DEPARTAMENTO DE ENGENHARIA ELÉTRICA

BI – Master

Otimização e Planejamento de Alocação de Recursos Humanos em Projetos de Software usando Algoritmos Genéticos Co-Evolucionários com Lógica Fuzzy

Autor: Thiago Melo Bezerra

Professores Orientadores: André Vargas Abs da Cruz & Jesus Domech Moré

Autor: Thiago Melo Bezerra

Otimização e Planejamento de Alocação de Recursos Humanos em Projetos de Software usando Algoritmos Genéticos Co-Evolucionários com Lógica Fuzzy

Monografia apresentada ao Programa de Pós-Graduação em Business Intelligence da PUC-Rio como requisito parcial para obtenção do título de Especialista em Business Intelligence. Aprovada pela Comissão Examinadora abaixo assinada.

Orientadores: Prof. André Vargas Abs da Cruz

Prof. Jesus Domech Moré

Rio de janeiro

05/11/2010

Agradecimentos

Agradeço aos meus orientadores, André Vargas e Jesus Domech, pois sem eles este trabalho não seria possível.

Agradeço aos meus pais e irmãos pelo apoio.

Agradeço a minha noiva pela paciência e apoio.

Agradeço a equipe de desenvolvimento de software com quem trabalho pelo apoio e pela imensa contribuição neste trabalho.

Sumário

1

Introdução

Com o crescimento do uso da tecnologia da informação no auxilio, controle e otimização dos processos de negócios das empresas a demanda de construção de software para estas áreas tem crescido muito. Com isso construir software de qualidade, com menos riscos, menos custos, com qualidade e mais rápidos tem se tornado cada vez mais estratégico.

Em função das mais diversas tecnologias, de diversos níveis proficiência nas mais variadas habilidades dos recursos humanos, dos mais diversos requisitos de software e em uma cobrança constante para menores custos e tempo de projeto, a tarefa de gerir os recursos para um projeto de software tornar-se árdua para o gerente de projeto.

No mercado há muito softwares para gerenciamento do ciclo de projeto com partes de gestão de recursos. Exemplos: Microsoft Project, Rational Plan, Enterprise Architect, TABA (COPPE-RIO) e etc.

No entanto dar inteligência a alocação destes recursos ao longo do ciclo do projeto, inferir melhorias, aperfeiçoar aspectos são características que poucos softwares possuem.

Outro aspecto importante a ser visto com essa crescente demanda por software é capacidade de se fazer estimativas, sejam elas de custo, prazo, dificuldade e tamanho de um software.

Há diversas técnicas para se mensurar estes aspectos no âmbito da gerência de projeto das quais podem ser citadas como principais: ***APF (Análise por Ponto de Função) e o Método COCOMO (Constructive Cost Model)***. Apesar de estas técnicas serem largamente usadas as mesmas apresentam diversos pontos fracos principalmente em refletirem os fatores subjacentes ao desenvolvimento do projeto, como observado por ***KEMERER F. (1987) [1]***, o que acaba por as levarem a não ter uma acurácia próxima do real.

Este trabalho propõe unir a capacidade dos Algoritmos Genéticos em resolver problemas de planejamento como ***Cruz (2003)[2]*** e ***Marco Aurélio e Thiago Souza[3]*** demonstraram com a capacidade de captar conceitos nebulosos

do pensamento humano da Lógica Fuzzy para a captação das particularidades dos recursos humanos disponíveis, a fim de auxiliar o gerente a planejar e estimar com maior acurácia.

Este trabalho começa por contextualizar o leitor sobre os problemas em se planejar um projeto de software com uma explanação sobre as técnicas hoje utilizadas e suas fraquezas. Logo em seguida será explanado o que são os Algoritmos Genéticos bem como a Lógica Fuzzy. Por fim serão mostrados o modelo de solução proposto neste trabalho e o estudo de caso da aplicação do modelo sobre um projeto real já construído por uma fábrica de software visando evidenciar a eficácia do modelo.

2

Planejamento de um projeto de software

Planejar um projeto de software é uma tarefa complexa, pois envolve muitas escolhas por parte do gerente de projeto sendo algumas delas até contraditórias. O gerente de projeto tem como meta construir um planejamento que combine recursos (humanos e maquinário) com tarefas a fim de aperfeiçoar o alcance de seus objetivos. Existem diversos autores que abordam o tema com ligeiras alterações de conceito como:

- ***PMI (Project Mangement Institute) (2004)[4] - Define gestão e planejamento de projetos, tão simplesmente, como sendo o processo através do qual se aplicam conhecimentos, capacidades, instrumentos e técnicas às atividades do projeto de forma a satisfazer as necessidades e expectativas dos diversos stakeholders envolvidos no mesmo.***
- ***Turner (1993)[5] - Entende que a gestão e planejamento de projetos é um processo através do qual um projeto é levado a uma conclusão e este possui três dimensões: objetivos (âmbito, organização, qualidade, custe e tempo); processo de gestão (planejar, organizar, implementar, controlar) e por fim níveis(integrativo e estratégico/táctico).***

Apesar destas pequenas diferenças um projeto pode ser representado por grafo acíclico e direcionado, consistindo em um conjunto de tarefas e $T = \{T1, T2, T3, ..., Tn\}$, e um conjunto de precedências $P = \{(Pij);\ i \neq j,\ 1 \leq i \leq n,\ 1 \leq j \leq n\}$, onde $Pij=1$ quando a tarefa i deve ser executada e finalizada antes da tarefa j iniciar, e zero caso contrário como o proposto por ***Marco Aurélio e Thiago Souza [3]***.Este forma de representação será explorada mais tarde neste trabalho quando for discutido o modelo de solução proposto.

Estimar tempo, esforço, custo, recursos é sem dúvida uma das maiores dificuldades que um gerente de projeto vai enfrentar, pois há uma infinidade de variáveis a se considerar. Existem muitas técnicas consagradas no mercado que tentam traçar essas estimativas para auxiliar o gerente em vislumbrar possíveis riscos, demandas e até mesmo a viabilidade do projeto. Dentre elas pode se destacar a APF e a COCOMO que serão brevemente descritas no próximo capítulo, pois são as mais utilizadas. Apesar de estas técnicas serem consagradas as mesmas deixam de captar os aspectos específicos daqueles quês estão a sua disposição no projeto, os recursos humanos. O Foco deste trabalho é exatamente introduzir no processo de estimativa as características daqueles que constroem o projeto para tornar estas estimativas o mais próximo da realidade.

2.1

Técnicas de estimativas de projeto

2.1.1 Análise por Pontos de Função – APF

A Análise por Pontos de Função (APF) mede o tamanho do software pela quantificação de suas funcionalidades, baseadas no projeto lógico ou a partir do modelo de dados segundo a visão e os requisitos do usuário final como descrito pelo ***IFPUG [6]***. Suas principais características são: ser independente da tecnologia, ser aplicável desde o início do sistema, apoiar a análise de produtividade e qualidade e estimar o tamanho do software com uma unidade de medida padrão.

Os seguintes passos devem ser observados para mensuração de tamanho do software utilizando esta abordagem segundo o ***IFPUG [6]***

1. Estabelecer o objeto da contagem (projetos de desenvolvimento, projetos de manutenção ou contagem de uma aplicação).
2. Determinar a fronteira de medição (a fronteira de medição deve ser sempre determinada sob o ponto de vista do usuário).
3. Contar as funções de dados, divididos em Arquivos Lógicos Internos (ALIs - que são grupos lógicos de dados mantidos dentro da fronteira da aplicação) e Arquivos de Interface Externa (AIEs – arquivos somente referenciados pela aplicação).
4. Contar as funções transacionais, divididos em Entradas Externas (EEs), Saídas Externas (SEs) e Consultas Externas (CEs).
5. Determinar o Fator de Ajuste, conjunto de 14 características que influenciarão a complexidade do software.
6. Determinar o tamanho do projeto (considera as funções de dados, transacionais, fatores de ajuste e tipo de projeto).

Cada função de dado ou transacional terá um peso diferente dependente de sua complexidade. Diversas tabelas baseadas na quantidade de elementos de dados, de registros e de arquivos referenciados são utilizadas para determinar a complexidade de cada função em Baixa, Média ou Alta.

O resultado da contagem de funções de dados e transacionais é uma medida chamada de contagem não ajustada (NoPF não ajustado), pois não

considera detalhes que afetam o produto e sua construção. O ajuste na mensuração é efetuado através do Fator de Ajuste determinado.

O Fator de Ajuste é calculado considerando outros fatores globais que afetam o tamanho funcional de um sistema. Estes fatores estão relacionados com características da aplicação:

Observações:

- Nível de influência de cada uma das 14 características, a serem observadas< varia de 0 a 5, correspondendo a uma escala de influência que parte da hipótese de nenhuma influência(0) até o grau de influência máxima (5).
- As características gerais do sistema podem influenciar no seu tamanho variando no intervalo de –35% a +35%. Isto implica em um intervalo de variação para o fator de ajuste da ordem de 0,65 a 1,35.

O fator de ajuste é responsável pela correção das distorções da etapa anterior. Baseia-se nas características gerais do sistema, correlacionando-as com uma tabela de referência que possui 14 itens, e determina o valor do nível de influência de cada item no dimensionamento do sistema.

