UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
COORDENADORIA DO CURSO DE ENGENHARIA DE SOFTWARE

ANDREI WELLITON COREZOLLA

UMA ABORDAGEM PARA PRIORIZAÇÃO DE REQUISITOS BASEADA EM BUSCA

TRABALHO DE CONCLUSÃO DE CURSO

DOIS VIZINHOS

2019

ANDREI WELLITON COREZOLLA

UMA ABORDAGEM PARA PRIORIZAÇÃO DE REQUISITOS BASEADA EM BUSCA

Trabalho de Conclusão de Curso apresentado como requisito parcial à obtenção do título de Bacharel em Engenharia de Software, da Universidade Tecnológica Federal do Paraná.

Orientador: Prof. Dr. Francisco Carlos Monteiro Souza

DOIS VIZINHOS

2019

TERMO DE APROVAÇÃO

Uma abordagem para priorização de requisitos baseada em busca

por

Andrei Welliton Corezolla

Este Trabalho de Conclusão de Curso foi apresentado em 26 de Novembro de 2019 como requisito parcial para a obtenção do título de Bacharel em Engenharia de Software. O(a) candidato(a) foi arguido(a) pela Banca Examinadora composta pelos professores abaixo assinados. Após deliberação, a Banca Examinadora considerou o trabalho aprovado.

Francisco Carlos Monteiro Souza
Presidente da Banca

Alinne Cristinne Correa Souza
Membro Titular

Rodolfo Adamshuk Silva
Membro Titular

* A Folha de Aprovação assinada encontra-se na Coordenação do Curso

AGRADECIMENTOS

Agradeço primeiramente a Deus por ter me mantido na trilha certa durante este projeto de pesquisa com saúde e forças para chegar até o final.

Agradeço aos meus pais pelo incentivo aos estudos e pelo apoio incondicional.

Agradeço de forma especial à minha namorada Tamiris, que sempre esteve ao meu lado durante o meu percurso acadêmico. Agradeço pelo amor incondicional, amizade, paciência e incentivo para não desistir dessa caminhada.

Sou grato pela confiança depositada na minha proposta de projeto pelo meu professor Francisco Carlos Monteiro Souza, orientador do meu trabalho. Obrigado por me manter motivado durante todo o processo.

Por último, quero agradecer também à esta universidade, seu corpo docente, direção e administração que oportunizaram a janela que hoje vislumbro um horizonte superior.

RESUMO

Corezolla, A. W. UMA ABORDAGEM PARA PRIORIZAÇÃO DE REQUISITOS BASEADA EM BUSCA. 78 f. Trabalho de Conclusão de Curso – Coordenadoria do Curso de Engenharia de Software, Universidade Tecnológica Federal do Paraná. Dois Vizinhos, 2019.

A necessidade de desenvolvimento e entrega de soluções de software com qualidade em um curto espaço de tempo é um desafio enfrentado pelas empresas, visto que muitas vezes a qualidade de um software é prejudicada por conta desses fatores. Por consequência a entrega de funcionalidades que pouco agregam valor a versão do projeto se mostra um problema. Saber escolher os melhores conjuntos de requisitos para compor a versão do sistema é uma atividade custosa, visto que métodos tradicionais de priorização se mostram ineficazes. Com base nesta problemática, este estudo tem como objetivo apresentar uma abordagem de priorização de requisitos utilizando algorítimo genético a fim de encontrar soluções alternativas que possam auxiliar no processo de decisão de priorização de requisitos, durante o processo de desenvolvimento de software. Os experimentos permitiram avaliar a eficácia da abordagem proposto qualitativamente e quantitativamente, gerando resultados satisfatórios, que indicam a viabilidade do seu uso podendo assim contribuir com o desenvolvimento de projetos de software, tanto no meio acadêmico, como no meio corporativo.

Palavras-chave: Engenharia de Software baseada em Busca, Priorização de Requisitos

ABSTRACT

Corezolla, A. W. A SEARCH-BASED REQUIREMENTS PRIORITIZATION APPROACH. 78 f. Trabalho de Conclusão de Curso – Coordenadoria do Curso de Engenharia de Software, Universidade Tecnológica Federal do Paraná. Dois Vizinhos, 2019.

The demand for developing and delivering quality software solutions in a short period of time is a challenge facing companies, as software quality is often inhibited by these factors. Consequently, the delivery of features that add less value to the project version proves to be a problem. Knowing how to choose the best sets of requirements to compose the system version is a costly activity, as traditional prioritization methods are ineffective. Based on this problem, this study aims to present a requirements prioritization approach using a genetic algorithm in order to find alternative solutions that can assist in the requirements prioritization decision process during the software development process. The experiments evaluated the effectiveness of the proposed approach qualitatively and quantitatively, generating satisfactory results, which indicate the feasibility of its use and can thus contribute to the development of software projects, both in academia and in the corporate environment.

Keywords: Search-Based Software Engineering, Requirements Prioritization

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

AG	Algoritmo Genético
AHP	Analytic Hierachy Process
ER	Engenharia de Requisitos
ES	Engenharia de Software
HC	Hill Climbing
IA	Inteligência Artificial
MONRP	Multi-Objective Next Release Problem
MS	Mapeamento Sistemático
NRP	Next Release Problem
NSGA	Non-dominated Sorting Genetic Algorithm
NSGA-II	Non-dominated Sorting Genetic Algorithm-II
SBSE	Search Based Software Engineering
UTFPR	Universidade Tecnológica Federal do Paraná
$Cf1$	Configuração 1
$Cf2$	Configuração 2
GUT-AG	Matriz Gut - AG
TS-AG	Theme Screening-AG
TVS-AG	Tempo *versus* Satisfação-AG
MT-AG	Menor Tempo-AG
G	Gravidade
U	Urgência
T	Tendência
Pr	Prioridade
Dep	Dependência
Tp	Tempo de espera
Tr	Tempo de execução
I	Importancia
H	Horas de desenvolvimento
QPs	Questões de pesquisa

LISTA DE SÍMBOLOS

$\sum$	Letra Sigma
$\leq$	Menor ou Igual
$\neq$	Diferente
μ	Letra Mu
$\subseteq$	Subconjunto

SUMÁRIO

1 INTRODUÇÃO

A Engenharia de Software (ES) é a criação e o uso de princípios sólidos da engenharia a fim de se obter um software que seja confiável, funcione de maneira eficiente, tenha qualidade e baixo custo (PRESSMAN, 2011). Além disso, ela está atrelada a outras atividades como gerenciamento de projeto, desenvolvimento de ferramentas e metodologias que apoiem a produção de software (SOMMERVILLE, 2010).

Nesta produção várias atividades são abordadas, dentre elas está a Engenharia de Requisitos (ER), a qual trata das especificações do software, visando a compreensão do problema a ser resolvido por parte dos analistas ou engenheiros de software. Entender as necessidades do cliente, no momento da construção do software, é essencial para um projeto bem sucedido (PRESSMAN, 2011).

A priorização de requisitos trata-se de uma atividade que tem o objetivo de definir quais serão os requisitos, ou conjunto deles, a serem executados em uma determinada ordem. Nessa etapa é presenciada uma grande dificuldade por parte das empresas desenvolvedoras de software, pois garantir quais conjuntos de funções ou mudanças no software que serão contemplados primeiro torna-se uma tarefa difícil e exaustiva. Portanto, um planejamento adequado consiste em uma atividade complexa, visto o envolvimento de diversos fatores e tomadas de decisões que podem impactar drasticamente no projeto de software.

Com o crescimento das demandas do mercado de software e a necessidade de entrega de produtos em um curto espaço de tempo, estimulam as empresas a procurarem por soluções que auxiliem na seleção dos requisitos que mais agregaram valor em seu produto para aquele momento. Portanto, definir os requisitos que vão compor a próxima versão do sistema se mostra um desafio. Entender a necessidade do cliente e também entregar um produto que disponibilize qualidade, além das funções desejadas e ao mesmo tempo traga lucros para empresa torna-se um objetivo cada vez mais distante de se alcançar.

Dessa forma, a utilização de metodologias de priorização de requisitos ganham

o seu espaço. Essas metodologias auxiliam nas tomadas de decisões, sendo um aspecto primordial dentro do desenvolvimento de software visto que a utilização de recursos de um projeto cresce exponencialmente de acordo com a sua complexidade.

1.1 MOTIVAÇÃO

Com a crescente demanda do mercado de software, a priorização de requisitos torna-se um desafio e a utilização de ferramentas que possam agregar nesse processo se torna cada vez mais relevante. Fatores como a falta de profissionais capacitados e familiarizados com o mercado também agravam esse cenário.

