Revista Eletrônica de Sistemas de Informação
ISSN 1677-3071

v. 11, n. 1
jan-jun 2012

doi:10.5329/RESI.2012.1101

Sumário

ISSN: 1677-3071

Esta revista é (e sempre foi) eletrônica para ajudar a proteger o meio ambiente, mas, caso deseje imprimir esse artigo, saiba que ele foi editorado com uma fonte mais ecológica, a *Eco Sans*, que gasta menos tinta.

APLICAÇÃO DE LÓGICA *FUZZY* NA ESTIMATIVA DE PRAZO DE PROJETOS DE SOFTWARE

FUZZI LOGIC APPLICATION IN SOFTWARE PROJECTS' TIME ESTIMATION

(artigo submetido em julho de 2011)

Rúbia Eliza de O. Schultz Ascari
Professora do Curso de Informática
Universidade Tecnológica Federal do Paraná
(UTFPR) - Campus Pato Branco
rubia@utfpr.edu.br

Beatriz Terezinha Borsoi
Professora do Curso de Informática
Universidade Tecnológica Federal do Paraná
(UTFPR) - Campus Pato Branco
beatriz@utfpr.edu.br

Kathya Silvia Collazos Linares
Professora do Curso de Informática
Universidade Tecnológica Federal do Paraná
(UTFPR) - Campus Pato Branco
kathya@utfpr.edu.br

Luiz Fernando Toscan
Aluno do Curso de Informática
Universidade Tecnológica Federal do Paraná
(UTFPR) - Campus Pato Branco
kathya@utfpr.edu.br

ABSTRACT

Estimation of the time required for software development is a difficult task of software project development. Function Point Analysis (FPA) is among the most commonly used techniques to estimate the size of software system projects or software systems. Several studies have already proposed 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 of fuzzy logic to combine factors to adjust the time estimated to implement requirements of software systems. A software system was developed to automate the time estimation that is adjusted using factors combined as fuzzy rules.

Keywords: software estimation; function point analysis; fuzzy logic; software metrics.

RESUMO

Estimar o tempo necessário para o desenvolvimento de projeto de *software* é uma tarefa difícil. A Análise de Pontos de Função (APF) está entre as técnicas mais comumente utilizadas para estimar o tamanho de projetos de sistemas de *software*. Vários estudos já propuseram extensões à APF, principalmente visando a maior precisão dos seus resultados para sistemas com complexidade algorítmica maior. Este trabalho propõe o uso de conceitos de lógica *fuzzy* para combinar fatores para ajustar o tempo estimado para implementar um sistema de *software*. Um *software* foi desenvolvido para automatizar a estimativa de prazo que é ajustado pelo uso de fatores combinados em regras *fuzzy*.

Palavras-chave: estimativa de *software*; análise de pontos de função; lógica *fuzzy*; métricas de software.

1 INTRODUÇÃO

Alfred Pietrasanta do Instituto de Pesquisa em Sistemas da IBM (*International Business Machines*) argumentou em 1968 que, na época, não havia uma solução rápida e fácil para o problema de estimativa de recursos (LAIRD, 2006). Três décadas mais tarde, Briand *et al.* (1999) observaram que apesar do grande número de fatores de custo identificados e considerados e da possibilidade de uma base de dados ampla e bem organizada disponível, muitas incertezas na estimativa de custo de *software* persistiam. Em 2001, Lewis (2001) se referia à estimativa como um dos problemas práticos mais importantes da Engenharia de *Software*. Em 2006, Laird (2006) acrescentou que há limitações na habilidade de estimar com precisão, devido às incertezas inerentes aos projetos de *software*. E, mais recentemente, Liao (2009) observou que a complexidade do *software*, as experiências históricas deficientes, a falta de ferramentas e os erros humanos fazem com que sejam obtidas estimativas de projeto de *software* ainda muito distantes da realidade.

Estimativas em projetos de *software* podem ser utilizadas para apoiar o planejamento do projeto pelo suporte na definição de custos, de recursos e de prazo necessários e do valor do *software* produzido. Assim, encontrar solução aos problemas relacionados às estimativas ou mesmo melhorar soluções existentes representam maneiras de contribuir com as fábricas de *software* para que seus projetos sejam concluídos no prazo, orçamento e qualidade planejados.

Albrecht (1979) define três critérios básicos para avaliar projetos de desenvolvimento de *software*: devem ser finalizados no tempo estipulado, com o custo definido e satisfazer o cliente, no sentido de o sistema atender aos seus interesses e expectativas. O cliente estipula os objetivos (ou requisitos) que o sistema deve atender e o projeto de desenvolvimento deve fazer com que esses objetivos sejam alcançados dentro do prazo e do custo estipulados e com a qualidade requerida.

Muitas técnicas (incluindo as denominações procedimento, padrão, métrica, modelo, metodologia) como, por exemplo, pontos de função (FPCPM, 1999) têm como base o esforço necessário para desenvolver o *software* que é utilizado para determinar prazo, custo e qualidade, dentre outros. Os custos também podem ser determinados a partir do prazo necessário para desenvolver o *software*. O cronograma, que expressa o prazo, permite ainda definir marcos de verificação durante a realização do projeto.

Quanto ao momento de realizar a estimativa, o indicado é que a fase de projeto (do ciclo de vida clássico proposto em Pressman (2005)) esteja concluída, ou que pelo menos haja um esboço razoavelmente claro dessa etapa (MELLER, 2002). Diferente do proposto em Pressman (2005) e Meller (2002), neste trabalho pontos de função são utilizados na fase de requisitos, pois considera-se que, quanto mais no início do ciclo de vida do *software* a estimativa for realizada, mais útil ela poderá ser. Isso se justi-

ficaria pelo seu uso no planejamento do projeto, na definição do contrato com o cliente e na alocação de recursos, por exemplo. Contudo, quanto mais avançado estiver o desenvolvimento do *software*, maior é a probabilidade de a estimativa ser mais precisa. Isso porque os dados utilizados para realizar a estimativa estarão mais completos e precisos.

Uma desvantagem apontada à técnica de pontos de função é que ela não considera apectos como os relacionados à equipe de desenvolvimento (incluindo domínio das tecnologias utilizadas, conhecimento e experiência em análise, projeto e desenvolvimento e mudanças que podem causar impacto) e a complexidade do *software* a ser desenvolvido. Além disso, a contagem é realizada com base nas funcionalidades que o usuário estabelece ou percebe para o sistema, podendo não ser considerada a complexidade dos esforços técnicos e/ou tecnológicos realizados. Pontos de função não consideram, tampouco, o reuso de artefatos de *software*, o uso de tecnologias como ambientes integrados de desenvolvimento de *software* e a geração automatizada de código, dentre outros.

Como forma de contribuir para a estimativa de prazo de projetos de *software*, neste trabalho é proposta uma complementação à metodologia publicada em Borsoi *et al.* (2010). Por meio dessa metodologia, o prazo é obtido a partir da especificação dos requisitos do ponto de vista do usuário, tendo como base pontos de função (FPCPM, 1999). A complementação proposta se refere à definição de fatores modificadores do prazo estimado e ao uso de lógica *fuzzy*. Os fatores modificadores de prazo propostos neste trabalho representam uma maneira de atender aos aspectos e requisitos não abrangidos pela técnica de pontos de função, incluindo a experiência e o conhecimento da equipe. Os fatores modificadores definem o percentual de aumento ou de redução do prazo obtido a partir dos requisitos do sistema. Esse percentual é obtido por meio da aplicação de lógica f*uzzy*. Essa lógica é utilizada para combinar esses fatores em regras com o objetivo de tornar a estimativa de prazo mais precisa, já que, com essa combinação, é possível considerar as influências desses fatores entre si. O uso de lógica *fuzzy* visa a facilitar o processo de realizar estimativa para o usuário, uma vez que a entrada é baseada em variáveis linguísticas (pouco e bastante, por exemplo) que são convertidas para constantes (valores crip). A combinação dessas variáveis resulta no percentual de ajuste do prazo definido com base nos fatores definidores de prazo, os requisitos do sistema.

Este texto está organizado em seções. Esta é a primeira e apresenta o contexto no qual o trabalho se insere, o problema e forma de solução proposta. A Seção 2 apresenta o referencial teórico centrado em estimativa de *software* e a Seção 3, o referencial teórico sobre sistemas *fuzzy*. Na Seção 4 estão os principais trabalhos relacionados a este. Na Seção 5 é apresentada a metodologia utilizada para o desenvolvimento do trabalho e na Seção 6, a proposta deste trabalho. E, por fim, estão a conclusão do trabalho e as referências.

2 ESTIMATIVA DE SOFTWARE

Estimar, em Engenharia de *Software*, consiste em determinar (prever) prazo, recursos e esforço necessários para desenvolver um projeto de *software* (PRESSMAN, 2005). Reinaldo e Filipakis (2009) acrescentam o objetivo de determinar o tamanho do produto e o custo à organização.

