De 11 a 14 - NOVEMBRO DE 2024
Inteligência Artificial na Gestão de Operações: Limitações e possibilidades
XXXI SIMPEP
www.simpep.feb.unesp.br

PROJETO E IMPLEMENTAÇÃO DE UM SOFTWARE BASEADO EM *HESITANT FUZZY LINGUISTIC* TOPSIS PARA APOIAR A TOMADA DE DECISÃO MULTICRITÉRIO

DAHAN POIEL LIMA SCHUSTER - dan.plschuster@gmail.com
UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ – UTFPR – CURITIBA

TAINARA SILVA NOVAES - tainaranovaes@alunos.utfpr.edu.br
UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ – UTFPR – CURITIBA

VIVIANE RUOTOLO - vivianeruotolo@alunos.utfpr.edu.br
UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ – UTFPR – CURITIBA

LUCYANO CAMPOS MARTINS - lucyanocm@gmail.com
UNIVERSIDADE FEDERAL DO TOCANTIS – UFT - PALMAS

FRANCISCO RODRIGUES LIMA JUNIOR - frjunior@utfpr.edu.br
UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ – UTFPR - CURITIBA

ÁREA: *6. ENGENHARIA ORGANIZACIONAL*
SUBÁREA: *6.7 – GESTÃO DA TECNOLOGIA*

RESUMO: *EM AMBIENTES CORPORATIVOS, OS PROCESSOS DE TOMADA DE DECISÕES ADQUIREM UM PAPEL DE EXTREMA RELEVÂNCIA, POIS HÁ DIVERSAS DECISÕES QUE POSSUEM O POTENCIAL TANTO DE IMPULSIONAR QUANTO DE PREJUDICAR A EMPRESA. AINDA QUE EXISTAM DIVERSOS SOFTWARES BASEADOS EM TÉCNICAS MULTICRITÉRIO, A PARTIR DE UM LEVANTAMENTO REALIZADO, VERIFICOU-SE A ESCASSEZ DE SOFTWARES DE APOIO À DECISÃO MULTICRITÉRIO BASEADOS NA TEORIA DOS CONJUNTOS FUZZY E NAS EXTENSÕES MAIS RECENTES DESTA TEORIA. NESSE CONTEXTO, ESTE OBJETIVA DESENVOLVER UM NOVO SOFTWARE DE APOIO À TOMADA DE DECISÃO SOB INCERTEZA. ESSE SOFTWARE AUTOMATIZARÁ A APLICAÇÃO DO MÉTODO HESITANT FUZZY LINGUISTIC TOPSIS, TORNANDO-O ACESSÍVEL MESMO PARA NÃO ESPECIALISTAS EM LÓGICA FUZZY. O PROCESSO DE DESENVOLVIMENTO ENVOLVEU A ANÁLISE E REFINAMENTO DOS REQUISITOS, USANDO TÉCNICAS COMO CONSULTA A ESPECIALISTAS, STORYBOARDS, FLUXOGRAMAS E WIREFRAMES. O DESIGN FINAL INCORPOROU AS MELHORES PRÁTICAS DE USABILIDADE, INCLUINDO AS LEIS DE KRUG, PARA CRIAR UMA INTERFACE INTUITIVA. O PROJETO RESULTOU EM UM SOFTWARE QUE TORNA A RESOLUÇÃO DE PROBLEMAS DE DECISÃO MULTICRITÉRIO MAIS FÁCIL, ACESSÍVEL E EFICAZ, MESMO EM CENÁRIOS COM VÁRIOS DECISORES. O SOFTWARE POSSUI ALTO POTENCIAL DE APLICAÇÃO EM DIVERSOS TIPOS DE PROBLEMAS DE TOMADA DE DECISÃO SOB INCERTEZA.*

PALAVRAS-CHAVES: *TOMADA DE DECISÃO MULTICRITÉRIO; HESITANT FUZZY LINGUISTIC TOPSIS; ENGENHARIA DE SOFTWARE.*

1. INTRODUÇÃO

A tomada de decisões em ambientes empresariais frequentemente exige a consideração de diversos critérios, cada um com seu próprio grau de importância. Nesse contexto, as técnicas de decisão multicritério desempenham um papel vital ao fornecer estruturas robustas para avaliar e priorizar alternativas para a solução de problemas (Ruotolo et al., 2024). Contudo, frequentemente, colocar em prática essas teorias demanda um conhecimento especializado, o que pode representar um desafio para diversos profissionais e decisores. Verifica-se assim a importância de integrar a Engenharia de Software com conhecimentos da área de Pesquisa Operacional a fim de se obter ferramentas computacionais de apoio, que facilitem a aplicação de técnicas de decisão e promovam a racionalidade nas decisões organizacionais (Moraes; Lima, 2017; Ruotolo et al., 2024).

Para apoiar os gestores em processos decisórios complexos, praticantes e acadêmicos desenvolveram softwares de apoio à decisão multicritério, capazes de auxiliar diversas áreas da gestão empresarial. Esses softwares são aplicáveis em problemas que envolvam a avaliação de duas ou mais alternativas, com base em dois ou mais critérios de decisão (Montes et al., 2015). Alguns deles também consideram as opiniões de diversos participantes no processo decisório, oferecendo suporte à tomada de decisão em grupo.

Ainda que existam diversos softwares baseados em técnicas multicritério tradicionais, como M-MACBETH, *Expert Choice* e *Super Decisions*, a partir de um levantamento realizado pelos autores do presente estudo em bases de periódicos e de patentes, verificou-se a escassez de softwares de apoio à decisão multicritério baseados na Teoria dos Conjuntos *Fuzzy* e nas extensões mais recentes desta teoria, como *Hesitant Fuzzy Linguistic Term Sets* - HFLTSs (Hamdan; Cheaitou, 2017; Dymova et al., 2022; Tahri et al., 2022). Os métodos baseados em HFLTSs se mostram efetivos para apoiar problemas de tomada de decisão sob incerteza e hesitação, especialmente em situações em que não há informações completas para tomada de decisão e as avaliações das alternativas e critérios são baseadas em julgamentos de especialistas envolvidos no problema (Rodríguez; Martínez; Herrera, 2013; Lima; Oliveira; Resende, 2023).

