidade. Para cada uma das linhas dessas matrizes, geraram-se números fuzzy para cada um de seus termos lingüísticos.

Após os experimentos realizados, verificou-se que os números *fuzzy* trapezoidais são os que melhor preservam os valores utilizados nas matrizes de complexidade na FPA, além de contornar as dificuldades apresentadas em S_1 (item 3.1).

Um número *fuzzy* trapezoidal pode ser representado por $\tilde{N}(a, m, n, b)$, cujas funções de pertinência são apresentados na Equação (4). 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 $\mu_{\tilde{A}}(x) = 1$, conforme Figura 4.1.

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

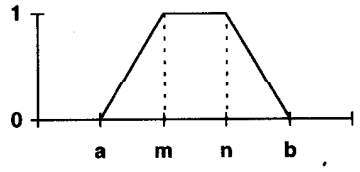

Figura 4.1: Número *fuzzy* trapezoidal

A FFPA consiste nas quatro etapas seguintes (LIMA, FARIAS & BELCHIOR, 2001a; LIMA, FARIAS & BELCHIOR, 2001b):

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 desses novos termos lingüísticos;

iv. defuzificação dos valores dos termos lingüísticos da FFPA em pontos por função.

Primeira Etapa

Nesta etapa, os números *fuzzy* trapezoidais foram gerados para cada termo lingüístico 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 (4).

Seja o seguinte exemplo, a partir da matriz de complexidade de um ILF, Tabela 3.1, com 2 a 5 números de registros lógicos (RETs), considerando o termo lingüístico *média*, que pode ser observado na Figura 4.3. Neste caso, $m = 20$ e $n = (20+51)/2 = 36$. Os valores de a e b são 11 e 51, respectivamente.

Segunda Etapa

Esta etapa veio minimizar os problemas mencionados em S_2 (subseção 3.1), 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 (GRAUEL, 1999), as funções que se encaixam nas duas últimas células da matriz possuem complexidade *alta*. A fim de manter o uso dos valores utilizados pelo (GRAUEL, 1999), decidiu-se 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 3.1. 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, para simplificar a condução deste trabalho. Para cada um dos cinco tipos de função pertencentes à FPA deverá ser calculado o valor correspondente de k. A partir da obtenção do valor de k, por exemplo, a Tabela 3.1 (subseção 3.1) foi estendida (parte sombreada) para a Tabela 4.1 abaixo.

Tabela 4.1: Matriz de complexidade de um ILF estendida

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

Tomando-se o valor de $k = 82$, conforme calculado acima, a partir dos dados da Tabela 4.1, são apresentadas, nas três figuras a seguir (Figura 4.2, Figura 4.3 e Figura 4.4), as funções de pertinência dos números *fuzzy* trapezoidais, para cada uma das três linhas da matriz de complexidade de um ILF, segundo o modelo exposto.

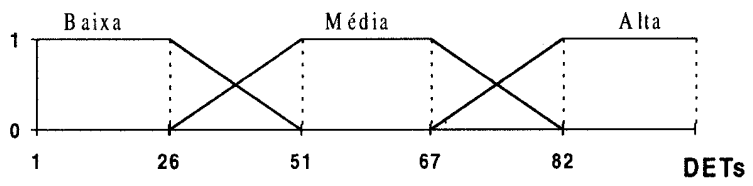

Figura 4.2: Funções de pertinência dos números *fuzzy* definidos para os ILFs com 1 RET

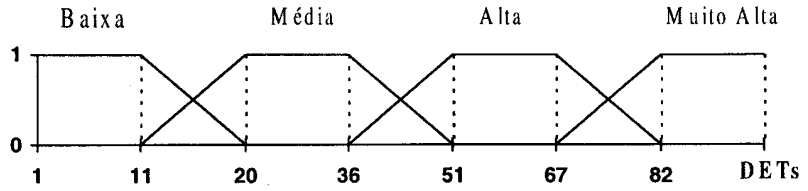

Figura 4.3: Funções de pertinência dos números *fuzzy* definidos para os ILFs com 2 a 5 RETs