Mesmo com a utilização das metodologias ágeis, ainda são encontradas diculdades durante o processo de desenvolvimento de um software e, dentre elas, priorizar requisitos ganha destaque, saber o momento ideal que cada requisito deve ser contemplado se mostra uma tarefa complexa (PRESSMAN, 2011). A incorreta priorização de requisitos prejudica o software, pois entregar funcionalidades que não serão utilizadas e não agregarão valor podem causar prejuízos, insatisfação e aversão ao produto. Tendo em vista essas circunstância, surge a necessidade de explorar alternativas para sanar tal problema encontrado dentro da área de ER.

1.2 OBJETIVOS

Assim, o trabalho tem como objetivo desenvolver uma abordagem baseada em busca para a priorização de requisitos de software que possa auxiliar nas tomadas de decisões. Para isso, conceitos de ER e técnicas da Inteligência Artificial (IA), mais especificamente a Engenharia de Software Baseada em Busca (do termo inglês, *Search Basead Software Engineering - SBSE*), serão utilizados para solução do problema abordado. A pesquisa terá como foco a utilização de técnicas de busca ou metaheurísticas que têm se mostrado bem sucedidas em diferentes domínios.

Para alcançar o objetivo geral desta monografia, alguns objetivos específicos foram definidos:

- Avaliar e selecionar técnicas de priorização de requisitos tradicionais;
- Implementar diferentes técnicas de busca;
- Propor uma ou mais funções objetivos;

1.3 ESTRUTURA DA MONOGRAFIA

O restante desta monografia está organizada da seguinte forma. O Capítulo 2 trata dos aspectos conceituais relevantes para elaboração do trabalho. O Capítulo 3 apresenta o mapeamento sistemático, destacando como foi conduzido os resultados obtidos. No Capítulo 4 é apresentado a proposta definida para esse trabalho, juntamente com a explicação da abordagem. Em seguida o Capítulo 5 apresenta os resultados obtidos da abordagem proposta e por fim no Capítulo 6 é apresentado as considerações finais.

2 ASPECTOS CONCEITUAIS

Para levantar as informações necessárias, foram realizadas pesquisas e estudos focando nos objetivos deste trabalho. Entre as informações levantadas destacam-se conceitos da área de ES tais como ER, priorização de requisitos, algoritmos e técnicas de buscas dentro da IA.

O capitulo está distribuído da seguinte forma. Na Seção 2.1 são sintetizados os principais conceitos em relação a ER. Na 2.2 são apresentados conceitos em relação a SBSE, onde também são apresentados seus conceitos em relação as técnicas de busca e priorização.

2.1 ENGENHARIA DE REQUISITOS

A ER é uma sub-área da ES, que consiste no levantamento de informações relevantes para o desenvolvimento de um software junto ao usuário ou as partes envolvidas, sendo assim uma das fases primordiais no desenvolvimento de um software. Pressman (2011) cita que a ER estabelece uma base sólida para o projeto e desenvolvimento do software, sem ela o software resultante possui grandes possibilidades de não atender as necessidades do cliente.

A ER pode ser contextualizada como um processo que busca a produção de um documento de requisitos que auxiliará a construção do software. Sua importância está fortemente atrelada a qualidade do produto entregue ao usuário final. O objetivo da ER é fornecer técnicas e métodos que auxiliem nas fases de elicitação, especificação, modelagem, validação e gerenciamento de requisitos.

Uma das principais atividades da ER consiste na elicitação e documentação dos requisitos. A sua execução de maneira correta assegura às equipes de desenvolvimento consistência na elaboração do software, de tal modo que possa atender as expectativas do cliente e prevenir contra custos desnecessários durante a elaboração do projeto.

No entanto um problema constante que surge neste processo de construção de um software é a compreensão das necessidades do cliente. Saber identificar quais serão os objetivos do sistema, as necessidades de negócio, e como será a utilização do sistema no dia-a-dia, exige muita experiência dos analistas de sistemas, afim de abstrair essas informações, visto que muitas vezes o cliente não sabe o que realmente precisa no software. Portanto, saber identificar as suas reais necessidades e entender seus problemas minimizam as chances de erros na construção do software.

Para um melhor entendimento em relação a importância dos requisitos de software em um projeto, alguns autores explanam as suas considerações, como por exemplo Sommerville (2010), o qual classifica os requisitos como os artefatos definidos nas fases iniciais de um projeto, os quais servem como as especificações que devem ser implementado no software.

A ER é subdividida em quatro fases principais, segundo Sommerville (2010), essas fases podem ser descritas como:

I) Estudo da viabilidade: constitui-se na realização do levantamento e análise da viabilidade dos requisitos, nessa fase são avaliados os riscos, prazos, custos e impactos entre outros fatores que afetam diretamente o projeto.

II) Elicitação de requisitos: consiste no processo de derivação dos requisitos do sistema, além de realizar discussões com os usuários para o entendimento de suas solicitações.

III) Especificação de requisitos: apresenta a atividade de traduzir as informações obtidas durante a análise em um documento que define um conjunto de requisitos. Existem dois tipos que podem ser incluídas nesse documento, os requisitos dos usuários, que são declarações abstratas dos requisitos do sistema para o cliente e usuário final; e os requisitos de sistema que são uma descrição mais detalhada da funcionalidade a ser provida.

IV) Validação: essa atividade verifica os requisitos quanto ao realismo. Nessa fase é analisado a consistência e a completude do projeto, na qual os erros na documentação de requisitos são descobertos, corrigidos e atualizados.

Pode-se perceber que muitos problemas ocorrem na construção e elaboração de um projeto de software, isso está atrelado a falta de entendimento referente às necessidades do cliente devido a má documentação dos requisitos em sua fase inicial. Por isso, a elicitação dos requisitos e a priorização se tornam relevantes no desenvolvimento de um software. Neste contexto, o trabalho apresentado busca soluções que possam auxiliar a superar tais

limitações. A seguir será detalhado mais a fundo os conceitos dos assuntos referidos.

2.1.1 ELICITAÇÃO DE REQUISITOS

A elicitação de requisitos, ou levantamento de requisitos, consiste em uma das principais atividade desenvolvida na ER. Nessa atividade são extraídas as informações junto ao cliente referente às sua necessidades diante do software. Por meio da elicitação de requisitos busca-se entender as perspectivas do cliente em relação a finalidade, regras de negócio e usabilidade do sistema.

Segundo Johnson (1996), essa fase tem o intuito de descobrir os requisitos do sistema, que muitas vezes são obscuros, vagos e confusos. No entanto, essa atividade pode se tornar um pouco desafiadora devido ao fato de que muitas vezes o cliente não sabe o que realmente necessita para o momento, ou até mesmo ele sabe, porém não consegue expressar de uma forma compreensível. Assim, o papel do analista de requisitos é fundamental, visto que o mesmo possui o conhecimento necessário para identificar, interpretar e documentar as necessidades do cliente.

Apesar das contribuições do analista, é notável que uma parte significativa dos problemas do desenvolvimento do software ocorre na elicitação de requisitos. Uma elicitação inadequada pode trazer consequências que podem levar ao fracasso do projeto, pois realizar as correções de requisitos demanda tempo e retrabalho e, por consequência, elevam os custos da equipe de desenvolvimento, além do descontentamento do cliente. De acordo com LARMAN (2004), a indústria de software está repleta de projetos fracassados que não atenderam as expectativas dos clientes.

Silva (2011) cita que o processo de elicitação de requisitos dispõe de técnicas que podem ser utilizadas durante o processo de levantamento de requisitos, tais técnicas podem ser definidas como:

- **entrevistas (fechadas e abertas):** preferencialmente devem ser aplicadas com diferentes *stakeholders* para obter um entendimento preciso sobre os requisitos do sistema;
- **cenários:** é usado como uma técnica de representação de situações durante a análise de um sistema, a fim de expressar os objetivos dos usuários desse sistema;
- **protótipos:** define uma versão inicial do sistema para experimentação. Os protótipos das telas serviriam como uma linguagem mais informal e acessível aos futuros usuários;

- **diagramas de caso de uso:** servindo como referência para as atividades de implementação;

Kotonya e Sommerville (1998) mencionam que a elicitação de requisitos não se trata apenas de uma atividade que envolve perguntas ao usuário, ela também envolve uma análise de domínio de processos e de negócios cuidadosa na organização onde o software será implantado.

Sommerville (2016) apresenta um modelo de processo de elicitação de requisitos exibido na Figura 1, nesse modelo de processos existem quatro fases, sendo elas:

Figura 1: Processo de elicitação e análise de requisitos

Fonte: Adaptada de Sommerville (2016)

1. **Descoberta de requisitos:** nessa fase são realizadas as interações com os *stakeholders* a fim de descobrir os requisitos que o software deve conter, para isso são utilizadas técnicas que auxiliam a elicitação dos requisitos.

