UNIVERSIDADE ESTADUAL DO CEARÁ

CENTRO DE CIÊNCIAS E TECNOLOGIA

MESTRADO EM CIÊNCIA DA COMPUTAÇÃO

DAYVISON CHAVES LIMA

UM FRAMEWORK *FUZZY* PARA PRIORIZAÇÃO DE REQUISITOS DE *SOFTWARE*

FORTALEZA, CEARÁ
2011

DAYVISON CHAVES LIMA

UM FRAMEWORK *FUZZY* PARA PRIORIZAÇÃO DE REQUISITOS DE *SOFTWARE*

Dissertação submetida à Comissão Examinadora do Programa de Pós-Graduação Acadêmica em Ciência da Computação da Universidade Estadual do Ceará, como requisito para obtenção do titulo de Mestre em Ciência da Computação.

Orientação: Prof. Dr Gustavo Augusto Lima de Campos; Prof. Dr Jerffeson Teixeira de Souza

FORTALEZA, CEARÁ
2011

L732f Lima, Dayvison Chaves
Uma framework fuzzy para priorização de requisitos de software / Dayvison Chaves Lima. — Fortaleza, 2011.
74 p. : il.
Orientação: Prof. Dr. Gustavo de Lima Campos,
Prof. Dr. Jefferson Teixeira de Souza.
Dissertação (Mestrado Acadêmico em Ciências da Computação) – Universidade Estadual do Ceará, Centro de Ciências e Tecnologia. Área de Concentração: Inteligência Computacional.
1. Priorização de requisitos. 2. Lógica Fuzzy. 3. Aplicações SBSE. I. Universidade Estadual do Ceará, Centro de Ciências e Tecnologia.

CDD: 004

DAYVISON CHAVES LIMA

UM FRAMEWORK *FUZZY* PARA PRIORIZAÇÃO DE REQUISITOS DE *SOFTWARE*

Dissertação submetida à Comissão Examinadora do Programa de Pós-Graduação Acadêmica em Ciência da Computação da Universidade Estadual do Ceará, como requisito para obtenção do titulo de Mestre em Ciência da Computação.

Aprovada em __/__/ 2011

BANCA EXAMINADORA

_____

Prof. Dr. Gustavo Augusto Lima de Campos (Orientador)
Universidade Estadual do Ceará – UECE

_____

Prof. Dr. Jerffeson Teixeira de Souza (Orientador)
Universidade Estadual do Ceará – UECE

_____

Profa. Dra. Mariela Inés Cortés
Universidade Estadual do Ceará – UECE

_____

Prof. Dr. Pedro de Alcântara dos Santos Neto
Universidade Federal do Piauí – UFPI

AGRADECIMENTOS

A Deus, por tudo. Antes, agora e sempre.

À minha esposa Érica, pela constante compreensão, dedicação, motivação e amor.

A dona Creusa e seu Aluízio [*in Memorian*], meus queridos pais, pelo apoio, por acreditarem mais uma vez nos meus estudos, e a minha irmã, por toda a ajuda e motivação.

Aos meus amigos, representados aqui pelo Paulo Roberto Pessoa, pela ajuda, conselhos e apoio durante esse período de estudo intenso.

Aos colegas e professores do mestrado, pelo apoio ao longo do curso, pelas ideias e discussões que sempre acrescentaram algo de valioso na construção do conhecimento.

Ao professor Gustavo Campos, pela orientação, paciência e incentivo ao longo de todo o mestrado.

Ao professor Jerffeson Souza, por toda a ajuda e inspiração durante o curso.

Aos professores Pedro de Alcântara e Mariela Inés Cortés pela gentileza em avaliar este trabalho.

Ao Banco do Nordeste do Brasil – BNB pelo apoio e liberação durante os meses de abril a junho de 2011.

“Não se deixem vencer pelo mal, mas vençam o mal com o bem.”
Romanos 12:21– Nova Versão Internacional

RESUMO

Uma das questões mais importantes em um projeto de desenvolvimento de software é a priorização de seus requisitos. Esta tarefa é realizada para indicar a ordem da implementação dos requisitos. Este problema possui aspectos nebulosos, dessa forma, os conceitos da Lógica Fuzzy podem ser utilizados para lidar adequadamente com esse desafio. O objetivo desse trabalho é apresentar um framework para auxiliar a tomada de decisão na priorização de requisitos em um processo de desenvolvimento de software, lidando inclusive com dados de origem ambígua e imprecisa.

Palavras-chave: Priorização de Requisitos. Lógica Fuzzy. Aplicações SBSE.

ABSTRACT

One of the most important issues in a software development project is the requirements prioritization. This task is used to indicate an order for the implementation of the requirements. This problem has nebulous aspects, therefore Fuzzy Logic concepts can be used to properly represent and tackle the task. The objective of this work is to present a formal framework to aid the decision making in prioritizing requirements in a software development process, including ambiguous and vague data.

Key-words: Requirements Prioritization. Fuzzy Logic. SBSE Applications.

LISTA DE ABREVIATURAS E SIGLAS

AHP Software	*Analytic Hierarchy Process* – Processo de Análise Hierárquica
ERP	*Enterprise Resource Planning* – Sistema de Gestão Empresarial
FLC	*Fuzzy Logic Controler* – Sistema de Controle *Fuzzy*
GA	*Genetic Algorithm* – Algoritmo Genético
GRASP	*Greed Randmized Adaptative Search Procedure* – Procedimento de Busca Gulosa Adaptativa
IGA	*Iterative Genetic Algorithm* – Algoritmo Genético Iterativo
NSGA-II	Non-dominated Sorting Genetic Algorithm-II – Algoritmo Genético de Ordenação não Dominante - II
QFD	*Quality Funcion Deployment* – Implantação de Função de Qualidade
SAD	Sistema de Apoio a Decisão
SBSE	*Search Based Software Engineering* – Otimização em Engenharia de
SHC	*Simple Hill Climbing* – Subida de Encosta Simples
TAA	Terminal de Auto Atendimento

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE EQUAÇÕES

SUMÁRIO

1 INTRODUÇÃO

Um dos primeiros passos no projeto de desenvolvimento de um produto de *software*, independente da metodologia utilizada, é o levantamento das necessidades de todos os envolvidos com o produto (SOMMERVILLE, 2009). As pessoas que utilizarão essa ferramenta, ou que receberão resultados dela, são responsáveis diretos por definir suas características e regras de funcionamento. Esses tais são conhecidos como *stakeholders*. Dessa forma, após investigar, identificar e catalogar as necessidades dos *stakeholders*, o projeto possui o conjunto inicial dos requisitos do produto. O próximo desafio é construir o *software* respeitando os requisitos elencados. Para isso, é necessário avaliar e definir a prioridade de cada um dos requisitos. Esse passo é importante, pois ajuda na identificação daqueles que apresentam maior risco ao projeto, ou que certamente apresentarão maior dificuldade de atendimento ou até os que ainda contêm características duvidosas ou não definidas. Requisitos que possuem essas características são bons candidatos a receber uma avaliação de priorização mais alta. Dessa forma, é necessário garantir que os requisitos mais importantes ou críticos sejam escolhidos para as primeiras *releases* do produto, para que eles contribuam na mitigação dos riscos do projeto e reflitam as interdependências entre os componentes de software existentes.

Para realizar a tarefa de priorização dos requisitos, o responsável pelo projeto deverá observar questões como orçamento disponível, prazo de entrega, tamanho da equipe alocada e o nível de importância de cada requisito para os *stakeholders*. Dessa forma, a priorização torna-se um desafio que possui alto nível de subjetividade, tendo em vista que, além de atender as metas do projeto, como custo e cronograma, existe ainda a necessidade de satisfazer pessoas envolvidas nos mais diversos níveis de uma organização. Na prática, busca-se atender aqueles indivíduos de maior importância na hierarquia organizacional, tendo em vista que os mesmos possuem uma visão estratégica das metas da corporação, ou aqueles que possuem maior domínio no conhecimento do produto em questão (SCHWABER, 2004).

1.1 Definição do Problema

Diariamente todos passam por situações onde é necessário realizar escolhas entre alternativas disponíveis, como qual comida escolher em um restaurante *self-service*, que DVD alugar ou qual o ônibus escolher para chegar ao centro da cidade. Quando o número de escolhas possíveis é pequeno, essa tarefa é resolvida sem muitas dificuldades. Porém, quando a decisão envolve dezenas ou centenas de possibilidades, a tarefa torna-se bem mais difícil.

Uma das maneiras de realizar uma escolha é priorizar entre as diversas alternativas disponíveis e escolher aquela de mais alta prioridade. Esse desafio também é simples quando são considerados poucos aspectos no processo de priorização. Ao alugar um apartamento, por exemplo, considerar apenas o valor do aluguel como critério de decisão é uma forma simples de realizar a escolha. Quando, porém, são considerados outros fatores como quantidade de quartos, estado de conservação do imóvel, nível de violência do bairro, proximidade com o emprego, a decisão não é realizada de forma tão simples quanto anteriormente. Durante a construção de um produto de *software*, as escolhas realizadas seguem o mesmo princípio, onde, por exemplo, a funcionalidade de maior importância para os usuários finais pode não ser a mais prioritária para o patrocinador do projeto quando outro aspecto como custo de desenvolvimento é considerado. A priorização de requisitos surge então para auxiliar a lidar com esse complexo problema de decisão (AURUM e WOHLIN, 2005).

Segundo (SCHULMEYER e MCMANUS, 1999), o nível de qualidade de um produto de software é determinado pela capacidade do produto em atender as necessidades de todos os envolvidos, ou *stakeholders*. A priorização de requisitos é utilizada durante a realização de tarefas como ajudar os *stakeholders* a decidir quais são os principais requisitos do sistema, selecionar o conjunto de requisitos a ser implementado nas *releases* do produto e definir a ordem entre requisitos que atendam a metas muitas vezes conflitantes como cronograma, orçamento, equipe disponível e qualidade. Assim, a priorização deve ser encarada como um processo estratégico, tendo em vista que a ordem de implementação dos requisitos do produto é capaz de definir o ganho ou perda de mercado, conforme descrito em (AURUM e WOHLIN, 2003). Dessa forma, priorizar requisitos pode ser definido segundo (RUHE e EBERLEIN, 2002)

como o desafio de selecionar um conjunto de requisitos dentre os conjuntos disponíveis a fim de que todos os interesses diferentes em questão, como limitações técnicas e preferências dos *stakeholders* sejam atendidos.

O entendimento dos requisitos torna-se mais claro à medida que o produto é desenvolvido, dessa forma, o processo de priorização não está atrelado a uma fase específica da construção do produto, podendo ser realizado durante todo o processo de desenvolvimento com diferentes níveis de avaliação para cada requisito. É interessante ressaltar a importância do entendimento dos requisitos e aspectos avaliados como importância, custo, cronograma e riscos, por parte dos *stakeholders*. Estudos em (LEHTOLA e KUJALA, 2004) mostram que os resultados da priorização são influenciados diretamente pelo nível de compreensão dessas variáveis.

Existem algumas técnicas disponíveis para realizar a priorização. Essas podem ser divididas basicamente em duas categorias: métodos quantitativos e negociação. Na abordagem por métodos são atribuídos valores numéricos a diferentes aspectos dos requisitos, enquanto que a abordagem por negociação busca alcançar o equilíbrio entre as avaliações dos requisitos realizadas por diferentes *stakeholders,* respeitando o papel desempenhando por cada um dentro de suas empresas.

Na abordagem baseada em métodos quantitativos, os requisitos são priorizados segundo os aspectos do produto. Um aspecto é um atributo que pode ser utilizado na priorização, como, por exemplo, a importância de um requisito para um *stakeholder,* ou custo, risco e cronograma. Os aspectos estão relacionados entre si, de forma que mudanças nos valores de um podem gerar alterações em outros. Dessa forma, é importante conhecer as relações de dependência entre os aspectos de um produto, tendo em vista que eles também variam entre cada projeto. Considerando o salário de um desenvolvedor, por exemplo, onde esse fator influencia diretamente o orçamento do projeto. No caso de uma empresa que trabalha como fábrica de software, em que os funcionários são pagos pelo valor homem/hora alocado ao projeto, cada hora além do previsto dedicada ao projeto, torna o orçamento do mesmo mais caro. Porém, no caso em que o *software* é desenvolvido para uso interno, a modalidade de pagamento praticada passa a ser salário fixo, assim independente de quanto tempo o funcionário esteja no projeto, o custo para a empresa é o mesmo. Dessa forma, o valor do aspecto está relacionado ao contexto do projeto.

Considerando a abordagem por negociação, o foco principal está na definição das metas do projeto, que segundo (LAMSWEERDE, 2001) são definidas como objetivos a

serem alcançados pelo *software* em desenvolvimento através de interação entre as características definidas pelos *stakeholders*, ou seja, as tarefas e procedimentos a serem automatizados, e as regras determinadas pelo ambiente de execução, como por exemplo, os termos linguísticos utilizados no sistema, no caso de um produto direcionado para públicos específicos como juristas ou médicos.

1.2 Motivação

Em muitos projetos de desenvolvimento de software, a tarefa de priorização é delegada para terceiros ou postergada ao máximo. Em muitos casos, o *software* construído é tecnicamente bom, mas não consegue atender de maneira apropriada as necessidades dos *stakeholders*. Estudos realizados em (STANDISH GROUP, 2009) mostraram que 32% dos projetos de desenvolvimento de *software* pesquisados obtiveram êxito em entregar o produto dentro do prazo, orçamento e com as características requisitadas. Já em 42% dos casos, o produto foi entregue com atrasos no cronograma ou acima do orçamento disponível, enquanto que 24% foram cancelados ou o produto entregue nunca foi utilizado. As principais causas desses números foram problemas na captura, especificação e validação dos requisitos, em que funcionalidades de alta importância para os envolvidos nem mesmo chegaram a fazer parte do núcleo principal do produto. Essa questão aparece, em muitos casos, devido à dificuldade em obter um consenso entre os envolvidos sobre a importância de um requisito em detrimento a outro.

Algumas técnicas podem ser utilizadas para auxiliar essa tarefa, como *Quality Function Deployment*, *Analytic Hierarchy Process* e Comparação *PairWise*. A questão é que a avaliação acaba sendo feita, em muitos casos, através da atribuição de valores numéricos. Dessa forma limita-se bastante a capacidade de expressar o pensamento e linguagem humanos. Os resultados alcançados em alguns casos, não refletem as metas estabelecidas pelo projeto de desenvolvimento do produto de software (STANDISH GROUP, 2009). Isso mais uma vez mostra a dificuldade em se realizar a priorização através de números. Alguns fatores que precisam ser considerados na realização da tarefa são as metas da corporação, metas do projeto, a forma como os requisitos são

capazes de atender as metas e qual a avaliação dos *stakeholders* sobre todos os requisitos. Deve-se considerar também que *stakeholders* em posições diferentes dentro da organização possuem ponderações díspares na avaliação dos requisitos.

Desse modo, pretende-se explorar a capacidade de representação do conhecimento através da Lógica *Fuzzy* para modelar as metas, requisitos e *stakeholders* envolvidos com um produto de *software*. Deseja-se também priorizar os requisitos de forma que estes atendam as metas estabelecidas e estejam alinhados com as expectativas dos diferentes *stakeholders* dispostos em níveis de hierarquia diferentes.

1.3 Objetivo Geral

A pesquisa tem como objetivo geral contribuir com a construção de um framework para resolver os problemas da priorização de requisitos de software. Deseja-se criar uma ferramenta onde seja possível transcrever as necessidades dos envolvidos no desenvolvimento de um projeto de software em uma lista de prioridades.