Processo de cálculo:

1. Avaliar o impacto de cada uma das 14 características em relação ao sistema que está sendo avaliado, atribuindo pontuação de 0 a 5 para cada característica.
2. Calcular o nível de influência através da soma dos pontos obtidos em cada uma das 14 características.
3. Aplicar a seguinte fórmula: Fator de Ajuste = (NI * 0,01) + 0,65 onde: NI = somatório da pontuação atribuída a cada uma das 14 características, refletindo o nível de Influência global no dimensionamento do sistema.

As características gerais do sistema são:

1. Comunicação de dados
2. Funções distribuídas
3. Desempenho
4. Configuração do equipamento
5. Volume de transações
6. Entrada de dados on-line
7. Interface com o usuário.

8. Atualização on-line
9. Processamento complexo
10. Reusabilidade
11. Facilidade de implantação
12. Facilidade operacional
13. Múltiplos locais
14. Facilidade de mudanças (flexibilidade).

2.1.2 COCOMO – Constructive Cost Model

O COCOMO é um método que busca estimar esforço, prazo, custo e tamanho de equipe, necessários ao desenvolvimento do software, desde que se tenha como premissa, a dimensão do mesmo. Este método foi desenvolvido por ***Barry Boehm[7]***, a partir de uma pesquisa de campo compreendendo 63 projetos de grande porte, foi proposto em 1981***[7]***. Existem três modelos neste método:

- COCOMO Básico – versão aplicável a grande maioria de projetos de software: pequenos e médios projetos de software desenvolvidos internamente. A equação de esforço neste modelo foi definida como:

$$MM = C(KDSI)^k$$

  Onde :

  MM = Número de homens mês (152 horas trabalhadas).
  C = Uma constante.
  KDSI = Milhares de instruções de fontes entregues.
  K = Uma constante.

- COCOMO Intermediário – Adiciona fatores ao método Básico como restrições de hardware, qualificação, experiência do pessoal e etc que ponderam a equação.
- COCOMO Detalhado – Apresenta as mesmas técnicas do COCOMO intermediário exceto que o projeto é divido em quatro fases: Design do produto, Design detalhado, Codificação/ Teste de unidade e Testes de Integração sendo que os fatores ponderativos são estimados e aplicados a cada fase do projeto e não como um todo.

O COCOMO também categoriza os projetos de software em três tipos fundamentais: Modo Orgânico (Convencional), Modo Difuso e Modo Restrito.

Para cada modelo e tipo são definidas equações para estimativas de esforço, prazo e quantitativo de pessoal.

3

Técnicas de Inteligência Computacional

A inteligência computacional é o estudo do design de agentes inteligentes. Um agente é algo que funciona em um ambiente que faz alguma coisa. Agentes de incluir vermes, cães, termostatos, aviões, pessoas, organizações e sociedade.

Um inteligente agente é um sistema que atua de forma inteligente: O que ele faz é apropriado para sua circunstância e seu objetivo, é flexível a ambientes e metas em mudança, ele aprende a partir da experiência, ele faz escolhas adequadas dadas as limitações da percepção e computação finita.

O objetivo central científicos de inteligência computacional é entender os princípios que tornam possível o comportamento inteligente, em sistemas naturais ou artificiais. A principal hipótese é que o raciocínio é a computação. O objetivo central é a de engenharia especificar os métodos de design de artefatos inteligentes uteis.

Como ***David, Alan e Randy [8]*** descreveram acima, as técnicas de inteligência computacional são um grande trunfo para lidar com problemas extremamente complexos como, por exemplo, no caso de um planejamento de recursos humanos para software. A grande maioria para não dizer todas as técnicas computacionais é baseada em aspectos humanos e naturais como, por exemplo: Adaptação, raciocínio, percepção, evolução, aprendizado.

Neste trabalho duas dessas técnicas foram empregadas, são elas: Os Algoritmos Genéticos e a Lógica Fuzzy que serão detalhadas a seguir.

3.1

Algoritmos Genéticos

Os Algoritmos Evolucionários usam modelos computacionais dos processos naturais de evolução como uma ferramenta para resolver problemas. Apesar de haver uma grande variedade de modelos computacionais propostos, todos eles têm em comum o conceito de simulação da evolução das espécies através de seleção, mutação e reprodução, processos estes que dependem do "desempenho" dos indivíduos dessa espécie dentro da população ***[9]***.

Os Algoritmos Genéticos são uma classe particular de Algoritmos Evolucionários fundamentada principalmente por ***John Henry Holland[10]*** que usam técnicas inspiradas na biologia como hereditariedade, mutação, seleção natural e recombinação.

Um algoritmo genético pode tem funcionamento básico descrito ao algoritmo abaixo:

```
procedure algoritmo_genético
begin
    t = 0 ; primeira geração
    inicializa P(t) ; população inicial aleatória
    avalia P(t) ; calcula f(i) p/ cada indivíduo
    while (not condição_parada) do
    begin
        t = t + 1 ; próxima geração
        seleciona P(t) de P(t-1)
        altera P(t) ; crossover e mutação
        avalia P(t) ; calcula f(i) p/ cada indivíduo
    end
end
```

Como pode ser visto no algoritmo acima os passos para realização do funcionamento básico de um GA são:

1. Uma população inicial deve ser criada aleatoriamente ou por qualquer outra heurística. Em GA os membros de uma população são os cromossomos onde cada uma representa uma possível solução do problema em questão. Um cromossomo pode ter variados tipos de representação da solução do problema em questão como, por exemplo, binária, baseado em ordem e etc.
2. Avaliam-se cada individuo (cromossomo) através de uma função de avaliação onde esta a ponte de ligação entre os GA e o problema real. O valor apurado pela função é chamado de aptidão (fitness) do cromossomo. Essa aptidão tem pesos sobre muitas coisas em um GA como, por exemplo, que tem mais preferência em ser um dos pais da próxima geração além de definir de métrica de qualidade da solução encontrada pelo GA.
3. Ocorre um processo de seleção dos progenitores da próxima geração baseados na aptidão dando prioridade aqueles quem tem mais aptidão.
4. Ocorre o processo de recombinação e mutação dos filhos obtidos no passo 3.No âmbito dos GA`s a recombinação(cross-over) e a mutação são chamados de operadores e são eles os responsáveis por modificar o cromossomo introduzindo novas possibilidades de solução.
5. Recomeçar do passo 2.

Para se aplicar algoritmos evolucionários com sucesso em problemas com complexidade cada vez maior, torna-se necessário introduzir noções explícitas de modularidade nas soluções para que elas disponham de oportunidades razoáveis de evoluir na forma de subcomponentes co-adaptados ***(Potter & Jong, 2000) [11]***. Existem duas razões principais pelas quais algoritmos evolucionários convencionais não são totalmente adequados para resolver este tipo de problema. Em primeiro lugar, os algoritmos genéticos convencionais impedem, a longo prazo, a preservação de certos componentes da solução pois, por estarem codificados por completo em um indivíduo, eles são avaliados como um todo e apenas os subcomponentes que pertencem a indivíduos com avaliações altas serão preservados. Em segundo lugar, o fato da representação estar relacionada a uma solução completa e por não haver interações entre os membros da

população, não existe pressão evolucionária para a ocorrência de co-adaptação, ou seja, não existe pressão para a adaptação de um subcomponente dada a ocorrência de uma mudança em outro subcomponente ***(Cruz, 2003)[2]***.

Com finalidade de resolver este problema foi desenvolvido uma técnica que combina duas ou mais espécies (os subcomponentes da solução) em uma única solução, fazendo com cada espécie evolua separadamente sem a pressão de ter que se adaptar aos demais subcomponentes. Esta técnica é chamada de Algoritmos Genéticos Co-Evolucionários, e em particular para este trabalho o modelo cooperativo. A figura 1 descreve o modelo do mesmo.

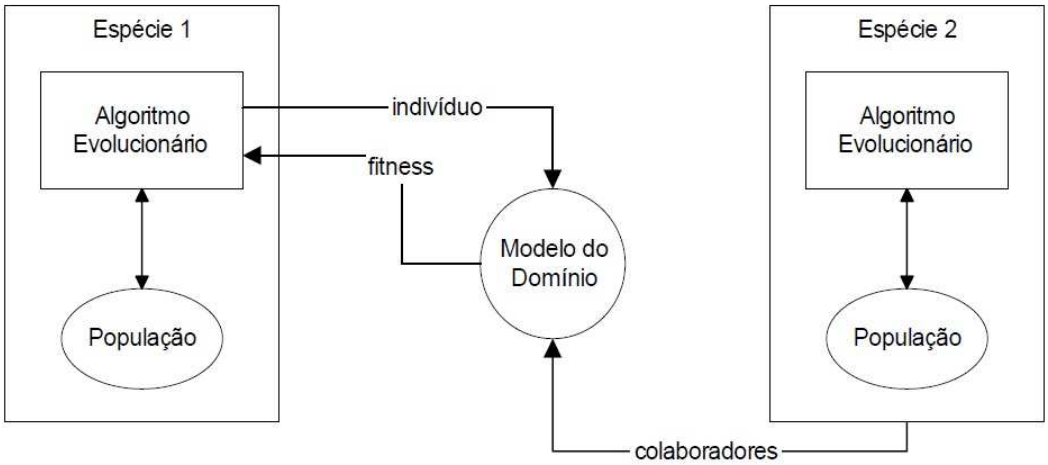


Figura 1 – Modelo Algoritmo Genético Co-Evolucionário Cooperativo

Cada espécie evolui em sua própria população (pois como já foi mencionado, elas são isoladas geneticamente) e se adaptam ao ambiente através de repetidas aplicações do algoritmo evolucionário. Para se calcular o fitness dos indivíduos de uma determinada espécie, deve-se submetê-lo ao modelo do domínio (que contém a função de avaliação) juntamente com um ou mais colaboradores de cada uma das outras espécies (de modo a formar a solução completa) ***(Cruz 2003) [2]***.