2. **Classificação e organização de requisitos:** essa atividade usa a coleção de requisitos não estruturados, agrupa requisitos relacionados e os organiza em grupos coerentes. A forma mais comum de agrupar os requisitos é o uso de um modelo de arquitetura do sistema para identificar subsistemas e associar requisitos a cada subsistema. Na prática, a ER e projeto da arquitetura não podem ser atividades completamente separadas.

3. **Priorização e negociação de requisitos:** nessa atividade tem como objetivo resolver conflitos, porque tendem a existir conflitos entre diferentes *stakeholders*. Esse processo trata as principais necessidades de cada usuário, atribuindo prioridades

aos requisitos a fim de garantir que os requisitos mais críticos sejam atendidos o mais rápido possível.

4. **Especificação de requisitos:** os requisitos são documentados e inseridos no próximo ciclo da espiral do processo.

Como pode-se perceber, Sommerville (2016) trata o processo de elicitação de requisitos em um formato de espiral interativa com constantes *feedbacks* nas atividades. O ciclo do processo começa com a descoberta de requisitos e termina com sua documentação. Nesse processo, inevitavelmente, os *stakeholders* envolvidos mostram opiniões diferentes sobre a importância e a prioridade dos requisitos, por consequência, essas opiniões entram em conflitos constantes. Diante desse cenário, é importante saber negociar com os envolvidos, a fim de que todos os requisitos estabelecidos sejam cumpridos em relação ao projeto.

2.1.2 PRIORIZAÇÃO DE REQUISITOS

Como foi apresentado na subseção anterior, os requisitos sofrem várias alterações em seu ciclo de elicitação. Assim, tratar e priorizar eles de maneira correta, torna-se uma tarefa complexa, uma vez que requer muita determinação e paciência, sem mencionar o trabalho de lidar com diferentes interesses das partes envolvidas.

A priorização de requisitos é uma atividade que consiste em definir quais serão os requisitos, ou conjunto deles, a serem executados em uma determinada ordem. Nessa etapa é presenciado uma grande dificuldade pelas empresas desenvolvedoras de software, pois garantir quais conjuntos de funções ou mudanças no software serão executados primeiro, se torna uma tarefa exaustiva. Portanto, um planejamento adequado se torna crucial, por envolver diversos fatores e tomadas de decisões que podem impactar drasticamente no projeto de software.

O significado da priorização de requisitos pode ser conceituado de várias formas. Na literatura vários autores explanam suas interpretações. Sommerville (2010) interpreta a priorização de requisitos como uma das tarefas mais significativas para os tomadores de decisão dentro de um projeto. Por outro lado, Hudaib et al. (2018) trata como o principal processo em ES, fornecendo a ordem perfeita de implementação dos requisitos para planejar versões de software e fornecer funcionalidade desejáveis aos clientes.

A priorização de requisitos pode ser vista como a importância que tais requisitos têm diante de um projeto de software. Muitas vezes a priorização ocorre dado o momento

ou interesse das partes envolvidas, como por exemplo, um cliente que representa um maior valor de negócio para a empresa, a fim de satisfazer as suas necessidades, as suas solicitações podem ser atendidas com maior prioridade. Entretanto tais requisitos podem representar algo significativo apenas para esse cliente, e o seu valor agregado ao sistema se torna baixo. Nesse contexto, é importante destacar que o tempo de desenvolvimento e os recursos utilizados para contemplar as necessidades deste cliente, representam um lucro muito baixo.

No contexto de uma empresa desenvolvedora de software é comum perceber que muitas vezes a quantidade de requisitos supera a quantidade de tempo, recurso e orçamentos disponíveis para a contemplação dos mesmos. Por isso, existe a necessidade de priorizar-se os requisitos, selecionando os que tendem atingir a maior satisfação dos clientes, ao ser implementado (MULLA, 2012).

Várias abordagens têm sido propostas na literatura para apoiar a priorização de requisitos em um projeto, na Tabela 1 pode-se verificar alguns exemplos de técnicas utilizadas. Entretanto com as fortes demandas dos projetos e a complexidade dos mesmos, técnicas tradicionais de priorização de requisitos perderam a sua eficácia, trazendo um grande problema em relação a priorização de requisitos.

Porém, nos últimos anos, com o crescimento dos estudo relacionados a área de ES fez com que o uso de técnicas de otimização se tornasse um meio favorável para a realização da priorização de requisitos. A aplicação de metaheurísticas na ES deu a origem a uma nova área de pesquisa denominada Engenharia de Software baseada em Busca do inglês, *Search Based Software Engineering (SBSE)* que auxiliou na construção de soluções da ER.

2.1.3 TÉCNICAS TRADICIONAIS DE PRIORIZAÇÃO DE REQUISITOS

Dentre varias técnicas de priorização existentes algumas destacam-se pela sua simplicidade e bom desempenho, as técnicas Matriz GUT e *Theme screening* servem de bons exemplos.

A Matriz GUT é uma ferramenta de gestão utilizada na priorização de problemas,ela é comumente utilizada quando se necessita uma tomada de decisão mais complexa, onde vários outros fatores estão incluídos como as dependências que cada operação possui. Essa técnica foi desenvolvida nos anos 80 por Charles Kepner e Benjamin Tregoe, especialistas na resolução de problemas organizacionais, sua finalidade era auxiliar nas decisões mais

complexas, decisões essas que envolvem dificuldades na visão de administradores de empresas. Para tratar dessa complexidade, a matriz GUT classifica cada problema de acordo com a Gravidade, Urgência e Tendência, gerando a sigla (GUT) (CAMARGO, 2018).

- Gravidade (G): trata do impacto que o problema gerará nos envolvidos, podendo ser os colaboradores, os processos, tarefas, resultados da empresa etc. A análise é feita nos efeitos que o problema, caso não seja resolvido, acarretará em médio e longo prazo.
- Urgência (U): é o prazo, ou o tempo disponível para a resolução do problema. Quanto menor o tempo, mais urgente será o problema que deverá ser resolvido.
- Tendência (T): trata da probabilidade (ou do potencial) que o problema tem de crescer com o passar do tempo.

A principal vantagem dessa matriz de priorização está em trazer uma avaliação quantitativa dos problemas de uma área ou da organização como um todo, possibilitando que sejam priorizadas ações corretivas e preventivas para que o problema seja eliminado parcialmente ou em sua totalidade (CAMARGO, 2018).

Os fatores trabalhados com a Matriz GUT (Gravidade, Urgência e Tendência) são pontuados de 1 a 5,conforme apresentados na a tabela 2. A combinação dessas pontuações definirá quais ações serão prioritárias. Essa combinação é feita com um cálculo de multiplicação dos três fatores (G) x (U) x (T). Portanto, o resultado com maior pontuação no Método GUT será o de 125 pontos e o menor, 1.

Por outro lado a técnica *Theme screening* apresenta um conceito diferente de classificação o qual se destaca por ser mutável e possuir uma aplicação imparcial e de fácil entendimento. Está técnica consiste em ordenar as funcionalidades com base em temas de negócio utilizados como fatores de comparação, conforme ilustrado na Figura 2 (MASSARI; VIDAL, 2018).

Para aplicar é preciso definir os critérios de comparação e garantir que todos são válidos em cada requisito em questão. Devem ser definidos no mínimo cinco e no máximo nove critérios, onde, um deles deve classificado como tema base e servir de referência para a pontuação dos outros. O tema base recebe pontuação zero enquanto os mais importantes recebem um sinal e positivo e o menos importantes um sinal de negativo.

O valor de cada item priorizado é obtido por meio da soma dos sinais negativos e positivos, ao final, ordenando do maior para o menor temos o resultado da priorização.

Figura 2: Exemplo da técnica *Theme screening*

	Critérios						
	Complexidade	Esforço	ROI	Integração	Orçamento	Total	Prioridade
Acesso diferenciado para assinantes	-	-	+	+	+	+1	02
Disponibilizar vídeos sobre o produto	-	+	+	-	-	-1	04
Pagar com cartão de crédito	0	0	0	0	0	0	03
Integração com mídias sociais	+	+	+	-	+	+2	01
Disponibilizar acesso gratuito	-	-	+	-	+	-1	04

Fonte: Adaptado de (MASSARI; VIDAL, 2018).

2.2 ENGENHARIA DE SOFTWARE BASEADA EM BUSCA

A SBSE foi utilizado pela primeira vez em 2001 por HARMAN e JONES (2001). A definição de SBSE pode ser vista como a busca de soluções apropriadas dentro de um espaço possível de soluções, a fim de resolver problemas encontrados na ES. Esses problemas são solucionados com a utilização de metaheurísticas em suas abordagens.

Geralmente esta área é envolvida por problemas mais complexos, oque torna a sua exploração exaustiva. Para minimizar esses problemas são sugeridas técnicas metaheurıstica, que utiliza métricas (que podem ser chamadas de função de adequação, função de custo, função objetiva ou medida de qualidade) para medir qualitativamente as possíveis soluções válidas (Harman; Clark, 2004).