1.4 Objetivo Específico

O objetivo desse trabalho é propor um framework capaz de representar metas, requisitos e *stakeholders* de um produto, utilizando conceitos da Lógica *Fuzzy*. O mesmo será realizado com foco na tarefa de priorização de requisitos. Será levado em consideração durante a avaliação de cada requisito a sua capacidade em atender as metas definidas no projeto de desenvolvimento de software e sua avaliação estimada por *stakeholders*.

1.5 Estrutura da Dissertação

O presente trabalho está organizado da seguinte forma: no segundo capítulo são apresentados os conceitos fundamentais relacionados à lógica *fuzzy* para compreensão do tema como definição de conjuntos, operações, variáveis, relações, medidas de similaridade e sistemas de controle ou inferência. No terceiro capítulo são apresentados os trabalhos voltados a priorização de requisitos relacionando os principais métodos de priorização, suas similaridades e diferenças com o trabalho proposto. O capítulo quatro é dedicado à apresentação do *framework* proposto. Nesse capítulo, são utilizados os conceitos apresentados anteriormente para propor uma abordagem de priorização de requisitos baseado em metas e que atenda as necessidades dos *stakeholders* de um produto. O algoritmo de priorização é apresentado em detalhes juntamente com um exemplo ilustrativo. No capítulo cinco são apresentados os testes realizados a partir da implementação do *framework* e os resultados obtidos. Já no capítulo de Conclusão são apresentadas as observações finais sobre a dissertação e o direcionamento para trabalhos futuros.

2 TRABALHOS RELACIONADOS

Trabalhos na área de priorização de requisitos estão distribuídos em algumas áreas do conhecimento como Sistemas de Apoio a Decisão (SAD) e a própria Engenharia de Software, especificamente. Na área de SAD, (BELLMAN e ZADEH, 1970) propõe um método quantitativo de priorização e classificação de alternativas possíveis respeitando metas e restrições *fuzzy*. Uma decisão *fuzzy* é então vista como uma interseção entre pontos que representam essas alternativas no universo definido. Esses valores *fuzzy* são defuzificados e convertidos em uma abordagem de programação inteira. Essa abordagem se mostrou eficiente no contexto onde foi utilizada, porém não levou em consideração a avaliação das alternativas por parte dos *stakeholders* como forma de aprimorar o critério de avaliação final.

Outra metodologia para priorização é *Analytic Hierarchy Process* ou AHP proposta por Saaty (SAATY, 1990) e aplicada à priorização de requisitos por Karlsson (KARLSSON, 1996). Sua utilização consiste em basicamente comparar todos os pares de requisitos possíveis e decidir qual dos dois possui maior prioridade (SAATY, 2008). Essa comparação é realizada utilizando como base algum aspecto do projeto, como custo, por exemplo. O número de comparações realizadas é igual a $n \cdot (n-1)/2$, onde n é o número de requisitos. Um dos problemas encontrados com essa metodologia é a dificuldade em realizar a avaliação dos requisitos (MEAD, 2008), tendo em vista ser necessário valorar um requisito com base em alguma escala numérica disponível. Outro ponto importante é que o tempo para avaliação dos requisitos é proporcional à quantidade dos mesmos. Dessa forma, em um cenário de projetos de grande porte com muitos requisitos, a metodologia pode ser tornar ineficiente, conforme (KARLSSON, WOHLIN e REGNELL, 1998).

Quality Function Deployment inicialmente desenvolvido por Akao no final dos anos 60 (AKAO, 1972) (CHAN e WU, 2002) busca priorizar requisitos traduzindo-os em especificações do *design* da aplicação (FRANCESCHINI e ROSSETO, 2002). Para alcançar esse tipo de especificação, é definida uma matriz de relacionamentos onde os requisitos definidos pelos *stakeholders* e aqueles especificados como características técnicas do produto são confrontados. Para cada um deles é definido um grau de importância baseado em uma escala numérica com valores restritos, por exemplo {1,3 e

9}. Outra abordagem é substituir os números por símbolos. O valor final da prioridade de um requisito é alcançado através do cálculo da média entre o valor atribuído a ele pelos *stakeholders* e o peso dado às suas características técnicas relacionadas na matriz. Independente de qual escala seja adotada, (LARICHEV, MOSHKOVICH, *et al.*, 1993) diz que o *stakeholder* realiza uma avaliação numérica dos requisitos, da mesma forma que o AHP, tendo em vista que esse tipo de avaliação não é natural à linguagem humana e pode gerar um produto não convergente as suas necessidades dos mesmos.

A Otimização em Engenharia de Software do inglês Search Based Software Engineering - SBSE é uma área, definida em (HARMAN e JONES, 2001) que propõe a modelagem e resolução de problemas complexos de engenharia de Software utilizando técnicas de busca como as meta-heurísticas. Dentre os desafios já tratados, o problema da priorização de requisitos tem recebido uma atenção especial recentemente. Atualmente dez por cento dos trabalhos na área de SBSE são voltados à gestão de projetos ou gestão de requisitos, onde o problema da priorização das características de software está inserido (ZHANG_B, 2010).

O problema da próxima *release* definido em (BAGNALL, RAYWARD-SMITH e WHITTLEY, 2001) tem como objetivo priorizar requisitos a fim de planejar as próximas *releases* do produto. O desafio é selecionar, dentre os requisitos disponíveis, aqueles que possam ser entregues levando em consideração restrições como orçamento e prazo e buscando atender as principais metas ou demandas estabelecidas pelos *stakeholders*. Esse é um problema da classe *NP-difícil* e pode ser solucionado através de abordagens exatas como programação linear inteira ou aproximadas, como métodos de busca. Nos métodos exatos, o problema é definido como uma aproximação do problema da mochila. Já nos métodos de busca, algumas soluções foram geradas utilizando algoritmos como GRASP, para busca gulosa, SHC ou subida de encosta, têmpera simulada e algoritmos genéticos (BAGNALL, RAYWARD-SMITH e WHITTLEY, 2001).

Em (FEATHER e MENZIES, 2002) é apresentada uma abordagem iterativa de negociação onde a próxima *release* é composta por requisitos submetidos a um processo de decisão composto de três fases: execução, sumarização e decisão. Na primeira fase os requisitos são selecionados randomicamente de forma a diversificar o espaço de soluções. Na fase de sumarização, um *framework* de modelagem de iteração entre requisitos é utilizado para selecionar os requisitos de maior risco e as alternativas de mitigação para cada risco de melhor custo/benefício. Na fase de sumarização, uma

ferramenta de aprendizagem é utilizada para restringir o conjunto de solução obtida no passo anterior com base em um conjunto de regras que atendam as metas definidas no projeto, como, por exemplo, “selecionar o subconjunto de menor custo”. Após sucessivas iterações, essa solução se aproxima do conjunto ótimo de requisitos para uma *release*. Porém apresenta a avaliação dos requisitos e metas do projeto ainda é feita de forma numérica e a influência dos *stakeholders* sobre o conjunto é realizada apenas durante a seleção inicial de exemplos.

Em (BAKER, HARMAN, *et al.*, 2006) o problema da priorização de requisitos é tratado através de uma abordagem que compara o desempenho de uma priorização feita por humanos, um algoritmo guloso de busca e a meta-heurística da Têmpera Simulada para selecionar os componentes de software de uma *release*. O resultado mostra que a Têmpera possui melhor desempenho. Essa abordagem utiliza apenas o fator custo para realizar a priorização.

Finkelstein (FINKELSTEIN, HARMAN, *et al.*, 2009) apresenta uma abordagem multiobjectivo para o problema. Nesse trabalho é utilizando o algoritmo NSGA-II para calcular a compensação ou *trade-offs* de priorizar requisitos, mutias vezes conflitantes, baseado em metas definidas por vários *stakeholders*. A avaliação de cada requisito é realizada através da atribuição de valores numéricos e os *stakeholders* possuem mesmo nível de importância dentro da abordagem.

Zhang em (ZHANG_A, 2010) expõe os benefícios da abordagem multi-objetiva na priorização de requisitos definindo como objetivo do problema a maximização da satisfação dos *stakeholders*. É importante notar também que, diferente de outras abordagens na SBSE até então, o custo do projeto é tratado como um objetivo, não como uma restrição.

No trabalho de (TONELLA, SUSI e PALMA, 2010) é apresentado um algoritmo genético iterativo (IGA) baseado em comparações em pares que apresenta melhor desempenho que um GA convencional.

O trabalho de (BRASIL, FREITAS, *et al.*, 2010), apresenta algumas abordagens para tratamento do problema de priorização de requisitos utilizando Algoritmos Genéticos. Essa abordagem, porém não considera os objetivos difusos e imprecisão humana, tal como a avaliação linguística dos interessados. Esses objetivos podem, por exemplo, expressar as metas de um projeto de desenvolvimento de software como nível de satisfação dos envolvidos, entrega do produto dentro do cronograma e orçamento

acordados, atender os requisitos de qualidade definidos e medir o nível de satisfação do time do projeto.

Uma abordagem para priorização de requisitos utilizando um sistema de apoio à decisão foi proposta em (GAUR e SONI, 2010). O algoritmo definido auxilia os *stakeholders* na análise dos requisitos conflitantes conforme metas e restrições definidas. Os requisitos que não estejam em conflito uns com os outros não são considerados na avaliação. As metas e restrições são tratadas como conjuntos *fuzzy*, representados através de funções triangulares, enquanto que os requisitos são representados através de pesos numéricos. Essa avaliação resulta em um número que é utilizado para definir qual o nível de prioridade de cada requisito. Os *stakeholders* possuem mesmo nível de prioridade, de forma que não é possível diferenciar uma avaliação realizada por alguém da área técnica, como um desenvolvedor, por exemplo, de outro relacionado à área do negócio ou estratégica da empresa. A influência da prioridade dos requisitos nas metas é outro fator que não foi considerado.

3 FUNDAMENTAÇÃO TEÓRICA

A seguir, são destacados os fundamentos teóricos sobre os quais o *framework* é proposto. São descritas as noções de matemática *fuzzy*, bem como suas relações, operações e medida de similaridade. Esta abordagem emprega a notação padrão de conjuntos, denotando conjuntos *fuzzy* por letras maiúsculas e índices, e família de todos os possíveis subconjuntos nebulosos de um universo de discurso *U* por *F(U)*. Seja o intervalo de números reais *L = [0, 1]*.

3.1 Conjuntos Fuzzy

Representar as necessidades humanas através de uma linguagem capaz de transcrever todas ou a maior parte de suas peculiaridades foi sempre um grande desafio. Em meados dos anos sessenta, (ZADEH, 1965) foi o primeiro a propor uma abordagem que define níveis de relacionamento entre objetos, os chamados Conjuntos *Fuzzy*. Através do uso de variáveis linguísticas, como "bom", muito "bom", "ruim" ou "muito ruim", é possível permitir a um indivíduo expressar suas necessidades através de palavras ao invés de números. O valor de uma variável linguística pode ser quantificado através de operações matemáticas. Dessa forma, através dos conjuntos *fuzzy* é possível representar o quão um elemento está relacionado com outro, alcançando uma descrição e entendimento melhor da linguagem humana.

Conjuntos *fuzzy* são aglomerações onde os elementos possuem algum tipo de relacionamento entre si. Esse relacionamento é descrito por uma função de pertinência. Essa função, não se limita apenas a informar se certo elemento faz parte ou não de um conjunto de forma booleana, onde a resposta para a pergunta seria sim ou não. Mas define que dado um espaço de pontos X e um elemento x pertencente a X, um conjunto fuzzy A, a função $f_{A(x)}$ associa a cada ponto em X um valor real fechado variando entre 0 e 1. Dessa forma, quanto mais próximo de 1 for o valor de $f_{A(x)}$, maior será o nível de

pertinência de x com A. Assim, conjuntos *fuzzy* podem ser definidos como $\mu_A(x) : U \rightarrow [0,1]$, onde o nível de pertinência é contínuo, não apenas um valor binário como *pertence* ou *não-pertence*. Formalmente, a definição pode ser vista como mostrado na Equação 1. Segundo (LEE, 2005)**,** Um conjunto *fuzzy* A em um conjunto universo U é um conjunto de pares ordenados de um elemento genérico *x* e seu grau de pertinência $\mu_A(x)$.

$$A = \{(x, \mu_A(x)) \mid x \in U\} \quad (1)$$

Equação 1 – Definição do Conjunto *Fuzzy*

Ainda conforme (LEE, 2005), caso o conjunto seja discreto, os elementos podem ser enumerados, junto ao seu grau de pertinência, conforme a Equação 2:

$$A = \sum_i \mu_A(x_i)/x_i \quad (2)$$

Equação 2 – Enumeração dos Elementos de um Conjunto *Fuzzy*

Onde o somatório refere-se à notação de união e $\mu_A(x_i)/x_i$ representa o elemento x_i que pertence ao conjunto *fuzzy* A com grau $\mu_A(x_i)$. Como exemplo de conjunto *fuzzy*, pode ser citado o conjunto universo discreto composto por indivíduos com idades entre 1 e 20 anos, *U* = {1, 2, 3, 4, ..., 20}. Os conjuntos *fuzzy* A e B na Figura 1 representam crianças e adolescentes respectivamente.

O conjunto que representa as crianças é formado pelos elementos e seus respectivos graus de pertinência A = 0.4/2 + 0.6/3 + 0.8/4 + 1.0/5 + 1.0/6 + 1.0/7 + 0.8/8 + 0.6/9 + 0.4/10, já o conjunto que representa os adolescentes tem a seguinte configuração B = 0.1/7 + 0.2/8 + 0.4/9 + 0.6/10 + 0.8/11 + 0.9/12 + 1.0/13 + 1.0/14. É interessante notar que a idade 8 anos possui nível de participação 0.8 no caso do conjunto de crianças e 0.2 para a representação de adolescentes. Caso as funções que representam a pertinência a esses conjuntos fossem contínuas, o conjunto crianças seria representado por uma função trapezoidal, enquanto que o adolescentes teria como representação uma função triangular.

As funções de pertinência podem ser representadas através de funções triangulares, trapezoidais, quadráticas ou gaussianas, como pode ser visto no exemplo na Figura 2. As funções lineares, triangulares e trapezoidais, são as mais populares

devido à simplicidade de implementação, enquanto que o custo computacional exigido por outras funções não reflete melhoria na qualidade dos valores representados (YEN e LANGARI, 1998).

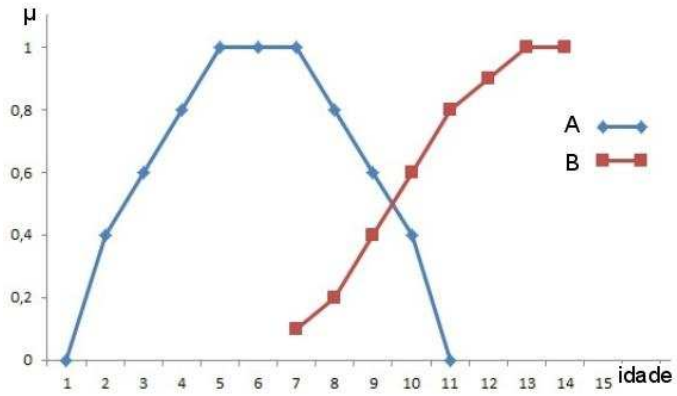

Figura 1 – Exemplo de um Conjunto *Fuzzy*

A fim de expandir o poder de representação de um conjunto *fuzzy*, definindo certo nível de incerteza no valor da função de pertinência de um determinado conjunto, tem-se o conceito de Conjunto Fuzzy Tipo-2, indicando conjuntos *fuzzy* cujos elementos são também conjuntos *fuzzy*. Dessa forma, o conjunto crianças poderia ser representado conforme a teoria cognitiva de Piaget definida em (PIAGET, 1971) que classifica o desenvolvimento infantil de 0 a 11 anos de idade em quatro fases distintas através do seguinte conjunto A = {Sensório Motor, Pré-operacional, Operatório Concreto, Operatório Formal}, onde cada um dos elementos do conjunto A é também um conjunto fuzzy. Tomando como exemplo, o conjunto Sensório Motor, definido como crianças de 0 a 2 anos seria representado como: Sensório Motor = 0.4/0, 1.0/1, 0.3/2.