O tamanho, definido a partir dos requisitos do sistema, sejam funcionais ou não funcionais, e a qualidade esperada para o *software* podem ser utilizados como base para estimar prazo (cronograma), recursos (pessoas, equipamentos, tecnologias, ambientes), esforço (a quantidade de trabalho necessária) e custo (valor financeiro).

Na literatura são encontradas várias técnicas para realizar estimativas. Desde técnicas mais genéricas e clássicas como a contagem de linhas de código (PARK, 1992) até as mais específicas, como a medida de pontos de função para *software* desenvolvido com a linguagem Java (KUSUMOTO *et al.*, 2002), ou mesmo a API Points (ALMEIDA e LUZ, 2008).

Nasir (2006) categoriza as técnicas de estimativa em: a) abordagens paramétricas, como pontos de função e linhas de código; e b) abordagens heurísticas que abrangem estimativas orientadas ao aprendizado e baseadas em conhecimento especialista. Já Meller (2002) e Keung (2009) as agrupam nas categorias:

a) julgamento especialista – ou parecer técnico que tem como base a experiência pessoal dos desenvolvedores. Delphi e Wideband-Delphi são exemplos de técnicas utilizadas para fazer julgamento especialista.

b) analogia – as estimativas para projetos novos são realizadas com base em um histórico de projetos, considerando semelhanças em termos de características entre os projetos. Para utilizar analogia é necessário que haja uma adequada caracterização e documentação do histórico de projetos para identificação de semelhanças ou equivalências entre eles.

c) modelos algorítmicos – são definidos com base em procedimentos, contagens e cálculos. Exemplos: linhas de código, modelo de estimativa de Putnam, modelo de custo construtivo (COCOMO), pontos de função, pontos de particularidade, ciência do *software* de Halstead e número ciclomático de McCabe. Linhas de código, modelo de estimativa de Putnam e COCOMO são baseados na contagem das linhas de código. Pontos de função objetiva medir a funcionalidade do *software*, medindo o tamanho do que o *software* faz (DEKKERS, 1999). Ciência do *software* de Halstead tem como base o número de operadores e operandos (KAN, 1995). O número ciclomático de McCabe é calculado a partir da representação do fluxo de controle de um programa (PRESSMAN, 2005).

As estimativas são, geralmente, realizadas com base nas características do sistema. Contudo, outros fatores podem ser considerados. A API Points (ALMEIDA e LUZ, 2008), por exemplo, que é um método para estimar tamanho e esforço para desenvolvimento de uma API (*Application Programming Interface*), determina o tamanho do projeto com base em

doi:10.5329/RESI.2012.1101008

características externas do sistema. Esse método tem como base a identificação das restrições e a escolha do percentual de complexidade. As restrições são fatores que podem limitar o desenvolvimento do projeto: suposições no planejamento, dependências externas e internas, riscos associados à API a ser desenvolvida e a quantidade de métodos.

Cada técnica possui vantagens e desvantagens em relação às demais e características e aplicabilidades específicas. Estimativas baseadas na contagem de linhas de código, por exemplo, requerem conhecimento adequado da tecnologia empregada ou das linhas de código necessárias para implementar os requisitos do *software*, além de experiência considerável no desenvolvimento de projetos. Laird (2006) destaca bons resultados obtidos com opinião de especialistas, pontos de casos de uso e de função e recomenda usar técnicas combinadas, incluindo analogia. Para Liao (2009) estimativas baseadas na opinião de especialistas e em analogia podem ser utilizadas após a definição do problema.

O sistema computacional desenvolvido para realizar estimativas como parte da proposta deste trabalho combina as técnicas baseadas em analogia e algorítmica, mas está fundamentado em pontos de função. O prazo é calculado com base nas funcionalidades do *software*. O prazo estimado é obtido a partir da indicação de tempo para realizar cada agrupamento de atividades definido pelo especialista. E um processo algorítmico utilizando lógica *fuzzy* é empregado para combinar os fatores de ajuste do prazo que consideram o contexto do processo de desenvolvimento de *software*, incluindo a equipe e o ambiente.

A subseção a seguir trata especificamente de pontos de função, por ser essa técnica importante na fundamentação da proposta deste trabalho.

2.1 PONTOS DE FUNÇÃO

Análise de Pontos de Função (APF) é uma técnica para medir projetos de *software* a partir de uma perspectiva funcional (FPCPM, 1999). APF é independente de métodos, tecnologias, equipamentos e outras condições necessárias para implementar o sistema. Essa técnica visa a estabelecer uma medida de tamanho, considerando a funcionalidade implementada do ponto de vista do usuário. Ela pode ser usada para estimar o tamanho do projeto de desenvolvimento ou de melhoria de sistemas de *software*. Pontos de função é a unidade de medida de tamanho dessa técnica.

Tavares, Carvalho e Castro (2002) definem a medição de pontos de função a partir da especificação de requisitos. Esses autores utilizam casos de uso e diagramas de classe como base para a contagem. De acordo com a técnica APF, uma aplicação de *software*, vista sob a ótica do usuário, é um conjunto de funções ou atividades do negócio que o beneficiam na realização de suas tarefas (TAVARES, CARVALHO, CASTRO, 2002). O manual da APF (FPCPM, 1999) explicita que uma visão do usuário representa as necessidades de negócio desse usuário na sua linguagem. Essas necessidades são traduzidas ou representadas de maneira a prover uma

solução computacional para o problema. Assim, a contagem de pontos de função é realizada representando a informação em uma linguagem que é comum para usuários e desenvolvedores do sistema.

Zheng *et al.* (2009) com base na FPCPM (1999) definem os seguintes passos para calcular pontos de função: determinar o tipo de contagem, identificar escopo e fronteiras da aplicação, contar funções de dados e de transação, determinar o fator de ajuste e calcular pontos de função ajustados, que são explicados a seguir:

a) determinar o tipo de contagem – definir o tipo de projeto, que pode ser, por exemplo, de desenvolvimento ou de melhoria.

b) identificar o escopo de contagem e as fronteiras da aplicação – definir a abrangência do que será contado e o tipo de sistema.

c) contar as funções de dados e de transação – determinar os pontos de função não ajustados por agrupamento de funções do tipo *dado* e *de transação* que são: entradas e saídas externas, cuja complexidade é obtida pela quantidade de arquivos lógicos e de dados elementares referenciados; e consulta externa, arquivos lógicos internos e arquivos de interface externa, cuja complexidade é calculada a partir da quantidade de registros lógicos e de dados elementares referenciados.

As funções do tipo *dado* (arquivo lógico interno e de interface externa) representam as funcionalidades fornecidas ao usuário do sistema visando a atender os requisitos internos e externos da aplicação que são referentes a dados. As funções do tipo transação (entradas externas, saídas externas e consultas externas) representam as funcionalidades fornecidas ao usuário que realizam operações no sistema, executando ações (inclusão, alteração, exclusão, consulta) sobre as informações do banco de dados.

O FPCPM (1999) propõe cinco funções entre dados e transações e tabelas para calcular os pontos de função não ajustados a partir de cada uma dessas funções. A NESMA, que é uma extensão do método apresentado pela FPCPM, propõe três tipos de contagem de pontos de função: a contagem indicativa, a contagem estimada e a contagem detalhada (NESMA, 2009).

d) determinar o valor do fator de ajuste – que pode ser obtido a partir de 14 características definidas pela FPCPM (1999) por meio da fórmula:

valor de ajuste = (soma dos graus de influência dos fatores de ajuste * 0,01) + 0,65

Roetzheim (2000) cita uma série de fatores ambientais de projeto que podem ter influência na definição de estimativas com base em pontos de função, como: capacidade do analista, experiência nas aplicações e na linguagem de programação e ferramentas, continuidade de pessoal, capacidade e experiência do gerente. Esses fatores são outra forma de determinar o ajuste na estimativa obtida.

O modelo COCOMO intermediário (BOEHM, FAIRLEY, 2000) considera como fatores de ajuste, um conjunto de atributos direcionadores do custo agrupados em quatro categorias: produto, *hardware*, pessoal e projeto. Cada um desses atributos deve ser classificado de acordo com uma escala que varia de "muito baixo" a "extremamente elevado" (em importância e valor). O produto de todos os resultados de multiplicadores de esforços é chamado de fator de ajustamento de esforço (PRESSMAN, 2005).

Jones (2008) inclui como fatores de ajuste, as entidades e os relacionamentos presentes nos arquivos lógicos, tipos de consultas, algoritmos, cronograma, custo, nível de qualidade, plataformas de *hardware* e de *software* utilizadas, critérios de segurança e desempenho, treinamento e instalação. Outros requisitos como reuso ou uso de componentes prontos e conhecimento e experiência da equipe também podem influenciar na determinação de prazo para desenvolver um projeto de *software*.

e) calcular os pontos de função ajustados – uma forma de cálculo dos pontos de função ajustados é obtida pelo fator de ajuste multiplicado por pontos de função não ajustados, assim:

pontos de função ajustados = pontos de função não ajustados * fator de ajuste

Definições, regras de contagem e classificação, tratamento de exceções e exemplos práticos relacionados a pontos de função podem ser encontrados em FPCPM (1999).