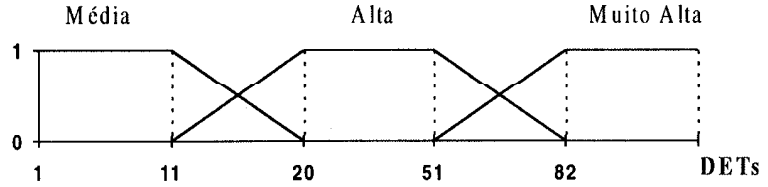

Figura 4.4: Funções de pertinência dos números *fuzzy* definidos para os ILFs com 6 ou mais RETs

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 5, isto fica melhor evidenciado.

Terceira Etapa

Em FPA, a cada termo lingüístico t_i, dos *n* termos utilizados (*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 $\mu_{\tilde{N}}(x) = 1$. A partir destes dados, obtém-se o valor do ponto por função p_m do novo termo lingüístico (*muito alta*), como se segue:

- $x_i = p_{i+1} - p_i$ (5)
- $r = x_{i+1} - x_i$ (6)
- $x_n = x_1 + (n-1) \cdot r$ (7)
- $p_m = x_n + p_n$ (8)

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

Quarta Etapa

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".

$$p_d = \mu_{\tilde{N}}(x).p_i + \overline{\mu}_{\tilde{N}}(x).p_{i+1} \qquad (9)$$

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.

5 Estudos de Casos

O modelo FFPA está sendo 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, desenvolvidos principalmente em *Natural 2*. A Tabela 5.1 apresenta estimativas em FPA e FFPA de alguns desses sistemas. As estimativas de prazo (em dias) para programar esses sistemas foram calculadas de acordo com os dados fornecidos por JONES (1996), 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 se constituem em manutenções.

Tabela 5.1: Estimativas em FPA e FFPA

Projetos	Pontos em FPA padrão	Estimativa de prazo de programação (FPA padrão)	Prazo real de programação	Erro (%)	Pontos em FFPA	Estimativa de prazo de programação (FFPA)	Erro (%)
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

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 realizada 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 (ILF/EIF), k = 18(EI) e k = 22(EO/EQ)), com o objetivo de 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. A Tabela 5.2 a seguir apresenta o desvio médio para os diversos valores de k.

Tabela 5.2: Estimativas para os diferentes valores de k

k			DM (%)
ILF – EIF	EI	EO - 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	14,55

Conforme os dados da Tabela 5.2, a combinação de valores de *k* iguais a (53, 18 e 23) 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. A Figura 5.1 apresenta graficamente os desvios médios obtidos em função da variação dos valores de *k*.

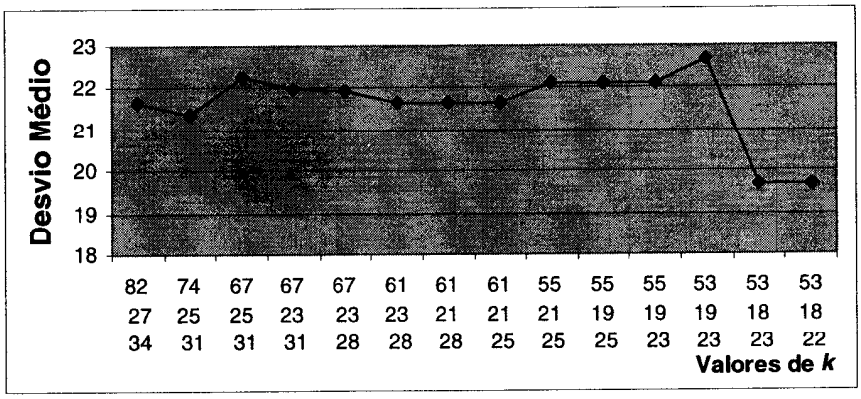

Figura 5.1: Desvio Médio das Estimativas para diferentes valores de *k*

6 Conclusão

Este trabalho, utilizando conceitos e propriedades da teoria dos conjuntos *fuzzy*, estendeu a FPA padrão (Análise de Pontos por Função) em FFPA (Análise de Pontos por Função *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 pontos por função da aplicação reflita os resultados da manutenção. Isto permite um melhor gerenciamento da evolução do sistema;
- construção de um protótipo, que automatiza o cálculo de pontos por função usando FFPA.