Em geral esses problemas são evidentes em atividades da ER, tal como a atividade de priorização de requisitos, visto que a existência de uma elevada quantidade de requisitos com diferentes níveis de complexidade, designa a utilização de uma solução otimizada que torne essa atividade menos exaustiva. Para solucionar tais problemas, a aplicação de algoritmos baseados em buscas se tornaram cada vez mais frequentes, apresentando resultados consideráveis em relação aos problemas que antes, sem o uso da SBSE, se mostravam complexos em encontrar soluções.

Neste contexto, o uso de técnicas automatizadas para solucionar tais problemas na área de ER tem sido eficazes, e consequentemente, a utilização da SBSE ganha seu

espaço na busca de soluções.

2.2.1 ALGORITMO GENÉTICO

O Algoritmo Genético (AG) foi proposto por Holland (1975) na Universidade de Michigan. O princípio da pesquisa era se fundamentar em processos de evolução das espécies para formalizar um algoritmo e aplicar conceitos da evolução (conhecidos como operadores genéticos, tais como, genes, cromossomos, seleção, cruzamento e mutação) para resolver problemas que possuem um amplo espaço de busca.

Um dos princípios fundamentais em AGs, é a evolução, a qual proporciona que uma população de indivíduos (soluções candidatas) se modifique até representar a melhor solução encontrada. Os indivíduos com as melhores aptidões genéticas têm maiores chances de sobrevivência e de produzirem filhos (novas soluções) cada vez mais aptos, enquanto indivíduos menos aptos tendem a ser eliminados da população pela seleção natural. Sendo assim um AG imita os mecanismos da evolução natural para resolver problemas de otimização.

O AG utiliza um conjunto de cromossomos (também podem ser denominados como indivíduos), onde um cromossomo representa a abstrata de uma solução para o problema. Cada cromossomo é formado por genes, os quais carregam uma informação para a resolução do problema. A Figura 3 representa ilustrativamente um cromossomo.

Figura 3: Representação de um cromossomo

Fonte: Autoria própria

De acordo com Mitchell (2002). o desempenho do AG depende muito da sua configuração como, por exemplo, o tamanho da população. Esse parâmetro está diretamente relacionado ao problema que se pretende resolver. Caso seja definida uma população com muitos indivíduos, a aplicação do AG poderá ter um elevado custo computacional. No entanto, caso ela seja muito pequena ocorrerá uma redução nas possibilidades de respostas para o problema, ou seja, não haverá variedades de soluções candidatas.

No mundo real, a capacidade de sobrevivência, em um meio, de um indivíduo demonstra sua aptidão de prosperar, e por consequência, procriar gerando assim novos descendentes. Porém caso o indivíduo não seja capaz de adaptar-se ao meio, certamente

sua descendência estará comprometida, visto que suas características genéticas não serão transmitidas às próximas gerações, tendendo a sua extinção.

Para medir o nível de adaptabilidade do indivíduo, o mesmo é avaliado através de um função de avaliação, também chamada de função de aptidão ou *fitness*, essa função tem como entrada o individuo que será avaliado. No processo de seleção, os indivíduos que obtiverem uma avaliação alta, possuirão mais chances de serem escolhidos para a reprodução, em comparação com os indivíduos que possuírem um valor mais baixo. Nesse processo é crucial que todos os indivíduos possam ser selecionados, pois, indivíduos com uma baixa avaliação podem apresentar alguma característica que seja propícia para encontrar a melhor solução do problema, pois se apenas os indivíduos com as melhores avaliações evoluírem, a população tenderá a ser formada por indivíduos cada vez mais semelhantes, reduzindo a diversidade da população, gerando o efeito da convergência (SANTOS, 2017).

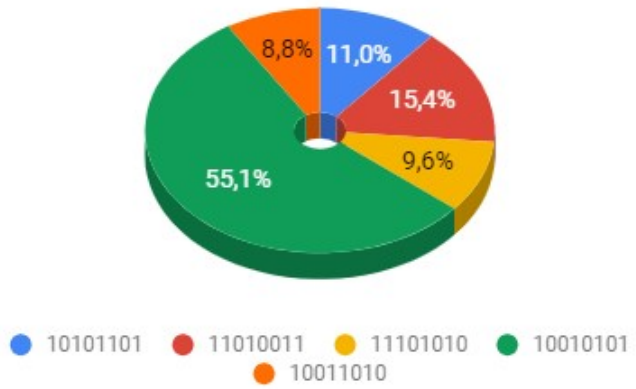
Para realizar a evolução da população são aplicados os operadores genéticos de seleção, reprodução e mutação. Os mesmos podem ser descritos como a base do funcionamento do AG. Mitchell (2002) cita que para a realizar o cruzamento dos indivíduos de uma população, é necessário aplicar um método para seleção dos mesmos, levando em consideração o seu valor de aptidão. Para tal procedimento existem diversas estratégias de seleção de indivíduos. Algumas técnicas de seleção serão apresentadas a seguir:

- **Roleta:** Se trata do método amplamente utilizado, no qual seleção dos cromossomos ocorrem de forma proporcional ao seu valor de aptidão, sendo que os cromossomos de uma população são representados em uma roleta, onde ocupam um espaço proporcional ao seu valor de aptidão. Esse método simula uma roleta, onde cada indivíduo possui a probabilidade correspondente a fatia total dividida pelo seu valor de aptidão. Sendo assim indivíduos com maior aptidão ocuparam maior espaço da fração. Nesse método a roleta pode ser girada quantas vezes forem necessárias para a obtenção de boas seleções.

  No exemplo da Tabela 3, podemos ver que o indivíduo 10010101 possui 55,15% da somatória de *fitness*, por tanto ele possui maior espaço da fração da roleta, em contra partida o indivíduo 10011010 possui a menor fração da roleta, com apenas 8,82% da fração, como mostra a Figura 4.

- **Cruzamento:** O processo de cruzamento é realizado após a seleção. Nesta fase ocorre a troca de segmentos entre pares de cromossomos selecionados para originar

Figura 4: Probabilidade de indivíduos para seleção da roleta

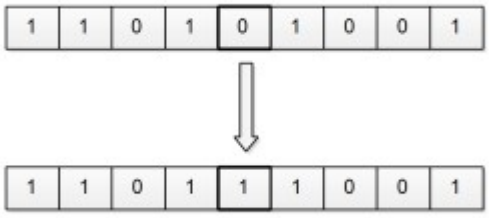


Fonte: Autoria própria

os novos indivíduos que virão a formar a população da geração seguinte. O objetivo desse processo, é gerar novos descendentes, através da troca de características entre os cromossomos, propagando assim as características positivas dos indivíduos mais aptos.

- **Mutação:** A mutação tem como objetivo realizar uma alteração aleatória de um gene do cromossomo. A realização desse processo efetua uma mudança cromossômica do indivíduo como o aparecimento de uma característica inexistentes nas gerações passadas. Esse processo garante maior diversidade entre os indivíduos. De tal modo o processo de mutação funciona como um operador de manutenção da diversidade genética, pois modifica as características de um indivíduo aleatoriamente como apresentado na Figura 5.

Figura 5: Exemplo da operação de mutação



Fonte: Adaptada de SOUZA (2017)

Para assegurar que determinados indivíduos não se dispersem durante as gerações, existem estratégias, como o elitismo, que consiste em preservar n indivíduos da população corrente, esta estratégia garante que características importantes para a solução não se

percam. Essa estratégia garante que o desempenho do AG sempre aumente no decorrer das execuções, porém, dependendo do seu uso, o elitismo pode ocasionar uma convergência prematura na busca (SOUZA, 2017).

Além da sua formulação clássica, existem outras formulações de AGs, como por exemplo, o NSGA e o NSGA-II que são utilizados para tratar de problemas multiobjetivos.

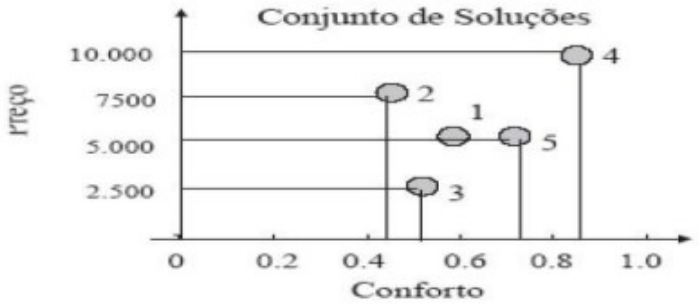
O Algoritmo Genético de Ordenação Não Dominada-II ou do termo inglês *Non-dominated Sorting Genetic Algorithm-II* (NSGA-II) foi primeiramente proposto por (Deb et al., 2002). Esse algoritmo se trata de um aprimoração do algoritmo, *Non-dominated Sorting Genetic Algorithm* (NSGA), desenvolvido por (SRINIVAS; DEB, 1994). A principal diferença, em relação a versão anterior, é que o NSGA-II incorpora elitismo para manter as soluções da melhor frente encontrados. A classificação de cada indivíduo é baseada no nível de não-dominação.