3 LÓGICA *FUZZY*

A utilização de conjuntos *fuzzy* para lidar com conceitos inexatos foi inicialmente proposta por Zadeh em 1965, motivado pelo fato de que muitas classes de objetos existentes no mundo físico não apresentam critérios de pertinência definidos com precisão (ZADEH, 1965, 1975, 1983). Alguns exemplos dessas classes são "temperatura alta", "declive elevado" e mesmo os conceitos de "floresta" ou "cidade". Para classes como essas, a lógica clássica (de Aristóteles) não permite a existência de uma transição gradual entre a pertinência completa e a não pertinência de um elemento a um determinado conjunto.

Na teoria dos conjuntos *fuzzy* é feita uma generalização da função característica, originando uma função de pertinência, que determina com que grau um objeto *x* pertence a um conjunto *A* no universo em questão (FONTE, 2004). Nessas condições, seja *X* um conjunto de objetos representando o universo, então um conjunto *A* é definido por um conjunto de pares ordenados de um elemento genérico *x* e sua função de pertinência μA(x). Essa relação pode ser expressa pela equação:

$$A = \{(x, \mu_A(x)) \mid x \in X\}$$

Ao contrário dos conjuntos clássicos, na teoria dos conjuntos *fuzzy*, a transição entre pertencer e não pertencer a um determinado conjunto é gradual. O conjunto *fuzzy* pode ser entendido como uma espécie de predicado lógico (condição) cujos valores percorrem o intervalo entre 0 e 1, representado por [0, 1].

Uma relação *fuzzy*, considerada uma generalização das relações tradicionais, representa um conjunto *fuzzy* associando cada elemento do produto cartesiano ou par (x,y), por exemplo, a um grau de pertinência definido no intervalo unitário [0; 1]. Assim, considerando dois universos X e Y quaisquer, uma relação *fuzzy* R é vista como uma generalização do produto cartesiano clássico $X \times Y \rightarrow \{0,1\}$. Essa relação *fuzzy* é dada por:

$$R=\{((x,y), \mu R(x,y)) \mid (x,y) \in X \times Y\}$$

Neste trabalho, as relações *fuzzy* são consideradas no formato matricial e são utilizadas para combinar pares de fatores definidos como modificadores ou de ajuste de prazo. O conteúdo das células da matriz representa o nível de ocorrência de cada fator no projeto analisado. São geradas relações *fuzzy* que representam o grau de influência dos fatores modificadores em questão ao prazo previamente estimado para o desenvolvimento do projeto de *software*, denominado prazo não ajustado. As relações *fuzzy* definidas são apresentadas na Seção 6.

A interpretação das regras *fuzzy* como relações *fuzzy* apropriadas permite a investigação de diferentes esquemas de raciocínio *fuzzy* (raciocínio aproximado) (PEDRYCZ e GOMIDE, 1998). O raciocínio *fuzzy* corresponde a uma metodologia de inferência que utiliza conceitos e ferramentas da lógica *fuzzy* para chegar a uma conclusão partindo-se de uma dada premissa. Desta forma, de posse de um conjunto de regras de proposições e conclusões (chamadas de regras linguísticas), combinadas por operadores *fuzzy*, pode-se inferir um conjunto *fuzzy*, do qual é possível extrair um valor numérico que representa o resultado final da análise.

Neste trabalho conceitos de lógica *fuzzy* são utilizados para ajustar o prazo de desenvolvimento de projeto de *software* obtido com base nos requisitos do sistema, que representam as funcionalidades requeridas ou definidas pelo usuário.

4 TRABALHOS RELACIONADOS

São considerados trabalhos relacionados a este, os que se referem a estimativa de projeto de *software* tendo como base pontos de função e o uso de lógica *fuzzy*. Diversos autores já empregaram conceitos de lógica *fuzzy* com o objetivo de aprimorar os resultados obtidos com técnicas de estimativas de esforço, tamanho e custo para desenvolvimento de *software*, tais como análise de pontos de função e análise de pontos de caso de uso.

Yau e Tsoi (1994) sugerem que, devido à incerteza comumente encontrada nas fases iniciais do desenvolvimento de projetos, ao se apli-

car uma métrica, seria mais fácil determinar o grau de influência das características do sistema a ser desenvolvido usando descrições cognitivas em vez de valores numéricos. Este foi um dos trabalhos preliminares que combinaram métricas e conceitos de lógica *fuzzy*.

Lima Júnior, Farias e Belchior (2003) ressaltam que a técnica de pontos de função não possui uma continuidade na classificação de suas funcionalidades. Eles propõem a Análise de Pontos de Função *Fuzzy*, baseada em uma classificação contínua e gradual das funcionalidades de um sistema, usando valores *fuzzy* no lugar da tradicional tabela de classificação. Esses autores indicam que o uso de termos linguísticos permite a substituição dos intervalos clássicos por conjuntos *fuzzy*, gerando uma extensão do modelo original. Eles apresentam também vantagens da extensão proposta: o modelo torna-se mais genérico, simulando o raciocínio humano na interpretação dos termos linguísticos. A transição de um termo linguístico para outro adjacente torna-se gradual, ao invés da mudança abrupta existente na técnica original.

Braz e Vergilio (2004) propõem uma métrica baseada em casos de uso, denominada Pontos por Tamanho de Caso de Uso *Fuzzy* (*Fuzzy Use Case Size Points*-FUSP), que usa conceitos da teoria de conjuntos *fuzzy* para criar uma classificação gradual das seções de um caso de uso, eliminando alguns problemas relacionados à classificação por meio de tabela de categorias. No estudo de caso apresentado por esses autores, a técnica proposta apresentou melhores resultados que a Análise de Pontos por Caso de Uso. Contudo, esses autores ressaltam que novos estudos de casos devem ser feitos para avaliar os resultados da aplicação da técnica, considerando modificações ou extensões na tabela de classificação FUSP, aumentando o número de elementos considerados ou a diferença entre a complexidade das categorias.

Xia, Capretz e Ho (2008) utilizam uma técnica denominada neuro-*fuzzy*, que incorpora a habilidade de aprendizado das redes neurais e a de capturar conhecimento humano da lógica *fuzzy*. Essa técnica visa a ajustar os valores propostos pela técnica de pontos de função. A proposta deste trabalho, ainda que esteja fundamentada em pontos de função, define fatores de ajustes e a sua combinação em regras *fuzzy*.

Fu *et al.* (2010) propõem um método de análise de pontos de função que combina regras *fuzzy* e redes neurais do tipo *backpropagation* com o objetivo de estimar tamanho de *software*. Os resultados obtidos mostram que o método pode eliminar a descontinuidade entre classes de diferentes complexidades da análise de pontos de função, aproveitando melhor os dados históricos, que realçam a exatidão da avaliação.

Com o objetivo de resolver o problema de descontinuidade da classe de complexidade da análise de pontos de função, Chen *et al.* (2010) propõem combinar a teoria *fuzzy* e métodos de interpolação visando a apresentar uma análise de pontos de função com interpolação-*fuzzy*. Os resultados obtidos indicam que o método pode estimar com mais exatidão

o número de pontos de função e obter melhor desempenho na operabilidade prática.

Diferentemente do proposto em Braz e Vergílio (2004), neste trabalho o cálculo de prazo não ajustado é definido a partir dos requisitos do ponto de vista do usuário do sistema, ou seja, as funcionalidades pretendidas para o sistema. Assim, é possível gerar a estimativa na fase inicial do ciclo de vida do *software*, permitindo utilizá-la na definição do planejamento do projeto, alocação de recursos e na negociação de contrato com o cliente.

No sistema desenvolvido como resultado deste trabalho, a descontinuidade existente entre classes de diferentes complexidades da técnica de análise de pontos de função é tratada com o uso de lógica *fuzzy*. Assim como é proposto por Fu *et al.* (2010).

A possibilidade de o usuário informar os tempos padrão a serem utilizados para gerar a previsão de prazo não ajustado para desenvolvimento do *software*, é vista como um diferencial em relação às propostas dos autores referidos nesta seção. O contexto do ambiente, a estrutura organizacional e características da equipe são considerados no cálculo do prazo estimado para um projeto de *software*. Após ser definido o prazo com base nos requisitos do sistema, são empregadas técnicas da lógica *fuzzy* para combinar fatores de ajustes (modificadores de prazo), cruzando informações desses fatores e gerando um prazo ajustado possivelmente mais próximo ao real.