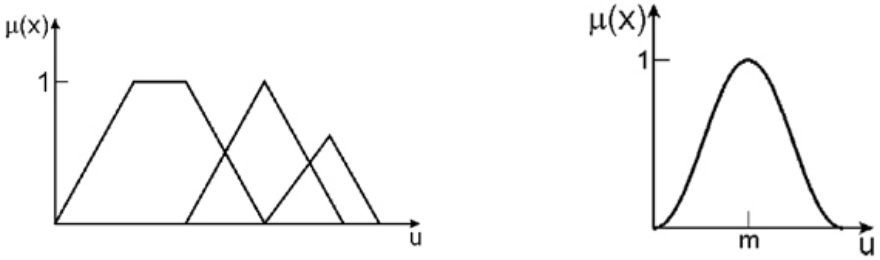

Figura 2 – Exemplo de funções de pertinência Trapezoidal, Triangular e Gaussiana

3.2 Operações Fuzzy

As operações básicas definidas em conjuntos *fuzzy* são as mesmas para conjuntos numéricos ou *crisp*. Basicamente, as três operações principais são Complemento, União ou Máximo e Intercessão ou Mínimo (LEE, 2005).

A operação de complemento é definida da mesma forma de um conjunto não *fuzzy*, onde o complemento do conjunto *A* é representado por $\bar{A}$, conforme Equação 3.

$$\mu_{\bar{A}}(x) = 1 - \ \mu_A(x), \forall\, x \ \in X \tag{3}$$

Equação 3 – Complemento de um Conjunto *Fuzzy*

Na operação de União ou Máximo, o conjunto final é resultado do maior valor da função de pertinência entre dois conjuntos A e B, conforme definido na Equação 4.

$$\mu_{A \cup B}(x) = Max\ [\mu_A(x), \mu_B(x)] \tag{4}$$

Equação 4 – União entre Conjuntos *Fuzzy*

Já na operação de Interseção, o resultado é o conjunto é formado pelo menor valor da função de pertinência entre os conjuntos A e B. A operação é definida na Equação 5. A Figura 3 mostra um exemplo de representação das principais operações em conjuntos *fuzzy*.

$$\mu_{A \cap B}(x) = Min\ [\mu_A(x), \mu_B(x)] \tag{5}$$

Equação 5 – Interseção entre Conjuntos *Fuzzy*

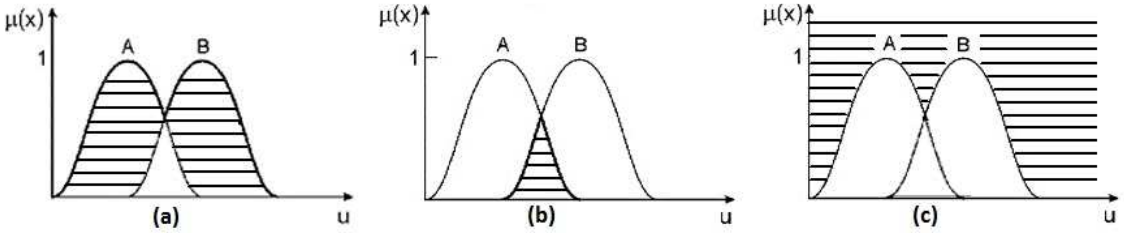

Figura 3 – Exemplo das operações união (a), intercessão (b) e complemento (c)

Para esse trabalho será utilizado também o conceito da operação de pseudo complemento que está definida segundo (FERNÁNDEZ, SUÁREZ e GIL, 1992) como um operador φ: $LxL \rightarrow L$, onde $L = [0, 1]$, e a e b $\in L$, representado na Equação 6.

$$\varphi(a,b) = \begin{cases} 1, & a \leq b \\ b, & a > b \end{cases} \tag{6}$$

Equação 6 – Operador Pseudo Complemento

Outras operações que podem ser utilizadas entre conjuntos *fuzzy* são as que determinam a distância entre eles. Para calcular essa informação, três medidas podem ser utilizadas, as distâncias de Hamming, Euclidiana e Minkowski (LEE, 2005).

A distância de Hamming, ou simétrica, é definida conforme a Equação 7, onde é assumido que X é um conjunto universal de n elementos. Essa medida representa o conceito matemático de distância, onde $d(A,B) \geq 0$, $d(A,B) = d(B,A)$, $d(A,C) \leq d(A,B) + d(B,C)$ e $d(A,A) = 0$.

$$d(A,B) = \sum_{i=1,\ x_i \in X}^{n} |\mu_A(x_i) - \mu_B(x_i)| \tag{7}$$

Equação 7 – Distância de Hamming

Já a distância Euclidiana representa o segmento de reta que une dois pontos ou conjunto de pontos no espaço euclidiano n-dimensional. É derivada da definição proposta por Hamming anteriormente e é representada pela Equação 8.

$$d(A,B) = \sqrt{\sum_{i=1,\ x_i \in X}^{n} \left(\mu_A(x_i) - \mu_B(x_i)\right)^2} \tag{8}$$

Equação 8 – Distância Euclidiana

A distância de Minkowski generaliza as outras duas medidas apresentadas anteriormente. A Equação 9 define a distância em questão, de forma que, para $w = 1$, é

configurada a distância de Hamming, enquanto que no caso de *w = 2* tem-se a definição da distância Euclidiana.

$$d(A,B) = \left(\sum_{i=1,\, x_i \,\epsilon\, X}^{n} |\mu_A(x_i) - \mu_B(x_i)|^w \right)^{1/w} \quad , w \,\epsilon\, [1, \infty] \tag{9}$$

Equação 9 – Distância de Minkowski

3.3 Variáveis Linguísticas

Uma variável linguística é definida conforme (LEE, 2005) como uma variável cujo valor é expresso semanticamente por um termo linguístico e uma função de pertinência. A representação de uma variável linguística é caracterizada através da quíntupla $\{n, T, U, m(t)\}$, onde *n* representa o nome da variável, *T* o conjunto de termos linguísticos de *n*, *U* é o universo do discurso onde os valores de *n* são representados e *m(t)* é a função semântica que realiza o mapeamento entre cada valor de $t \in T$ e um conjunto *fuzzy* em *U* (WILHELM e PARSAEI, 1991).

A Figura 4 apresenta um exemplo de representação de um conjunto *fuzzy* através de variáveis linguísticas. O nome da variável *n* é Pessoa, o conjunto dos termos linguísticos possíveis $t \in T$ para essa variável é $\{criança, adolescente, adulto\}$, o universo do discurso *U* é o intervalo de idades compreendido entre 0 e 30 anos, e $m(t)$ é a função de pertinência dos valores de idade aos conjuntos representados.

Os termos linguísticos utilizados não atribuem um significado quantitativo a Pessoa, porém representam alguma semântica na sua classificação. Ao mencionar, por exemplo, que “Fulano já é um adolescente!”, não se sabe qual a idade do sujeito em questão, porém, a informação contida na sentença é suficiente para quantizar uma idade aproximada do indivíduo.

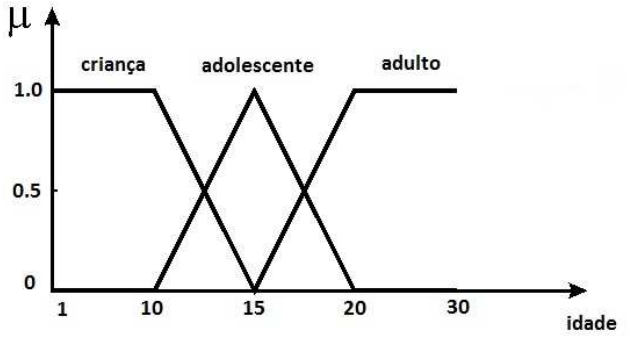

Figura 4 – Exemplo de representação da variável linguística Pessoa

3.4 Relações Fuzzy

Uma relação entre conjuntos *crisp* ou numéricos A e B, pode ser definida a partir de uma função de pertinência de A em B $\mu_R(x,y)$, para $x \in A$ e $y \in B$ conforme a Equação 10, onde essa relação referencia A x B ao conjunto {0,1}, ou seja $\mu_R : A\ x\ B \rightarrow \{0,1\}$.

$$d\mu_R(x,y) = \begin{cases} 1\ iff\ (x,y) \in \mathbb{R} \\ 0\ iff\ (x,y) \notin \mathbb{R} \end{cases} \quad (10)$$

Equação 10 – Relação Crisp

Já uma relação *fuzzy*, conforme (LEE, 2005) possui uma função de pertinência cujo valor está contido dentro do intervalo [0,1], dessa forma, $\mu_R : A\ x\ B \rightarrow [0,1]$. A relação *fuzzy* é definida conforme a Equação 11.

$$R = \left\{\left((x,y), \mu_R(x,y)\right) \middle| \mu_R(x,y) \geq 0, \quad x \in A, y \in B\right\} \quad (11)$$

Equação 11 – Relação Fuzzy

Onde a semântica de $\mu_R(x,y)$ é interpretada como a força do relacionamento entre *x* e *y*. Dessa forma, se $\mu_R(x,y) > \mu_R(x',y')$, então (x,y) é mais fortemente relacionada que (x',y'). Um exemplo de relações crisp e *fuzzy* é mostrado na Figura 5, onde **(a)**

representa a relação crisp e **(b)** a relação *fuzzy*. Dessa forma, tomando como exemplo o valor da relação entre os elementos a e c, no caso crisp, $\mu_R(a,c)$ é igual a 1. Ou seja, existe uma relação entre os elementos. No caso da relação *fuzzy* dos elementos citados, $\mu_R(a,c)$ é igual a 0.8, indicando que a relação existe e que possui um grau de relacionamento em 0.8. Já para os elementos b e d, $\mu_R(b,d) = 0$ em ambos os casos, indicando que os elementos não possuem uma relação.

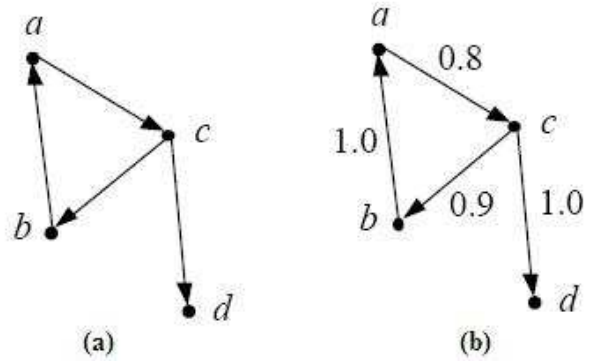

Figura 5 – Exemplo de relação crisp (a) e fuzzy (b)

Uma das operações mais importantes relativa as relações fuzzy é a operação de Inferência. Esta é resultante da interpretação de um conjunto de regras linguísticas a partir de variáveis de entrada. As regras são apresentadas no formato $Se\ \{condição\}\ então\ \{consequencia\}$ ou seja, dado a ocorrência da condição descrita em $\{condição\}$, quais fatos podem ser conhecidos $\{consequencia\}$. Como exemplo tem-se a relação $R: Se\ x\ é\ A\ então\ y\ é\ B$, também abreviada por $R: A \rightarrow B$. A expressão pode ser vista como uma relação entre as duas variáveis x e y. R pode ser interpretado então como um Conjunto *Fuzzy* com uma função de pertinência bidimensional conforme definido na Equação 12.

$$\mu_R(x,y) = f\big(\mu_A(x), \mu_B(y)\big) \tag{12}$$

Equação 12 – Relação de Inferência

Onde a função f é conhecida como Função de Inferência *Fuzzy*, ou Função de Implicação *Fuzzy* (LEE, 2005), de forma que o nível de pertinência de x em A e de y em B é transformado em um nível de pertinência de (x,y) em A x B.

Para que a inferência da função f seja possível, é necessário o uso de um operador de inferência. Um dos mais utilizado é o operador de Mamdani (MAMDANI e ASSILIAN, 1999) que interpreta a operação de inferência *fuzzy* como uma operação de mínimo ou interseção, conforme definido na Equação 13.

$$R: A \rightarrow B = \int_{X \times Y} \mu_A(x) \wedge \mu_B(y)/(x,y) \tag{13}$$

Equação 13 – Operador de Inferência de Mamdani

3.5 Números Fuzzy

Números *fuzzy* são uma extensão dos números reais de forma que enquanto os últimos têm seu valor associado a um único número, os primeiros possuem um conjunto de possíveis valores representados por sua função de pertinência no intervalo [0,1]. Para definir um número *fuzzy* é necessário que algumas outras definições sejam realizadas, como Conjunto de Suporte, Normalizado, Corte-α e Convexo.

Observando o exemplo da Figura 6, verifica-se que cada subconjunto de *U* (criança, jovem, adulto ou sênior) está relacionado aos elementos do conjunto universo *U* através de níveis de pertinência. Dessa forma, o conjunto suporte de um conjunto *fuzzy* A pode ser definido como sendo o conjunto crisp de elementos *U* cujo grau ou nível de pertinência em *A* seja maior que 0. A definição formal é mostrada na Equação 14. Aplicando a definição para o conjunto *jovem*, temos que *Suporte(jovem) = {15,25,35,45}*.

$$Suporte(A) = \{x \in U | \mu_A(x) > 0\} \tag{14}$$

Equação 14 – Definição de Conjunto de Suporte

U = {5, 15, 25, 35, 45, 55, 65, 75, 85}

idade	criança	jovem	adulto	senior
5	0	0	0	0
15	0	0.2	0.1	0
25	0	1	0.9	0
35	0	0.8	1	0
45	0	0.4	1	0.1
55	0	0.1	1	0.2
65	0	0	1	0.6
75	0	0	1	1
85	0	0	1	1

Figura 6 – Exemplo de Conjunto Fuzzy. Adaptado de (LEE, 2005)

O Conjunto Normalizado pode ser definido como o conjunto que possui o valor máximo de uma determinada função de pertinência. Supondo que a função de pertinência do exemplo da Figura 6 tenha valor máximo ou altura igual a 1, os conjunto *jovem*, *adulto* e *sênior* são ditos Normalizados. Já o conjunto *criança* não pode ser definido desta maneira por não possuir o valor máximo em nenhuma de suas relações.

No caso do Conjunto de Corte-α, os elementos que o compõe são aqueles cujo grau de pertinência é maior ou igual a α. Dessa forma, o conjunto pode ser definido conforme (LEE, 2005) na Equação 15. Seguindo o exemplo acima, o Conjunto de Corte-α para o conjunto *fuzzy adulto* onde α = *0.5*, temos $adulto_{0.5} = \{25, 35, 45, 55, 65, 75, 85\}$.

$$A_\alpha = \{x \epsilon U | \mu_A(x) \geq \alpha\} \tag{15}$$

Equação 15 – Definição de Conjunto de Corte-α

Um conjunto A é Convexo considerando inicialmente um conjunto crisp em R^n, onde n é o espaço Euclidiano n-dimensional, se as seguintes propriedades forem satisfeitas (LEE, 2005):

i. Dois pontos arbitrários s e r estão definidos em A.
   $r = (r_i | i \in N_n)$ e $s = (s_i | i \in N_n)$, onde $N \in \mathbb{N}$.

ii. Para um número real λ entre 0 e 1, um ponto t está incluído em A e é definido da seguinte forma: $t = (\lambda r_i + (1 - \lambda) s_i | i \in N_n)$. Se um ponto em linha reta conecta dois pontos s e r em A, ele também está em A.

A Figura 7 apresenta exemplos de conjuntos convexos. Os conjuntos A1, A2 e A3 são convexos, enquanto que os conjuntos A4, A5 e A6 não o são. É importante notar

que visivelmente os conjuntos não convexos estão em desacordo com a propriedade ii definida acima, de forma que é possível conectar dois pontos desses conjuntos em linha reta e alguns pontos dessa linha não pertençam ao conjunto.