O modelo proposto está sendo refinado, através da utilização de uma grande base histórica de sistemas governamentais calculados em FPA. Este estudo será conduzido também com a utilização de outras bases históricas em outros domínios de aplicação.

Referências

ABRAN, A.; ROBILLARD, P. N. Function points analysis: an empirical study of its measurement processes. *IEEE Transactions on Software Engineering*, Los Alamitos, v. 22, n. 12, p. 895-910, Dec. 1996.

ALBRECHT, A.J. Measuring applications development productivity. In: IBM APPLIC. DEV. JOINT SHARE/GUIDE SYMPOSIUM, 1979, Monterey. *Anais...* Monterey: IBM, 1979, p. 10-25.

ALBRECHT, A. J.; GAFFNEY, J. E. Software functions, source lines of code, and development effort prediction: a software science validation. *IEEE Transactions on Software Engineering*, Los Alamitos, v. 9, n. 6, p. 639-648, Nov. 1983.

APRIL, A.; ABRAN, A. Industrial research in software maintenance: development of productivity models. In: GUIDE SUMMER'95 CONFERENCE AND SOLUTIONS FAIR. 1995, Boston. *Anais...* Boston: Quality & Research, 1995, p. 5-43.

BELCHIOR, A. D. *Um modelo para avaliação da qualidade de software*. 1997. 185 f. Tese (Doutorado em Ciências da Computação) - COPPE, Universidade Federal do Rio de Janeiro, Rio de janeiro.

BOEHM, B. W. *Software engineering economics*. Englewood Cliffs: Prentice-Hall, 1981, 187 p.

BOURQUE, P.; MAYA M.; ABRAM, A. A sizing measure for adaptative maintenance work products. In: IFPUG SPRING CONFERENCE, 1995, Nice. *Anais...* Nice: IEEE, 1995, p. 286-294.

DESHARNAIS, J. M. Adjustment model for function point scope factors – a statistical study. In: IFPUG SPRING CONFERENCE, 1990, Orlando. *Anais...* Orlando; IFPUG, 1990, p. 22-35.

DUBOIS, D.; PRADE, H. Fuzzy sets in approximate reasoning, Part 1: Inference with possibility distributions. *Fuzzy Sets and Systems*, Amsterdam, v. 40, n. 8, p. 143-202, Aug. 1991.

GRAUEL, A. Analytical and structural considerations in fuzzy modeling. *Fuzzy Sets and Systems*, Oxford, v. 101, n. 5, p. 205-206, June 1999.

HALSTEAD, M.H. *Elements of software science*. New York: Elsevier, 1977.

IFPUG. FPCPM - Function point counting practices manual. Westerville, 1999, 141 p.

JONES, C. Programming languages table. Software Productivity Research Inc., mar. 1996. Disponível em: <http:/ www.theadvisors.com/langcomparison.htm>. Acesso em: 03 maio 2002.

KANTROWITZ, M. *et al.* Answers to questions about Fuzzy logic and Fuzzy expert systems. Carnegie Mellon, School of Computer Science, Pittsburgh, mar. 1997. Disponível em: < http://www-cgi.cs.cmu.edu/afs/cs/project/ai-repository/ ai/areas/fuzzy/faq/ fuzzy.faq>. Acesso em: 03 maio 2002.

KEMMERER, C. F. An empirical validation of software cost estimation models. *Comm. of ACM*, New York, v. 30, n. 5, p. 416-429, May 1987

KLIR, G. J.; YUAN, B. *Fuzzy sets and Fuzzy logic: theory and applications*. New Jersey: Prentice Hall, 1995, 155 p.