3.2

Lógica Fuzzy

Muitas vezes nós não conseguimos expressar com números reais com exatidão a representação de idéias como: muito quente, bom, pouco tempo e etc. O intelecto humano tem a capacidade de entender e raciocinar em cima destes conceitos imprecisos. A Lógica Fuzzy tem como finalidade o estudo dos princípios formais do raciocínio aproximado ***[12]***.

Para considerar as incertezas e ambigüidades existentes nos problemas em que o conhecimento do especialista é diretamente requerido, Fuzzy tem se provado ser uma ferramenta efetiva para levar em consideração o impreciso. A maioria destes problemas são complexos devido ao número de variáveis envolvidas e a dificuldade de quantificação destas variáveis. Por vezes, diversas destas grandezas têm seus valores estabelecidos através de processos subjetivos.

A Lógica Fuzzy é formalmente definida pela Teoria de Conjuntos Fuzzy concebida por ***Lotfi A. Zadeh[13]*** com a finalidade de definir um modelo matemático para tratar as informações de caráter impreciso ou vago.

Na teoria de conjuntos clássica, um subconjunto C de um conjunto S é definido como um mapeamento de elementos de S nos elementos do conjunto {0, 1}. Esse mapeamento pode ser expresso por pares ordenados, em que o primeiro elemento do par é representativo de um dos elementos do conjunto S, e o segundo, um elemento do conjunto {0, 1}. Esses valores 1 e 0 representam respectivamente, a pertinência e a não-pertinência de um elemento de S em C (ou a verdade ou falsidade da afirmação de que um elemento de S pertence a C).

Analogamente, um subconjunto fuzzy F de um conjunto S pode ser definido como um conjunto de pares ordenados, em que o primeiro elemento do par pertence a S e o segundo elemento [0,1]. Essa união de valores define um mapeamento entre os elementos do conjunto S e os valores no intervalo [0,1]. O valor 0 é usado para representar a completa pertinência. Os valores entre 0 e 1

são usados para representar graus de pertinência intermediários dos elementos do conjunto S em F ***[12].***

Um conjunto fuzzy A é caracterizado pelo par $\left(x, \mu_A(x)\right)$, no qual x é a variável, continua ou discreta, do universo em estudo, e μ_A é uma função cuja imagem está contida em [0,1] ***[12]***.

A Lógica Fuzzy é muito extensa, porém no intere deste trabalho duas áreas da mesma são necessárias: Números Fuzzy e Aritmética Fuzzy.

Pode-se definir um número fuzzy como um conjunto fuzzy normalizado com função de pertinência convexa e que tem como universo de discurso uma reta real ***[12]***. Existem diversos tipos de números fuzzy ***(KAUFMANN e GUPTA) [14],*** porém para este trabalho foram utilizados apenas os números fuzzy triangulares (no inglês Triangle Fuzzy Numbers - TFN).

Os números fuzzy triangulares têm o seguinte modelo matemático:

$$f(x; a_1, a_2, a_3) \begin{cases} 0; & x < a_1 \\ (x - a_1)/(a_2 - a_1) & a_1 \leq x < a_2 \\ (a_3 - x)/(a_3 - a_2) & a_2 \leq x < a_3 \\ 0; & a_3 \leq x \end{cases}$$

Restrição $\quad a_1 < a_2 < a_3$

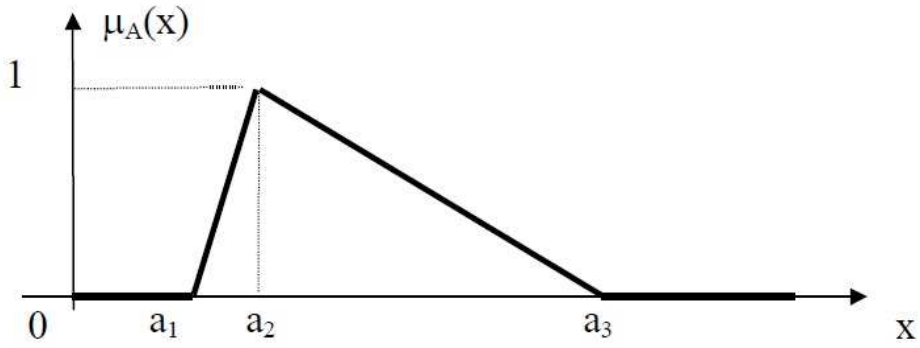


Figura 2 – Representação gráfica de um Número Fuzzy Triangular - TFN

A base dos TFN é denominada suporte do conjunto fuzzy, também conhecido amplitude**.** A amplitude está diretamente associada à confiança que se tem no valor da função de pertinência da variável x. Ou seja, quanto menor a amplitude do intervalo, maior confiança dos dados; e de modo análogo, quanto maior a amplitude do intervalo, menor a confiança nos dados.

Os valores de a1, a2, a3 podem se interpretados como:

- a1 - valor pessimista
- a2 - valor provável
- a3 - valor otimista

Existem diversas operações aritméticas fuzzy análogas as operações aritméticas comuns, contudo serão apresentadas as pertinentes no intere deste trabalho.

A multiplicação de um número fuzzy por uma constante escalar C é $(C \cdot a_1, C \cdot a_2, C \cdot a_3)$, que é número fuzzy triangular. Seja A um TFN igual a (2,4,5) e C igual a 2 então teremos um TFN resultante da multiplicação igual a (4,8,10). A divisão de um TFN por uma constante C pode ser vista como a operação inversa da multiplicação, como por exemplo: Seja A um TFN igual a (2,4,6) e C igual 2 então teremos TFN resultante da divisão igual a (1,2,3) ***[12]***.

O cálculo para se estabelecer a distância entre dois números fuzzy é largamente estudado, assim sendo, há diversas formas de se calcular a mesma, cada forma tem prós e contras que podem levar a situações que contrariam a lógica humana como, por exemplo, dois números fuzzy que graficamente estão próximos são dados como distantes pelos métodos.

O trabalho proposto por ***Tran e Duckstein[15]*** apresenta um método que permite evitar muitas anomalias que ocorrem em métodos “convencionais” de cálculo de distância entre dois números fuzzy. A equação proposta por eles abaixo denota o cálculo de distância entre dois números fuzzy triangulares A(a1,a2,a3) e B(b1,b2,b3) com pesos iguais para diferentes alfa-cortes.

$$(a_2-b_2)^2+\frac{1}{2}(a_2-b_2)[(a_3+a_1)-(b_3+b_1)]+\frac{1}{9}[(a_3-a_2)^2+(a_2-a_1)^2+(b_3-b_2)^2+(b_2-b_1)^2]$$
$$-\frac{1}{9}[(a_2-a_1)(a_3-a_2)+(b_2-b_1)(b_3-b_2)]+\frac{1}{6}(2a_2-a_1-a_3)(2b_2-b_1-b_3)$$

Figura 3 – Equação da distância entre dois números fuzzy

Ordenar números fuzzy não é uma tarefa simplória. Existem diversos estudos nesta área, afinal de contas, ordenar números fuzzy pode significar uma tomada de decisão acertada ou não quando se tem diversos critérios representados por esses números. No intere deste trabalho foi escolhida a forma descrita por ***KAUFMANN e GUPTA[14]*** que é definido por três critérios de avaliação. Esses critérios são usados um seguido do outro caso seus antecessores não tenham sido suficientes para a avaliação.

O primeiro critério consiste em definir, para um número fuzzy triangular, um “ordinary representative”, ou seja, definir um número real que represente o número fuzzy triangular correspondente. Este número é calculado conforme a equação na figura abaixo:

$$\hat{A} = \frac{a_1 + 2a_2 + a_3}{4}$$

Figura 4 – Representação do número fuzzy em um número real

Caso ocorra um empate no primeiro critério o segundo critério a ser utilizado consiste em usar o número a2, cuja pertinência é igual a um. O número fuzzy triangular que possuir o maior a2 será considerado o maior.

Por fim, ainda havendo empate no segundo critério, o terceiro critério a ser utilizado é divergência entre o valor de a2 será considerado maior aquele que possuir maior divergência entre o seu valor de a2. A divergência de um TFN é calculada fazendo-se a3 - a1.

O último aspecto da lógica fuzzy usado neste trabalho é o grau de inclusão. O grau de inclusão mensura o quanto um conjunto fuzzy está incluso em outro. A equação que denota este grau foi proposta por ***KOSKO, B[16]*** e pode ser vista na figura abaixo:

Card(A) = Cardinalidade do conjunto fuzzy A que é igual ao somatório de todos os graus de pertinência do conjunto fuzzy A.

$$S(A,B) = \frac{1}{Card(A)}\left\{Card(A) - \sum_{x \in X} \max\{0, A(x) - B(x)\}\right\}$$

Figura 5 – Equação do grau de inclusão

4

Metodologia

Neste trabalho houve a necessidade de se obter muita informação oriunda diretamente das pessoas envolvidas nos projetos usados como estudo de caso. O modo como estas informações foram captadas, o que elas significam e como foram tratadas para realização deste trabalho é o tema deste capítulo.