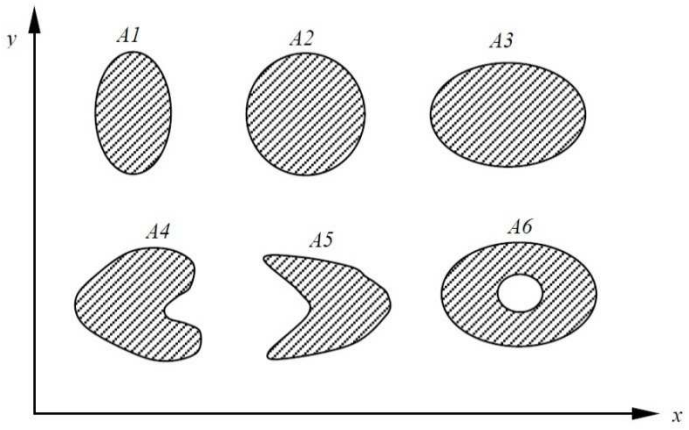

Figura 7 – Exemplos de conjuntos convexos e não convexos

Assim, um conjunto *fuzzy A* é convexo se todos os seus Conjuntos de Corte-α forem convexos. Dessa forma, se a relação $\mu_A(t) \geq Min[\mu_A(r), \mu_A(s)]$, onde $t = \lambda r + (1-\lambda)s, r, s \in \mathbb{R}^n,\ \lambda \in [0,1]$ se confirmar então o conjunto *A* é convexo.

Definição de Número Fuzzy. Se um conjunto *fuzzy* for normalizado, convexo e sua função de pertinência for contínua em $\mathbb{R}$, então esse conjunto é um "número *fuzzy*". Dessa forma, um número *fuzzy* representa um intervalo de números reais cujas fronteiras são *fuzzy*.

Os números *fuzzy* podem ser representados através de funções triangulares ou trapezoidais. No caso das triangulares, o número *fuzzy A* é representado através de três pontos $A = (a_1,\ a_2, a_3)$ e sua função de pertinência é definida conforme a Equação 16.

$$\mu_A(x) = \begin{cases} 0, & x < a_1 \\ \dfrac{x - a_1}{a_2 - a_1}, & a_1 \leq x \geq a_2 \\ \dfrac{x - a_1}{a_2 - a_1}, & a_1 \leq x \geq a_2 \\ 0, & x > a_3 \end{cases} \tag{16}$$

Equação 16 – Número Fuzzy Triangular

Já a representação de números *fuzzy* através da função trapezoidal é definida pela Equação 17. Essa forma é alcançada devido à existência de pontos cujo valor da função

de pertinência é máximo e igual a 1 ($\alpha = 1$). A Figura 8 mostra a representação de um Número *Fuzzy* Triangular **(a)** e um Trapezoidal **(b)**.

$$\mu_A(x) = \begin{cases} 0, & x < a_1 \\ \dfrac{x - a_1}{a_2 - a_1}, & a_1 \leq x \leq a_2 \\ 1, & a_2 \leq x \leq a_3 \\ \dfrac{a_4 - x}{a_4 - a_3}, & a_3 \leq x \leq a_4 \\ 0, & x > a_4 \end{cases} \quad (17)$$

Equação 17 – Número Fuzzy Trapezoidal

Nota-se que, para o Número *Fuzzy* Trapezoidal, se $a_2 = a_3$, tem-se a transformação deste em um Número *Fuzzy* Triangular.

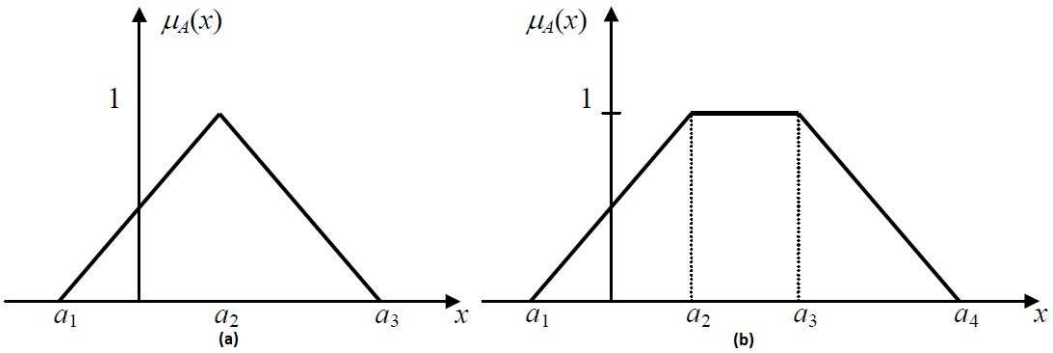

Figura 8 – Representação de Número Fuzzy Triangular (a) e Trapezoidal (b)

3.6 Medidas de Similaridade

De acordo com (XUECHENG, 1992), a entropia de um conjunto *fuzzy* é uma medida que define o quão nebuloso é o conjunto. Ela é uma função que atribuí um valor a cada subconjunto *fuzzy* de um universo de discurso. Este valor caracteriza o grau de nebulosidade do subconjunto *fuzzy*. Existem medidas de entropia geradas a partir de medidas de distância de conjuntos *fuzzy*. Uma medida de distância entre dois conjuntos *fuzzy* é uma medida que descreve a diferença entre esses. Da mesma forma que à

distância, a medida de similaridade entre dois conjuntos nebulosos é uma medida que indica a paridade entre os conjuntos.

Para esse trabalho será utilizado o conceito de medida de similaridade definido a partir do conceito de distância por (XUECHENG, 1992) na Equação 18:

$$dis^p(C_1, C_2) = (\sum_{i=1}^{U} |C_1(u_i) - C_2 u_i|^p)^{1/p} \Big/ L^{1/p} \tag{18}$$

Equação 18 – Distância definida por **(XUECHENG, 1992)**

Onde $p \geq 1$, n denota a cardinalidade do universo de discurso discreto U, e $u_i \in U$. Dessa forma, é possível obter a seguinte medida de similaridade correspondente, representada na Equação 19:

$$sim^p(C_1, C_2) = 1 - \ dis^p(C_1, C_2), \forall\ C_1, C_2\ \in U\ F(U) \tag{19}$$

Equação 19 – Medida de Similaridade

3.7 Sistemas Fuzzy

Sistemas de Controle baseado em lógica *Fuzzy* provêm uma forma eficiente de entendimento das características do mundo real. Um Sistema *Fuzzy* ou Controlador de Lógica *Fuzzy* (FLC) é definido, segundo (LEE, 2005), como um conjunto de estratégias de controle linguístico baseado em regras. É considerada uma metodologia de controle eficiente quando o processo de controle é tão complexo para ser analisado por meio de técnicas quantitativas ou quando os recursos disponíveis para realizar as avaliações são incertos. A Figura 9 a seguir apresenta a arquitetura de um sistema *fuzzy*, que pode ser dividida em três fases distintas: Fuzificação, Inferência e Defuzificação.

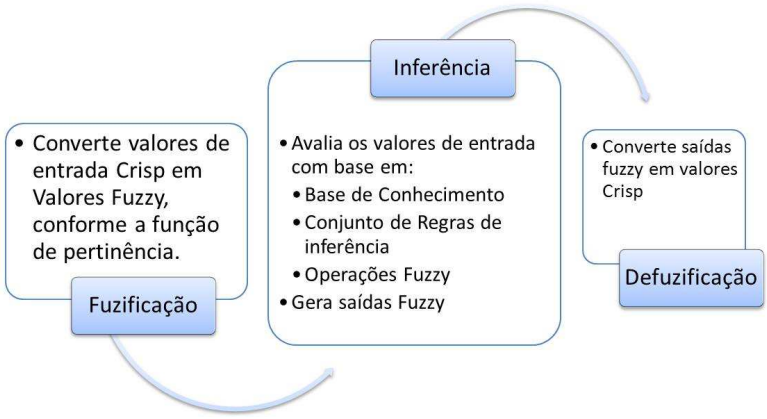

Figura 9 – Sistema Fuzzy

Na fase de Fuzificação, o sistema *fuzzy* transforma os valores de entrada crisp em valores *fuzzy*. Isso é feito a partir das funções de pertinência definidas para cada um dos conjuntos *fuzzy* participantes do sistema. O valor crisp gerado está definido dentro dos limites do universo de discurso estabelecido.

O segundo momento dentro do FLC é conhecido como Inferência. A partir da base de conhecimento cadastrada no sistema, as variáveis de entrada são avaliadas tomando como base um conjunto de regras de inferência no formato $Se\ \{condição\}\ então\ \{consequencia\}$, conforme apresentado na Figura 10 bem como um conjunto específico de operações *fuzzy* como união, intercessão, complemento ou negação e implicação. O resultado da avaliação é expresso através de variáveis de saída *fuzzy*.

	Premise		Conclusion
if	(importancia_stk == muito_baixo & similaridade_req == muito_baixo)	->	importancia_req = muito_baixo
if	(importancia_stk == muito_baixo & similaridade_req == baixo)	->	importancia_req = muito_baixo
if	(importancia_stk == muito_baixo & similaridade_req == medio)	->	importancia_req = baixo
if	(importancia_stk == muito_baixo & similaridade_req == alto)	->	importancia_req = medio
if	(importancia_stk == muito_baixo & similaridade_req == muito_alto)	->	importancia_req = medio
if	(importancia_stk == baixo & similaridade_req == muito_baixo)	->	importancia_req = muito_baixo
if	(importancia_stk == baixo & similaridade_req == baixo)	->	importancia_req = baixo
if	(importancia_stk == baixo & similaridade_req == medio)	->	importancia_req = baixo
if	(importancia_stk == baixo & similaridade_req == alto)	->	importancia_req = medio
if	(importancia_stk == baixo & similaridade_req == muito_alto)	->	importancia_req = alto
if	(importancia_stk == medio & similaridade_req == muito_baixo)	->	importancia_req = baixo
if	(importancia_stk == medio & similaridade_req == baixo)	->	importancia_req = baixo
if	(importancia_stk == medio & similaridade_req == medio)	->	importancia_req = medio
if	(importancia_stk == medio & similaridade_req == alto)	->	importancia_req = alto
if	(importancia_stk == medio & similaridade_req == muito_alto)	->	importancia_req = alto
if	(importancia_stk == alto & similaridade_req == muito_baixo)	->	importancia_req = baixo
if	(importancia_stk == alto & similaridade_req == baixo)	->	importancia_req = medio
if	(importancia_stk == alto & similaridade_req == medio)	->	importancia_req = medio
if	(importancia_stk == alto & similaridade_req == alto)	->	importancia_req = alto
if	(importancia_stk == alto & similaridade_req == muito_alto)	->	importancia_req = muito_alto
if	(importancia_stk == muito_alto & similaridade_req == muito_baixo)	->	importancia_req = baixo
if	(importancia_stk == muito_alto & similaridade_req == baixo)	->	importancia_req = medio
if	(importancia_stk == muito_alto & similaridade_req == medio)	->	importancia_req = alto
if	(importancia_stk == muito_alto & similaridade_req == alto)	->	importancia_req = muito_alto
if	(importancia_stk == muito_alto & similaridade_req == muito_alto)	->	importancia_req = muito_alto

Figura 10 – Regras utilizadas durante a avaliação

A última etapa do processo denominada Defuzificação é responsável por converter as saídas geradas na fase de Inferência em valores reais ou *crisp*. Para isso, é necessário o uso de uma função de defuzificação. Uma das funções mais utilizada é a Centro de Área. Essa função gera o centro de gravidade de um conjunto *fuzzy C* (LEE, 2005). A Equação 20 define essa operação para o caso de um conjunto universo discreto. A ilustração da função é apresentada na Figura 11.

$$z_0 = \frac{\sum_{j=1}^{n} \mu_c(z_j).\, z_j}{\sum_{j=1}^{n} \mu_c(z_j)} \tag{20}$$

Equação 20 – Método de Defuzificação Centro de Área

Onde *n* é o número de níveis de quantização da variável de saída e *C* é o conjunto *fuzzy* definido na dimensão *(z)*.

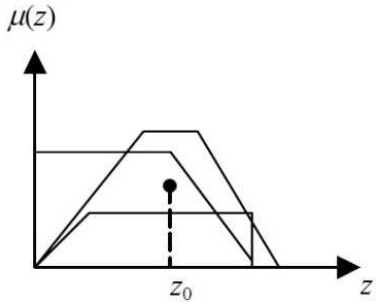

Figura 11 – Centro de Área

4 ABORDAGEM PROPOSTA

A abordagem proposta trata-se de um framework com um algoritmo associado que é capaz de realizar a priorização de requisitos de software guiados por um conjunto de informações. Essas são descritas em termos linguísticos oriundos do discurso das pessoas que estão envolvidas com o processo de desenvolvimento do software e, mais especificamente, na etapa de priorização dos requisitos. Este capítulo visa apresentar a abordagem. Seção 4.3.1 apresenta a formulação do problema de priorização que foi abordado neste trabalho. A Seção 4.3.2 apresenta o framework composto por um conjunto de definições específicas, fundamentadas nas definições apresentadas no capítulo de Fundamentação Teórica. Finalmente as Seções 4.1 e 4.3.1 utilizam as definições específicas e apresentam o algoritmo de priorização.

4.1 O Problema de Priorização de Requisitos

O problema de priorização proposto neste trabalho considera que estão disponíveis um conjunto de informações relacionadas às metas estratégicas de uma organização, expressas em termo de uma situação desejada, aos requisitos que são desejados por diversos *stakeholders* e aos próprios *stakeholders* no contexto da organização. Mais especificamente, este problema pode ser inicialmente formulado como segue:

Dados: (a) um conjunto de requisitos que são desejados por um subconjunto dos *stakeholders* que compõem uma organização; (b) uma descrição linguística da situação desejada para a organização, em termos de metas estratégicas e valores de importância associados; (c) uma descrição linguística do alcance de cada requisito em termos de cada meta na situação desejada; (d) descrição linguística dos valores de importância associados por cada um dos *stakeholders* a cada requisito; (e) descrição linguística dos

valores de importância associados a cada um dos *stakeholders* no contexto da organização.

Tarefa: encontrar uma ordem de prioridade dos requisitos de software desejados pelos *stakeholders* que contribua com a realização das metas estratégicas descritas na situação desejada da organização, considerando os valores de importância de cada requisito para cada *stakeholder* e os valores de importância de cada *stakeholder* no contexto da organização.

As duas próximas seções a seguir apresentam um esboço do algoritmo de priorização para resolver o problema, os principais componentes envolvidos nesta primeira formulação e as definições formais correspondentes.

4.2 Esboço de uma Abordagem *Fuzzy* para o Problema

A abordagem desenvolvida nesta dissertação utiliza a informação linguística disponível na definição do problema e determina a ordem de prioridade de cada um dos requisitos disponíveis. A Figura 12 esquematiza a informação disponível no problema definido. Onde o operador de pseudo complemento φ é utilizado para representar a relação fuzzy entre um valor aspirado em uma meta e o seu valor de importância, bem como, a relação entre o valor de importância de um requisito, definido pelos stakeholders, e o valor prometido (realização prometida) pelo requisito em relação a uma meta.