O algoritmo consiste em realizar a comparação de duas soluções, obtendo a população filha, o qual é usado para prosseguir com a execução do problema. No torneio uma solução domina a outra quando tem uma melhor aptidão, e se caso as soluções possuam a mesma aptidão, domina a que tiver melhor distância. Assim, é realizada a mutação e recombinação dos indivíduos. Para garantir o elitismo, os conjuntos pai e filho são aglomerados na mesma população para aplicar a classificação não dominada em frentes de dominância (DELINSKI, 2017).

Ticona (2003) explica em um conceito ilustrativo, em que dado um problema multiobjetivo, onde se tem dois objetivos, como a compra de um carro, onde se busca minimizar o valor de custo e ao mesmo tempo maximizar o conforto. A Figura 6 ilustra uma opção de compra de um carro, onde são apresentados 5 opções de possíveis compra. Diretamente descarta-se a opção 1 pois a solução 5 fornece mais conforto por um preço menor, a solução 2 segue a mesma lógica, assim tem-se 3 soluções possíveis: 3,4,5, que representam boas alternativas. Em termos quantitativos, nenhuma opção é melhor que a outra, pois uma opção possui mais conforto que a outra, porém o seu custo é mais alto e vice-versa. Com isso percebemos que quanto maior o conforto maior o seu preço, e quanto menor o preço menor será o conforto. Nesse contexto é dito que uma solução domina a outra caso seus valores forem superiores em todos os sentidos.

Por exemplo, a solução 5 domina a solução 1. Por outro lado a solução 5 não é dominada por nenhuma outra. Isso também ocorre com as soluções 3 e 4, como não se sabe a dominância relativa de cada objetivo, pode-se dizer que as soluções 3, 4, e 5 são igualmente boas. Portanto, existe um conjunto de soluções ótimas, este conjunto

Figura 6: Conjunto de Soluções



Fonte: Adaptada de Ticona (2003)

é chamado de conjunto não dominado.As soluções 1, e 2 formam o conjunto dominado (TICONA, 2003).

Com tudo, os algoritmos possuem vantagens e desvantagens, dentre elas estão, problemas relacionados ao parâmetro de tamanho da população presente na primeira versão do NSGA, tal foi resolvido e não se encontra na segunda versão (NSGA-II) porém, a utilização da distância de aglomerados, da segunda versão só funciona no espaço objetivo.

2.2.2 FORMULAÇÃO DE PRIORIZAÇÃO DE REQUISITOS COMO BUSCAS

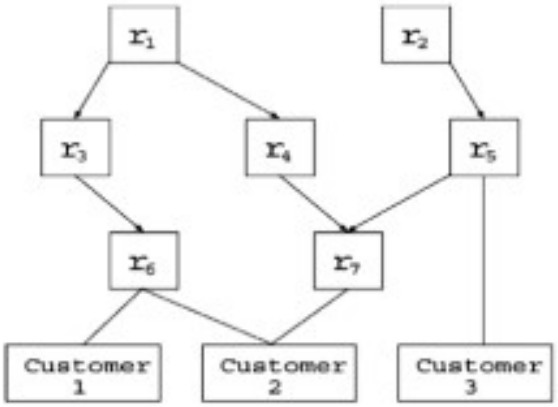
Nos últimos 10 anos, vários autores publicaram estudos em relação a SBSE, Zhang, Finkelstein e Harman (2008), por exemplo, citam que a SBSE utiliza técnicas para a solução de problemas de otimização durante a fase de análise de requisitos. O uso dessas técnicas de busca oferecem grandes vantagens nas escolhas para o tomador de decisão.

Segundo Harman (2007), a SBSE, está sendo aplicada a em diversos problemas de ES. Além disso, o autor também explica algumas maneiras de como a SBSE foi aplicada para a priorização de requisitos, as quais tem se mostrado muito eficientes. O mesmo autor ainda relata, que a SBSE se torna interessante quando comparada com outras abordagens, uma vez que apresenta uma gama de soluções automatizadas e semi-automatizado adaptativas que podem ser utilizadas em condições que envolvem problemas grandes e complexas com múltiplos objetivos.

Em relação a aplicação da SBSE na ER, o primeiro a propor uma modelagem para os problemas enfrentados na priorização de requisitos foi Bagnall, Rayward-Smith e Whittley (2001) ele apresentou a modelagem *Next Release Problem (NRP)*. Sua modelagem propôs a representação de um conjunto de variáveis binárias $R = \{r_1, ..., r_n\}$ e $C =$

$\{c_1, ..., c_n\}$ representando conjunto de requisitos e clientes, respectivamente. Para cada requisito r_i. há um custo v_i associado e para cada cliente c_i há um valor representando a importância que o cliente tem para a empresa w_i. Para cada cliente c_i existe um subconjunto ($R_i \subset R$), os requisitos solicitado pelo cliente c_i. A Figura 7 mostra a representação de dependência entre os requisitos. Com o conjunto de vértices sendo o conjunto de requisitos e uma aresta (r_i, r_j) para cada r_i que é pré-requisito de r_j. (SILVA, 2015)

Figura 7: Representação de dependência entre requisitos



Fonte: Adaptada de Bagnall, Rayward-Smith e Whittley (2001)

Por fim uma consideração importante em seu modelo proposto é que existe um custo máximo V para o projeto, que deve ser respeitado. Isso pode ser traduzido na seguinte forma.

$$\sum_{i=1}^{n} v_i r_i \leq V \tag{1}$$

Enquanto, o objetivo do problema é maximizar a satisfação do cliente usando a seguinte formulação. Nesse contexto, o intuito dessa função é encontrar o melhor conjunto de requisitos respeitando o valor máximo extipulado ao projeto.

$$\sum_{i=1}^{m} w_i c_i \tag{2}$$

Em 2007 outra abordagem foi proposta por Zhang, Harman e Mansouri (2007), na qual foi chamada de *Multi-Objective Next Release Problem (MONRP)*, no qual o custo não é mais uma restrição e o conjunto a ser achado é aquele que visa maximizar a satisfação

dos clientes, minimizando o custo total. Esta formulação Multi-objetiva é apresentada abaixo (SILVA, 2015):

$$maximize \sum_{i=1}^{n} score_i.r_i$$

$$minimize \sum_{i=1}^{n} cost_i.r_i$$

Dada a formulação $R = \{r_1, r_2, ..., r_n\}$ as variáveis binárias correspondentes aos requisitos, com o valor de importância e custo representados por $score_i$ e $cost_i$, respectivamente. O valor de $score_i$ leva em consideração que os clientes podem selecionar diferentes valores de importância para cada requisito e a importância que o cliente tem para a empresa. De tal modo, se assume que $C = \{c_1, ..., c_n\}$ o conjuntos de clientes e $W = \{w_1, ..., w_n\}$ a importância que os clientes tem para a empresa. O cliente c_j dar para o requisito r_i o valor (r_i, c_j), em que esse valor é positivo quando o requisito r_i foi escolhido pelo cliente c_j . Dessa forma $score_i$ é dado por (SILVA, 2015):

$$score_i \sum_{j=1}^{m} w_j.value(r_i, c_j)$$

Na formulação MONPR não vai existir apenas uma solução ótima, mas sim um conjunto delas.

Tabela 1: Métodos de priorização de requisitos

Método	Descrição
Árvore de busca binária	Algoritmo utilizado para a busca de informações. Consiste em selecionar um dos requisitos de uma lista e colocá-lo no nó raiz, a partir daí cada requisito restante na lista deve ser comparado com cada nó da árvore em relação a importância. Ao final obtém-se a árvore de requisitos priorizados .
Atribuição numérica	Utiliza uma escala de 1 a 5, os stakeholders devem identificar a qual nível da escala cada requisito corresponde.
Método dos 100 pontos	A cada *stakeholder* são fornecidos 100 pontos que devem ser utilizados em favor dos requisitos mais importantes com base na percepção de cada um.
MoSCoW	Consiste na classificação dos requisitos com base em seu valor de negócio, cada requisitos recebe uma definição de prioridade podendo ser classificada como, *Must Have* (Deve Ter), *Should Have* (Deveria Ter), *Could Have* (Poderia Ter), *Won't Have for Now* (Não Terá por Enquanto.
Theme Screening	Consiste em ordenar as funcionalidades com base em temas de negócio utilizados como fatores de comparação. O valor de cada demanda priorizada é obtido por meio da soma dos sinais negativos e positivos. Ao final, ordenando do maior para o menor resultando na priorização dos requisitos.
Matriz GUT	Classifica os requisitos de acordo com a sua Gravidade, Urgência e Tendência. Cada aspecto da tabela recebe um valor de 1 a 5 que representa o nível de prioridade, após isso é multiplicado os valores das colunas formando o valor que demonstra a sua prioridade.