5 MÉTODO DE DESENVOLVIMENTO EMPREGADO

Para o cálculo do prazo de desenvolvimento de um projeto foram definidos os fatores modificadores de prazo, considerados como os que poderiam alterar o prazo obtido a partir dos fatores definidores de prazo propostos em Borsoi *et al.* (2010). Esses fatores modificadores foram agrupados em: relacionados ao produto (*software* a ser desenvolvido), à equipe, ao conhecimento do negócio, ao uso de ferramentas, métodos e modelos e ao reuso de artefatos.

A partir desses fatores foram realizados testes para identificar se o cálculo do tempo poderia ser realizado por meio de interpolação em uma reta, ou seja, se os tempos para cada um dos fatores modificadores poderiam ser definidos de forma incremental contínua. Para fazer essa verificação foi definida a equação da reta a partir do número de campos de uma tabela e o respectivo tempo indicado por um especialista para implementar manutenção de dados complexa. Esse tipo de manutenção é um dos requisitos definidores de prazo propostos em Borsoi *et al.* (2010) que considera a quantidade de campos de uma tabela em banco de dados. A Figura 1 exemplifica esse procedimento de cálculo e análise para a implementação de manutenção de dados complexa (com validação, referências cruzadas, campos calculados, buscas inclusive com filtros), utilizando o tempo de 3 horas para até 5 campos e, para entre 6 e 15 campos, de 3 a 5 horas.

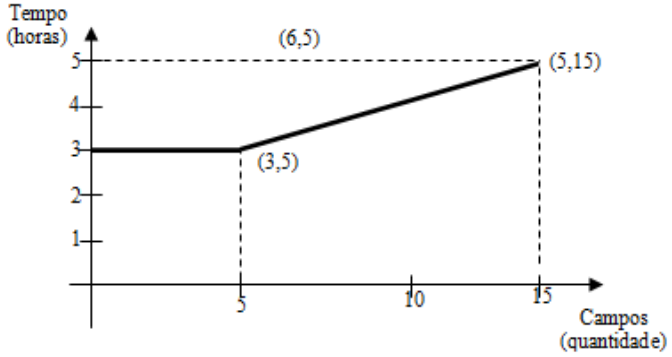


Figura 1. Interpolação sobre uma reta
Fonte: elaborada pelos autores

A equação da reta representada na Figura 1 é:

$$\frac{y_0 - 5}{x_0 - 15} = \frac{2}{10}$$

$$y_0 = \frac{2}{10} x_0 + 2$$

Para calcular o tempo que é necessário para implementar a manutenção explicitada para 6 campos de tabela (manutenção de dados complexa), obtém-se:

$$y_0 = \frac{2(6)}{10} + 2$$

$$y_0 = 3{,}2$$

Logo: serão necessárias 3,2 horas para uma manutenção de dados com 6 campos.

Quando se associam tempos com as atividades consideradas para estimativa, os dados são fornecidos de forma pontual. Desse modo valores intermediários precisam ser estimados, por exemplo, por meio de interpolação.

Cada intervalo, que se refere a uma determinada quantidade de campos de tabelas, associa-se com um valor de tempo, fazendo que a mudança de um dado número de campos para outro seja abrupta. Para suavizar essa mudança é utilizado o conceito de lógica *fuzzy* com regras diretas entre número de campos e tempos, por exemplo. Assim, se *campos* é C1, então *tempo* é T1. E constrõem-se funções de pertinência para a variável "campos", que representam a entrada e funções de pertinência para a variável "tempo" que representam a saída.

Para que fosse possível determinar se seria mais adequado realizar o cálculo a partir de interpolação em uma equação de reta ou com o uso de lógica *fuzzy*, as mesmas entradas do exemplo da Figura 1 foram utilizadas para definir os valores por meio de lógica *fuzzy*, ou seja, com o uso de funções de pertinência. O procedimento utilizado para obter o tempo em

decorrência do uso de funções de pertinência é apresentado nas Figuras 2 e 3, onde “µC” e “µT” significam o grau de pertinência para *campos* e *tempo* respectivamente.

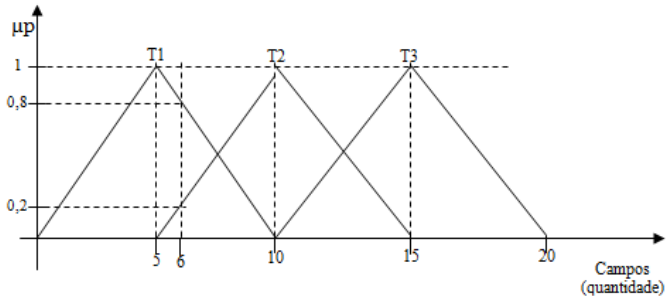


Figura 2. Funções de pertinência triangulares
Fonte: elaborada pelos autores

Os resultados gerados pelas funções de pertinência são os valores *fuzzy* (entradas) a partir dos quais são definidos os valores *crisp* (saídas). A saída se refere ao valor obtido para a definição do tempo necessário para implementar a respectiva funcionalidade do sistema.

Para obtenção das saídas *crisp*, que são tempos, foram utilizadas funções *singleton*, como representado na Figura 3. Esse tipo de função foi utilizado porque é esperado um valor pontual (valor específico de tempo) e não uma faixa de valores para a saída.

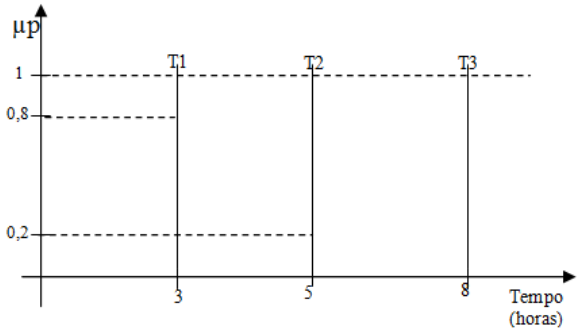


Figura 3. Função de pertinência impulso (*singleton*)
Fonte: elaborada pelos autores

O cálculo a seguir exemplifica a forma de obter o tempo para implementação de manutenção de dados complexa (com validação, referências cruzadas, campos calculados, buscas, inclusive com filtros) tendo como base as representações das Figuras 2 e 3.

$$t = \frac{3h(0{,}8) + 4h(0{,}3)}{0{,}8 + 0{,}3}$$

$$t = 3{,}27 horas$$

A resposta ao questionamento se era necessário utilizar funções de pertinência ou apenas obter os valores por interpolação de uma reta foi

obtida por meio dos dados exemplificados nas Figuras 2 e 3. Por meio da análise dessas figuras verificou-se que:

a) Os valores obtidos com a equação da reta, por interpolação, são proporcionais. Assim, utilizando essa equação o tempo aumentaria proporcionalmente ao número de campos;

b) Os tempos definidos pelos especialistas, nos dados utilizados para teste, não aumentam proporcionalmente ao acréscimo de número de campos como ocorre na representação de interpolação da reta;

c) Os valores obtidos utilizando lógica *fuzzy* (testados no aplicativo MatLab) são não proporcionais e se aproximam mais adequadamente aos valores informados pelos especialistas.

Assim, concluiu-se que é mais adequado utilizar lógica *fuzzy*, gerando as funções de pertinência para calcular os tempos. Desta forma, a partir das entradas *crisp* (valores padrão definidos pelo especialista) foram definidas as variáveis *fuzzy* para o sistema. E lógica *fuzzy* foi utilizada para agrupar fatores modificadores em regras, visto que a relação entre os fatores não é completamente linear.

O procedimento de cálculo proposto (etapas mostradas na Figura 4) é similar ao adotado pela técnica de pontos de função e é constituído de duas partes:

a) cálculo de fatores definidores de prazo que estão relacionados aos requisitos do sistema;

b) cálculo dos fatores modificadores de prazo (ou de ajuste), incluindo aspectos da equipe e do ambiente.

A definição do prazo não ajustado é obtida com base nos requisitos do sistema. Esse prazo é ajustado a partir de fatores modificadores que são agrupados em duplas. Cada dupla é considerada como variável linguística e é representada pelas suas funções de pertinência, gerando regras *fuzzy*. A execução das regras resulta em valores que definem o percentual de ajuste ao prazo obtido a partir dos requisitos do sistema. O conhecimento do especialista foi utilizado para definir as variáveis linguísticas, as regras e o percentual de ajuste resultante de cada regra.

O procedimento utilizado para obter o prazo estimado foi organizado em etapas, representadas na Figura 4. A ênfase desse procedimento está na representação da obtenção do percentual de ajuste que os fatores modificadores determinam sobre o prazo que é definido com base nas funcionalidades do *software*.