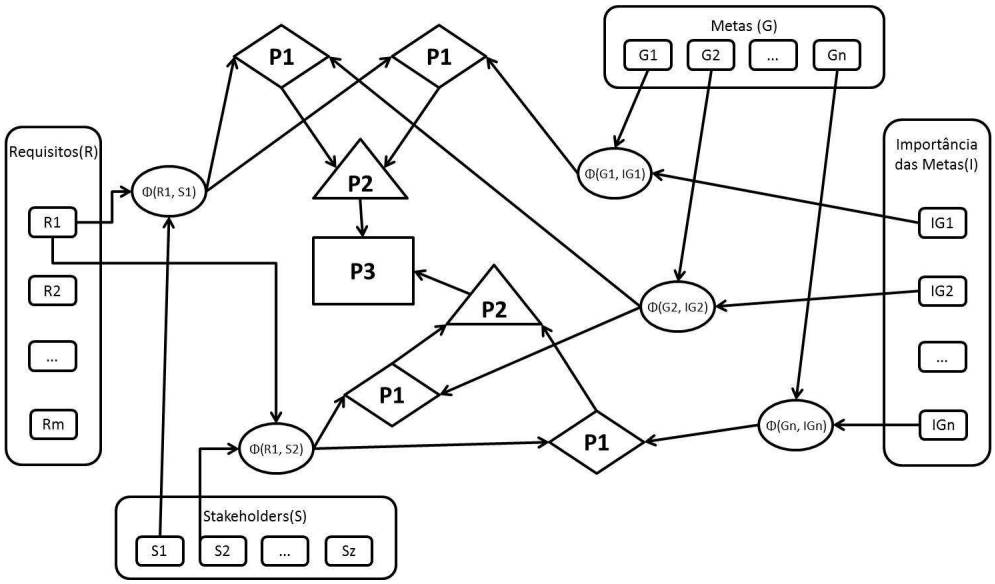

Figura 12 – Esboço de uma Abordagem Fuzzy para o Problema

De posse destas informações a abordagem deve ser capaz de realizar três passos principais até a obtenção de uma solução, ou seja:

(P1) Para cada um dos requisitos preferidos por cada *stakeholder* da organização, calcular um valor de avaliação indicando o quanto o requisito está alinhado com a situação desejada do projeto.

(P2) Para cada um dos requisitos preferidos por cada *stakeholder* da organização, considerando o valor de avaliação correspondente obtido no passo (P1) e a importância do stakeholder para a organização, calcular um valor de prioridade do requisito para o stakeholder;

(P3) Para cada um dos requisitos desejados pelos *stakeholders*, considerando os valores de prioridades calculados no passo (P2), calcular o valor de prioridade final do requisito.

A última seção deste capítulo emprega as definições apresentadas na próxima seção para formalizar os passos (P1), (P2), P(3) e gerar o algoritmo capaz de realizar a tarefa estabelecida no problema, isto é, encontrar a ordem de prioridade adequada (satisfatória) que deve ser seguida na implementação dos requisitos.

4.3 Framework

Neste framework as letras maiúsculas representam conjuntos, conjuntos fuzzy, e relações fuzzy, *U = 0.0 + 0.1 + ...+... +1.0* é o universo de discurso na definição de conjuntos fuzzy, *L = [0,1]* é o intervalo que define os valores de participação ou pertinência e *F(U)* descreve a família de conjuntos *fuzzy* definidos em *U*. É importante notar que o sinal de adição representa a união *fuzzy* dos elementos do conjunto *U*, que também pode ser representado da seguinte forma: *U = {0.0, 0.1, ..., 1.0}*.

4.3.1 Definições

A primeira parte do framework permite a representação de Metas Fuzzy, Situação Desejada Fuzzy e Requisitos Fuzzy de um *stakeholder*. Um par (variável, nível linguístico de aspiração) define uma meta *fuzzy*. A definição a seguir formaliza a noção de Meta *Fuzzy*.

Definição 1. Uma meta G_m é dita *fuzzy* se o nível de aspiração A_m, atribuído a um atributo de uma situação desejada é descrito em termos de um subconjunto *fuzzy* de *U*:

$$G_m = \left(X_{m,}, A_m = \sum_{i=1}^{L} A_m\,(u_i)/u_i \right) \tag{21}$$

Equação 21 – Definição de Meta Fuzzy

Onde X_m é o nome da *m*-ésima variável linguística que representa o *m*-ésimo atributo de uma situação desejada; *L* é o limite dos elementos em *U*; A_m é o valor linguístico de X_m que pertence a função *T(Aspiração)* e representa o nível de aspiração atribuído a X_m; e $A_m(u_i)$ é o nível de participação de $u_i \in U$ no conjunto *fuzzy* que define $A_m \in F(U)$.

Os valores das variáveis linguísticas são sentenças na linguagem humana formadas por termos que representam níveis de aspiração como, por exemplo, *T(Aspiração)* = ... baixo + médio + ... + alto + ... Por sua vez, esses níveis são representados matematicamente por conjuntos *Fuzzy* definidos em *U*, como, por exemplo, os conjuntos *baixo = [1.0, 1.0, 0.9, 0.7, 0.5, 0.2, 0.0, 0.0, 0.0, 0.0, 0.0]* e *alto = [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.3, 0.5, 1.0, 1.0, 1.0]).* Geralmente mais de uma meta é utilizada para especificar uma situação desejada. Cada meta contribui em um nível diferente na realização da situação. A definição a seguir formaliza a noção de situação (estratégica) desejada.

Definição 2. Uma situação é denominada *fuzzy* se puder ser representada por um conjunto *fuzzy* tipo-2 em termos de metas *fuzzy* G_m e níveis de importância *fuzzy* I_m das metas na situação desejada *DS* (m = 1, ..., M):

$$DS = \sum_{m=1}^{M} I_m / G_m \tag{22}$$

Equação 22 – Definição de Situação Fuzzy

Onde $I_m \in F(U)$ é um valor linguístico de importância em *T(Importância).*

Da mesma forma que *T(Aspiração),* o conjunto de termos em *T(Importância)* e sua família de conjuntos *fuzzy* associada devem ser definidos adequadamente considerando as descrições linguísticas presentes na formulação do problema. Da mesma forma, considerando a situação desejada, são conhecidos os requisitos mais importantes para cada *stakeholder* e a opinião destes em termos da realização prometida por estes requisitos em termos das metas *fuzzy* que devem ser satisfeitas na situação desejada. A definição a seguir formaliza o conceito de requisito *fuzzy* para um *stakeholder*.

Definição 3. Um requisito Re_{rs} é dito *fuzzy* para um *stakeholder s* se for representado por um conjunto *fuzzy* tipo-2 descrito em termos de pares (X_m, R_{mr}) e níveis de participação *fuzzy* I_{ms} (m = 1, ..., M):

$$Re_{rs} = \sum_{m=1}^{M} I_{ms}/(X_m, R_{mr}) \quad (23)$$

Equação 23 – Definição de Requisito Fuzzy

Onde X_m é o nome da *m*-ésima variável linguística empregada na representaçãodo *m*-ésimo atributo de uma situação desejada *fuzzy*; $R_{mr} \in F(U)$ são valores linguísticos de X_m que pertencem a função *T(Aspiração)* e representam os níveis realizados no atributo X_m em uma meta *fuzzy* G_m dada a implementação do requisito, e $I_{ms} \in F(U)$ é um valor linguístico de importância em *T(Importância)* descrevendo a importância do requisito para o *stakeholder s*.

Da mesma maneira que um *stakeholder* descreve os valores de importância dos requisitos que deseja na formulação, a organização descreve os valores de importância associados aos diversos *stakerholders* que a compõem e estão envolvidos com o processo de priorização. A definição a seguir formaliza a noção de *stakeholder fuzzy*.

Definição 4. Uma *organização Org* é dita *fuzzy* se for representada por um conjunto *fuzzy* tipo-2 descrito em termos dos *stakeholders* St_s que a compõem e de seus valores de importância *fuzzy* I_s (s = 1, ..., Ns) no contexto da organização:

$$Org = \sum_{s=1}^{Ns} I_s/St_s \quad (24)$$

Equação 24 – Definição de Organização Fuzzy

Onde $I_s \in F(U)$ é um valor linguístico de importância em *T(Importância)*.

4.3.2 Função Avaliação dos Requisitos dos *Stakeholders*

A abordagem esboçada na Seção 4.2, mais especificamente o passo (P1), considera que existe uma função capaz de medir a similaridade entre uma determinada situação desejada pela organização, conforme Definição 2, e as opiniões de cada um dos

stakeholders a respeito da realização prometida nas metas de uma situação desejada, por cada um dos requisitos importantes de sua preferência, conforme Definição 3. Assim, essa função deve levar em consideração os valores linguísticos de importância e de níveis aspirados nas metas *fuzzy* de uma situação desejada *fuzzy*, e os valores de importância e de realização prometida por um requisito *fuzzy* pretendido por um o *stakeholder* em particular.

A utilização desta função para cada um dos requisitos *r* de um *stakeholder s*, permite obter um valor de avaliação que indica o quanto um requisito está alinhado com a situação desejada do projeto e, ainda, uma certa "sinergia" entre aquilo que a organização e o *stakeholder* desejam.

$$f_{rs}(Re_{rs}, DS) = \sum_{m=1}^{M} \alpha(D1_{mrs}, D2_m)/M \tag{25}$$

Equação 25 – Função Avaliação

Onde $D1_{mrs}$ e $D2_m$ são relações *fuzzy* em *F(UxU)* que descrevem respectivamente a importância da realização prometida pelo requisito *r* na meta *m* para o stakeholder *s* e a importância do nível aspirado na meta *m* da organização; e α é a medida de similaridade entre as duas relações *fuzzy*.

Neste trabalho foi empregada a medida de similaridade associada à distância euclidiana *fuzzy* com *p=2*, definida pela Equação 8 no Capítulo 3, ou seja:

$$\alpha(D1_{mrs}, D2_m) = 1 - \left(\sum_{i=1}^{Nu}\sum_{j=1}^{Nu}\left|D1_{mrs}(u_i, u_j) - D2_m(u_i, u_j)\right|^2\right)^{1/2} \Big/ Nu \tag{26}$$

Equação 26 – Similaridade entre Conjuntos Fuzzy

e o operador pseudo-complemento, definido de acordo com a Equação 6 no Capítulo 3, para expressar o significado das relações *fuzzy* envolvidas, ou seja:

$$D1_{mrs}(u_i, u_j) = \varphi(I_{ms}(u_i), R_{mr}(u_j)) \quad (27)$$

Equação 27 – Operador pseudo-complemento aplicado à relação entre requisito r na meta m para o *stakeholder*

e

$$D2_m(u_i, u_j) = \varphi(I_m(u_i), A_m(u_j)) \quad (28)$$

Equação 28 – Operador pseudo-complemento aplicado à relação entre a importância do nível aspirado na meta m da organização

Onde $(u_i, u_j) \in UxU$.

Vale ressaltar que o segundo termo da diferença na Equação 26 trata-se da distância Euclidiana entre conjuntos *fuzzy* (LAZIM e ABU OSMAN, 2009). Sendo interessante ainda notar que: α: $F(UxU)xF(UxU) \rightarrow [0, 1]$. Portanto, à medida que a similaridade entre $D1_{mrs}$ e $D2_m$ aumenta, f_s também aumenta. Nesse caso, para cada meta m = 1, ..., M, se $D1_{mrs} = D2_m$ então f_s chega ao valor máximo e é igual a 1. Da mesma forma, na proporção que o valor de similaridade diminui, f_s decresce, chegando ao valor mínimo de 0. Essas propriedades permitem avaliar e priorizar os requisitos desejados por cada *stakeholder* levando em consideração a situação desejada da organização.

Além do mais, vale ressaltar também, a utilização do operador pseudo-complemento para o cálculo das relações *fuzzy* se justifica levando em consideração que o valor calculado pelo mesmo prioriza os valores de importância associados aos requisitos preferidos pelos *stakeholders* e às metas estratégicas, em detrimento dos níveis realizados prometidos pelos requisitos e aspirados nestas metas.

4.3.3 Sistema de Inferência *Fuzzy*

O Sistema de Inferência *Fuzzy* utilizado nesse trabalho segue a especificação definida na Figura 9 do Capítulo 3. Nesse sistema, definiram-se como configuração de

entrada duas variáveis uma de valor crisp “Similaridade do Requisito” cujo valor é calculado pelo algoritmo de priorização, e “Importância do *Stakeholder*”, no contexto da organização, um número *Fuzzy* dado como entrada no início da avaliação.

Na etapa de “Fuzificação” o sistema converte o valor da variável “Similaridade do Requisito” para um valor *fuzzy* conforme a função de pertinência do conjunto Universo definido para o problema. Nesse caso, a função escolhida para representar valores relativos à “similaridade” foi a triangular conforme definido na Equação 16 da seção 3.5. Já para representação de valores referentes à “importância”, a função escolhida foi a trapezoidal seguindo especificação da Equação 17 na seção 3.5.

Durante a fase de “Inferência”, as variáveis de entrada são confrontadas com um conjunto de regras *fuzzy* no formato $se\ \{condição\}\ então\ \{consequência\}$ que calculam o valor de saída da variável *fuzzy* “Importância do Requisito”. Para realizar esse cálculo, um conjunto de operadores é utilizado. Para as operações de Negação, União, Interseção e Inferência, são utilizados respectivamente Complemento, definido na Equação 3, Máximo, Equação 4, Mínimo, Equação 5, e o operador de Mamdani, Equação 13, todos definidos no capítulo 3.

Na última etapa realizada pelo sistema de inferência, a variável de saída “Importância do Requisito” calculada no passo anterior, é defuzificada gerando assim um valor crisp correspondente. A Figura 13 fornece uma visão geral sobre os passos realizados pelo sistema de inferência.

Figura 13 – Sistema de Inferência Fuzzy

4.4 Algoritmo

Esta seção apresenta o algoritmo de avaliação dos requisitos utilizado dentro do framework. Os passos realizados podem ser vistos na Figura 14. Nela percebe-se que as entradas necessárias para a priorização de requisitos são a lista dos requisitos, definido em Equação 23, o conjunto de metas que compões a situação desejada, Equação 21 e Equação 22, bem como a lista de importâncias fuzzy de cada meta e finalmente o conjunto contendo os *stakeholders*, e suas importâncias no contexto da organização, conforme a Equação 24.

Algoritmo 1: Priorização de Requisitos

```
entradas: requisitos //conjunto de requisitos a serem priorizados
          metas //conjunto de metas fuzzy que compõem a situação desejada
          importancias //conjunto de importancias fuzzy de cada meta
          stakeholders //conjunto de stakeholders

1  relacaoMetasImportancia ← calcularRelacaoMetasImportancia(metas, importancias);
2  relacaoRequisitosStakeholder ← calcularRelacaoRequisitosStakeholders(requisitos, stakeholders);
3  para i de 1 até tamanho(requisitos) faça
4  |       diferencas ← calculardiferencas(relacaoRequisitosStakeholder[i], relacaoMetasImportancia);
5  |       similaridades ← calcularSimilaridade(diferencas);
6  |       avaliacoesFuzzy ← calcularAvaliacoesFuzzy(similaridades, stakeholders);
7  |       requisitos[i].avaliacao ← calcularMaxMin(avaliacoesFuzzy);
8  fim do para
9  exibir requisitos ordenados por sua avaliacao
```

Figura 14 – Algoritmo de Priorização de Requisitos

Para descrever cada passo do algoritmo de priorização de requisitos, utilizaremos como entrada as informações na Figura 15. Nela percebe-se que o conjunto universo (U) utilizado é discreto e composto por apenas dois valores U = {0.0,1.0}. Cada variável linguística representada possui dois valores de U, como no caso de Médio = {0.5/0.0+ 0.5/1.0}, onde 0.5/0 e 0.5/1.0 representa a contribuição do conjunto Médio em U com 0.5 para o valor 0.0 e 0.5 para o valor 1.0, ou seja, Médio = {0.5, 0.5} e o sinal “+”

representa a união *fuzzy*. Dessa forma, os exemplos mostrados a seguir terão como informação inicial os dados da referida imagem.