Dentre os métodos multicritério existentes, o HFL-TOPSIS (*Technique for Order of Preference by Similarity to Ideal Solution*), proposto por Beg e Rashid (2013), se destaca por ser uma abordagem para tomada de decisão em grupo, que permite aos decisores o uso de termos linguísticos (ex.: baixo, médio, alto) e de expressões linguísticas (ex. “entre baixo e médio”) para expressar suas preferências. Essa abordagem de computação com palavras é

adequada para apoiar decisões sob incerteza e hesitação, caracterizadas pela escassez de informações e pelo uso de julgamentos subjetivos de especialistas (Lima; Oliveira; Resende, 2023). Contudo, atualmente o uso do HFL-TOPSIS é dificultado pela ausência de um software com uma interface gráfica que possibilite o uso por usuários não especialistas em lógica *fuzzy*.

A ausência de softwares baseados em HFL-TOPSIS contribui para que as aplicações sejam realizadas por meio de planilhas e códigos de programação, que são difíceis de serem replicados por gestores. O uso de técnicas HFLTSs demanda um nível de *expertise* considerável para manipular as planilhas de forma eficaz, tornando-se uma barreira significativa para aqueles que não possuem conhecimento aprofundado nessa área. Além disso, a ausência de ferramentas que ofereçam uma interface gráfica amigável limita a acessibilidade, dificulta a edição dos modelos e a análise de sensibilidade do modelo.

Portanto, este trabalho tem como objetivo desenvolver um software que simplifique significativamente a aplicação do método HFL-TOPSIS para processos decisórios em grupo, tornando-o acessível e intuitivo para usuários que não possuem *expertise* em lógica *fuzzy*. Assim, o presente estudo apresenta os resultados do processo de definição de requisitos e da interface do software, assim como da implementação e testes do sistema. A seção a seguir detalha os procedimentos utilizados neste estudo.

2. MÉTODO

Neste estudo foram utilizados princípios de desenvolvimento ágil de *software*, com o uso da metodologia *Scrum* (Sbrocco; Macedo, 2012), para permitir uma abordagem iterativa e incremental. Adotou-se *Sprints* de duas a três semanas, com reuniões de revisão e planejamento, garantindo a entrega de funcionalidades em cada iteração. A equipe do projeto foi formada por um Professor Coordenador, da área de Engenharia de Produção; um Professor Colaborador, da área de Sistemas de Informação, responsável pela orientação em termos de desenvolvimento de software; e três Graduandas em Engenharia de Computação, responsáveis por diversas atividades, desde a concepção do software até a implementação computacional. As etapas do projeto foram estruturadas como segue:

a) Estudo da literatura existente – baseou-se no material bibliográfico extraído a partir de consultas ao INPI (Instituto Nacional da Propriedade Industrial) e a bases de periódicos. Foram consultados livros e artigos sobre Engenharia de Software, Qualidade de Software, Tomada de Decisão Multicritério, HFLTS e HFL-TOPSIS, utilizando as bases de dados *Scopus*, *Scielo*, *Web of Science* e *Emerald* Insight. Nesta etapa, também foi realizado um

levantamento de softwares com interface gráfica, para tomada de decisão multicritério, baseados em técnicas *fuzzy*. Como resultado, foram identificados e estudados os sistemas apresentados em Estrella et al. (2014), Montes et al. (2015), Hamdan e Cheaitou (2017) e Dymova et al. (2022), o que subsidiou o levantamento de requisitos funcionais e não funcionais que deveriam ser supridos pelo novo software a ser desenvolvido;

b) Definição e refinamento dos requisitos – juntamente com os resultados da etapa "a", a definição das características desejadas para o produto foi feita com base na consulta a três especialistas, usando *Google Meet* e *Google Forms*. Esses especialistas são pesquisadores da área de Pesquisa Operacional e possuem entre 3 e 10 anos de experiência no uso de HFLTSs. Eles foram consultados a respeito dos requisitos relevantes para esse tipo de software. Também atribuíram um grau de importância para cada requisito, que poderia ser "muito baixo", "baixo", "médio", "alto" e "muito alto". Essas avaliações foram computadas utilizando o método *Fuzzy* TOPSIS (Ruotolo et al., 2024) a fim de criar um *ranking* que indicasse a prioridade de cada requisito, para assim incluir apenas os requisitos prioritários na primeira versão do software. O *Fuzzy* TOPSIS foi escolhido por ser um dos métodos multicritério mais utilizados na literatura sobre priorização de requisitos de software e por permitir o uso de julgamentos linguísticos. Na sequência, empregou-se uma técnica *storyboard* para o refinamento dos requisitos previamente definidos (Sbrocco; Macedo, 2012), o que possibilitou a representação descritiva de todo o cenário e dos casos de uso;

c) Seleção de ferramentas - a escolha das tecnologias usadas para desenvolvimento do sistema levou em conta o conhecimento prévio da equipe e a capacidade das ferramentas de lidar com as particularidades da implementação, como: precisão dos cálculos, criação de interfaces adaptáveis a diferentes telas e facilidade de manutenção. Com isto em mente, foi escolhida a linguagem *Python*, para calcular o resultado do modelo a partir dos dados inseridos, e *TypeScript* em conjunto com a biblioteca *React* para criação da interface de usuário. Como ferramenta de design, utilizou-se a plataforma Figma, por esta ter todas as funcionalidades necessárias para prototipação de interfaces e ser gratuita;

d) Projeto da interface ao usuário - utilizando a ferramenta *Miro*, elaborou-se um fluxograma para traçar a jornada do usuário durante o uso do software, abrangendo desde o momento inicial de entrada no sistema até a sua saída. Esse fluxograma representa uma síntese do que foi delineado nos *storyboards*, oferecendo uma visão holística do fluxo de trabalho do usuário. Com base no fluxograma de jornada do usuário, realizou-se a criação da Interface de Usuário (*User Interface - UI*) e da Experiência do Usuário (*User Experience UX*) do sistema. Seguindo as recomendações apresentadas por Atoum (2023), adotou-se diretrizes

de usabilidade, as quais constituem um conjunto de princípios que auxiliam na avaliação da qualidade da interface do usuário em termos de usabilidade. A primeira versão do design do software foi desenvolvida por meio da técnica de *wireframe*, que consiste em um protótipo de baixo detalhamento focado nas principais funcionalidades do sistema e que fornece uma concepção inicial de ideia de software. Seguido à criação das telas em formato de *wireframes*, o passo subsequente consistiu no desenvolvimento dos elementos visuais exclusivos para o software, empregando técnicas de design de UI/UX usando a ferramenta Figma;