Neste trabalho foram usados dois projetos reais de uma empresa de desenvolvimento de softwares para área do mercado financeiro. As equipes que desenvolveram estes projetos eram compostas de profissionais que desempenham um ou mais dos seguintes papéis: gerente de projeto, analista de sistemas, analista de negócio, arquiteto de software e desenvolvedor.

Os gerentes de projetos foram os responsáveis por planejar o curso do projeto, por decidir quem participaria do mesmo, por decidir qual seria seu escopo, por delegar quem e o que os *stakeholders* (envolvidos no projeto) fariam.

Os analistas de sistemas foram os responsáveis pelo levantamento de requisitos junto ao cliente, ou seja, que necessidades os clientes tinham e que problemas que eles possuíam na área alvo do projeto o software os ajudaria a sanar ou suprir ajuda.

Os analistas de negócio foram os especialistas no na área de negócio do cliente e ajudavam os analistas a definir as soluções para o cliente e orientavam os desenvolvedores em possíveis dúvidas em relação à área de negócio alvo do projeto.

Os arquitetos de software foram os especialistas nas tecnologias usadas para construir as soluções de negócio propostas. Eles tiveram como propósito também definir a arquitetura global do software de modo a tornar viável do ponto de vista de desempenho, segurança, escalabilidade e usável. Eles definiram as tecnologias que seriam utilizadas e por também ajudaram a sanar possíveis

dúvidas dos desenvolvedores sobre as melhores práticas de como desenvolver o software para o projeto em questão.

Os desenvolvedores foram os responsáveis por codificar o software para o projeto, utilizando tudo o definido pelos analistas de sistema e arquitetos.

As tarefas foram definidas pelo gerente de projeto tomando como base todo o trabalho dos analistas de sistemas e dos arquitetos de software.

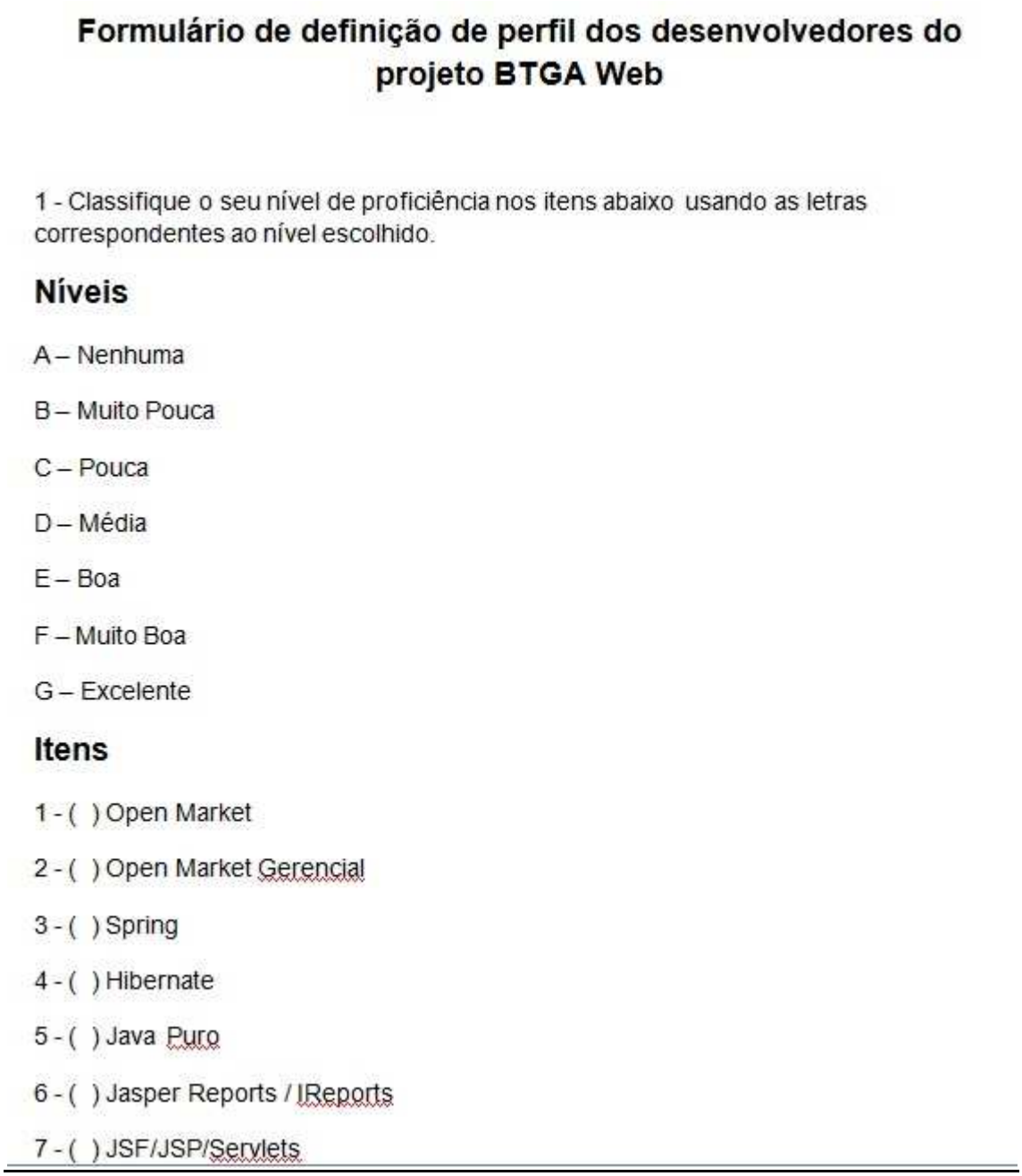
No intere deste trabalho fora pedido a todas essas pessoas que definissem algumas métricas e respondessem alguns questionários.

Fora pedido aos analistas de negócio e arquitetos que definem que habilidades (de agora em diante eles serão chamados de especialistas neste trabalho) seriam necessárias para se completar cada tarefa. Para cada habilidade foi requisitado que os especialistas também definissem o nível de proficiência ideal para a resolução da tarefa. Essa métrica foi definida como um valor lingüístico fuzzy sendo referida neste trabalho como NPI de agora em diante. Seus valores lingüísticos e respectivas representações em número fuzzy triangulares (TFN) possíveis são:

- Valor lingüístico: Nenhum / TFN: (0,0,0)
- Valor lingüístico: Muito pouca / TFN: (0,15; 0,25; 0,35)
- Valor lingüístico: Pouca / TFN: (0,30;0,40;0,50)
- Valor lingüístico: Média / TFN: (0,45;0,55;0,65)
- Valor lingüístico: Boa / TFN: (0,60;0,70;0,80)
- Valor lingüístico: Muito Boa / TFN: (0,75;0,85;0,95)
- Valor lingüístico: Excelente / TFN: (0,9;0,95;1)

Como em cada equipe havia apenas um analista de negócio e um arquiteto de software não fora necessário um tratamento dos dados para combinar múltiplas opiniões.

Fora pedido aos desenvolvedores que preenchem dois formulários; o primeiro teve como objetivo descrever as habilidades dos desenvolvedores e para tal também se valeu da lógica fuzzy com seus termos lingüísticos. O formulário contém todas as habilidades relacionadas ao projeto em que eles estão participando. Este formulário também contém perguntas para captar o valor hora de cada desenvolvedor. Um trecho do formulário pode ser visto abaixo:



Formulário de definição de perfil dos desenvolvedores do projeto BTGA Web

1 - Classifique o seu nível de proficiência nos itens abaixo usando as letras correspondentes ao nível escolhido.