LIMA, O. S. J.; FARIAS, P. P. M.; BELCHIOR, A. D. Fuzzy functions points analysis. In: EUROPEAN CONFERENCE ON SOFTWARE MEASUREMENT AND ICT CONTROL, 4., 2001, Heildeberg. *Anais...* Heildeberg: FESMA-DASMA, 2001, p.161-172.

LIMA, O. S. J.; FARIAS, P. P. M.; BELCHIOR, A. D. Modelo Fuzzy para análise de pontos por função. In: CONFERENCIA INTERNACIONAL DE TECNOLOGIA DE SOFTWARE, 12., 2001, Curitiba. *Anais...* Curitiba: Centro Internacional de Tecnologia de Software, 2001, p. 96-109.

PEDRIYCZ, W.; GOMIDE, F. *An introduction to Fuzzy sets – analysis and design*. London: The MIT Press, 1998, 465 p.

PUTNAM, L.H. A general empirical solution to the marco software sizing and estimation problem. *IEEE Transactions on Software Engineering*, Los Alamitos, v. 4, n. 4, p. 345-361, July 1978.

SMITH, L. Function point analysis and its uses. Predicate Logic, INC., San Diego, 1997. Disponível em: <http:// www.predicate.com/up-fp.html>. Acesso em 03 maio 2002.

TEODORESCU, H. N.; KANDEL, A.; SCHNEIDER, M. Fuzzy modeling and dynamics. *Fuzzy Sets and Systems*, Oxford, v. 106, n. 20, p. 1-2, Jan. 1999.

TURKSEN, I. B. Measurement of membership functions and their acquisition. *Fuzzy Sets and Systems*, Amsterdam, v. 40, n. 8, Aug. 1991.

UNIVERSITY OF QUEBEC. FFP - Full function points: counting practices manual. Montreal, 1997.

WANG, P.; TAN, S. Soft computing and Fuzzy logic. *Soft Computing*, Berlin, v. 1, n. 4, p. 35-41, Apr. 1997.

ZADEH, L. A. Fuzzy sets, information and control. *Fuzzy Sets and Systems*, Oxford, v. 8, n. 5, p. 338-353, May. 1965.

ZIMMERMANN, H. J. *Fuzzy set theory and its applications*. Boston: Kluwer, 1991, 399 p.

ZITOUNI, M.; ABRAN A. A model to evaluate an improve the quality of software maintenance process. In: INTERNATIONAL CONFERENCE ON SOFTWARE QUALITY, 6., 1996, Ottawa. *Anais...* Ottawa: ASQC´S Software Division, 1996, p. 238-258.

Osias Souza Lima Júnior
Graduado em Informática pela Universidade de Fortaleza em 1996. Mestre em Informática Aplicada pela Universidade de Fortaleza em 2002. Atualmente ocupa o posto de Analista de Informática no SERPRO.

Pedro Porfírio Muniz Farias
Graduado em Ciências da Computação pela Universidade Estadual do Ceará em 1988. Mestre em engenharia de Sistemas da Computação pelo Instituto Militar de Engenharia em 1991. Doutor em Engenharia de Sistemas da Computação pela Universidade Federal do Rio de Janeiro em 1998. Atualmente ocupa o posto de Professor Titular junto à Coordenação do Curso de Informática e ao Mestrado em Informática Aplicada da Universidade de Fortaleza.

Arnaldo Dias Belchior
Graduado em Egenharia Civil pela Universidade Federal do Ceará em 1983. Mestre em Engenharia de Sistemas da Computação pela Universidade Federal do Rio de Janeiro em 1992. Doutor em Engenharia de Sistemas da Computação pela Universidade Federal do Rio de Janeiro em 1997. Atualmente ocupa o posto de Professor Titular junto à Coordenação do Curso de Informática e ao Mestrado em Informática Aplicada da Universidade de Fortaleza.