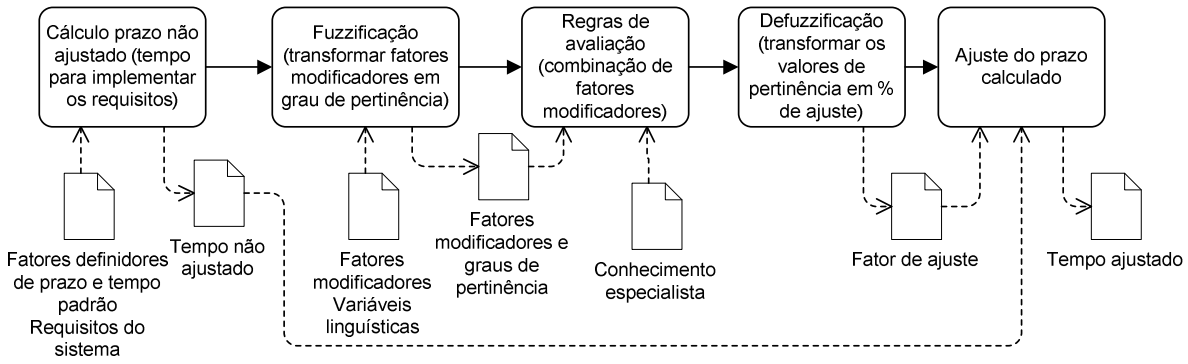


Figura 4. Etapas para o cálculo do prazo ajustado
Fonte: elaborada pelos autores

Na Figura 4, os retângulos com cantos arredondados representam atividades (que se referem às etapas) e as setas de traço contínuo que os interligam representam a sequência da sua realização. As setas de traço pontilhado ligam as atividades aos artefatos que elas produzem ou utilizam. Os artefatos são representados por retângulos com o canto superior direito simulando uma dobra.

Para obtenção do prazo a partir dos requisitos do sistema, o tempo padrão para implementar os requisitos é definido pelo usuário. A combinação de regras gera um percentual de aumento sobre o prazo estimado a partir dos fatores definidores.

Para o cálculo do prazo não ajustado (primeira etapa do rocedimento exposto na Figura 4), os fatores definidores de prazo (entradas para o sistema implementado) foram agrupados em manutenção de dados, geração de relatórios, interação com periféricos e processamento. Esses agrupamentos e as respectivas quantidades associadas estão apresentados resumidamente no Quadro 1. Para o cálculo da estimativa de prazo não ajustado é feito o somatório dos prazos obtidos em cada um desses subgrupos, utilizando a fórmula:

Tempo não ajustado = ∑dos tempos obtidos dos fatores definidores de prazo

No Quadro 1, os grupos representam as categorias de requisitos de um sistema de *software*, os subgrupos definem os fatores que são pontuados e a quantidade indica quantos elementos compõem uma unidade de fator.

Como entrada para a segunda etapa do procedimento da Figura 4 estão os fatores modificadores, que são representados por variáveis linguísticas no sistema desenvolvido. O Quadro 2 apresenta os fatores propostos como os que podem alterar o prazo obtido a partir dos fatores definidores de prazo. Esses fatores ajustam o prazo definido pelos requisitos considerando o contexto. O contexto se refere às condições em que o sistema é implementado. A coluna “% de influência” indica a escala de influência no ajuste de prazo do referido fator e varia de acordo com a importância de cada fator no cálculo da estimativa.

Grupos	Fatores definidores de prazo – Quantidade
Manutenção de dados	1. Simples (inclusão, exclusão, alteração, consulta simples em banco de dados): a) até 5 campos; b) 6 a 15 campos; c) + de 15 campos.
	2. Complexa (com validação, referências cruzadas, campos calculados, buscas inclusive com filtros): a) até 5 campos; b) 6 a 15 campos; c) + de 15 campos.
Geração de relatórios	1. Simples (listagem de cadastros): a) 1 tabela; b) 2 a 5 tabelas; c) + de 5 tabelas.
	2. Complexo (campos calculados, *joins*, *unions*, *subselects*, filtros, agrupamentos, ordenação, referência cruzada, gráficos): a) 1 tabela; b) 2 a 5 tabelas; c) + de 5 tabelas.
	3. Com geração de arquivos externos para exportação de dados: a) 1 tabela; b) 2 a 5 tabelas; c) + de 5 tabelas.
Interação com periféricos	1. Acesso a sistemas externos (validação em banco de dados de outro sistema, interface/protocolo de comunicação): 1 tabela; b) 2 a 5 tabelas; c) + de 5 tabelas.
	2. Interação com dispositivos como leitores de biometria, código de barras, obtenção de dados de dispositivos (sensores): 1 tabela; b) 2 a 5 tabelas; c) + de 5 tabelas.
	3. Envio de comandos para periféricos (sensores, atuadores): 1 tabela; b) 2 a 5 tabelas; c) + de 5 tabelas.
Processamento (transações)	1. Validação de acesso (*login*)
	2. Cálculos com ou sem consulta a banco de dados, conversão de dados: a) 1 cálculo; b) 2 a 5 cálculos; c) + de 5 cálculos
	3. Cálculos para atender a legislação: Conversão de dados: a) 1 cálculo; b) 2 a 5 cálculos; c) + de 5 cálculos.
	4. Processamento interno complexo, lógico e matemático extensivo: a) 1 processamento; b) 2 a 5 processamentos; c) + de 5 processamentos.
	5. Segurança como SSL ou código para criptografia.
	6. Requisitos não funcionais relevantes: restrições de desempenho, restrições de memória, portabilidade, confiabilidade das operaçoes realizadas, integração com sistemas existentes: a) 1 requisito; b) 2 a 4 requistios; c) 5 a 8 requisitos; d) + de 8 requisitos.
	7. Bancos de dados distintos com peculiaridades em sentenças SQL para cada banco: a) 1 tabela; b) 2 a 5 tabelas; c) + de 5 tabelas.

Quadro 1. Fatores definidores de prazo
Fonte: Borsoi *et al.* (2010)

Os fatores apresentados na coluna “Fator” do Quadro 2 foram agrupados definindo regras *fuzzy*, gerando uma maneira mais adequada de determinar a influência desses fatores no ajuste do prazo contado a partir dos requisitos do sistema. Essas regras são definidas na terceira etapa da representação da Figura 4. As regras representam combinações de fatores modificadores e são apresentadas no Quadro 3, exceto a regra F1, que se refere à redução do tempo estimado, enquanto todas as outras ocasionam o aumento deste tempo. Optou-se por um relacionamento entre dois fatores para simplificar as variáveis linguísticas geradas e a definição pelo especialista das variáveis linguísticas resultantes, conforme exemplificado no Quadro 4. Esses fatores são informados pelo usuário indicando um valor em uma escala pré-definida. Essa escala determina a faixa de percentual de ajuste que o respectivo fator pode determinar no prazo calculado com base nos requisitos do sistema.

Grupo	Fator	Descrição	% de influência
Relacionados ao produto (o sistema a ser desenvolvido)	F1 - Reuso de artefatos	Redução do tempo calculado, pelo uso de artefatos prontos, incluindo código e documentos de projeto.	Escala 0,5 – 10
	F2 Confiabilidade das operações realizadas pelo sistema.	Aumento do tempo calculado se houver confiabilidade e risco envolvidos. Refere-se à criticidade e risco das operações realizadas pelo sistema.	Escala 0,5 – 10
	F3 Integração com sistemas existentes	Aumento do tempo calculado se há necessidade de integração entre sistemas.	Escala 0,5 – 5
Relacionados à equipe	F4 Conhecimento e experiência da equipe em análise e projeto	Aumento do tempo calculado pela falta de conhecimento e experiência em análise e projeto.	Escala 0,5 – 30
	F5 Conhecimento e experiência da equipe nas tecnologias utilizadas	Aumento do tempo calculado pela falta de conhecimento e experiência no uso das tecnologias a serem utilizadas no desenvolvimento do projeto.	Escala 0,5 – 20
	F6 Mudanças na equipe	Aumento do tempo calculado se previstas mudanças na equipe que possam causar impacto negativo.	Escala 0,5 – 20
Relacionados ao domínio do negócio	F7 O conhecimento da equipe do negócio informatizado.	Aumento do tempo calculado pela falta de conhecimento do domínio de negócio do sistema.	Escala 0,5 – 10
Relacionados ao uso de ferramentas, métodos e modelos	F8 Uso de processos	Aumento do tempo calculado pela inexistência de processos e procedimentos (inclui uso de modelos de qualidade) bem definidos e documentados.	Escala 0,5 – 5
	F9 Uso de modelos de artefatos	Aumento do tempo calculado por não haver uso de modelos de artefatos, padronização de codificação e outros.	Escala 0,5 – 15

Quadro 2. Fatores modificadores de prazo
Fonte: Borsoi *et al.* (2010)

Para todas as regras foi considerada uma escala entre 0 e *n*, sendo zero definido quando não há influência e *n* quando a influência é máxima. Essa escala foi utilizada para facilitar a determinação da abrangência das variáveis linguísticas, ou seja, um relacionamento entre as entradas e as saídas do sistema *fuzzy*. As variáveis definidas para entrada dos fatores no sistema são: pouco, médio, bastante, significando o quanto há do referido fator. Também foi considerado que um dos fatores de cada regra poderia ter mais influência que o outro. A definição do fator que teria mais influência foi feita com base no conhecimento do especialista.