Níveis

A – Nenhuma

B – Muito Pouca

C – Pouca

D – Média

E – Boa

F – Muito Boa

G – Excelente

Itens

1 - ( ) Open Market

2 - ( ) Open Market Gerencial

3 - ( ) Spring

4 - ( ) Hibernate

5 - ( ) Java Puro

6 - ( ) Jasper Reports / IReports

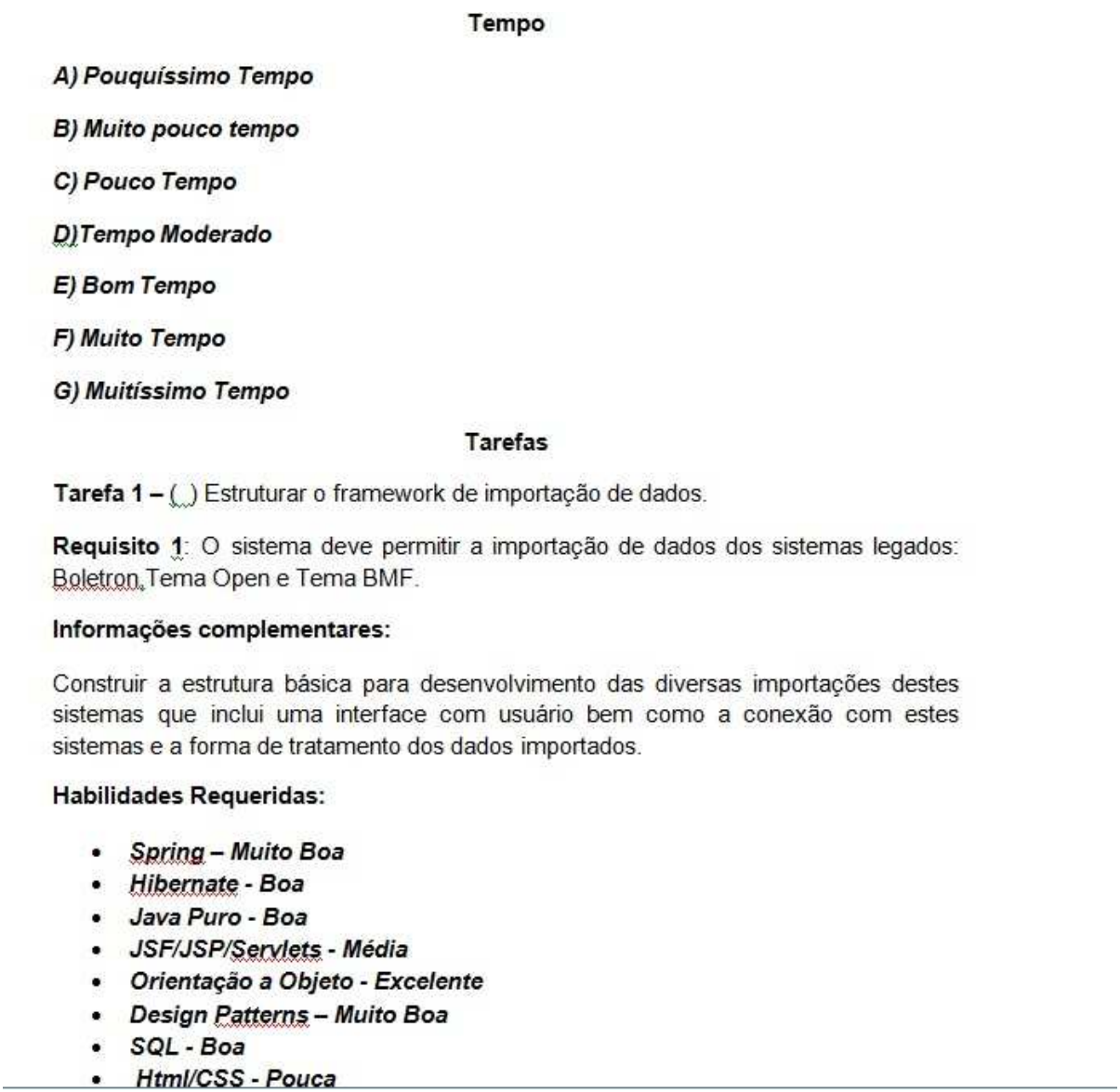
7 - ( ) JSF/JSP/Servlets

Figura 6 - Formulário de perfil de desenvolvedores

Os números fuzzy triangulares associados a cada valor lingüístico são os mesmos definidos para os NPIs anteriormente.

O segundo formulário dados aos desenvolvedores tem como objetivo captar como eles definem o tempo que levariam para concluir uma tarefa baseada nas habilidades requeridas pelas mesmas bem como seus NPIs. Este formulário é um dos componentes do modelo de estimativa de tempo

aproximado (Acrônimo – META) que é uma das peças centrais do modelo proposto por este trabalho detalhado no capitulo cinco. A figura abaixo mostra um trecho do formulário:



Tempo

A) Pouquíssimo Tempo

B) Muito pouco tempo

C) Pouco Tempo

D)Tempo Moderado

E) Bom Tempo

F) Muito Tempo

G) Muitíssimo Tempo

Tarefas

Tarefa 1 – ( ) Estruturar o framework de importação de dados.

Requisito 1: O sistema deve permitir a importação de dados dos sistemas legados: Boletron, Tema Open e Tema BMF.

Informações complementares:

Construir a estrutura básica para desenvolvimento das diversas importações destes sistemas que inclui uma interface com usuário bem como a conexão com estes sistemas e a forma de tratamento dos dados importados.

Habilidades Requeridas:

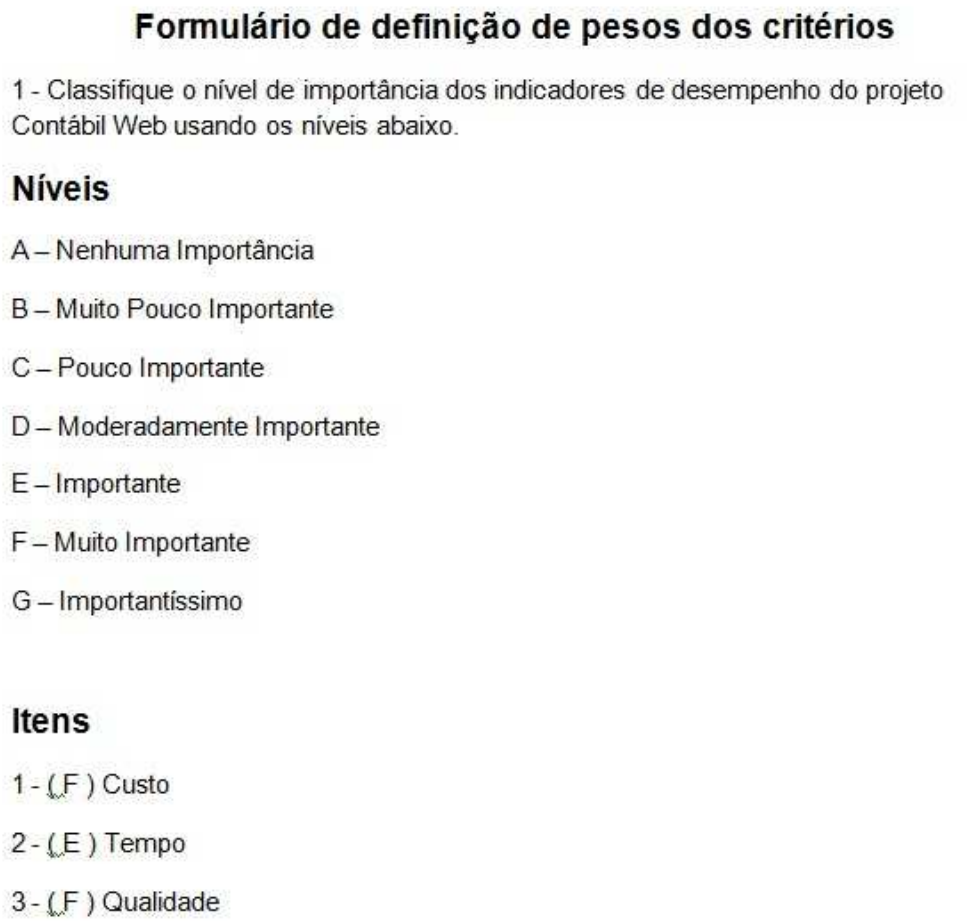
- ***Spring – Muito Boa***
- ***Hibernate - Boa***
- ***Java Puro - Boa***
- ***JSF/JSP/Servlets - Média***
- ***Orientação a Objeto - Excelente***
- ***Design Patterns – Muito Boa***
- ***SQL - Boa***
- ***Html/CSS - Pouca***

Figura 7 – Trecho do formulário de avaliação do tempo

Ao gerente de projeto fora pedido que fossem preenchidos dois formulários. O primeiro formulário questionou o gerente sobre os pesos dos critérios de avaliação de desempenho de um projeto. Os critérios de desempenho tinham seus pesos avaliados com variáveis lingüísticas que estão descritas abaixo:

- Valor lingüístico: Nenhuma importância / TFN: (0,0,0)
- Valor lingüístico: Muito pouco importante / TFN: (0,15; 0,25; 0,35)
- Valor lingüístico: Pouco importante / TFN: (0,30;0,40;0,50)
- Valor lingüístico: Moderadamente importante / TFN: (0,45;0,55;0,65)
- Valor lingüístico: Importante / TFN: (0,60;0,70;0,80)
- Valor lingüístico: Muito Importante / TFN: (0,75;0,85;0,95)
- Valor lingüístico: Importantíssimo / TFN: (0,9;0,95;1)

Um trecho do formulário pode ser visto na figura abaixo:



Formulário de definição de pesos dos critérios

1 - Classifique o nível de importância dos indicadores de desempenho do projeto Contábil Web usando os níveis abaixo.