INFORMAÇÕES DE ENTRADA

Conjunto Universo (U) = {0.0, 1.0}
Valor Fuzzy Zero(0) = {0.0/0.0 + 0.0/0.0} Valor Fuzzy Médio (m) = {0.5/0.0 + 0.5/1.0}
Valor Fuzzy Baixo (b) = {0.9/0.0 + 0.0/1.0} Valor Fuzzy Alto (a) = {0.0/0.0 + 0.9/1.0}

Situação Desejada para um produto "P"			
Nível de Aspiração das Metas		Meta 1	Meta 2
		m	a
Importâncias das Metas		b	a
Nível de realização dos Requisitos nas Metas	Requisito 1	m	a
	Requisito 2	b	a
	Requisito 3	b	m

Stakeholders		
Importância dos Stakeholders para o produto "P"	Stakeholder 1	Stakeholder 2
	a	m
Importância dos Requisitos para os Stakeholders		
Requisito 1	a	b
Requisito 2	0	a
Requisito 3	a	a

Figura 15 – Informações de Entrada para Algoritmo de Priorização de Requisitos

As operações inicias realizadas no algoritmo de priorização são *calcularRelacaoMetasImportancia* e *calcularRelacaoRequisitosStakeholders* , linhas 1 e 2. Essas funções, embora diferentes na nomenclatura, são bastante semelhantes no funcionamento. Trate-se de aplicar o operador de pseudo-complemento φ definido na Equação 6 sobre os conjuntos avaliados. No caso da relação entre metas e importâncias, o operador será aplicado sobre o valor desejado em uma meta e sua respectiva importância. Já na relação requisitos e *stakeholders*, o operador será aplicado sobre cada valor de realização do requisito nas metas em relação à importância do requisito avaliada pelo *stakeholder*.

A Figura 16 apresenta um exemplo de utilização. Nela percebe-se que o resultado da utilização do operador gera uma matriz quadrada onde os elementos são iguais a “1”. Por exemplo, o caso envolvendo o valor Desejado na Meta 2 e o valor de Importância da Meta 2, obteve-se a matriz $\begin{vmatrix} 1 & 1 \\ 0 & 1 \end{vmatrix}$.

RELAÇÃO METAS x IMPORTANCIA

	IMP. META1	
META1	0,9	0
0,5	1	0
0,5	1	0

	IMP. META2	
META2	0	0,9
0	1	1
0,9	0	1

RELAÇÃO REQUISITO x IMPORTÂNCIA DO REQUISITO

REQUISITO 1

	META 1	
	STKAKEHOLDER 1	
REQUISITO 1	0	0,9
0,5	0	1
0,5	0	1

	META 2	
	STKAKEHOLDER 1	
REQUISITO 1	0	0,9
0	1	1
0,9	0	1

	META 1	
	STKAKEHOLDER 2	
REQUISITO 1	0,9	0
0,5	1	0
0,5	1	0

	META 2	
	STKAKEHOLDER 2	
REQUISITO 1	0,9	0
0	1	1
0,9	1	0

Figura 16 – Exemplo de uso do Operador de Pseudo-Complemento

Os próximos passos do algoritmo, linhas 3 a 8, realizam uma avaliação individual de cada requisito. A primeira operação realizada é *calcularDiferencas*. Esse procedimento é responsável por calcular a Distância Euclidiana conforme definido na Equação 8 entre as relações *fuzzy* que representam os requisitos dos stakeholders e as relações *fuzzy* que representam as metas e importâncias em uma situação desejada. O resultado obtido no exemplo pode ser visto na Figura 17. Nessa figura, por exemplo, está representado que a diferença entre R1M1STK1 e M1I1 é 0,6666667, onde R1M1STK1, significa a relação entre o Requisito 1, Meta 1 e *Stakeholder* 1 e M1I1 é a relação entre a Meta 1 e seu respectivo valor de Importância.

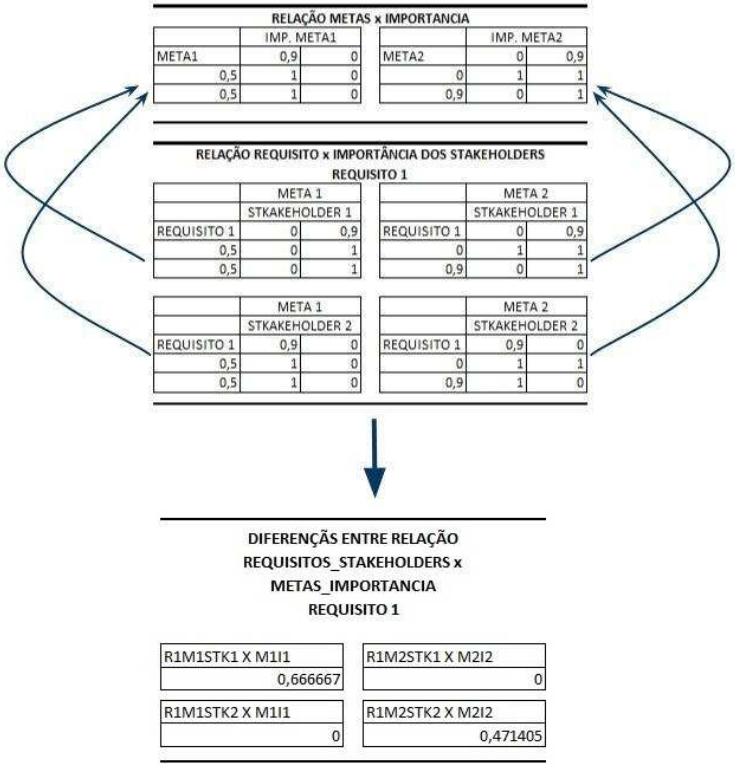

Figura 17 – Diferenças entre relação Requisitos/*Stakeholders* e Metas/Importâncias

A operação *calcularSimilaridades*, linha 5 do algoritmo, utiliza o conceito definido na Equação 26 que calcula a similaridade entre conjuntos *fuzzy*, para tal, são utilizados os valores das diferenças calculadas no passo anterior. O resultado dessa operação pode ser visto na Figura 18, onde SIMI(R1M2STK2 X M2I2) = 0,528595479 representa a similaridade entre as relações Requisito 1, Meta 2 e *Stakeholder* 2 e Meta 2, Importância da Meta 2. O valor de similaridade representa o nível de semelhança entre os conjuntos avaliados. Quanto maior a similaridade, maior a semelhança.

SIMILARIDADES

REQUISITO 1

SIMI(R1M1STK1 X M1I1)	SIMI(R1M2STK1 X M2I2)
0,333333333	1

SIMI(R1M1STK2 X M1I1)	SIMI(R1M2STK2 X M2I2)
1	0,528595479

REQUISITO 2

SIMI(R2M1STK1 X M1I1)	SIMI(R2M2STK1 X M2I2)
0,528595479	0,666666667

SIMI(R2M1STK2 X M1I1)	SIMI(R2M2STK2 X M2I2)
0,528595479	0,666666667

REQUISITO 3

SIMI(R1M1STK1 X M1I1)	SIMI(R1M2STK1 X M2I2)
0,422649731	0,666666667

SIMI(R1M1STK2 X M1I1)	SIMI(R1M2STK2 X M2I2)
0,422649731	0,666666667

Figura 18 – Cálculo das Similaridades para cada Requisito

A operação *calcularAvaliacoesFuzzy* utiliza o conceito de sistema *Fuzzy* definido em 3.7. Para tal, foi construído um sistema *Fuzzy* utilizando a ferramenta Xfuzzy 3.0 (INSTITUTO DE MICROELECTRONICA DE SEVILLA, 2003) onde foram definidas duas variáveis linguísticas de entrada similaridade_requisito, calculada no passo anterior do algoritmo, e importancia_stakeholder, importância do *stakeholder* no contexto da organização para o produto “P”, e uma variável linguística de saída avaliacao_requisito. Essas variáveis estão sujeitas a um conjunto de regras *fuzzy* que guiam o comportamento da avaliação. Os valores de entrada e as regras definidas podem ser vistos na Figura 19 e Tabela 1.

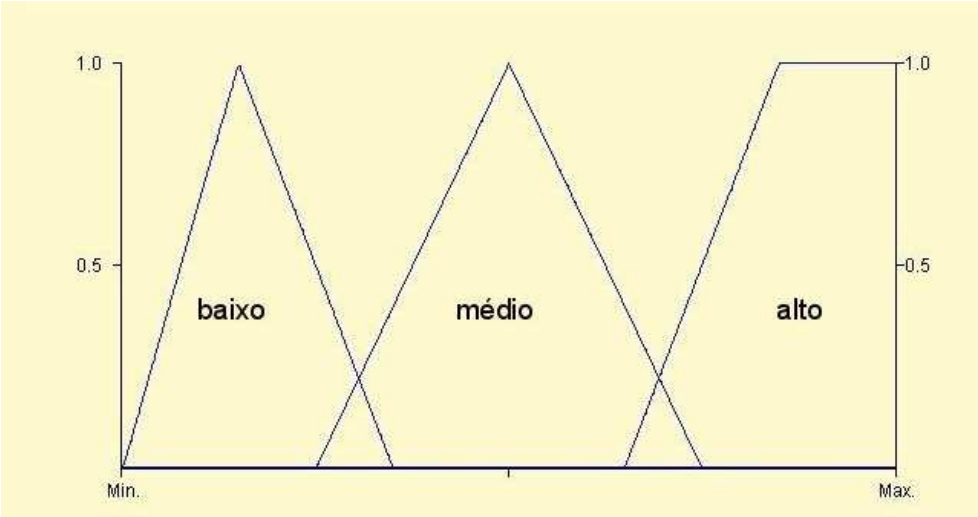

Figura 19 – Valores Fuzzy

Regra	Premissa	Conclusão
1	similaridade_requisito = baixo & importancia_stakeholder = baixo	avaliacao_requisito = baixo
2	similaridade_requisito = baixo & importancia_stakeholder = médio	avaliacao_requisito = baixo
3	similaridade_requisito = baixo & importancia_stakeholder = alto	avaliacao_requisito = médio
4	similaridade_requisito = médio & importancia_stakeholder = baixo	avaliacao_requisito = baixo
5	similaridade_requisito = médio & importancia_stakeholder = médio	avaliacao_requisito = médio
6	similaridade_requisito = médio & importancia_stakeholder = alto	avaliacao_requisito = médio
7	similaridade_requisito = alto & importancia_stakeholder = baixo	avaliacao_requisito = médio
8	similaridade_requisito = alto & importancia_stakeholder = médio	avaliacao_requisito = médio
9	similaridade_requisito = baixo & importancia_stakeholder = alto	avaliacao_requisito = alto

Tabela 1 – Regras Fuzzy

O valor de similaridade é fuzificado para cada um dos valores *fuzzy* possíveis de entrada, nesse caso, baixo, médio ou alto. Enquanto que o valor de importância do *stakeholder* já é recebido na entrada do algoritmo com seu valor *fuzzy* correspondente. O resultado da inferência *fuzzy* realizada sobre os valores de entrada é um número *fuzzy*, que, após ser defuzificado, é transformado em um valor real no intervalo [0.0 e 1.0] e retornado pela função *calcularAvaliacoesFuzzy*.

A Figura 20 apresenta os valores resultantes após a avaliação *fuzzy* para cada um dos valores de similaridades calculados no passo anterior do algoritmo, onde AVFUZZY(SIMI(R1M2STK1 X M2I2)), corresponde a avaliação *fuzzy* da similaridade calculado anteriormente para Requisito 1, Meta 2 e a importância do *stakeholder* 1 para o produto “P” em questão.

AVALIAÇÕES FUZZY

REQUISITO 1

AVFUZZY(SIMI(R1M1STK1 X M1I1))	AVFUZZY(SIMI(R1M2STK1 X M2I2))
0,5	0,870959

AVFUZZY(SIMI(R1M1STK2 X M1I1))	AVFUZZY(SIMI(R1M2STK2 X M2I2))
0,5	0,49

REQUISITO 2

AVFUZZY(SIMI(R2M1STK1 X M1I1))	AVFUZZY(SIMI(R2M2STK1 X M2I2))
0,5	0,539368

AVFUZZY(SIMI(R2M1STK2 X M1I1))	AVFUZZY(SIMI(R2M2STK2 X M2I2))
0,5	0,5

REQUISITO 3

AVFUZZY(SIMI(R3M1STK1 X M1I1))	AVFUZZY(SIMI(R3M2STK1 X M2I2))
0,5	0,530368

AVFUZZY(SIMI(R3M1STK2 X M1I1))	AVFUZZY(SIMI(R3M2STK2 X M2I2))
0,5	0,5

Figura 20 – Avaliações Fuzzy

A última operação realizada pelo algoritmo de priorização de requisitos é uma operação de agregação dos valores das avaliações *fuzzy* de cada requisito. Na linha 7 da Figura 14 tem-se a operação *calcularMaxMin* cujo resultado é um valor real compreendido no intervalo [0,1.0] que determina a avaliação final do requisito. A Figura 21 apresenta o resultado da aplicação da função.

MAXMIN

REQUISITO 1

AVFUZZY(SIMI(R1M1STK1 X M1I1))	AVFUZZY(SIMI(R1M2STK1 X M2I2))
0,5	0,870959

AVFUZZY(SIMI(R1M1STK2 X M1I1))	AVFUZZY(SIMI(R1M2STK2 X M2I2))
0,5	0,49

MAX(MIN(0,5 e 0,5), MIN(0,870959 e 0,49)) = 0,5

Figura 21 – Resultado da Agregação MAXMIN

Nessa figura, percebe-se que inicialmente é calculado o valor mínimo entre os valores de avaliação das similaridades entre requisitos, metas e *stakeholders* com suas respectivas importâncias. Após essa operação inicial, é calculado o valor máximo entre os valores resultantes da operação anterior. Esse valor corresponde à prioridade do requisito avaliado. Após a execução de todos os passos do algoritmo, os requisitos estão avaliados e prontos para serem ordenados segundo seu valor de prioridade.

5 AVALIAÇÃO DA ABORDAGEM

Este capítulo apresenta alguns estudos realizados com o algoritmo de priorização programado a partir do *framework* proposto no capítulo anterior. Ele está dividido em duas seções. Em cada uma delas, são apresentadas informações sobre o exemplo a ser avaliado, o problema de priorização específico, a solução produzida pelo algoritmo e uma análise dos resultados obtidos.

Foram realizados dois tipos de avaliações. Inicialmente, na Seção 5.1, foi estudado apenas o comportamento do framework relativo ao relacionamento entre metas que compõem uma situação desejada e os requisitos. Apenas as metas e requisitos possuem valores *fuzzy*, e o valor de importância das metas foi tratado como um valor real representado no intervalo [0, 1.0]. A segunda avaliação, Seção 5.2, foi mais abrangente, contemplando todo o conceito de representação do framework, onde metas, importância, requisitos e *stakeholders* são representados por meio de valores *fuzzy*. Já na Seção 5.3 buscou-se avaliar a capacidade de representação do framework, bem como verificar a possibilidade de alcançar a avaliação máxima dos requisitos.

5.1 Avaliação 1

A instância de avaliação utilizada inicialmente para realizar os testes com o *framework* para priorização de requisitos baseado em uma abordagem linguística *fuzzy* é composta por sete metas e nove requisitos. Metas como essas são frequentemente utilizadas em (COHN, 2010) para ajudar os *stakeholders* a refletir sobre as dimensões de sucesso em um projeto de desenvolvimento de software. O objetivo da avaliação é verificar a capacidade do framework de realizar a tarefa de priorização e avaliar seu nível de resposta à ocorrência de mudanças nos valores de avaliação dos requisitos e metas.