e) Projeto e implementação do sistema - a partir de um exemplo de uso do HFL-TOPSIS em *MS Excel*, foram identificadas as relações da aplicação, as quais permitiram a criação do diagrama de classes que orientou a implementação do sistema. Com relação à precisão numérica, foram implementados dois protótipos com base no exemplo de uso do HFL-TOPSIS: (1) Cálculo direto do método na linguagem de programação *web* PHP; e (2) Implantação orientada a objetos do modelo de decisão e seus respectivos cálculos em *Python*. Esses protótipos foram essenciais para traduzir o método HFL-TOPSIS em linguagem computacional. Por conseguinte, a comparação entre os dois protótipos serviu de embasamento para estabelecer o *Django*, um *framework* de programação *web* em *Python*, no *backend*. No *frontend*, utilizou-se do *TypeScript* e do *React*, uma biblioteca da linguagem *JavaScript* para páginas da *web*. Na implementação computacional, houve uma transição do protótipo em *Python* para a estrutura do *framework Django* a fim de uni-la com o *frontend* em *React,* o qual produziu o *design* idealizado no *Figma*. Ao longo das entregas, foram implementadas diversas funcionalidades, como a criação de modelos, a definição do problema, inserção dos julgamentos para os pesos dos critérios e para cada alternativa aos critérios, o ranqueamento dos resultados e a possibilidade de exportar e importar modelos em arquivos compartilhados;

f) Testes – após a conclusão da implementação, foram realizados diversos testes de uso do software para atestar a consistência dos resultados fornecidos. Nesses testes, foram utilizados os dados extraídos dos casos de aplicação apresentados em Beg e Rashid (2013) e Magalhães et al. (2022), que também utilizaram *Hesitant Fuzzy Linguistic* TOPSIS em problemas de decisão multicritério. Também foram criados dois casos simulados para verificar os resultados produzidos pelo modelo em situações em que todas as alternativas receberem pontuações máximas ou pontuações mínimas em todos os critérios. Na seção a seguir são apresentados os principais resultados.

3. O PROCESSO DE DESENVOLVIMENTO DE SOFTWARE

O desenvolvimento de novos produtos de software é um processo complexo, geralmente dividido em várias etapas. Essas etapas incluem diagnóstico, concepção, levantamento e análise de requisitos, desenvolvimento e manutenção (Pressman; Maxim, 2016). É crucial dedicar esforços às etapas iniciais para otimizar o uso de recursos e garantir que o produto final atenda às expectativas de qualidade dos clientes (Pacheco; García; Reyes, 2018; Ruotolo et al., 2024).

Há diversas abordagens metodológicas que apoiam o desenvolvimento de software. No Modelo Cascata, as fases do desenvolvimento de software são sequenciais, lineares e há dependências entre as etapas. O Modelo Espiral é inspirado em elementos do Cascata, mas inclui interações como mecanismo propulsor da adaptação contínua. Já o modelo *Rational Unified Process* (RUP) também é uma abordagem iterativa, mas baseada em componentes, que se baseia na colaboração entre desenvolvedores para produção de artefatos de qualidade. Há também metodologias ágeis como o *Scrum*, que divide o projeto em *sprints* (ciclos de trabalho), com entregas bem definidas, realizadas de forma incremental e com revisões contínuas (Pressman; Maxim, 2016; Sbrocco; Macedo, 2012).

No contexto do desenvolvimento de software, é crucial compreender os pilares que garantem o sucesso e o desempenho adequado de um sistema. Segundo Atoum (2023), a elicitação e a análise de requisitos desempenham um papel fundamental em assegurar que o software atenda às expectativas dos usuários em relação à experiência do mesmo. Além disso, a autora destaca que o tempo investido nessa etapa é de suma importância, pois aumenta a chance de alcançar resultados eficazes que possam satisfazer plenamente os futuros usuários. Portanto, uma base sólida em relação aos requisitos estabelecidos e a sua devida priorização desempenharam um papel relevante no avanço da implementação do *software*.

Em relação aos aspectos que orientam a definição dos requisitos de software, a área de Qualidade de Software apresenta orientações e modelos que tornam o processo de desenvolvimento mais efetivo. A qualidade de software é uma vertente da engenharia de software que, segundo Pressman e Maxim (2016), consiste na conformidade dos requisitos funcionais e do desempenho de um software a padrões de desenvolvimento documentados e características implícitas previstas em um bom produto de software.

Existem modelos de qualidade que permitem a avaliação de softwares usando critérios predefinidos. Esses modelos podem ser úteis no desenvolvimento de um novo software ou na avaliação da qualidade de produtos existentes (Moraes; Lima Junior, 2017). A série de normas

ISO/IEC 25000 - SQuaRE (*System and Software Quality Requirements and Evaluation*), é a mais relevante em termos de avaliação da qualidade de softwares (ISO, 2014). Ela sugere a definição de requisitos que levem em conta a adequação funcional, eficiência de desempenho, usabilidade, compatibilidade, confiança, segurança, manutenibilidade e portabilidade.

Conforme Atoum (2023) ainda ressalta, a qualidade dos requisitos que foram priorizados e validados desempenha um papel direto e significativo na construção da experiência do usuário. Quanto mais claras forem essas necessidades, maiores são as chances dos profissionais *designers* criarem uma experiência de usuário aprimorada. Além disso, ao simplificar os processos subjacentes e oferecer uma interface intuitiva, o software remove a barreira do conhecimento técnico especializado.

Há várias técnicas usadas para apoiar a elicitação de requisitos no desenvolvimento de software. Em um estudo de revisão sistemática sobre técnicas de elicitação de requisitos, Pacheco; García; Reyes (2018) constataram que as mais utilizadas têm sido entrevistas, prototipagem, análise de cenários, grupo focal e *brainstorming*. Como geralmente há muitos requisitos a considerar e também podem ocorrer mudanças nestes requisitos, é essencial adotar um método claro para priorização dos requisitos identificados, de modo a selecionar aqueles que refletem de fato as demandas do usuário (Ruotolo et al., 2024).