Níveis

A – Nenhuma Importância

B – Muito Pouco Importante

C – Pouco Importante

D – Moderadamente Importante

E – Importante

F – Muito Importante

G – Importantíssimo

Itens

1 - (F) Custo

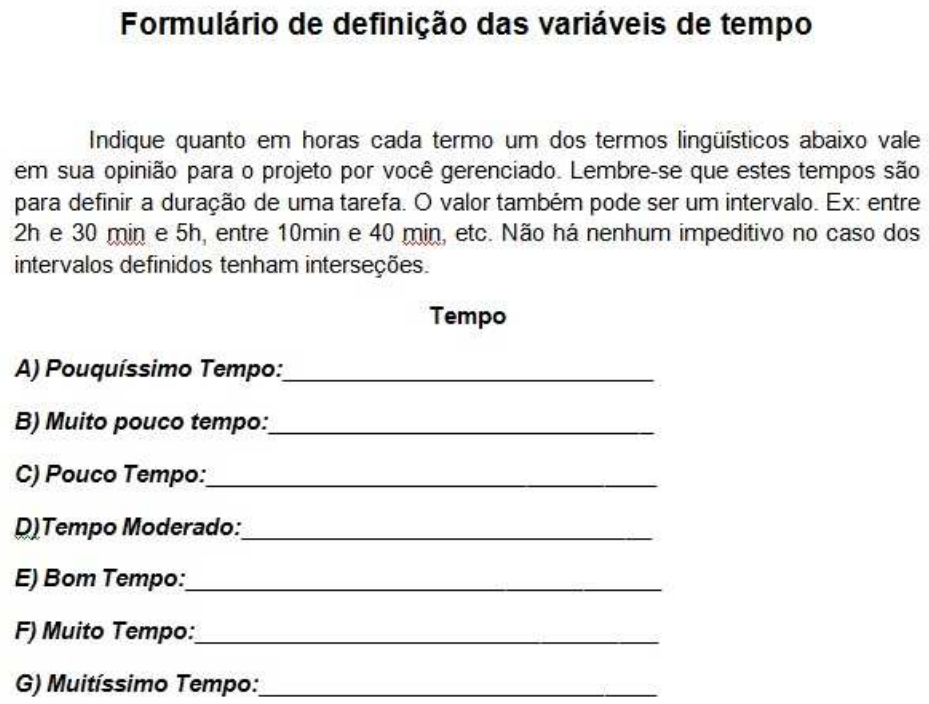
2 - (E) Tempo

3 - (F) Qualidade

Figura 8 – Trecho do formulário de avaliação dos pesos dos critérios de desempenho de um projeto

O segundo formulário questionou o gerente quanto à definição das variáveis lingüísticas do tempo de duração de uma tarefa no projeto por ele gerenciado estudado neste trabalho. As variáveis foram valoradas tanto em horas quanto em minutos pelos gerentes sendo todas elas transformadas em

minutos para a utilização neste trabalho. A figura abaixo mostra um trecho do formulário:



Formulário de definição das variáveis de tempo

Indique quanto em horas cada termo um dos termos lingüísticos abaixo vale em sua opinião para o projeto por você gerenciado. Lembre-se que estes tempos são para definir a duração de uma tarefa. O valor também pode ser um intervalo. Ex: entre 2h e 30 min e 5h, entre 10min e 40 min, etc. Não há nenhum impeditivo no caso dos intervalos definidos tenham interseções.

Tempo

A) Pouquíssimo Tempo:_____

B) Muito pouco tempo:_____

C) Pouco Tempo:_____

D)Tempo Moderado:_____

E) Bom Tempo:_____

F) Muito Tempo:_____

G) Muitíssimo Tempo:_____

Figura 9 – Trecho do formulário de definição dos valores das variáveis lingüísticas do tempo de duração de uma tarefa

5

Modelagem

A modelagem da solução para o problema de planejamento proposta neste trabalho será baseada em Algoritmos Genéticos Co-Evolucionários tendo como base o modelo proposto por ***(Cruz 2003) [2]*** com sua função de avaliação usando os aspectos, inerentes a cada recurso humano disponível para o projeto de software, captados pela Lógica Fuzzy em uma tentativa de aumentar a acurácia das diversas estimativas em torno do planejamento, em contraste aos métodos convencionais que ignoram esses aspectos. Nos capítulos que se seguem será descrito o modelo em detalhes bem como a metodologia de captação dos dados usados.

5.1

Modelo de Solução

Como o proposto por ***(Cruz 2003) [2]*** o planejamento foi encarado como um problema de ordem com restrição de precedência. Para não acarretar em perda de desempenho por parte do algoritmo com reparos e penalizações dos cromossomos partiu-se do principio que o algoritmo apenas geraria soluções legais que não violassem a ordem de precedência. Para tal um modelo baseado em grafos dirigidos fora utilizado.

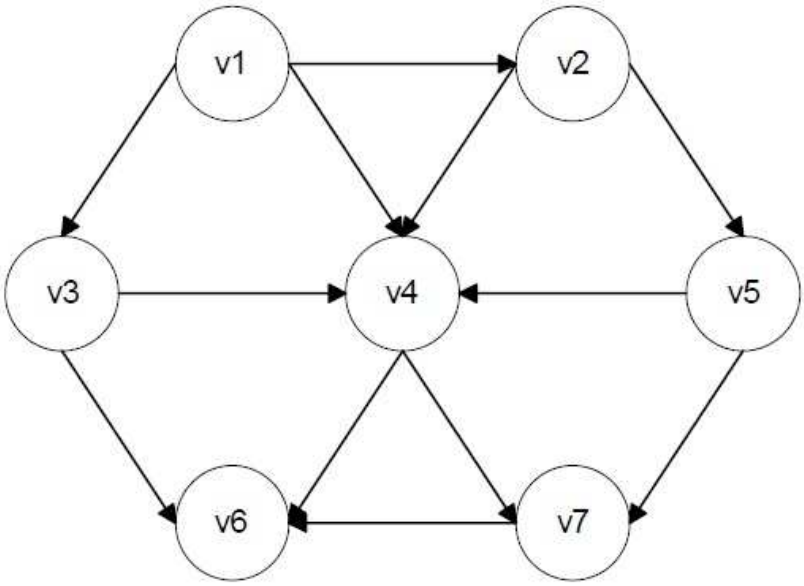


Figura 10 – Exemplo de grafo dirigido para definição de precedências.

Abaixo está o pseudocódigo do algoritmo usado para gerar uma solução de planejamento viável, ou seja, que siga as restrições impostas. Fica evidente que com o algoritmo que para que seja possível construir a solução o grafo não pode ser cíclico.

```
procedure: gerar planejamento válido
        input: grafo dirigido
        while (existir vértice)
                if (todo vértice tem um precedente) then grafo inválido: stop
                else    selecione um vértice v com a maior prioridade entre os que
não                     têm precedentes
                planejar v
                remover v do grafo e todos as ligações que partem dele
        end while
end procedure
```

Figura 11 – Pseudocódigo da geração de um planejamento válido

Visando a otimização do algoritmo para definir a prioridade das tarefas (vértices do grafo) foi especificado que as ordens dos genes no cromossomo definiram-se as prioridades dos mesmos evitando que o algoritmo se tivesse que percorrer todo o cromossomo verificando-se a existência de precedentes para cada tarefa o que seria custoso computacionalmente. A cada tarefa estão vinculadas as habilidades necessárias para realização das mesmas bem como os níveis de proficiências ideais (NPI) nessas habilidades definidos pelos os especialistas envolvidos no projeto. Para que uma tarefa tenha um tempo de realização seja o esperado por eles tanto quanto o grau de qualidade do trabalho realizado, então foi definida uma espécie chamada “Tarefa” com cromossomo descrito na figura 5.

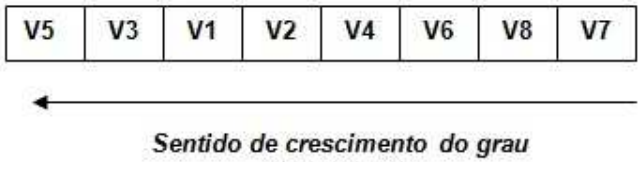


Figura 12 – Exemplo do cromossomo da espécie Tarefa

Os operadores genéticos usados para esta espécie foram os comumente usados em problemas de ordem como o problema do Caixeiro Viajante, Ex: Mutação Baseada em Ordem, Recombinação de Adjacências, Crossover Baseado em Ordem e etc..

Como o proposto por ***(Cruz 2003) [2]*** fora definida uma espécie chamada de “Recurso” que irá evoluir paralelamente da espécie Tarefa e será responsável por otimizar a lista a alocação de recursos para as tarefas. O cromossomo desta espécie pode ser descrito da seguinte forma: cada gene está associado a uma tarefa como na espécie Tarefa, porém cada gene contém uma lista de recursos ordenados pela prioridade que podem ser utilizados naquela tarefa. Associado a cada recurso estão seus níveis de proficiência (NP) em cada uma das habilidades necessárias para a realização do projeto (caso o recurso não possua uma das habilidades, ela apenas é omitida da lista referente ao recurso) o valor hora (VH) de trabalho do recurso e modelo de estimativa de tempo aproximado (META). O cromossomo pode ser exemplificado pela figura 6.

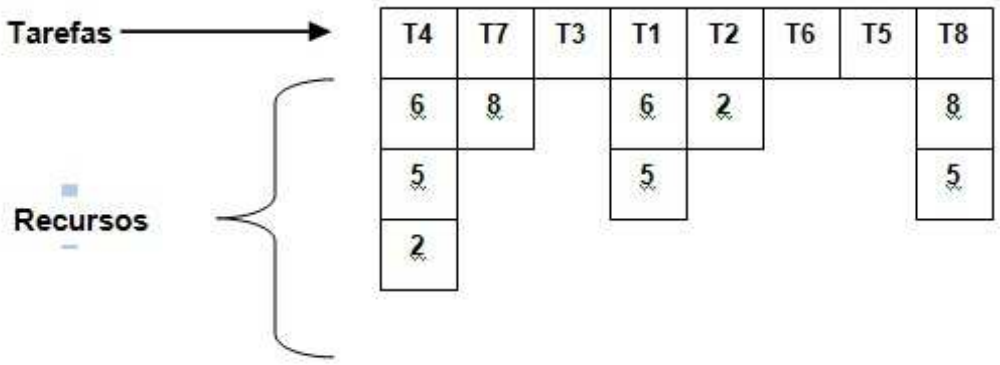


Figura 13 – Exemplo do cromossomo da espécie Recurso

Os operadores genéticos de crossover que atuaram nesta espécie não trocaram *locus* do gene entre os progenitores, más sim atuaram em cada lista de recursos separadamente. Em outras palavras quando dois progenitores são selecionados, primeiro se faz o crossover entre as listas da primeira tarefa de ambos, depois o mesmo procedimento é adotado em cada tarefa. De modo análogo os operadores de mutação nunca mudam o material genético entre as tarefas apenas atuando na lista de recursos de cada uma das tarefas.