Fonte: Adaptada de CORDEIRO (2010)

Tabela 2: Exemplo de classificação da Matriz Gut

Gravidade	Urgência	Tendência
1.Sem gravidade	1.Pode esperar	1.Não irá mudar
2.Pouco grave	2.Pouco urgente	2.Irá piorar a longo prazo
3.Grave	3.Urgente, merece atenção no curto prazo	3.Irá piorar a médio prazo
4.Muito grave	4.Muito urgente	4.Irá piorar a curto prazo
5.Extremamente grave	5.Necessidade de ação imediata	5.Irá piorar rapidamente

Fonte: Autoria própria

Tabela 3: Cálculo da porcentagem do fitness (roleta)

Individuo	***Fitness***	**Total**
10101101	15	11,03
11010011	21	15,44
11101010	13	9,56
10010101	75	55,15
10011010	12	8,82
Total	136	100,00

Fonte: Autoria própria

3 MAPEAMENTO SISTEMÁTICO

Esse capítulo refere-se ao Mapeamento Sistemático (MS), executado para concepção desse trabalho,o qual foi organizado da seguinte forma: A Seção 3.1 tem como objetivo detalhar o planejamento do MS. Na seção 3.2 a estratégia de busca e as fontes de busca utilizadas neste MS. Na seção 3.3 são descritos os critérios utilizados para apoiar a seleção dos estudos. A seção 3.4 apresenta a execução do MS, e por fim a seção 3.5 apresenta a síntese e a análise dos dados coletados.

Um MS é uma revisão de estudos primários, que visa identificar, avaliar e interpretar pesquisas de determinadas áreas ou interesses. Ele consiste em um estudo secundário que tem como objetivo classificar informações coletadas de um conjunto de estudos primários e por meio destes resultados, ajudar a identificar lacunas nestas áreas.

O MS presentado neste trabalho segue uma atualização dos resultados apresentados na Revisão Sistemática (RS), realizada por Pitangueira, Maciel e Barros (2015). Essa revisão buscava compreender as ações que estavam sendo tomadas para resolver os problemas relacionados com a seleção e priorização de requisitos utilizando SBSE. Seus estudos foram coletados durante o período de 2001 à 2013, ao todo somaram 39 estudos, sendo eles:

- The next release problem, (BAGNALL; RAYWARD-SMITH; WHITTLEY, 2001)
- Quantitative studies in software release planning under risk and resource constraints, (Ruhe; Greer, 2003)
- Software release planning: an evolutionary and iterative approach, (GREER DES E RUHE, 2004)
- Supporting software release planning decision for evolving systems, (SALIU OMOLADE E RUHE, 2005a)
- Determination of the next release of a software product: an approach using integer linear programming, (AKKER et al., 2005a)

- Software release planning for evolving systems, (SALIU; RUHE, 2005b)
- Flexible release planning using integer linear programming,(AKKER et al., 2005b)
- Search based approaches to component selection and prioritization for the next release problem , (BAKER et al., 2006)
- The multi-objective next release problem, (ZHANG; HARMAN; MANSOURI, 2007)
- Bi-objective release planning for evolving software, (SALIU; RUHE, 2007)
- Integrated requirement selection and scheduling for the release planning of a software product, (LI et al., 2007)
- Software product release planning through optimization and what-if analysis,(AKKER et al., 2008)
- A systematic approach for solving the wicked problem of software release planning, (RUHE et al., 2008)
- Search based data sensitivity analysis applied to requirement engineering, (HARMAN et al., 2009a)
- A study of the multi-objective next release problem,(DURILLO et al., 2009)
- A new approach to the software release planning, (COLARES et al., 2009)
- A search based approach to fairness analysis in requirement assignments to aid negotiation, mediation and decision making,(FINKELSTEIN et al., 2009)
- Search based optimization of requirements interaction management,(ZHANG; HARMAN, 2010)
- Using interactive GA for requirements prioritization, (TONELLA; SUSI; PALMA, 2010)
- An integrated approach for requirement selection and scheduling in software release planning ,(LI et al., 2010)
- Ant colony optimization for the next release problem, (SAGRADO; ÁGUILA; ORELLANA, 2010)
- A hybrid ACO algorithm for the next release problem,(JIANG et al., 2010a)

- Approximate backbone based multilevel algorithm for next release problem, (JIANG; XUAN; REN, 2010b)
- Comparing the performance of metaheuristics for the analysis of multi-stakeholder tradeoffs in requirements optimization , (ZHANG et al., 2011)
- Software next release planning approach through exact optimization, (FREITAS et al., 2011)
- A study of the bi-objective next release problem , (DURILLO et al., 2011)
- An ant colony optimization approach to the software release planning problem with dependent requirements, (SOUZA et al., 2011)
- A fuzzy approach to requirements prioritization, (LIMA et al., 2011)
- Software requirements selection using quantum inspired elitist multi-objective evolutionary algorithm, (KUMARI; SRINIVAS; GUPTA, 2012)
- Solving the large scale next release problem with a backbone-based multilevel algorithm,(XUAN et al., 2012)
- Evolutionary approaches for multi-objective next release problem, (CAI; WEI; HUANG, 2012)
- A multiobjective optimization approach to the software release planning with undefined number of releases and interdependent requirements,(BRASIL et al., 2012)
- Multi-objective optimization approaches to software release time determination,(LI; XIE; NG, 2012)
- Empirical evaluation of search based requirements interaction management,(ZHANG; HARMAN; LIM, 2013)
- Interactive requirements prioritization using a genetic algorithm,(TONELLA; SUSI; PALMA, 2013)
- A scenario-based robust model for the next release problem,(PAIXÃO; SOUZA, 2013a)
- A recoverable robust approach for the next release problem,(PAIXÃO; SOUZA, 2013b)

- Hill climbing and simulated annealing in large scale next release problem, (MAUŠA et al., 2013)
- A hybrid of decomposition and domination based evolutionary algorithm for multi-objective software next release problem, (CAI; WEI, 2013)

3.1 PLANEJAMENTO DO MAPEAMENTO SISTEMÁTICA

Como mencionado anteriormente, um MS é um estudo secundário que visa encontrar evidências de um tema primário a fim de encontrar lacunas a serem estudadas. O intuito desse MS é identificar abordagens relacionadas com técnicas baseadas em buscas para a priorização de requisitos. Para alcançar tais propósitos, esse MS baseou-se no processo proposto por Kitchenham et al. (2010), que é composta por três fases: *(i)* Planejamento; *(ii)* Condução; *(iii)* Análise.

Para a condução desse MS duas questões de pesquisas foram realizadas:

Q1- Quais técnicas baseadas em busca têm sido utilizadas para a priorização de requisitos de software?

Q2- Como os problemas de priorização de requisitos tem sido abordados nos estudos identificados?

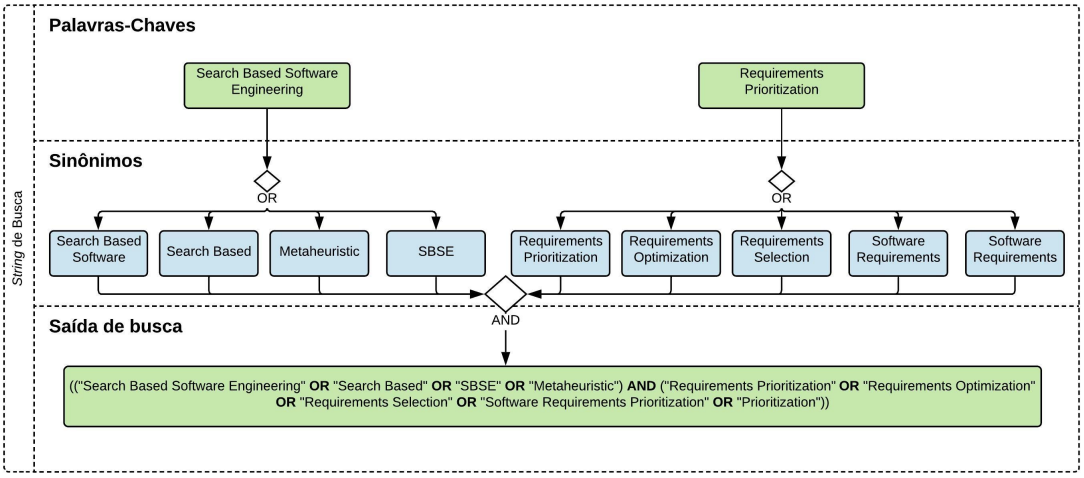
3.2 ESTRATÉGIA DE BUSCA

Após definidas as questões de pesquisas, a próxima etapa foi construir a *string* de busca e definir quais as bibliotecas digitais seriam utilizadas para execução do trabalho.