Uma vez que os requisitos de software escolhidos são determinantes na estrutura e no sucesso do produto, a priorização de requisitos deve ser feita de forma estruturada, podendo ser abordada como um problema de tomada de decisão (Pacheco, García & Reyes, 2018). Há diversos modelos decisórios que foram propostos na literatura para dar suporte a esse problema de priorização. No trabalho de revisão sistemática desenvolvido Achimugu et al. (2014), foram identificados 73 estudos que aplicaram métodos para priorização de requisitos de software, com os métodos AHP e QFD (*Quality Function Deployment*) estando entre os mais utilizados. Um estudo similar foi feito por Bukhsh et al. (2020), o qual identificou 102 trabalhos que desenvolveram ou utilizaram abordagens metodológicas para priorização de requisitos. Nesse caso, os métodos mais frequentemente adotados foram AHP e as técnicas baseadas na Teoria dos Conjuntos *Fuzzy*.

Dentre as técnicas *fuzzy*, o método *Fuzzy* TOPSIS se destaca pela facilidade de aplicação e por permitir o uso de uma quantidade ilimitada de alternativas e critérios. Além disso, as escalas utilizadas pelos decisores para avaliação dos elementos do problema são de fácil entendimento e menos complexas do que as escalas dos métodos AHP e *Fuzzy* AHP (Lima Junior; Carpinetti, 2015). Devido a essas vantagens de uso, o método *Fuzzy* TOPSIS foi escolhido para ser aplicado neste estudo.

4. RESULTADOS E DISCUSSÃO

4.1 Definição, priorização e refinamento dos requisitos

O levantamento de softwares prévios resultou na identificação de quatro softwares com interface gráfica que possibilitam o uso de técnicas *fuzzy* para decisão multicritério, propostos por Estrella et al. (2014), Montes et al. (2015), Hamdan e Cheaitou (2017) e Dymova et al. (2022). A análise da interface e da documentação desses softwares, em conjunto com a consulta a três especialistas em HFLTSs (nomeados E_1, E_2 e E_3), propiciaram a criação de uma listagem dos principais requisitos, os quais serviram como base para a implementação do novo software para decisão multicritério. O Quadro 1 apresenta uma lista dos 16 requisitos avaliados pelos especialistas. Esses requisitos estão relacionados a atributos de qualidade sugeridos pela norma de qualidade de software ISO/IEC 25000 (ISO, 2014). Cada especialista estimou os pesos dos requisitos de acordo com sua experiência e com as necessidades do projeto em questão. O Quadro 1 também mostra os pesos atribuídos aos decisores, o que foi feito considerando a experiência de cada um na área de HFLTS e o grau de responsabilidade sobre o projeto em questão.

QUADRO 1 – Avaliação dos requisitos para o projeto do software em questão

Características	**Requisitos**	**E_1**	**E_2**	**E_3**
Completude funcional	R_1. Permitir a atribuição de pesos aos critérios	Muito Alto (MA)	Alto	MA
	R_2. Uso de critérios qualitativos	MA	MA	MA
	R_3. Uso de critérios quantitativos	MA	MA	MA
	R_4. Avaliar critérios por valores numéricos	Alto	Baixo	Médio
	R_5. Avaliar pesos por julgamentos linguísticos	Alto	Alto	Alto
Operacionalidade	R_6. Escala com 5 ou 7 termos linguísticos	Alto	Alto	Alto
Acessibilidade	R_7. Ser auto explicativo em relação ao seu uso	Alto	MA	MA
	R_8. Fornecer suporte ao cadastro de usuários	Alto	Médio	Médio
Confidencialidade	R_9. O administrador poderá gerenciar usuários	MA	Alto	Alto
Instalabilidade	R_{10}. Funcionar em plataforma WEB	MA	MA	Baixo
Recuperabilidade	R_{11}. Salvar os dados em caso de falhas	Alto	Alto	Baixo
Modificabilidade	R_{12}. O código deve conter comentários que expliquem as etapas implementadas	Alto	MA	Alto
Apreensibilidade	R_{13}. Figura com visão geral das etapas de uso	Alto	MA	Alto
	R_{14}. Figura com as teorias matemáticas usadas	Alto	Médio	Médio
	R_{15}. Centralizar os elementos na mesma página	Médio	MA	Alto
Autenticidade	R_{16}. Salvar os dados de entrada e os resultados	Médio	Alto	Alto
Pesos atribuídos aos decisores:		MA	Alto	Alto

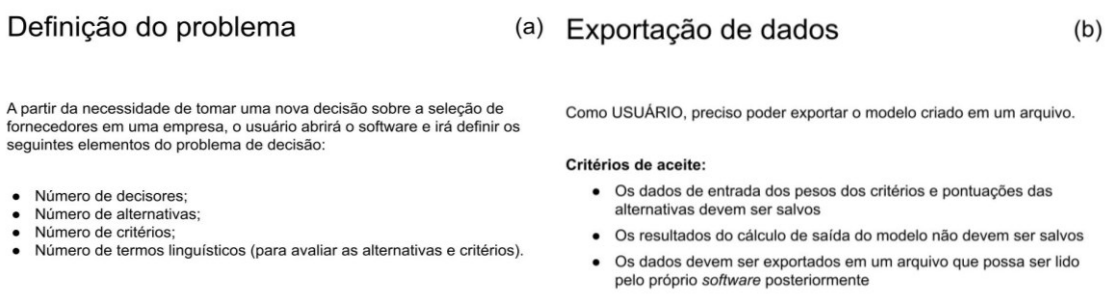
Fonte: Elaborado pelos autores (2024).

Os dados mostrados no Quadro 1 foram utilizados para entrada no método *Fuzzy* TOPSIS para gerar um ranqueamento que estabeleceu uma ordem de prioridade entre os requisitos elencados. Assim, a ordem de prioridade obtida foi: $R_2 = R_3 > R_1 > R_7 > R_9 > R_{12} = R_{13} > R_5 = R_6 > R_{10} = R_{15} > R_{16} > R_8 = R_{14} > R_{11} > R_4$. Assim, a aplicação do *Fuzzy* TOPSIS indicou que os requisitos prioritários dizem respeito a funcionalidades cruciais do software, como possibilitar a adoção de critérios qualitativos (R_2) e critérios qualitativos (R_3), além de permitir a consideração dos pesos dos critérios (R_1). O requisito R_7 também se destacou, evidenciando a importância de o sistema ser intuitivo.