As espécies trabalharam em um modelo cooperativo (discutido no capítulo 3.1) de modo a alcançar a solução do problema.

Como o individuo que representa a solução está dividido em duas espécies foi necessário o uso de um decodificador para transformar as duas espécies na solução. O pseudocódigo do decodificador pode ser exemplificado na figura 7.

```
procedure decodificador
    cromossomaT = cromossoma da espécia "Tarefa"
    cromossomaR = cromossoma da espécie "Recurso"
    grafo = grafo com restrições de precedência
    j = 0
    while ( J < tamanho(cromossomaT) )
        planejar tarefa j usando cromossomaT e grafo
        selecionar recurso usando cromossomaR
    end while
end procedure
```

Figura 14 – Pseudocódigo do decodificador de solução

Este trabalho propôs uma alteração na alocação de recursos do modelo proposto por ***Cruz [2] e Marco Aurélio e Thiago Souza[3]*** em se partir do principio que os tempos das tarefas já estivessem sido estimados independente dos recursos alocado para elas. No intere do objetivo deste trabalho em introduzir os aspectos dos recursos humanos disponíveis para projeto no algoritmo de planejamento, fora utilizado uma estimativa de tempo dinâmica para tarefa de acordo com o recurso alocado para ela através do Modelo de Previsão de Tempo Aproximado (META) do recurso.

O *META* é um modelo desenvolvido pelo autor deste trabalho para estimar aproximadamente o tempo que o recurso levaria para cumprir uma tarefa dada sua natureza. O *META* se valeu de termos lingüísticos para a definição de tempo e dos NPI representados por números fuzzy triangulares (TFN). Para exemplificar a lógica por trás do *META* será descrito os passos para elaboração do mesmo bem como o mecanismo usado por ele para estimar o tempo.

Através de 5 tarefas do projeto a ser planejado e com as habilidades requeridas com seus respectivos NPI definidos(explicados no capítulo anterior)

fora pedido aos recursos que estimassem o tempo que cada um deles levaria para cumprir cada uma destas 5 tarefas através de termos lingüísticos.

Diante desta definição o META segue os seguintes passos para a determinação do tempo de uma tarefa X qualquer a ser planejada:

1. Encontrar tarefa do formulário mais semelhante à tarefa x em questão através da comparação da soma das distâncias de cada NPI.
2. Encontrar o grau de inclusão do padrão de habilidade e NPI da tarefa X com a tarefa semelhante encontrada.
3. Usar grau de inclusão como usado como um fator de variação do tempo estimado para tarefa do formulário para estimar o tempo da tarefa X.

O grau de inclusão é usado na equação:

T(x) = T(s) / gi

Onde: T(x) é o tempo da tarefa em questão, T(s) é o tempo estimado da tarefa semelhante e gi é o grau de inclusão.

Como o visto no capítulo 3 o grau de inclusão é maior que 0 e menor ou igual a 1 , sendo assim, o tempo estimado para tarefa X será no mínimo igual ao estimado para tarefa semelhante quando o grau alcançar 1 e penalizado com crescimento a medida que esse grau tenda a zero.

Por fim o indivíduo decodificado e oriundo da cooperação dos cromossomas das duas espécies é avaliado por 3 critérios dado pela equação abaixo :

$$F = Validade * (\frac{W_t}{TP} + \frac{W_c}{CE} + \frac{W_q}{QE})$$

Validade representa se o planejamento está válido recebendo um como valor ou inválido recebendo zero como valor. Por um planejamento válido entende-se que todas as tarefas têm pelo menos um recurso alocado.

Wt, Wc e Wq representam os pesos do critérios de avaliação informados pelo gerente do projeto devidamente defuzzificados.

TP representa o tempo estimado do planejamento dado pela soma dos tempos das tarefas resultantes da alocação de recursos em conjunção do *META*.

CE representa o custo estimado do planejamento dado pela soma do valor hora de cada recurso alocado do projeto multiplicado pelos seus respectivos tempo de alocação.

QE representa a qualidade estimada do projeto. Essa medida é dada pela soma dos graus de inclusão do conjunto de NP do recurso no conjunto de NPI da tarefa por ele alocado dividido pelo total de tarefas. Como os NPI são uma estimativa dos especialistas do projeto para que a tarefa seja feita com uma qualidade aceitável em um tempo hábil, quanto mais esse valor tender a 1 melhor será avaliação.

6

Conclusão e Estudo de Caso

Para o estudo de caso foi construída uma ferramenta usando o framework GACOM para utilização de algoritmos genéticos na plataforma Microsoft – Dot Net. Essa ferramenta teve como objetivo realizar o melhor planejamento para o projeto real usado neste trabalho usando os critérios de avaliação já explicados nos capítulos anteriores.

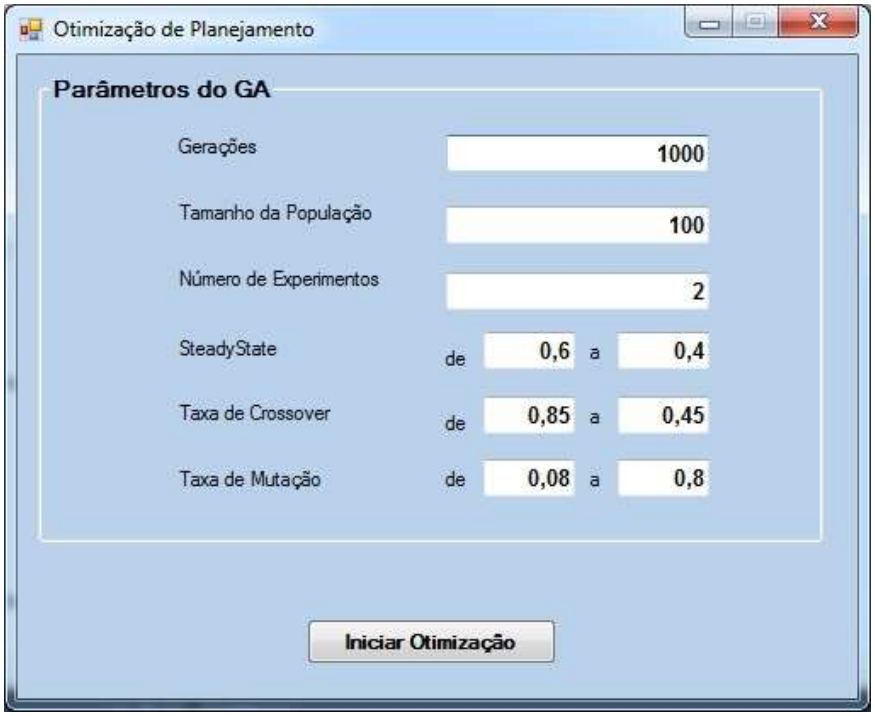


Figura 15 – Interface gráfica da ferramenta de otimização

Os dados utilizados foram coletados da ferramenta de Time Sheet da empresa dona do projeto real utilizado neste trabalho. Boa parte das estruturas de dados deste software foi aproveitada sendo acrescidas novas estruturas requeridas para a solução do problema. O modelo final dessas estruturas de dados pode ser visto nas figuras abaixo.