A *string* de busca foi construída por meio dos seguintes passos: *(i)* reconhecimento das palavras chaves retiradas das questões abordadas no trabalho, sendo elas, Engenharia de Software Baseada em busca e Priorização de Requisitos; *(ii)* identificação dos sinônimos das palavras-chaves; *(iii)* utilização do operador lógico *("OR")* entre os sinônimos identificados; *(iv)* utilização do operador lógico *("AND")* entre as palavras chaves de pesquisa. A Figura 8 descreve o processo de criação da *string* .

Com a *string* de busca definida, o passo seguinte foi identificar as fontes de busca eletrônicas. As fontes utilizadas estão dispostas na Tabela 4.

Figura 8: *String* de busca



Fonte: Autoria própria

Tabela 4: Fontes de busca eletrônicas

Fontes de Busca	Endereço Eletrônico
IEEE Xplore	*www.ieeexplore.com.br*
ACM Digital Library	*www.portal.acm.org*
Compendex	*www.engineeringvillage.com*
Scopus	*www.scopus.com*

Fonte: Autoria própria

3.3 CRITÉRIOS DE SELEÇÃO DOS ESTUDOS

Com o objetivo de identificar estudos relevantes para responder às questões de pesquisa propostas, alguns Critérios de Inclusão **(CI)** e Critérios de Exclusão **(CE)** foram estabelecidos:

- **Critérios de Inclusão:**
  - **CI1:** Estudos primários que apresentam uma abordagem das técnicas existentes para a seleção de requisitos de software e / ou priorização usando SBSE.
  - **CI2:** Estudos primários que apresentam a forma como os problemas de priorização tem sido abordados.
- **Critério de Exclusão:**
  - **CE1:** Estudos sobre priorização ou seleção de requisitos de software, que não usam técnicas de SBSE.

– **CE2:** Estudos primários repetidos

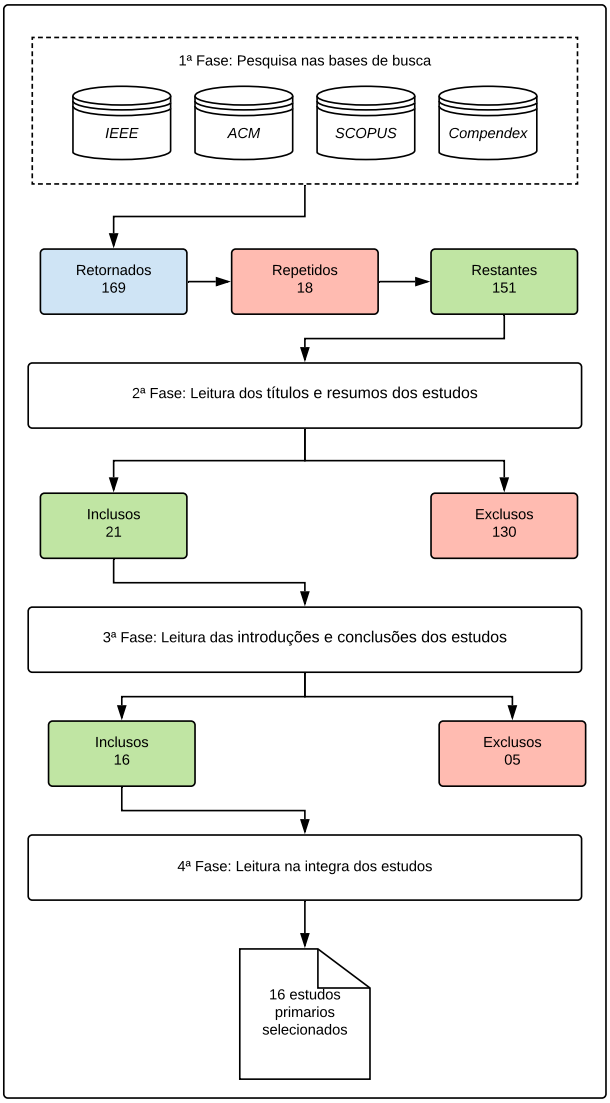
– **CE3:** Estudos com data de publicação inferior ao ano de 2013.

– **CE4:** Estudos primários com resultados incompletos.

3.4 EXECUÇÃO DO MAPEAMENTO SISTEMÁTICO

Com a aplicação da *string* de busca nas 4 bibliotecas eletrônicas, foram coletados os estudos, nos quais conduziram esse MS. A Figura 9 apresenta os passos executados na condução do MS.

Figura 9: Processo de seleção dos estudos



Fonte: Autoria própria

Na 1ª fase foram retornados 16 estudos no *IEEE*, 73 estudos no *Scopus*, 15 estudos

na *ACM* e 65 no *Compendex*, totalizando 169 estudos, desses estudos foram identificados 18 estudos repetidos os quais foram excluídos de acordo com os critérios estabelecidos, restando assim 151 estudos. Na 2ª fase, foram lidos os títulos e os resumos dos estudos, e neles aplicados os critérios de inclusão e exclusão, onde os estudos que não se encaixavam com o objetivo de pesquisa do trabalho foram descartados, restando assim 21 estudos incluso e 165 exclusos. Na 3ª fase foram lidas as introduções e conclusões dos estudos e dos 21 estudos restantes da fase anterior, 16 foram incluídos e 5 descartados. Na 4ª fase, os 16 estudos foram lidos na íntegra e todos foram incluídos, onde os mesmos são apresentados abaixo.

- Software requirements optimization using multi-objective quantum-inspired hybrid differential evolution, (KUMARI; SRINIVAS; GUPTA, 2013).
- Interactive requirements prioritization using a genetic algorithm,(TONELLA; SUSI; PALMA, 2013).
- Requirements Prioritization and Next-Release, (SUREKA, 2014).
- An Integer Linear Programming approach to the single and bi-objective Next Release Problem, (VEERAPEN et al., 2015).
- Differential evolution with Pareto tournament for the multi-objective next release problem, (CHAVES-GONZÁLEZ; PÉREZ-TOLEDANO, 2015).
- Multi-objective ant colony optimization for requirements selection, (SAGRADO; ÁGUILA; ORELLANA, 2015).
- Software requirement optimization using a multiobjective swarm intelligence evolutionary algorithm, (CHAVES-GONZALEZ; PEREZ-TOLEDANO; NAVASA, 2015a).
- Search based risk mitigation planning in project portfolio management , (XIAO et al., 2013).
- Incorporating preferences from multiple stakeholders in software requirements selection an interactive search-based approach, (PITANGUEIRA, 2015a).
- Teaching learning based optimization with Pareto tournament for the multiobjective software requirements selection, (CHAVES-GONZÁLEZ; PÉREZ-TOLEDANO; NAVASA, 2015b).

- Comparing the performance of quantum-inspired evolutionary algorithms for the solution of software requirements selection problem, (KUMARI; SRINIVAS, 2016).
- Minimizing the stakeholder dissatisfaction risk in requirement selection for next release planning , (PITANGUEIRA et al., 2017).
- Search-Based Uncertainty-Wise Requirements Prioritization, (LI et al., 2017).
- Towards multi-decision-maker requirements prioritisation via multi-objective optimisation, (KIFETEW et al., 2017).
- A multi-objective, risk-based approach for selecting software requirements, (AMARAL; ELIAS, 2018).
- Supporting Many-Objective Software Requirements Decision An Exploratory Study on the Next Release Problem, (GENG et al., 2018).

Como pode ser visto na Figura 9, a aplicação da *string* de busca nas bibliotecas eletrônicas retornaram um número considerável de estudos, sendo um total de 169, contudo, ao se aplicar os critérios de inclusão e exclusão restaram apenas 16 estudo classificados, pois a maioria dos estudos excluídos não atendiam às questões de pesquisas propostas neste trabalho.

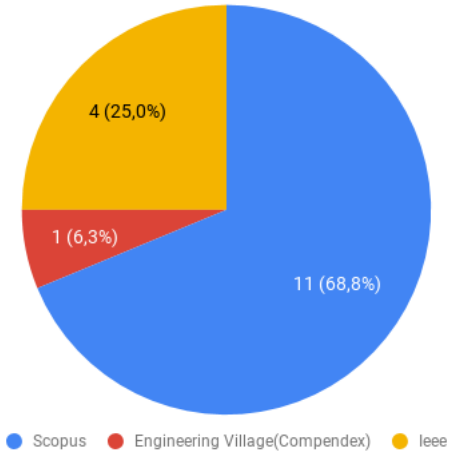
3.5 SÍNTESE E ANÁLISE DOS DADOS

Conforme mostrado anteriormente no processo de seleção de estudos, foram identificados 16 trabalhos relacionados com a priorização de requisitos utilizando a SBSE. Assim a extração dos dados mais relevantes sucedeu-se, a partir deles foram sintetizado as informações para responder as questões de pesquisa levantadas neste trabalho.