Regras: [Fator X][Fator Y]
[F2] Confiabilidade das operações realizadas pelo sistema. [F3] Integração com sistemas existentes.
[F2] Confiabilidade das operações realizadas pelo sistema. [F4] Conhecimento e experiência da equipe em análise e projeto.
[F5] Conhecimento e experiência da equipe nas tecnologias utilizadas. [F3] Integração com sistemas existentes.
[F4] Conhecimento e experiência da equipe em análise e projeto. [F5] Conhecimento e experiência da equipe nas tecnologias utilizadas.
[F4] Conhecimento e experiência da equipe em análise e projeto. [F6] Mudanças na equipe.
[F5] Conhecimento e experiência da equipe nas tecnologias utilizadas. [F6] Mudanças na equipe.
[F4] Conhecimento e experiência da equipe em análise e projeto. [F9] Uso de modelos de artefatos.
[F5] Conhecimento e experiência da equipe nas tecnologias utilizadas. [F8] Uso de processos.
[F5] Conhecimento e experiência da equipe nas tecnologias utilizadas. [F2] Confiabilidade das operações realizadas pelo sistema.
[F4] Conhecimento e experiência da equipe em análise e projeto. [F7] O conhecimento da equipe do negócio informatizado.
[F4] Conhecimento e experiência da equipe em análise e projeto. [F8] Uso de processos.
[F6] Mudanças na equipe. [F8] Uso de processos.
[F6] Mudanças na equipe. [F9] Uso de modelos de artefatos.
[F6] Mudanças na equipe. [F7] O conhecimento da equipe do negócio informatizado.

Quadro 3. Regras criadas com duplas de fatores modificadores de prazo
Fonte: elaborado pelos autores

Exemplo de regra gerada:

Regra [F2] e [F4]: Confiabilidade das operações *x* Conhecimento e experiência da equipe em análise e projeto.

Confiabilidade / Análise	Pouco	Médio	Bastante
Pouco	Muito grande	Grande	Médio
Médio	Grande	Médio	Pequeno
Bastante	Médio	Pequeno	Muito pequeno

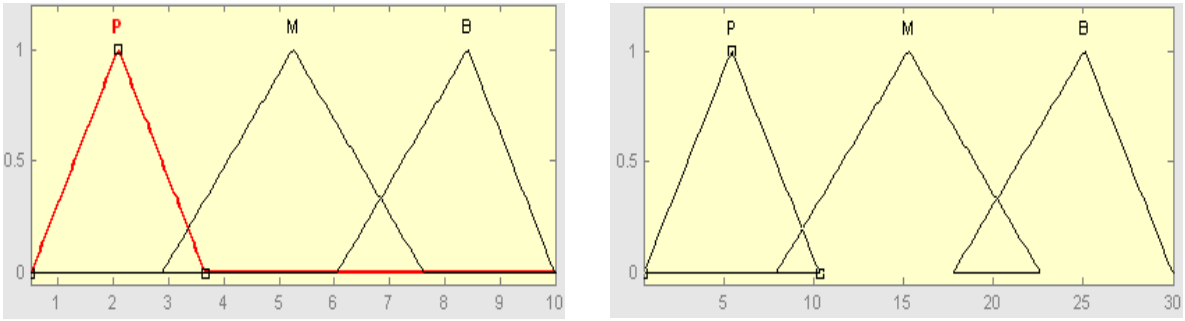
Quadro 4. Exemplo de regra composta por dois fatores modificadores de prazo
Fonte: elaborado pelos autores

A forma de interpretação e leitura do Quadro 4 é:

Se [F2] é *Pouco* e [F4] é *Pouco* então Saída (*Muito grande*).

Essa forma de leitura se aplica às nove combinações possíveis, entre as três alternativas de cada regra, que são geradas pela combinação de dois fatores (Quadro 3), como exemplificado no Quadro 4.

A Figura 5 contém a representação gráfica da variável de entrada para F2 (confiabilidade das operações realizadas pelo sistema), considerando a escala de zero a dez, e para a variável de entrada F4 (conhecimento e experiência da equipe em análise e projeto), considerando uma escala de zero a trinta.



(a) Confiabilidade das operações (F2) (b) Conhecimento em análise e projeto (F4)

Figura 5. Variáveis de entrada
Fonte: elaborada pelos autores

A defuzificação (fase 4 da Figura 4) ocorre tendo como base as variáveis de entrada da Figura 5 (a e b) para as quais são definidas as saídas para o sistema (Figura 6). A escala para as variáveis de saída foi definida com base na maior escala entre as entradas. A Figura 6 possui representação *singleton*, como justificado na representação da Figura 3.

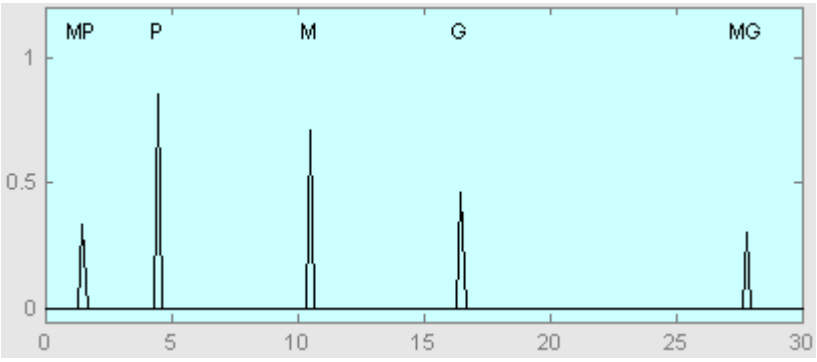


Figura 6. Variáveis de saída para o sistema
Fonte: elaborada pelos autores

Os percentuais de influência gerados pelas treze regras *fuzzy* (Quadro 3) são somados e então aplicados ao prazo não ajustado. Assim, o cálculo do prazo ajustado (última etapa da Figura 4) é realizado adotando-se a seguinte fórmula:

Prazo ajustado = Prazo não ajustado + (Prazo não ajustado * percentual de influência do somatório das regras *fuzzy* / 100)

6 SISTEMA COMPUTACIONAL DESENVOLVIDO

O *software* desenvolvido utiliza os fatores definidores apresentados no Quadro 1 para o cálculo do prazo não ajustado e os fatores modificadores expostos no Quadro 2 para gerar a estimativa de prazo final. Para cada um dos fatores modificadores, o usuário indica o valor de influência, tendo como base uma escala. As Figuras 7 a 9 apresentam telas do sistema desenvolvido. Nessas telas, o usuário indica a quantidade e o tempo de cada um dos respectivos fatores definidores e os valores considerados adequados para os fatores modificadores. Esses fatores modificadores são indicados considerando o contexto em que o projeto é desenvolvido. Os valores constantes nos campos de formulário dessas figuras se referem ao estudo de caso utilizado para exemplificar a proposta e o resultado deste trabalho.

O estudo de caso analisado se refere ao desenvolvimento de um módulo de um sistema com alguns cadastros e a geração de nota fiscal eletrônica. Esse módulo é parte de um sistema já em uso e desenvolvido pela mesma empresa, mas por outra equipe e há algum tempo. A equipe atual tem o conhecimento considerado necessário das tecnologias utilizadas. Nessa situação, a aplicação de contagem por pontos de função tendo como base arquivos de entrada e saída e transações pode não representar a realidade das condições de trabalho.

Nessa empresa as estimativas têm sido realizadas por um especialista, um analista sênior que trabalha há muito tempo na empresa. Para o cálculo, esse analista tem como base os requisitos do sistema, a estrutura organizacional e as características dos membros da equipe, bem como o contexto no qual o projeto está inserido. A estimativa calculada pelo especialista para o estudo de caso em questão é comparada à obtida por meio da utilização do sistema desenvolvido.

A Figura 7 apresenta a tela do sistema desenvolvido onde são informados os valores dos fatores definidores referentes ao estudo de caso analisado. No campo “tempo” é indicado o valor de tempo padrão utilizado para o cálculo. Os valores de tempo são definidos pelo usuário e podem ser armazenados e utilizados posteriormente, gerando assim uma espécie de *template*. Desta forma, é possível ter padrões para cálculo de acordo com especificidades do sistema ou características da equipe, por exemplo. O sistema também permite alterar os tempos informados como padrão, salvando-os como um novo padrão ou utilizando-os somente para o cálculo atual.