Por outro lado, os requisitos com menor prioridade foram R_4 e R_{11}, que se referem à tolerância a falhas e uso de valores numéricos como entradas. Os resultados foram apresentados aos especialistas, que endossaram a ordem de prioridade. Portanto, a equipe desenvolvedora optou por incluir na primeira versão do software os requisitos classificados entre a 1ª e a 8ª posição (R_2 a R_{16}). Portanto, os requisitos R_8, R_{14}, R_{11} e R_4 não foram incluídos nessa primeira versão.

Utilizando os requisitos prioritários como ponto de partida, foi construído um *storyboard* contendo uma sequência de passos que o usuário deve poder realizar no sistema, destacando as funcionalidades e as regras de negócio. Isso foi feito pela equipe de desenvolvedores juntamente com o especialista E_1. O *storyboard* tornou claro o fluxo a ser seguido pelo sistema em diferentes situações, como a criação de um novo modelo de decisão, a edição de um modelo existente e a importação ou exportação de modelos. As Figuras 1(a) e 1(b) apresentam uma pequena parte do *storyboard* desenvolvido.

FIGURA 1 - *Storyboard* para (a) definição do problema e (b) exportação de dados



Fonte: Elaborado pelos autores (2024).

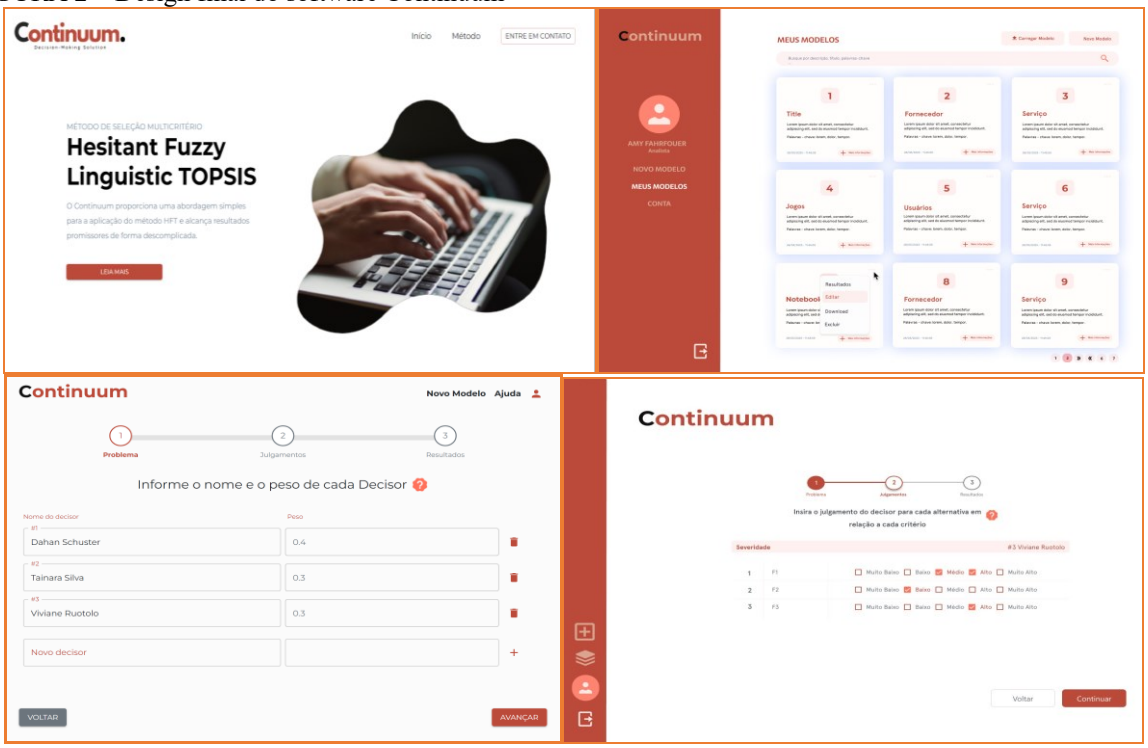
4.2 Projeto da interface ao usuário

O projeto da interface foi iniciado por meio do mapeamento da jornada do usuário, o qual foi baseado no *storyboard* desenvolvido anteriormente. Esse mapeamento descreve a

sequência de caminhos que o usuário pode percorrer na criação do modelo de aplicação, destacando o fluxo de criação de contas, o acesso à página inicial, a criação de modelos, a atualização de modelos e a obtenção de resultados.

Posteriormente, na etapa de design de interface, tornou-se evidente que antes de prosseguir para o design definitivo, seria benéfico criar um esboço inicial por meio de *wireframes* das telas principais do sistema. Isso foi feito com o propósito de determinar a disposição mais adequada dos elementos essenciais para a construção de modelos decisórios baseados em HFL-TOPSIS pelos futuros usuários. O design final foi desenvolvido com base nos esboços previamente criados, o que possibilitou um leiaute mais preciso e acessível. A representação final da interface gráfica do software pode ser observada na Figura 2, que exibe algumas telas do sistema.

FIGURA 2 – Design final do software Continuum



Fonte: Elaborado pelos autores (2024).