A Tabela 2 descreve alguns termos linguísticos e suas representações em conjuntos *fuzzy*. A Tabela 3, segundo a definição 1, descreve o cenário com sete atributos (X_m) que definem sete metas *fuzzy* (G_m). Conforme a definição 2, a tabela apresenta também a descrição da aspiração em cada meta e seu nível de importância, inicialmente real, não *fuzzy*, (μ_m), que define a situação desejada *(DS)* (22). Finalmente, a tabela descreve, de acordo com a definição 3 (23), o nível realizado de cada um dos requisitos (Re_m) em relação às metas em questão. As metas que definem a situação desejada são representadas pelo seguinte conjunto de valores *fuzzy* já definidos: [m, a, b, a, b, a, m].

Considerou-se, como exemplo, o desenvolvimento de um sistema para Terminais de Auto Atendimento (TAAs) onde os requisitos e metas do projeto foram mapeados. O atributo X_1 representa a meta "Entregar todas as funcionalidades de alta prioridade na primeira release do produto" e possui um nível desejado ou aspirado médio(m) para a referida meta. Os requisitos Re_4 e Re_7 são definidos como "Realizar Saque" e "Exibir mensagem de boas vindas" respectivamente. Neste exemplo, o requisito 4 realiza a meta X_1 em um alto nível (a), conforme pode ser percebido na Tabela 3.

Termo Linguístico	Representação do Conjunto *Fuzzy*
baixo (b)	[1.0, 1.0, 0.9, 0.7, 0.5, 0.2, 0.0, 0.0, 0.0, 0.0, 0.0]
médio (m)	[0.0, 0.0, 0.0, 0.8, 0.9, 1.0, 0.7, 0.0, 0.0, 0.0, 0.0]
alto (a)	[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.3, 0.5, 1.0, 1.0, 1.0]

Tabela 2 – Descrição de cada prioridade em termos linguísticos e sua representação em conjuntos fuzzy

Metas (X_m)	X_1	X_2	X_3	X_4	X_5	X_6	X_7
Importância (μ_m)	0.6	1.0	0.4	0.2	0.4	0.8	0.6
Situação Desejada (*DS*)	m	a	b	a	b	a	m
Re_1	m	a	b	a	b	a	m
Re_2	a	m	a	m	a	b	b
Re_3	a	a	m	a	b	m	a
Re_4	a	b	m	a	b	b	a
Re_5	m	b	b	m	a	a	m
Re_6	a	b	m	a	b	m	a
Re_7	b	m	m	a	b	a	a
Re_8	m	m	m	a	b	a	a
Re_9	a	m	m	a	b	a	a

Tabela 3 – Descrição de cada requisito fuzzy, nível de contribuição para alcance de cada meta e situação desejada de cada meta

Apenas para essa primeira avaliação, o valor de importância foi considerado numérico, não fuzzy, funcionando como um valor de ponderação. Dessa forma, as funções de avaliação assumem o formato definido nas equações Equação 29 e Equação 30 a seguir.

$$f_{eval(Re_r,DS)} = \sum_{m=1}^{M} \mu_m \alpha(R_{mr}, A_m) \Big/ \sum_{m=1}^{M} \mu_m \tag{29}$$

onde α é a medida de similaridade entre conjuntos fuzzy que, nessa avaliação assume o formato a seguir:

$$\alpha(R_{mr}, A_m) = 1 - \left(\sum_{i=1}^{L} |R_{mr}(u_i) - A_m u_i|^2\right)^{1/2} \Big/ L^{1/2} \tag{30}$$

onde L é o número de elementos no universo de discurso empregado para definir os valores fuzzy como, por exemplo na Tabela 2 foi adotado um universo de discurso discreto composto de onze elementos, ou seja, L = 11.

5.1.1 Resultados

A Tabela 4 descreve os resultados da execução de três testes. Em cada coluna, está identificado o requisito e um número real representando o nível de similaridade do requisito relativo às metas. Em cada teste foi alterado apenas o valor de uma variável a fim de verificar quais alterações seriam percebidas na lista priorizada.

O Teste 1 foi utilizado como controle, onde o requisito Re_1 foi propositalmente deixado igual a situação desejada em questão. Já no Teste 2, a situação desejada foi alterada para [m, **b**, b, a, b, a, m], onde apenas o nível desejado na meta em X_2 foi alterado de alto para baixo. No Teste 3, houve uma mudança na importância de G_4 saindo de 0.2 para 0.8.

Teste 1	Teste 2	Teste 3
Re_1,1.00	Re_5,0.88	Re_1,1.00
Re_5,0.68	Re_1,0.80	Re_8,0.69
Re_8,0.64	Re_8,0.67	Re_5,0.63
Re3,0.57	Re_7,0.58	Re_3,0.62
Re_7,0.55	Re_6,0.57	Re_7,0.61
Re_9,0.53	Re_9,0.56	Re_9,0.59
Re_6,0.37	Re_4,0.56	Re_6,0.45
Re_4,0.36	Re_3,0.37	Re_4,0.44
Re_2,0.26	Re_2,0.29	Re_2,0.26

Tabela 4 – Lista priorizada geradas a partir de cada teste. O primeiro valor trata-se do número do requisito e o segundo é sua similaridade resultante da avaliação relativa ao atendimento das metas do projeto

Conforme já esperado, no Teste 1 Re_1 alcançou 100% de similaridade com as metas, tendo em vista que este requisito realiza exatamente situação desejada *fuzzy* [m,a,b,a,b,a,m], esse resultado também foi percebido no Teste 3. Houve um reordenamento entre os requisitos nos Testes 2 e 3 influenciado pelas mudanças no nível aspirado e no valor da importância. No Teste 2, por exemplo, nenhum requisito atingiu o nível máximo de similaridade, tendo em vista que, com a mudança no nível aspirado na meta G2, o requisito Re_1, que realiza [m,**a**,b,a,b,a,m], não mais realiza a nova situação desejada *fuzzy* [m,**b**,b,a,b,a,m]. Para o Teste 3, o requisito 1 manteve-se no topo da lista, e o aumento no nível de importância de G_4 motivou o novo reordenamento. Caso todos os requisitos estivessem exatamente iguais a situação desejada, como Re_1, ocorreria 100% de similaridade em todos os casos.

5.2 Avaliação 2

Nessa segunda parte do processo de avaliação do framework, todos os valores envolvidos na representação das metas, importâncias, requisitos e *stakeholders* são *fuzzy*. Nesse caso, faz-se necessário avaliarmos a capacidade de representação dos valores linguísticos do framework, bem como sua capacidade de alcançar resultados satisfatórios. Dessa forma, iremos utilizar inicialmente como conjunto de controle a

mesma quantidade de requisitos e metas definidas na avaliação anterior, 9 requisitos e 7 metas que representarão a situação desejada dentro ainda do contexto de um sistema para TAAs. São acrescentados 3 stakeholders para completar todas as variáveis do framework. Os valores estão descritos nas Tabelas 5-7. É importante notar que o número de valores das variáveis linguísticas aumentou de três (baixo, médio, alto) para seis (zero, muito baixo, baixo, médio, alto e muito alto), possibilitando maior poder de representação.

Termo Linguístico	Representação do Conjunto *Fuzzy*
zero(0)	[1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
muito baixo(t)	[1.0, 0.7, 0.5, 0.1, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
baixo (b)	[0.0, 0.5, 1.0, 0.5, 0.2, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
médio (m)	[0.0, 0.0, 0.0, 0.6, 0.8, 1.0, 0.7, 0.5, 0.0, 0.0, 0.0]
alto (a)	[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.8, 1.0, 0.4, 0.0, 0.0]
muito alto(g)	[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.2, 0.6, 0.8, 1.0]

Tabela 5 – Descrição da nova representação dos termos linguísticos

Metas (X_m)	X_1	X_2	X_3	X_4	X_5	X_6	X_7
Importância	b	a	a	m	b	b	a
Situação Desejada (*DS*)	g	a	b	a	b	a	m
Re_1	g	a	b	a	b	a	m
Re_2	a	m	a	m	t	b	b
Re_3	b	m	b	b	b	b	b
Re_4	a	b	m	a	b	b	a
Re_5	m	b	t	m	a	a	m
Re_6	a	b	g	a	b	m	t
Re_7	b	m	m	a	g	a	a
Re_8	t	m	m	a	b	a	a
Re_9	a	t	g	t	g	t	t

Tabela 6 – Definição dos Requisitos, Metas e suas respectivas Importâncias

Stakeholders (St_r)	St_1	St_2	St_3
Importância para Produto	a	b	m
Re_1	a	a	a
Re_2	a	m	a
Re_3	t	m	0
Re_4	0	a	a
Re_5	m	0	0
Re_6	a	b	a
Re_7	m	a	a
Re_8	m	a	a
Re_9	m	b	0

Tabela 7 – Definição dos *Stakeholders*, Importância do *Stakeholder* para o Produto e Importâncias dos Requisitos para os *Stakeholders*

Conforme mencionado na Seção 4.3.3, o sistema *fuzzy* utilizado proposto no *framework* é composto por duas variáveis linguísticas de entrada, *similaridade_requisito* e *importância_stakeholder* e uma variável linguística de saída *avaliacao_requisito*. A Figura 22 apresenta os valores linguísticos das variáveis e seus respectivos conjuntos *fuzzy*.

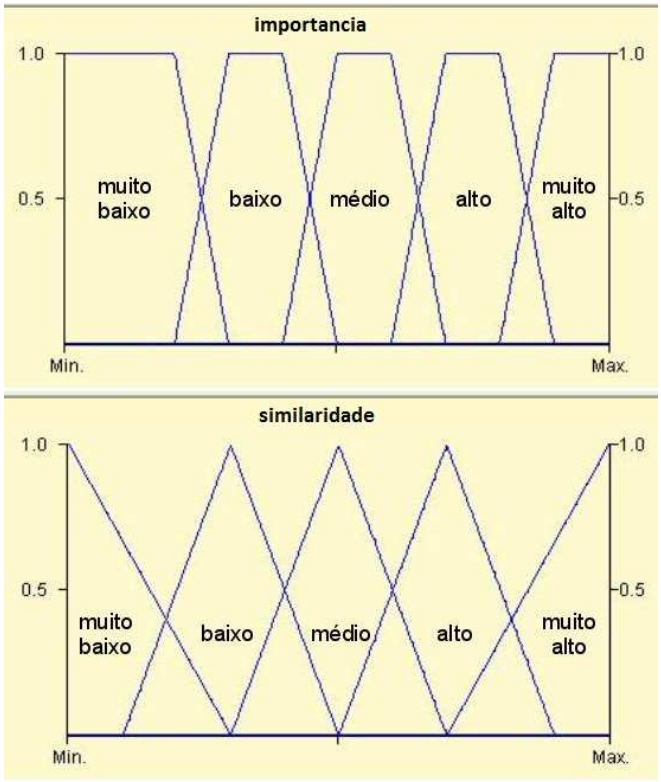

Figura 22 – Definição das variáveis de entrada e saída do sistema fuzzy

O conjunto de regras utilizado para definir o comportamento durante a avaliação *fuzzy* pode ser visto na Figura 10. É importante notar que durante a construção das regras, buscou-se um equilíbrio entre a avaliação da similaridade e importância do *stakeholder*, de forma que, por exemplo, caso um *stakeholder* associe uma importância muito alta a um requisito com similaridade baixa, a avaliação final do referido requisito será “médio”.

5.2.1 Resultados

Foram realizados testes a fim de avaliar mais uma vez a sensibilidade do framework às alterações em todos os seus componentes: Situação Desejada, Importância das Metas, Requisitos, Avaliação dos Requisitos por parte dos Stakeholders e Importância dos *Stakeholders*. Também foi avaliada a capacidade de representação do framework e se as soluções produzidas são satisfatórias. O Teste 1, avaliação de controle, foi realizado com as informações detalhadas na seção anterior. Os parágrafos a seguir descrevem os resultados alcançados em cada uma das instâncias.

No Teste 2, mostrado na Tabela 8, a situação desejada foi alterada em 4 metas de [g, a, b, a ,b, a ,m] para [**t**, **b**, **t**, **m**, b, a, m] gerando assim uma reordenação em todos os requisitos envolvidos na avalição em referência ao teste de controle.

Teste 1 (Controle)	Teste 2 – Situação Desejada
Re_1, 0.724	Re_1, 0.724
Re_6, 0.637	Re_4, 0.724
Re_5, 0.609	Re_2, 0.697
Re_4, 0.568	Re_6, 0.637
Re_7, 0.568	Re_5, 0.609
Re_8, 0.568	Re_7, 0.517
Re_3, 0.447	Re_8, 0.517
Re_2, 0.391	Re_3, 0.447
Re_9, 0.390	Re_9, 0.343

Tabela 8 – Teste 2: Alteração dos Valores da Situação Desejada (DS)

No Teste 3, detalhado na Tabela 9, o valor da Importância das metas foi alterada em apenas um valor de [b, a, a, m, b, b, a] para [b, a, a, m, **g**, b, a], indicando assim que a meta 5 deixou de ter “baixa” importância para “muito alta”. Essa alteração gerou, mais uma vez, uma reordenação em boa parte dos os requisitos envolvidos, bem como modificou também seus valores de avaliação. Resultados como este são esperados,

tendo em vista que todos os requisitos contribuem com algum valor para a realização da meta em questão, no caso a meta 5.

Teste 1 (Controle)	Teste 3 – Valor de Importância das Metas
Re_1, 0.724	Re_1, 0.724
Re_6, 0.637	Re_5, 0.609
Re_5, 0.609	Re_4, 0.568
Re_4, 0.568	Re_7, 0.568
Re_7, 0.568	Re_8, 0.568
Re_8, 0.568	Re_6, 0.522
Re_3, 0.447	Re_2, 0.391
Re_2, 0.391	Re_9, 0.390
Re_9, 0.390	Re_3, 0.383

Tabela 9 – Teste 3: Alteração do valor de Importância da Meta 5

No Teste 4, detalhado na Tabela 10, o valor do nível de realização do Requisito 9 na Meta 7 foi alterado de [a, t, g, t, g, t, t] para [a, t, g, t, g ,t, **m**], definindo assim uma nova ordem na priorização, onde o Requisito 9 passa a ocupar a posição que antes pertencia ao Requisito 2. É importante ainda perceber que com a alteração descrita, apenas o valor de avaliação do Requisito 9 foi alterado. Os valores dos outros requisitos permaneceram os mesmos. A alteração do valor de apenas um requisito é esperado, tendo em vista que o nível de realização do requisito é um valor que, nesta abordagem não interfere nos níveis de realização dos demais.

Teste 1 (Controle)	Teste 4 – Valor de Realização dos Requisitos
Re_1, 0.724	Re_1, 0.724
Re_6, 0.637	Re_6, 0.637
Re_5, 0.609	Re_5, 0.609
Re_4, 0.568	Re_4, 0.568
Re_7, 0.568	Re_7, 0.568
Re_8, 0.568	Re_8, 0.568
Re_3, 0.447	Re_3, 0.447
Re_2, 0.391	Re_9, 0.447
Re_9, 0.390	Re_2, 0.391

Tabela 10 – Teste 4: Alteração no Valor de Realização do Requisito 9 para Meta 7

A Tabela 11 apresenta o resultado do Teste 5 onde são alterados apenas os valores de importância dos requisitos para os *stakeholders* na Tabela 7. Ao modificar o valor de importância do Requisito 5 pelo *Stakeholder* 2 de "Zero" para "Alto" [a, m, m, a, **0**, b, a, a, b] para [a, m, m, a, **a**, b, a, a, b] o valor final de avaliação do requisito aumentou de 0.609 para 0.724, ocupando assim uma posição acima na lista de priorização. Esse resultado é esperado, tendo em vista que a avalidação de um requisito não interfere na avaliação de outros. É interessante notar que os outros requisitos permaneceram com os mesmos valores de avaliação.