As quatro primeiras abas (marcadas com um retângulo pontilhado na Figura 7) apresentam os agrupamentos de fatores definidores de prazo descritos em Borsoi *et al.* (2010). Esses fatores têm como base os requisitos do sistema. A manutenção de dados, por exemplo, pode ser simples

ou complexa, de acordo com a quantidade de tabelas manipuladas e o número de campos dessas tabelas.

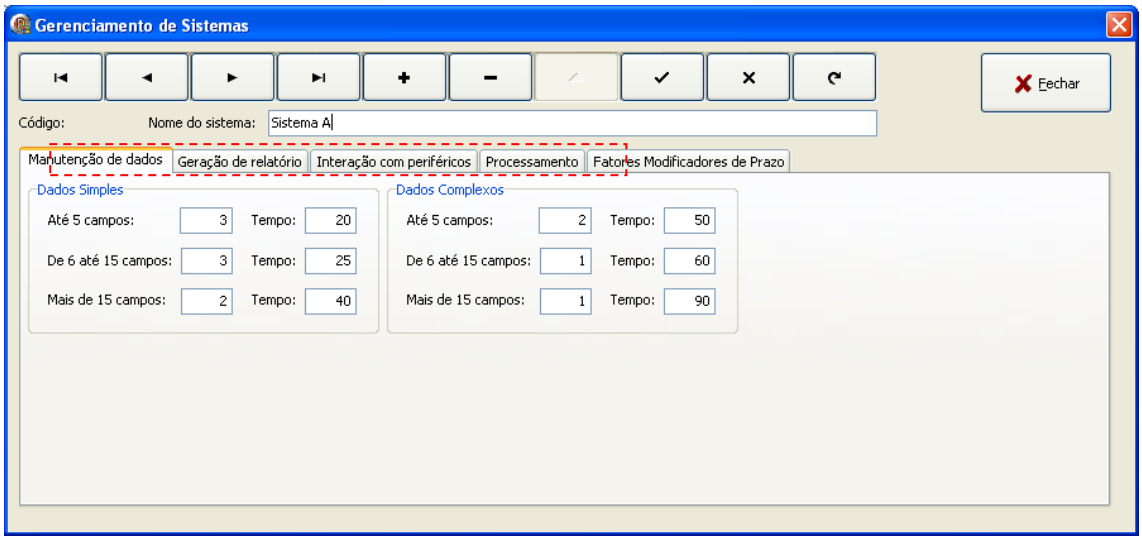


Figura 7. Fatores definidores de prazo
Fonte: elaborada pelos autores

A Figura 8 (aba com foco) apresenta a tela para o usuário informar os valores para os fatores modificadores de prazo por meio da indicação de um valor em uma escala pré-estabelecida. Esses fatores são combinados pelas regras apresentadas no Quadro 3 e de acordo com os procedimentos constantes na sequência a esse quadro. O resultado dessas combinações é o percentual de ajuste no prazo obtido com os fatores indicados na Figura 7.

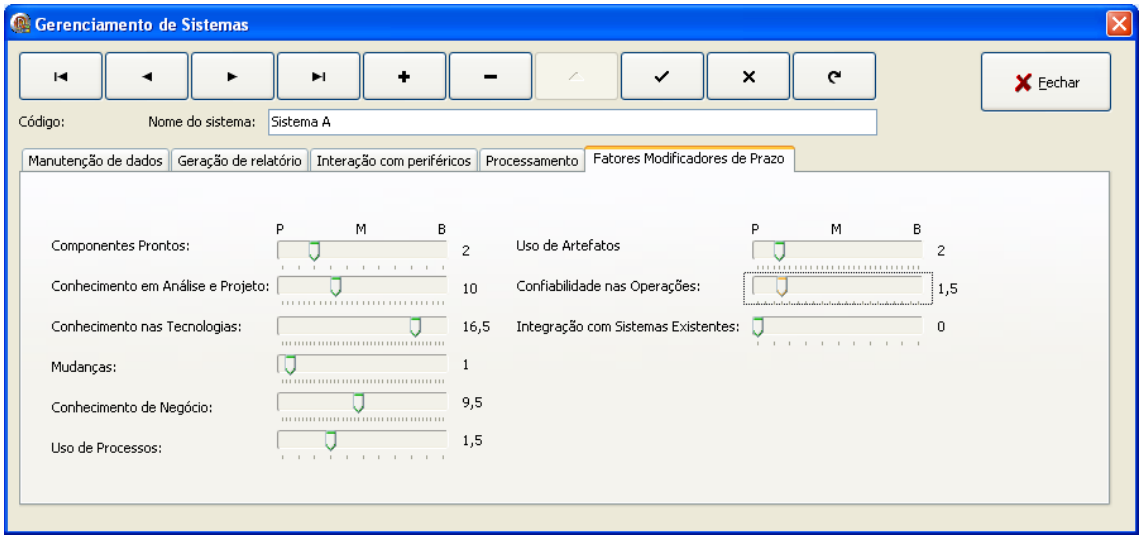


Figura 8. Fatores modificadores de prazo
Fonte: elaborada pelos autores

Na Figura 8, “P” sifinifica pouco, “M” médio e “B” bastante que representam os valores “fuzzy” para cada fator modificador. Paralelamente é mostrada a escala numérica (valores “crip”) utilizada para as entradas. Considera-se que pode ser mais fácil para quem está realizando a estima-

tiva definir que o conhecimento da equipe, por exemplo, é “pouco” da tecnologia sendo utilizada em vez de indicar 2 em uma escala que vai de 0,5 a 20.

A Figura 9 apresenta o prazo final ou ajustado obtido. Esse prazo é calculado a partir dos fatores definidores, dos fatores de ajustes e do tempo necessário para a modelagem do sistema. O tempo para modelagem é calculado automaticamente pelo sistema a partir dos fatores definidores e de ajuste.

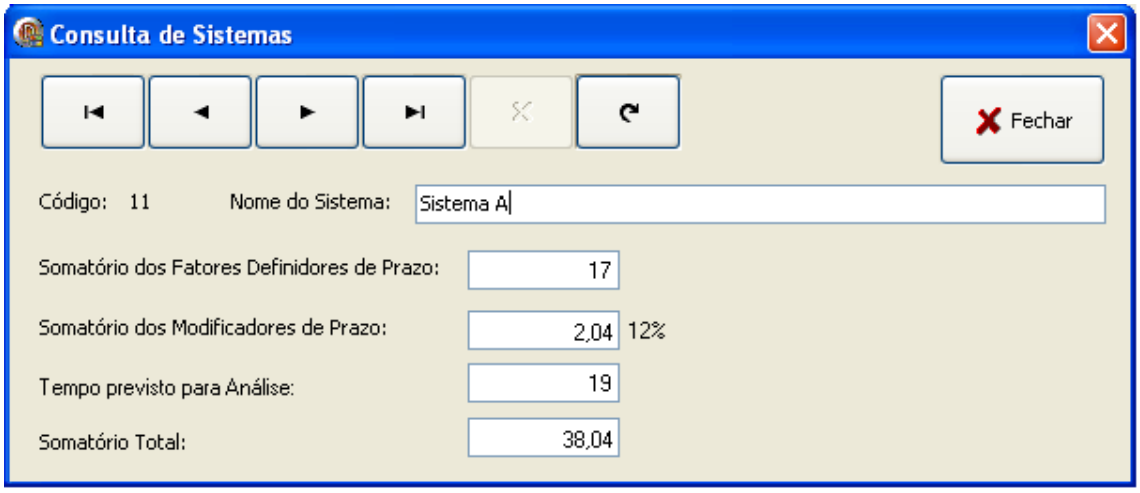


Figura 9. Prazo estimado
Fonte: elaborada pelos autores

O prazo calculado para o módulo de sistema utilizado como estudo de caso é de 38 horas, como calculado na tela mostrada na Figura 9. E esse foi praticamente o tempo efetivamente utilizado para a modelagem e a implementação do referido módulo de sistema. Esse módulo de sistema foi desenvolvido em uma semana, com jornada de 8 horas diárias.

7 CONSIDERAÇÕES FINAIS

O uso da lógica *fuzzy* para combinar fatores de ajuste de prazo oferece flexibilidade à atividade de estimativa de prazo. Isso ocorre porque o especialista não apenas pode decidir quais variáveis farão parte da estimativa como associar a elas um peso para cada avaliação específica.

O sistema desenvolvido permitiu colocar em prática o proposto neste trabalho, e com base no módulo de sistema utilizado como estudo de caso foi possível verificar que o tempo calculado pelo *software* foi bastante próximo ao estimado pelo especialista. O especialista utilizou vários fatores na estimativa, além dos especificados pela análise de pontos de função: arquivos e transações. Dentre esses fatores está a experiência e o conhecimento da equipe, o conhecimento das tecnologias utilizadas e o reuso de artefatos.