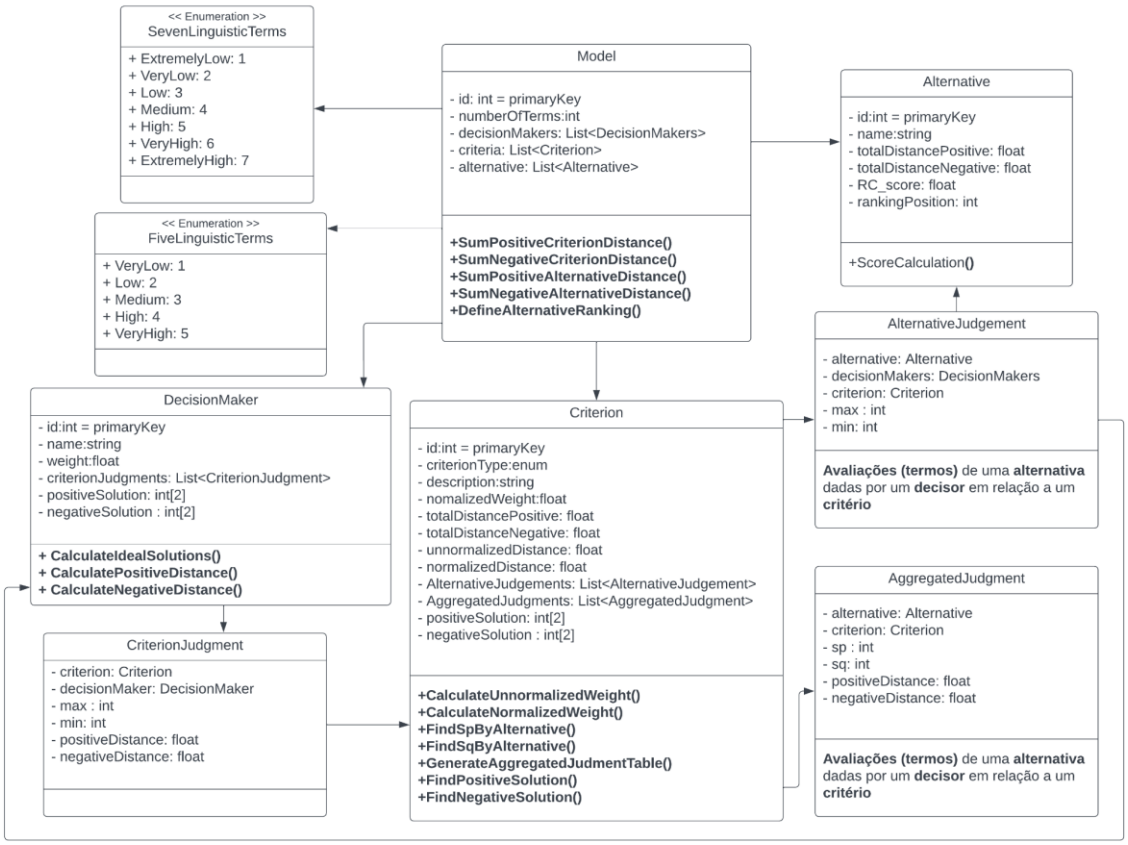
Dado que o objetivo principal era tornar mais amigável a aplicação do método HFL-TOPSIS, a adoção das melhores práticas de usabilidade desempenhou um papel fundamental. Algumas dessas práticas foram baseadas nas diretrizes amplamente reconhecidas conhecidas como Leis de Krug (2008). Essas diretrizes partem do princípio “Não me faça pensar”, fazendo referência à necessidade de os componentes da interface do software serem autoexplicativos e fáceis de usar, além de evitarem a perda de tempo do usuário e garantirem uma navegação ágil entre as funcionalidades do sistema. Com base nisso, todos os aspectos relacionados à usabilidade foram minuciosamente examinados durante o processo de criação,

abrangendo desde o tamanho das fontes, a seleção de cores, o posicionamento de elementos, até a estrutura de navegação e todos os elementos do *front end* de um site.

4.3 Projeto e Implementação do Sistema

Em relação à etapa de modelagem do software, a Figura 3 apresenta o diagrama de classes desenvolvido para gerenciar a criação de modelos decisórios baseados em HFL-TOPSIS. A classe *Model* representa o modelo com os seus decisores, critérios e alternativas, os quais compõem, respectivamente, as classes: *DecisionMaker, Criterion* e *Alternative.* Esses objetos são responsáveis por gerar o modelo, o qual se baseia em cálculos com os julgamentos dos decisores (*CriterionJudgment* e *AlternativeJudgment*). *SevenLinguisticTerms* e *FiveLinguisticTerms* são as escalas de julgamento disponíveis.

FIGURA 3 - Diagrama de classes desenvolvido para o software



Fonte: Elaborada pelos autores (2024).

Posteriormente, foram implementados dois protótipos para execução dos cálculos do HFL-TOPSIS, o que permitiu identificar a diferença de execução entre eles. Como é evidenciado na Figura 4, é notável que em *Python* obteve-se uma maior precisão numérica, apesar dos resultados serem bem semelhantes e com uma diferença de duas ou três casas

decimais entre os resultados. Por esse motivo, optou-se pelo *Python* e o *framework Django* para a execução matematicamente mais precisa do HFL-TOPSIS.

FIGURA 4 - Comparação entre os protótipos em *Python* e em *PHP* para o cálculo do peso dos critérios

CRITERION WEIGHT (PHP):

CRITERION	NORMALIZED
Custo	0.48031486431208
Qualidade	0.38884018940448
Ocorrencia	0.13084494628344

CRITERION WEIGHT (PYTHON):

CRITERION	NORMALIZED
Custo	0.4803148643120842
Qualidade	0.3888401894044807
Ocorrencia	0.13084494628343515

Fonte: Elaborado pelos autores (2024).

A primeira versão do *software* incluiu as funcionalidades essenciais do projeto. Muitas funções são executadas com o *React*. O *backend* é requisitado, apenas, na criação ou alteração do modelo, em que há a necessidade de refazer os cálculos. A etapa de criação do modelo decisório possui um total de quatro passos, tais quais: (1) descrição do modelo, (2) definição do problema, (3) inserção dos julgamentos dos decisores, e (4) ranqueamento dos resultados. Ao finalizar o terceiro passo, os dados inseridos são formatados no modelo *JavaScript Object Notation* (JSON) e enviados para a *Application Programming Interface* (API) em *Django*. A Figura 5 exemplifica um desses arquivos JSON. Nesse exemplo, o modelo é composto por um decisor, um critério, uma alternativa e cinco termos linguísticos a fim de demonstrar a estrutura de armazenamento dos dados. Nesse sentido, depois do *backend* executar a sua função, é retornado ao *frontend* um JSON com a mesma estrutura. Contudo, são adicionados os dados finais que permitem o ranqueamento no último passo.