Teste 1 (Controle)	Teste 5 – Valor de Avaliação dos Requisitos pelos Stakeholders
Re_1, 0.724	Re_1, 0.724
Re_6, 0.637	Re_5, 0.724
Re_5, 0.609	Re_6, 0.637
Re_4, 0.568	Re_4, 0.568
Re_7, 0.568	Re_7, 0.568
Re_8, 0.568	Re_8, 0.568
Re_3, 0.447	Re_3, 0.447
Re_2, 0.391	Re_2, 0.391
Re_9, 0.390	Re_9, 0.390

Tabela 11 – Teste 5: Alteração no Valor de Avaliação do Requisito 5 pelo *Stakeholder* 2

O Teste 6 demonstra como o valor de Importância do *Stakeholder* no contexto da organização para o produto influencia a priorização de requisitos. Nesse teste, foi alterado o valor de importância do *Stakeholder* 2 de "Baixo" [b] para "Alto" [**a**]. Essa operação modificou a avaliação de todos os requisitos, criando uma nova ordem na lista de priorização, conforme os resultados mostrados na Tabela 12.

Teste 1 (Controle)	Teste 5 – Valor de Importância dos *Stakeholders*
Re_1, 0.724	Re_1, 0.924
Re_6, 0.637	Re_7, 0.758
Re_5, 0.609	Re_8, 0.758
Re_4, 0.568	Re_4, 0.750
Re_7, 0.568	Re_5, 0.749
Re_8, 0.568	Re_6, 0.722
Re_3, 0.447	Re_3, 0.647

Re_2, 0.391	Re_9, 0.550
Re_9, 0.390	Re_2, 0.549

Tabela 12 – Teste 6: Alteração no Valor de Importância do *Stakeholder* 2

5.3 Avaliação 3

Após executar testes com alterações em todas as variáveis disponíveis no framework, esta seção busca validar sua capacidade de representação. No Teste 7, os valores linguísticos de representação [zero, muito baixo, baixo, médio, alto e muito alto] foram de descritos através de um único conjunto *fuzzy* conforme definido na Tabela 13. Os resultados alcançados estão catalogados na Tabela 14, onde se percebe que se não fossem utilizados valores diferentes para os termos linguísticos, os resultados não seriam satisfatórios, tendo em vista que todos os requisitos foram avaliados com a mesma prioridade.

Termo Linguístico	Representação do Conjunto *Fuzzy*
zero(0)	[0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1]
muito baixo(t)	[0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1]
baixo (b)	[0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1]
médio (m)	[0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1]
alto (a)	[0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1]
muito alto(g)	[0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1]

Tabela 13 – Representação dos termos linguísticos com mesmo valor

Teste 1 (Controle)	Teste 2	Teste 3	Teste 4	Teste 5	Teste 6
Re_1, 0.749	Re_1, 0.749	Re_1, 0.749	Re_1, 0.749	Re_1, 0.749	Re_1, 0.749
Re_2, 0.749	Re_2, 0.749	Re_2, 0.749	Re_2, 0.749	Re_2, 0.749	Re_2, 0.749
Re_3, 0.749	Re_3, 0.749	Re_3, 0.749	Re_3, 0.749	Re_3, 0.749	Re_3, 0.749
Re_4, 0.749	Re_4, 0.749	Re_4, 0.749	Re_4, 0.749	Re_4, 0.749	Re_4, 0.749
Re_5, 0.749	Re_5, 0.749	Re_5, 0.749	Re_5, 0.749	Re_5, 0.749	Re_5, 0.749
Re_6, 0.749	Re_6, 0.749	Re_6, 0.749	Re_6, 0.749	Re_6, 0.749	Re_6, 0.749
Re_7, 0.749	Re_7, 0.749	Re_7, 0.749	Re_7, 0.749	Re_7, 0.749	Re_7, 0.749
Re_8, 0.749	Re_8, 0.749	Re_8, 0.749	Re_8, 0.749	Re_8, 0.749	Re_8, 0.749
Re_9, 0.749	Re_9, 0.749	Re_9, 0.749	Re_9, 0.749	Re_9, 0.749	Re_9, 0.749

Tabela 14 – Teste 7: Todos os valores linguísticos com mesmo valor de representação

No teste 8, buscou-se obter uma solução em que cada requisito alcança avaliação final igual a “1”. Para isso, é necessário que: (1) o nível de realização de todos os requisitos seja igual aos níveis aspirados nas metas da situação desejada; (2) as avaliações de importância de cada requisito por parte de cada stakeholder e os valores de importâncias das metas, (3) e que os valores de importância dos stakeholders no contexto da organização sejam o máximo possível. As Tabelas 15 e 16 apresentam uma combinação destes valores. A Tabela 17 apresenta os resultados. Como era de se esperar, os valores de avaliação para os requisitos foram muito próximos do de um.

Metas (X_m)	X_1	X_2	X_3	X_4	X_5	X_6	X_7
Importância	b	a	a	m	b	b	a
Situação Desejada (*DS*)	g	a	b	a	b	a	m
Re_1	g	a	b	a	b	a	m
Re_2	g	a	b	a	b	a	m
Re_3	g	a	b	a	b	a	m
Re_4	g	a	b	a	b	a	m
Re_5	g	a	b	a	b	a	m
Re_6	g	a	b	a	b	a	m
Re_7	g	a	b	a	b	a	m
Re_8	g	a	b	a	b	a	m
Re_9	g	a	b	a	b	a	m

Tabela 15 – Todos os requisitos atendem as metas definidas

Stakeholders (St_r)	St_1	St_2	St_3
Importância para Produto	g	g	g
Re_1	g	g	g
Re_2	g	g	g
Re_3	g	g	g
Re_4	g	g	g
Re_5	g	g	g
Re_6	g	g	g
Re_7	g	g	g
Re_8	g	g	g
Re_9	g	g	g

Tabela 16 – Requisitos com avaliação máxima e *Stakeholders* com importância máxima

Teste 1 (Controle)	Teste 8 – Valores próximos do ótimo
Re_1,0.724	Re_1, 0.924
Re_6,0.637	Re_2, 0.924
Re_5,0.609	Re_3, 0.924
Re_4,0.568	Re_4, 0.924
Re_7,0.568	Re_5, 0.924
Re_8,0.568	Re_6, 0.924
Re_3,0.447	Re_7, 0.924
Re_2,0.391	Re_8, 0.924
Re_9,0.390	Re_9, 0.924

Tabela 17 – Resultados alcançados próximos da solução ótima

Percebe-se nesta abordagem que alterações nos valores de avaliação de cada requisito não interferem na avaliação final dos outros. Isso se repete também para o caso do nível de realização das metas. Este cenário, embora representado em ênfase quando se trata do problema de planejamento de releases (ZHANG_A, 2010) e em processos de tomada de decisão baseado em metas (DRIANKOV, 1987), pode ser inicialmente desconsiderado quando se trata de um problema de priorização e classificação de requisitos, como é o caso desta abordagem, e representa um desafio a ser avaliado em trabalhos futuros.

Diferentemente do observado na Seção 5.1.1, não foi possível alcançar o valor de avaliação máximo, tendo em vista que a avaliação final é realizada por um sistema de regras *fuzzy* baseado nas regras que descrevem a opinião subjetiva de um tomador de decisão. Apesar desta aproximação parecer uma desvantagem, a capacidade de representar os valores de importância como números *fuzzy*, a possibilidade de controle sobre a avaliação obtida através do conjunto de regras e os resultados alcançados bem próximos dos valores desejados, são vantagens que merecem ponderação.

6 CONCLUSÃO

Priorização de requisitos é uma tarefa difícil de ser realizada. Trabalhos anteriores trataram o problema sem levar em consideração seu nível de indefinição presente no processo de desenvolvimento de software. Esse trabalho propôs um framework guiado por metas fuzzy que se propõe a resolver esse problema de priorização.

Além de apresentar o framework, exemplos para avaliar sua eficácia foram descritos. Os resultados mostraram a capacidade da abordagem em relação à flexibilidade que pode ocorrer na priorização em qualquer uma de suas variáveis disponíveis. Como visto nos testes, a variação de uma configuração leva a uma nova solução proposta pelo framework.

Assim, os objetivos definidos nesse trabalho foram alcançados tendo em vista que foi formalizada uma abordagem de priorização de requisitos de software capaz de representar as necessidades dos envolvidos de maneira mais próxima da linguagem humana, e que realiza sua avaliação considerando as metas definidas para o produto, a capacidade dos requisitos em concretizar as metas, a avaliação dos requisitos por parte dos *stakeholders* e a importância dos mesmos para o produto.

Estudos futuros podem incluir os resultados da priorização levando em conta a interdependência entre metas, requisitos, bem como aplicação da abordagem em um projeto real.

REFERÊNCIAS BIBLIOGRÁFICAS

AKAO, Y. New product development and quality. **Standardisation and Quality Control**, 1972. 243-246.

AURUM, A.; WOHLIN, C. The Fundamental Nature of Requirements Engineering Activities as a Decision-Making Process. **Information and Software Technology**, 2003. 945-954.

AURUM, A.; WOHLIN, C. **Engineering and Managing Software Requirements**. 1ª. ed. [S.l.]: Springer, 2005.

BAGNALL, A. J.; RAYWARD-SMITH, V. J.; WHITTLEY, I. M. The next release problem. **Information and Software Tecnology**, 2001. 883-890.

BAKER, P. et al. Search Based Approaches to Component Selection and Prioritization for the Next Release Problem. **International Conference on Software Maintenance**, 2006. 176-185.

BELLMAN, R.; ZADEH, L. A. Decision-making in a fuzzy environment. **Management Science**, 1970. 141-154.

BRASIL, M. M. A. et al. A New Multiobjective Optimization Approach for Release Planning in Iterative and Incremental Software Development (in Portuguese). **Proceedings of the Brazilian Workshop on Optimization in Software Engineering (WOES '10)**, 2010. 30-30.

CHAN, L.-K.; WU, M.-L. Quality function deployment: A literature review. **European Journal of Operational Research**, 2002. 463–497.

COHN, M. Project Sucess Sliders. **Mountain Goat Software**, 2010. Disponivel em: <http://www.mountaingoatsoftware.com/tools/project-success>. Acesso em: dez. 2010.

DRIANKOV, D. An outline of a fuzzy sets approach to decision making with interdependent goals. **Fuzzy Sets and Systems**, North-Holand, 1987. 275-288.

FEATHER, M. S.; MENZIES, T. Converging on the Optimal Attainment of Requirements. **Proceedings of the IEEE Joint International Conference on Requirements Engineering**, 2002.

FERNÁNDEZ, M. J.; SUÁREZ , F.; GIL, P. Equations of fuzzy relations defined on fuzzy subsets. **Fuzzy Sets and Systems**, 24 Dezembro 1992. 319-336.

FINKELSTEIN, A. et al. A search based approach to fairness analysis in requirement assignments to aid negotiation, mediation and decision making. **Requirements Engineering Journal**, v. 14, p. 231-245, Dezembro 2009.

FRANCESCHINI, F.; ROSSETO, S. QFD: an interactive algorithm for the prioritization of product´s technical design characteristics. **Integrated Manufacturing Systems**, 2002. 69-75.

GAUR, V.; SONI, A. An Integrated Approach to Prioritize Requirements using Fuzzy Decision Making. **IACSIT International Journal of Engineering and Technology**, Agosto 2010.

HARMAN, M.; JONES, B. Search-based Software Engineering. **Information and Software Technology**, 43, Dezembro 2001. 833-839.

INSTITUTO DE MICROELECTRONICA DE SEVILLA. An Overview of Xfuzzy 3.0. **Fuzzy Logic Design Tool**, 2003. Disponivel em: <http://www2.imse-cnm.csic.es/Xfuzzy/Xfuzzy_3.0/index.html>. Acesso em: 30 Maio 2010.

KARLSSON, J. Software Requirements Prioritizing. **Proceedings of the Second International Conference on Requirements Engineering**, Abril 1996. 110-116.

KARLSSON, J.; WOHLIN, C.; REGNELL, B. An evaluation of methods for prioritizing software requirements. **Information and Software Technology**, 1998. 939–947.

LAMSWEERDE, A. Goal-oriented requirements engineering: a guided tour. **Proceedings of Fifth IEEE International Symposium on Requirements Engineering**, Toronto, 2001. 249-262.

LARICHEV, O. I. et al. Experiments comparing qualitative approaches to rank ordering of multiattribute alternatives. **Journal of Multi-Criteria Decision Analysis**, 1993. 5-26.

LAZIM, M. A.; ABU OSMAN, M. T. Measuring Teachers' Beliefs about Mathematics: A fuzzy Set. **International Journal of Social Sciences**, 4, n. 1, 2009. 39-43.

LEE, K. H. **First Course on Fuzzy Theory and Applications**. [S.l.]: [s.n.], v. 27, 2005. 335 p. ISBN 978-3-540-22988-9.

LEHTOLA, L.; KUJALA, S. Requirements prioritization challenges in practice. **Proc. of 5th Int´l Conf. On Product Focused Software Process Improvement (PROFES)**, 2004. 497-508.

MAMDANI, E. H.; ASSILIAN, S. An experiment in linguistic synthesis with a fuzzy logic controller. **International Journal of Human-Computer Studies - Special issue: 1969-1999**, Agosto 1999. 135-147.

MEAD, N. M. Requirements Prioritization Case Study Using AHP. **Build Security In**, 2008. Disponivel em: <https://buildsecurityin.us-cert.gov/bsi/articles/best-pratices/requirements/534-BSI.html>. Acesso em: Julho 2011.

PIAGET, J. **Genetic Epistemology**. [S.l.]: W. W. Norton & Company, 1971. ISBN 0393005968.

RUHE, G.; EBERLEIN, A. Quantitative WinWin: a new method for decision support in requirements negotiation. **Proceedings of the 14th international conference on Software engineering and knowledge engineering**, 2002. 159-166.

SAATY, T. L. How To Make A Decision: The Analytic Hierarchy Process. **European Journal of Operational Research**, Setembro 1990. 9-26.

SAATY, T. L. Relative Measurement and Its Generalization in Decision Making Why Pairwise Comparisons are Central in Mathematics for the Measurement of Intangible Factors The Analytic Hierarchy/Network Process. **RACSAM**, 2008. 251–318.

SCHULMEYER, G. G.; MCMANUS, J. I. **Handbook of Software Quality Assurance**. 3ª. ed. [S.l.]: Prentice Hall, 1999.

SCHWABER, K. **Agile Project Management with Scrum**. [S.l.]: Microsoft Press, 2004.

SOMMERVILLE, I. **Engenharia de Software**. 8. ed. São Paulo: Pearson Addison Wesley, 2009. ISBN 9788588639287.

STANDISH GROUP. Standish Newsroom - CHAOS 2009. **The Standish Group**, 2009. Disponivel em: <http://www1.standishgroup.com/newsroom/chaos_2009.php>. Acesso em: Julho 2011.

TONELLA, P.; SUSI, A.; PALMA, F. Using Interactive GA for Requirements Prioritization. **International Symposium on Search Based Software Engineering**, 2010. 57-66.

WILHELM , M. R.; PARSAEI, H. R. A Fuzzy Linguistic Approach to Implementing a Strategy for Computer Integrated Manufacturing. **Fuzzy Sets and Systems**, 42, 1991. 191-204.

XUECHENG, L. Entropy, distance measure and similarity measure of fuzzy sets and their relations. **Fuzzy Sets and Systems**, 24 Dezembro 1992. 305-318.

YEN, J.; LANGARI, R. **Fuzzy Logic:** Intelligence, Control, and Information. 1ª. ed. [S.l.]: [s.n.], 1998.

ZADEH, L. A. Fuzzy Sets. **Information and Control**, 1965. 338-353.

ZHANG_A, Y. Multi-Objective Search-based Requirements Selection and Optimisation. **Tese (Doutorado)**, Londres, Fevereiro 2010.

ZHANG_B, Y. Repository of Publications on Search Based Software Engineering. **SBSE Repository**, 2010. Disponivel em: <http://www.sebase.org/sbse/publications/repository.html>. Acesso em: 9 dez. 2010.