Considerar os fatores indicados nos Quadros 1 e 2, especialmente os modificadores, se mostrou relevante nos vários testes realizados para

identificá-los porque valores bastante distintos foram obtidos para a implementação das mesmas funcionalidades. São indícios de que outros aspectos influenciam na definição do prazo. Muitos desses aspectos não estão diretamente relacionados ao sistema, mas ao contexto ou ao ambiente em que o desenvolvimento do sistema ocorre.

Assim, os resultados obtidos com este primeiro teste em situação real são limitados, por ser um único estudo de caso, mas motivam a realização de novos testes, visando a verificar a eficácia da proposta em diferentes contextos e ambientes de desenvolvimento.

Como principal perspectiva futura deste trabalho está a realização de testes e a avaliação de sua efetividade por especialistas em gerar estimativas de prazo de projeto de sofware de empresas diferentes. Também podem ser realizados estudos comparativos com resultados obtidos por outras técnicas comparando os resultados com o tempo efetivamente incorrido no desenvolvimento de projetos de *software*. Outra perspectiva futura é flexibilizar ainda mais o sistema desenvolvido, permitindo ao usuário ajustar a escala para os fatores modificadores, de modo que as variáveis fuzzy “pouco”, “médio” e “bastante” permanecessem, mas fosse possível definir os limites superior e inferior dessa escala.

8 AGRADECIMENTOS

Os autores agradecem à Fundação Araucária pelo suporte financeiro ao projeto.

REFERÊNCIAS

ALBRECHT, Allan J. Measuring application development productivity. In: Joint Application Development Symposium. Monterey, California. *Anais...*, 1979.

ALMEIDA, Ana Carina M.; LUZ, Carla Renata R. Melhoria de um processo de estimativa de desenvolvimento de software através do emprego de metodologia e criação da técnica de estimativa - API points. In: Congresso Brasileiro de Gerência de Projeto. 3., 2008, Porto Alegre, *Anais...*, 2008.

BOEHM, Barry W., FAIRLEY, Richard E. Software estimation perspectives. *IEEE Software*, v. 17, n. 6, November 2000, p. 22-26, 2000.

BORSOI, Beatriz T.; LINARES, Kathya S. C.; ASCARI, Rúbia E. O. S.; ROCHA, Leandro G.; TOSCAN, Luiz F.; BOLO, Matheus M., Fatores definidores e modificadores em estimativa de prazo de projeto de software. In: Semana de Inovações em Sistemas de Informação (SIS2INFO). 11., 2010, Cascavel. *Anais...*, 2010.

BRAZ, Márcio Rodrigo; VERGILIO, Silvia R. Using fuzzy theory for effort estimation of object-oriented software. In: IEEE. International Conference on Tools with Artificial Intelligence (ICTAI'04). 16., *Anais...*, 2004.

BRIAND, Lionel C.; EMAM, Khaled El; SURMANN, Dagmar; WIECZOREK, Isabella; MAXWELL, Katrina D. An assessment and comparison of common software cost estimation modeling techiniques. In: International Conference on Software Engeneering (ICSE). 21., Los Angeles, California. *Anais...*, 1999.

CHEN, Qingzhang; CHENG, Rong; FANG, Shuojin; OU, Yanqiang. Study of function points analysis based on fuzzy-interpolation. *Journal of Computational Information Systems*, v. 6, n. 5, p. 1369-1375, 2010.

DEKKERS, Carol A. Pontos de função e medidas: O que é um ponto de função? Flórida, 1999. Disponível em: http://www.bfpug.com.br/Artigos/Dekkers-PontosDeFuncaoE Medidas.htm. Acesso em: 30/08/2001.

FONTE, Cidália Maria P. C. *Entidades geográficas difusas – métodos de construção e processamento*. Universidade de Coimbra. Portugal, 2004.

FPCPM. *Function point counting practices manual*. Versão 4.1.1. International Function Point Users Group. Disponível em: http://www.ifpug.org, 1999. Acesso em: 20/06/2011.

FU, Ya-fang; LIU, Xiao-dong; YANG, Ren-nong; DU, Yi-lin; LI, Yan-jie. A software size estimation method based on improved FPA. In: WRI World Congress on Software Engineering. (WCSE). 2., *Anais...*, 2010, p. 228-233.

JONES, Capers. Software sizing during requirements analysis. 2008. Disponível em: http://www.modernanalyst.com/Resources/Articles/tabid/115/articleType/ArticleView/articleId/512/Software-Sizing-During-Requirements-Analysis.aspx. Acesso em: 20/04/2011.

KAN, Stephen H. Metrics and models in software quality engineering. Boston: Addison-Wesley, 1995.

KEUNG, Jacky. Software development cost estimation using analogy: a review. In: Australian Software Engineering Conference. (ASWEC). *Anais...*, 2009, p. 327-336.

KUSUMOTO, Shinji; IMAGAWA, Masahiro; INOUE, Katsuro; MORIMOTO, Shauuma; MATSUSITA, Kouji; TSUDA, Michio. Function point measurement from java programs. In: International Conference on Software Engineering. 24., *Anais...*, Orlando, Flórida, 2002.

LAIRD, Linda M. The limitations of estimation. *IT Professional*, v. 8, n. 6, 2006. p. 40-45. http://dx.doi.org/10.1109/MITP.2006.149

LEWIS, John P. Large limits to software estimation. *ACM Software Engineering Notes*, v. 26, n. 4, jul 2001, p. 54-59. http://dx.doi.org/10.1145/505482.505490

LIAO, Hancheng. Research on software development estimation. In: IEEE International Conference on Granular Computing. (GRC '09). *Anais...*, 2009, p. 390-393.

LIMA JÚNIOR, Osias S.; FARIAS, Pedro Porfírio Muniz; BELCHIOR, Arnaldo Dias. A *fuzzy* model for function point analysis. *Software Quality Journal*, v. 11, p. 149-166, 2003. http://dx.doi.org/10.1023/A:1023716628585

MELLER, Maristela Corrêa. Modelos para estimar custos de software: estudo comparativo com softwares de pequeno porte. Dissertação de Mestrado. Universidade Federal de Santa Catarina. Santa Catarina, 2002.

NASIR, Mehwish. A survey of software estimation techniques and project planning practices. In: ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing (SNPD'06). 7., 2006, p. 305-310.

NESMA. Netherlands function point users group. 2009. Disponível em: http://www.nesma.nl/section/nesma/. Acesso em: 06/04/2011.

PARK, Robert E. Software size measurement: a framework for counting source statements. Pittsburgh, 1992. Disponível em: http://www.sei.cmu.edu/library/abstracts/reports/92tr020.cfm. Acesso em: 20/06/2011.

PEDRYCZ, Witold; GOMIDE, Fernando. *An introduction to fuzzy sets: analysis and design*. Cambridge: MIT Press, 1998.

PRESSMAN, Roger. *Engenharia de software*, Rio de Janeiro: McGraw-Hill, 2005.

REINALDO, Werley Teixeira; FILIPAKIS, Cristina D'Ornellas. Estimativa de tamanho de software utilizando APF e a abordagem NESMA. In: Encontro de Estudantes de Informática do Tocantins. 11., 2009, Palmas. *Anais...*, Palmas: Centro Universitário Luterano de Palmas, 2009. p. 151-160.

ROETZHEIM, William. Project cost adjustments. *SD Magazine*, nov 2000. Disponível em: http://drdobbs.com/184414671. Acesso em: 20/03/2011.

TAVARES, Helena Cristina A. B.; CARVALHO, Ana Elizabete S.; CASTRO, Jaelson F. B. Medição de pontos por função a partir da especificação de requisitos. In: Workshop em Engenharia de Requisitos. (WER). 2002, p. 278-298.

XIA, Wei; CAPRETZ, Luiz Fernando; HO, Danny. A Neuro-Fuzzy Model for Function Point Calibration. *WSEAS Transactions on Information Science & Applications*, v. 5, n. 1, January 2008, p. 22-30, 2008.

YAU, Chuk; TSOI, Raymond H. L. Assessing the fuzziness of general. System characteristics in estimating software size. *IEEE Transactions on Software Engineering*, p. 189-193, 1994.

ZADEH, Lotfi A. Fuzzy sets. *Information and Control*, v. 8, p. 338-352. 1965. http://dx.doi.org/10.1016/S0019-9958(65)90241-X

ZADEH, Lotfi A. The concept of a linguistic variable and its application to approximate reasoning. *Information Science*. New York: Elsevier Publishing Company Inc. n. 8 p. 199-249. 1975.

ZADEH, Lotfi A. Linguistic variables, approximate reasoning and dispositions. *Medical Informa*tics, v. 8, n. 3, p. 173-186. 1983. http://dx.doi.org/10.3109/14639238309016081

ZHENG, Yinhuan; WANG, Beizhan; ZHENG, Yilong; SHI, Liang. Estimation of software projects effort based on function point. In: International Conference on Computer Science & Education. 4., 2009, *Anais...*, p. 941-943, 2009.