FIGURA 5 - Exemplo de JSON utilizado no Continuum


```
{
  "id": "1dd6d183-fdf7-42e2-b645-082bc9df23ae",
  "info": {
    "description": "Modelinho basico",
    "keywords": ["fornecedores"],
    "name": "Modelinho",
    "searchString": "modelinho modelinho basico fornecedores"
  },
  "decisionMakers": {
      "id": "d755148e-644c-49b6-a668-1b20ee911554",
      "name": "Decisor1",
      "weight": 0.4,
      "criterion_judgments": {
          "criterion_id": "0a211b60-ee0c-46df-af7f-0cf06e32a165",
          "sp": 3,
          "sq": 4
      },
      "positive_ideal_solution": [0, 0],
      "negative_ideal_solution": [0, 0]
  },
  "criteria": {
      "id": "ddea19be-89a7-4186-b098-f0a94107f86b",
      "description": "Qualidade",
      "criterion_type": "benefit",
      "alternative_judgments": {
          "alternative_id": "a832ed90-b798-49f3-a063-94135fdae74a",
          "decision_maker_id": "d755148e-644c-49b6-a668-1b20ee911554",
          "min_value": 4,
          "max_value": 4
      }
  },
  "alternatives": {
      "id": "a832ed90-b798-49f3-a063-94135fdae74a",
      "name": "Fornecedor A",
      "ranking_position": 0,
      "score": 0,
      "total_distance_negative": 0,
      "total_distance_positive": 0
  },
  "linguisticScale": 5
}
```


Fonte: Elaborada pelos autores (2024).

Na etapa de implementação, também foram criadas rotas no sistema (caminhos no endereço WEB) para cada subetapa do modelo. A rota "*/new-model*" permite a inserção dos dados iniciais: nome, descrição e palavras-chave. Foi estabelecida a obrigatoriedade de preenchimento desses dados antes de avançar para o próximo passo. A rota "*/new-model/dms*"

possibilita a adição dos decisores do modelo, incluindo seus nomes e pesos. A soma total dos pesos é validada para garantir que seja igual a 1 antes de poder avançar.

Na rota "*/new-model/criteria*", é possível inserir os múltiplos critérios de escolha, juntamente com seus tipos (Benefício ou Custo). Ao menos um critério deve ser inserido para poder seguir para o próximo passo. A rota "*/new-model/alternatives*" possibilita a adição das alternativas a serem julgadas com base nos critérios inseridos. Também há a obrigatoriedade de ao menos uma alternativa para poder prosseguir. Já a rota "*/new-model/linguistic-scale*" permite a escolha da escala linguística do modelo, com opções de escala de 5 termos linguísticos (de "Muito baixo" a "Muito alto") ou 7 termos (de "Extremamente baixo" a "Extremamente alto"). Na rota "*/new-model/dm/judge-criteria-weight*", cada decisor insere seus julgamentos sobre o peso de cada critério. A etapa de julgamento das alternativas foi incluída na rota "*/new-model/dm/judge-alternatives*", e a rota "*/new-model/results*" apresenta as alternativas ranqueadas com base nos cálculos realizados pela API.

Uma tela secundária para listar os modelos do usuário também foi implementada, utilizando o recurso de salvamento local dos navegadores WEB, que permite persistir os dados da aplicação para recuperação futura, sem a necessidade de um banco de dados nesta primeira implementação. Nesta tela também foram adicionados botões para baixar os modelos criados e para importar um modelo baixado. Os arquivos utilizam o padrão JSON (*Javascript Object Notation*) para estruturação de dados.

A implementação computacional do software foi finalizada em Julho de 2024, com duração total do projeto de 1 ano e 6 meses. O último estágio consistirá na publicação do software na Internet e envolverá a criação da infraestrutura necessária para hospedar ambos os códigos da API escrita em Python e da interface feita em React, para que possam ser acessíveis *online*, o que será feito utilizando um servidor WEB configurado com uma ferramenta de hospedagem como Apache ou Nginx.

4.4 Testes

A Tabela 1 apresenta os resultados gerados pelo software em quatros casos de aplicação. Foram realizados diversos testes de uso do software para atestar a consistência dos resultados fornecidos. O caso 1 corresponde aos dados extraídos de Beg e Rashid (2013), com múltiplos decisores, mas que não aplica pesos aos critérios. O caso 2 se refere aos dados apresentados em Magalhães et al. (2022), que consistem em uma aplicação para avaliação de falhas em um processo de fabricação. Essa aplicação considera múltiplos decisores e critérios com pesos distintos. O caso 3 considera um decisor, com atribuição do valor "muito alto"

(maior valor da escala linguística) para todas as alternativas, em todos os critérios. O caso 4 é similar ao caso 3, mas com atribuição do valor "muito baixo" (menor valor da escala) para todas as alternativas, em todos os critérios.

TABELA 1 – Síntese dos resultados dos testes do software

Posição	**Caso 1**		**Beg e Rashid**	**Caso 2**		**Magalhães et al.**	**Caso 3**		**Caso 4**	
	A_i	$RC(A_i)$		A_i	$RC(A_i)$		A_i	$RC(A_i)$	A_i	$RC(A_i)$
1º	A_3	0,725	0,725	A_4	0,674	0,676	A_1=A_2=A_3	0	A_1=A_2=A_3	1
2º	A_1	0,60	0,60	A_{17}	0,655	0,657	-	-	-	-
3º	A_2	0,525	0,525	A_9	0,653	0,657	-	-	-	-
4º	A_4	0,25	0,25	A_2	0,611	0,615	-	-	-	-
5º	A_5	0,17	0,17	A_8	0,609	0,612	-	-	-	-

Fonte: Elaborado pelos autores

A ordenação produzida pelo Continuum nos quatro casos correspondeu à sequência esperada em cada situação. Embora as posições das alternativas sejam as mesmas, houve pequenas variações nos valores de RC(Ai) no caso 2, o que pode ser atribuído ao fato de que, ao contrário dos estudos de Magalhães et al. (2022), o algoritmo implementado no Continuum utiliza o método de distância Hamming ponderada. Ainda assim, os *rankings* das alternativas foram idênticos aos obtidos por Beg e Rashid (2013) e Magalhães et al. (2022). Nos casos 3 e 4, as alternativas empataram e receberam pontuação global coerente com as pontuações de entrada, como esperado. Isso indica a corretude da implementação computacional e a capacidade do software de fornecer resultados consistentes.

5. CONCLUSÃO

Este trabalho representa um esforço significativo para simplificar e democratizar a aplicação do método HFL-TOPSIS em processos decisórios organizacionais. O software desenvolvido torna o uso desse método acessível a profissionais e tomadores de decisão que não possuem conhecimento técnico especializado na área. O processo de desenvolvimento envolveu a análise detalhada dos requisitos, utilizando várias técnicas, desde *storyboards,* diagramas até *wireframes*, o que garantiu uma compreensão precisa das necessidades dos usuários. Além disso, a incorporação das melhores práticas de usabilidade, desempenhou um papel fundamental na criação de uma interface que dispensa suporte especializado. Isso resultou em uma interface gráfica que simplifica a aplicação de um método de decisão multicritério, permitindo que os usuários enfrentem problemas complexos com maior eficácia.