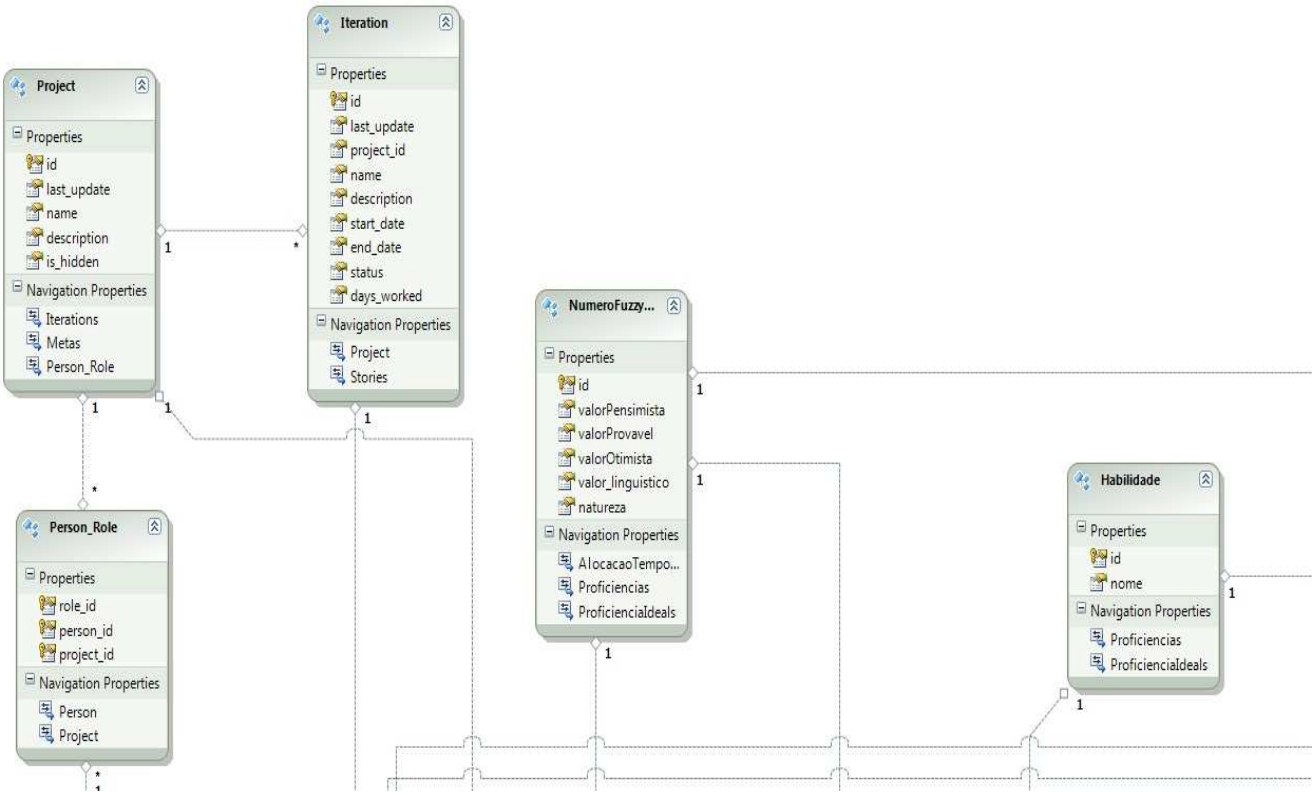


Figura 16 – Modelo de estrutura de dados da ferramenta

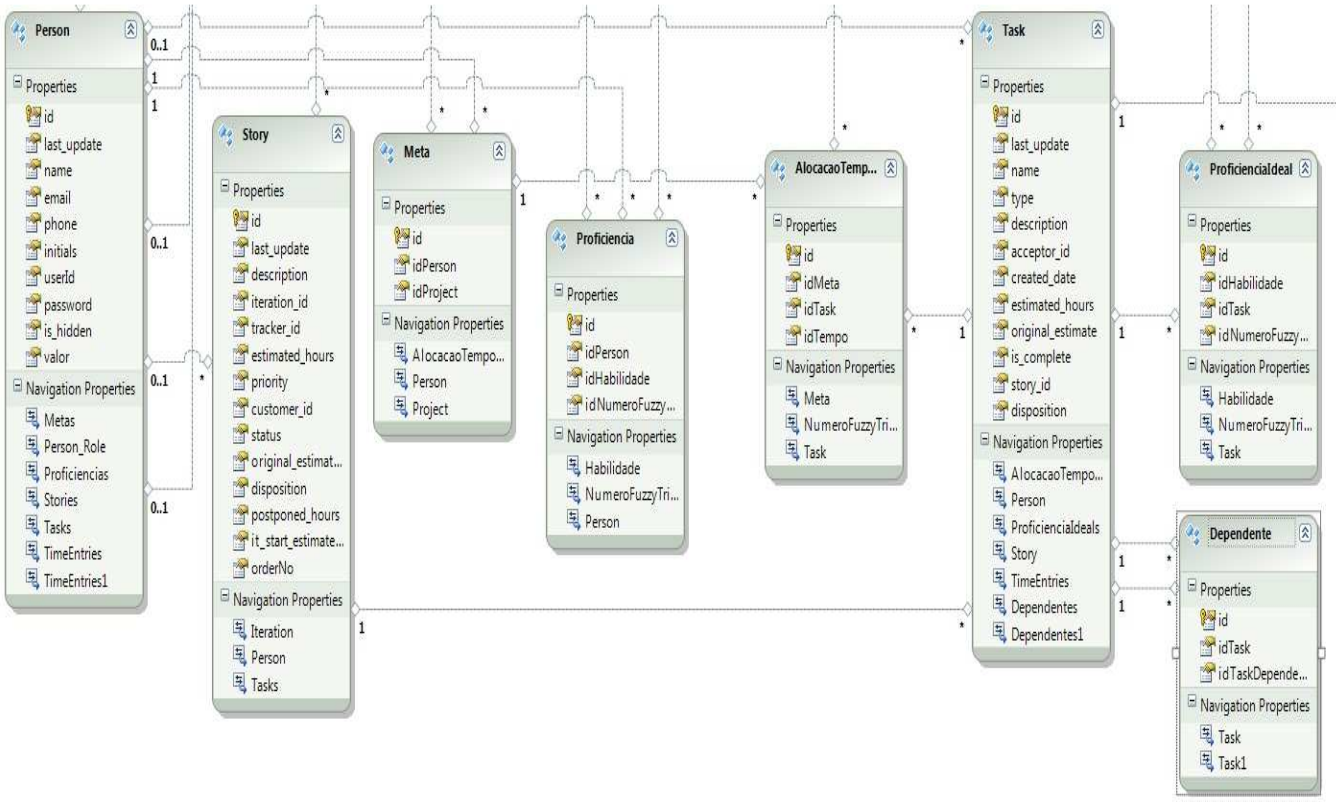


Figura 17 – Modelo de estrutura de dados da ferramenta

O melhor planejamento obtido pela ferramenta foi comparado com o planejamento real utilizado no projeto avaliado. O planejamento real teve como base a APF (Análise de pontos de função). A empresa tem como métrica um tempo 13,5 horas por ponto de função em projetos semelhantes ao projeto estudado. O planejamento obtido pela ferramenta teve uma duração de tempo em torno de 20% menos que o planejamento real, com um custo de em torno de 15% menos.

A qualidade não pode ser comparada com o projeto real, pois no projeto real o cliente final do projeto teria que avaliar o mesmo o que não ocorreu, pois o projeto ainda não foi entregue ao próprio.

O melhor planejamento foi obtido usando os operadores de mutação de inversão de posição e o de troca juntamente com os operadores de crossover de mapeamento parcial (PMX) e operador de ciclo (CX). Fora utilizado taxas de mutação e crossover interpoladas com inicio 8% e fim 50% para mutação e inicio 100% e fim 50% para o crossover com 50% de chance de seleção para ambos operadores. Foi realizada uma média de 100 gerações com 4 experimentos

utilizando 200 indivíduos cada uma. O Steady State utilizado teve uma interpolação entre 20% e 10% aproximadamente.

Este projeto teve como objetivo utilizar às características dos especificas dos colaboradores envolvidos no projeto a fim de deixar as métricas de estimativas o mais acuras possível contrastando com os métodos comumente utilizados nesta área que simplesmente descartam o individual em detrimento de médias e generalizações.

Como só possível utilizar um projeto real como base de comparação não foi possível analisar padrões de comportamento do algoritmo como variações do número de participantes, número de restrições de precedência das tarefas, número de tarefas, mais de um especialista por área o que acarretaria em métodos de mistura de opniões. Ficando esta analise para trabalhos futuros.

Em trabalho futuros pode-se utilizar métodos de captura mais elaborados como Redes Neurais, Sistema de inferência Fuzzy completos ou até mesmo uma mistura dos dois para se extrair as características e forma de pensar dos colaboradores do projeto.

Bibliografia

[1] Kemerer, Chirs F. “An Empirical Validation of Cost Software Estimation Models”, Comunications of the ACM, Vol . 30, no. 5, May 1987.

[2] Cruz, A. V. A. (2003), “Otimização de Planejamentos com Restrições de Precedência usando Algoritmos Genéticos e Co-Evolução Cooperativa”, Tese de Mestrado, DEE, PUCRio.

[3] Marco Aurélio C. Pacheco & Thiago Souza M. Guimarães. “Algoritmos Genéticos para Gerência de Projetos” – Technical Notes in Computacional Intelligence, ICA, PUC-Rio.

[4] A Guide to the Project Management Body of Knowledge, Third Edition (PMBOK Guides) - Project Management Institute; 3rd edition (November 2004).

[5] J. Rodney Turner - The Handbook of Project-Based Management - McGraw-Hill Companies (January 1, 1993)

[6] International Function Point User Group -http://www.ifpug.org/.

[7] Boehm, B.W. Software Engineering Economics. Prentice-Hall, Englewood Cliffs, N.J.. 1981.

[8] David Poole, Alan Mackworth, Randy Goebel - Computational Intelligence - A Logical Approach - Ed. Oxford University Press, USA (January 8, 1998) .

[9] Ricardo Linden – Algoritmos Genéticos 2º Edição (2008) - Ed: Brasport.

[10] J. H. Holland, “Outline for a logical theory of adaptive systems,” J. ACM, vol. 9, no. 3, pp. 297–314, July 1962

[11] POTTER, M. A.; DE JONG, K. A. A Cooperative Co-Evolutionary Approach to Function Optimization. The Third Parallel Problem Solving From Nature, Jerusalem, Israel, pp. 249-257, Springer-Verlag, 1994.

[12] Hime Aguiar e Oliveira Junior, André M. Cadeira, Maria Augusta S. Machado, Reinaldo C. Souza, Ricardo Tanshcheit – *Inteligência Computacional (2007)* – Ed: Thompson.

[13] Lotfi A. Zadeh - *Fuzzy sets. Information and Control*. 1965; 8: 338–353

[14] A. Kaufmann and M.M. Gupta, 1988. Fuzzy Mathematical Models in Engineering and Management Science, North Holland

[15] Liem Tran, Lucien Duckstein.. Comparison of fuzzy numbers using a fuzzy distance measure. Fuzzy Sets and Systems, v. 130, p. 331-341. 2002.

[16] KOSKO, B. *Neural networks and fuzzy systems*. Englewood Cliffs: Prentice-Hall, 1992.