Considerando os resultados da busca de softwares prévios realizada na etapa inicial deste estudo, pode-se afirmar que esse trabalho apresentou o primeiro software brasileiro para tomada de decisão baseado em HFLTSs com interface gráfica amigável para usuários não

especialistas. Há a intenção de disponibilizá-lo para uso de forma gratuita a docentes, pesquisadores e discentes de instituições públicas de ensino superior. Espera-se que isso possa auxiliar profissionais no suporte a processos decisórios, principalmente em contextos em que houver pouca informação para tomada de decisão. O software possui alto potencial de aplicação em diversos problemas de tomada de decisão sob incerteza, incluindo seleção de fornecedores, avaliação de riscos, priorização de projetos, seleção de funcionários, escolha de planos de *marketing*, entre outros problemas de seleção e ordenação de alternativas.

Uma limitação do presente estudo é que ainda não foi possível testar o uso do software de forma continuada. Assim que o sistema possuir usuários ativos e usos frequentes, será possível identificar os impactos do sistema e necessidades adicionais dos usuários. Estudos futuros podem aplicar o Continuum em problemas de tomada de decisão multicritério e comparar os resultados com aqueles fornecidos por outras ferramentas. Também podem desenvolver novos softwares de apoio à tomada de decisão multicritério, baseados em técnicas *fuzzy*, voltados para problemas de categorização de alternativas.

REFERÊNCIAS

ACHIMUGU, P.; SELAMAT, A.; IBRAHIM R.; MAHRIN, M.N. **A systematic literature review of software requirements prioritization research**. Information and Software Technology, v. 56, p. 568-585, 2014.

ATOUM, I. **Measurement of key performance indicators of user experience based on software requirements**. Science Of Computer Programming, v. 226, 102929, 2023.

BEG, I.; RASHID, T. **TOPSIS for hesitant fuzzy linguistic term sets**. International Journal of Intelligent Systems, v. 28, n. 12, 1162-1171, 2013.

DYMOVA, L.; KACZMAREK, K., SEVASTJANOV, P.; KULAWIK, J. **A Fuzzy Multiple Criteria Decision Making Approach with a Complete User Friendly Computer Implementation**. Entropy, v. 23, n. 2, p. 1–28, 2022.

ESTRELLA, F.; RODRÍGUEZ, R.; MARTINEZ, L. **A Hesitant Linguistic Fuzzy TOPSIS Approach Integrated into FLINTSTONES**. Atlantis Press. p. 799-806, 2015.

HAMDAN, S.; CHEAITOU, A. **Supplier selection and order allocation with green criteria: An MCDM and multi-objective optimization approach**. Computers & Operations Research, v. 81, p. 282–304, 2017.

ISO - International Organization for Standardization (2014). **ISO/IEC 25000: Software engineering - System and software Quality Requirements and Evaluation (SQuaRE)**. Acesso: Agosto de 2022. Disponível em: https://www.iso.org/standard/64764.html

KRUG, S. **Não Me Faça Pensar**: Uma Abordagem de Bom Senso à Usabilidade na Web. Alta Books, 2008.

LIMA JUNIOR, F. R.; OLIVEIRA, M. E. B.; RESENDE, C. H. L. **An Overview of Applications of Hesitant Fuzzy Linguistic Term Sets in Supply Chain Management**: The State of the Art and Future Directions. Mathematics, v. 11, p. 2814, 2023.

LIMA, F. R.; CARPINETTI, L. C. R. **Uma comparação entre os métodos TOPSIS e Fuzzy-TOPSIS no apoio à tomada de decisão multicritério para seleção de fornecedores**. Gestão & Produção, v. 22, n. 1, 17–34, 2015.

MAGALHAES, W.; LIMA JUNIOR, F. R.; GALO, N.; OKOSHI, C. Y.; PEINADO, J. **Proposição de um modelo para priorização de riscos baseado em FMEA e *Hesitant Fuzzy*-TOPSIS**. In: XLVI Encontro da ANPAD, 2022.

MONTES, R.; SÁNCHEZ, A. M.; VILLAR, P.; HERRERA, F. **A web tool to support decision making in the housing market using hesitant fuzzy linguistic term sets**. Applied Soft Computing, v. 35, p. 949-957, 2015.

MORAES, M.; LIMA, F. R. **Proposição e Aplicação de uma Metodologia baseada no AHP e na ISO/IEC 25000 para apoiar a Avaliação da Qualidade de Softwares de Gestão de Projetos**. Gestão Da Produção, Operações e Sistemas, v. 12, n. 2, p. 239-260, 2017.

PACHECO, C.; GARCÍA, I.; REYES, M. **Requirements elicitation techniques: a systematic literature review based on the maturity of the techniques**. IET Software, v. 12, n. 4, p. 365-378, 2022.

PRESSMAN, R. S.; MAXIM, B. R. **Engenharia de software**: uma abordagem profissional. 8 ed. Porto Alegre: AMGH, 2016.

RODRÍGUEZ, R. M.; MARTÍNEZ, L.; HERRERA, F. **A group decision making model dealing with comparative linguistic expressions based on hesitant fuzzy linguistic term sets**. Information Sciences, n. 241, p. 28-42, 2013.

RUOTOLO, V.; SCHUSTER, D.P.; NOVAES, T.S.; MARTINS, L.C.; LIMA JUNIOR, F.R. (2024). **Priorização de requisitos para um novo software de decisão multicritério: uma aplicação baseada em *Fuzzy* TOPSIS**. Exacta (no prelo), 2024.

SBROCCO, J.H.T.C.; MACEDO, P.C. **Metodologias Ágeis**: engenharia de software sob medida. 1 ed. Editora Érica: São Paulo, 2012.

TAHRI, M.; MAANAN, M.; TAHRI, H.; KAŠPAR, J.; PURWESTRI, R.; MOHAMMADI, Z.; MARUŠÁK, R. **New Fuzzy-AHP Matlab based graphical user interface (GUI) for a broad range of users: Sample applications in the environmental field**. Computers & Geosciences, v. 158, 104951, 2022.