Ao realizar a análise dos estudos, pode-se evidenciar que a maior concentração dos estudos, foram dispostos na biblioteca on-line *Scopus*, seguidos do *Compendex*, e *IEEE*. A Figura 10 apresenta os resultados referentes à concentração dos estudos. A partir dos 16 estudos analisados é possível observar que em 2015 o número de estudos publicados foi o dobro de 2013 e 2017. Na Figura 11 e notável que a atualização foi fundamental, uma vez que além dos 6 estudos selecionados por (PITANGUEIRA; MACIEL; BARROS, 2015) em 2013, mais 3 novos estudos foram incluidos neste mesmo ano.

Os estudo selecionados foram analisados de acordo com o objetivo principal deste estudo em relação as técnicas de busca para priorização de requisitos, bem como os

Figura 10: Bibliotecas on-line de busca



Fonte: Autoria própria

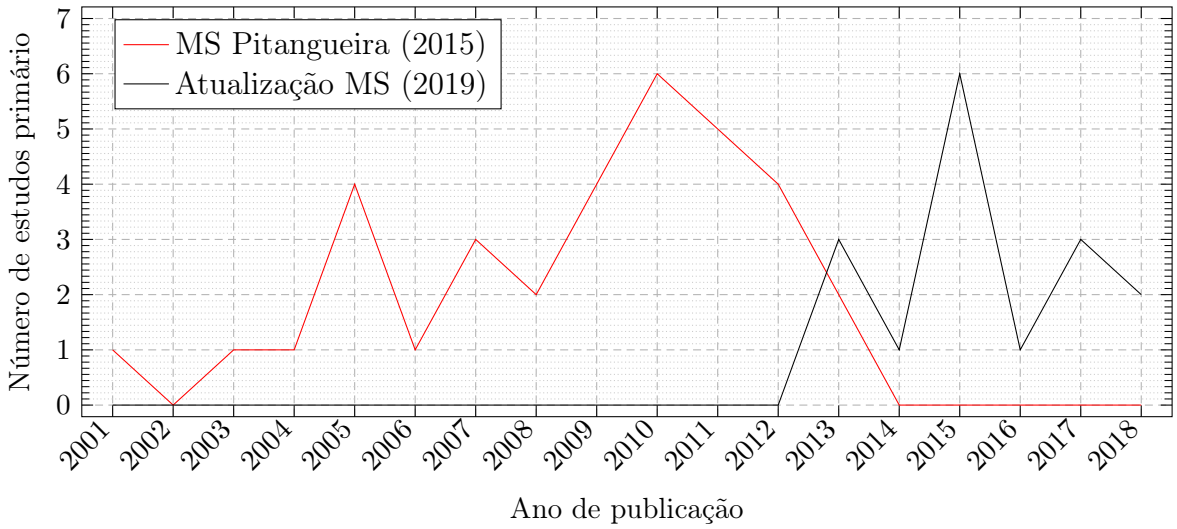
problemas de priorização de requisitos que estão sendo modelados. E necessário destacar que de acordo com os objetivos dos estudos, os mesmos podem ser classificados em 4 categorias:

- **C1 - NRP:** reúne estudos que a apresentam modelagem do problema como um único objetivo a ser otimizado.
- **C2 - MONRP:** contempla estudos que tratam a priorização como multiobjetivos.
- **C3 - NRP e MONRP:** contempla estudos que focam tanto na modelagem único objetivo quanto multiobjetivo.
- **C4 - Outros:** agrupa estudos que não estão relacionados com as categorias mencionadas anteriormente

Os resultados quantitativos desse analise esta apresentado na Figura 12, onde fica evidente que a categoria mais abordada é a MONPR presente em 57% dos estudos.

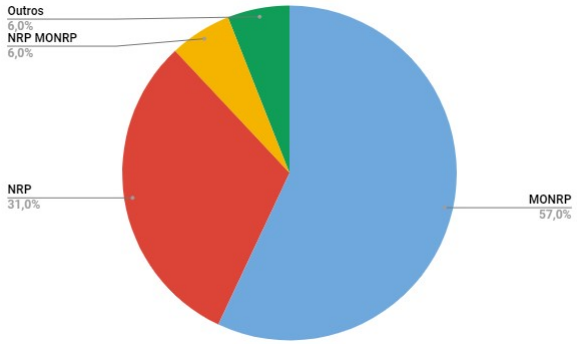
Com essa informação, é obtido a resposta da segunda questão apresentada neste trabalho (Q2) que visa identificar como os problemas de priorização de requisitos têm sido abordados nos estudos. Desta forma pode-se ressaltar o que Pitangueira, Maciel e Barros (2015) descreveu em seu trabalho ao citar o estudo de Zhang, Harman e Mansouri (2007) o qual formulou a proposta do MONRP, por essa abordagem ser *multi-objective* ela torna-se mais eficaz para ES do que a formulação *single-objective*. Por consequência o seu uso se tornou expressivo nos estudos.

Figura 11: Visao geral dos estudos primários incluídos neste MS e por Pitangueira et al. (2015) ao longo dos anos



Fonte: Autoria própria

Figura 12: Formulação dos problemas de priorização de requisitos

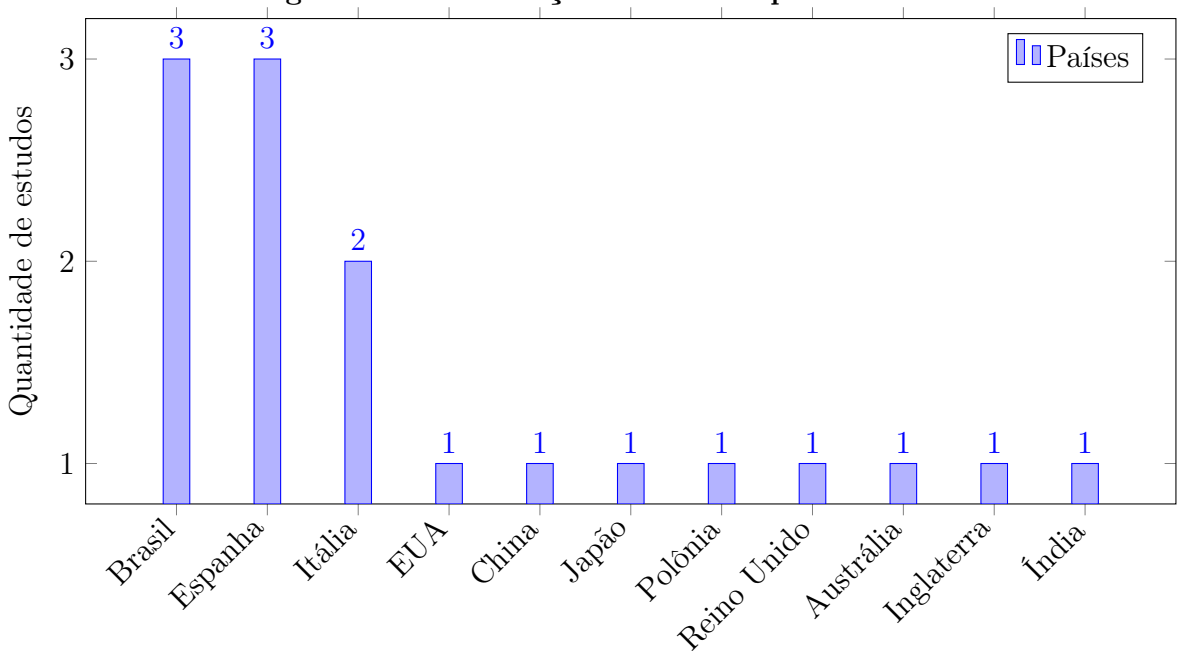


Fonte: Autoria própria

Na Figura 13, é apresentado a distribuição dos estudos por países. O eixo X do gráfico mostra a distribuição dos países onde os trabalhos foram realizados, e o eixo Y representa a quantidade de estudos incluídos em cada país. É possível notar a maior concentração de estudos relacionados à área, vindos do Brasil e Espanha, com 3 estudos cada, em seguida pela Itália, com 2 estudos.

Dos 3 trabalhos produzidos no Brasil, é possível destacar a Universidade Federal da Bahia (UFBa), a qual contribui com a produção de 2 trabalhos, sendo que um dos trabalhos foi produzido no ano de 2015 e o outro no ano de 2017. O terceiro trabalho, e o mais recente de todos, foi elaborado no ano de 2018 na Universidade Federal da Paraíba (UFPB).

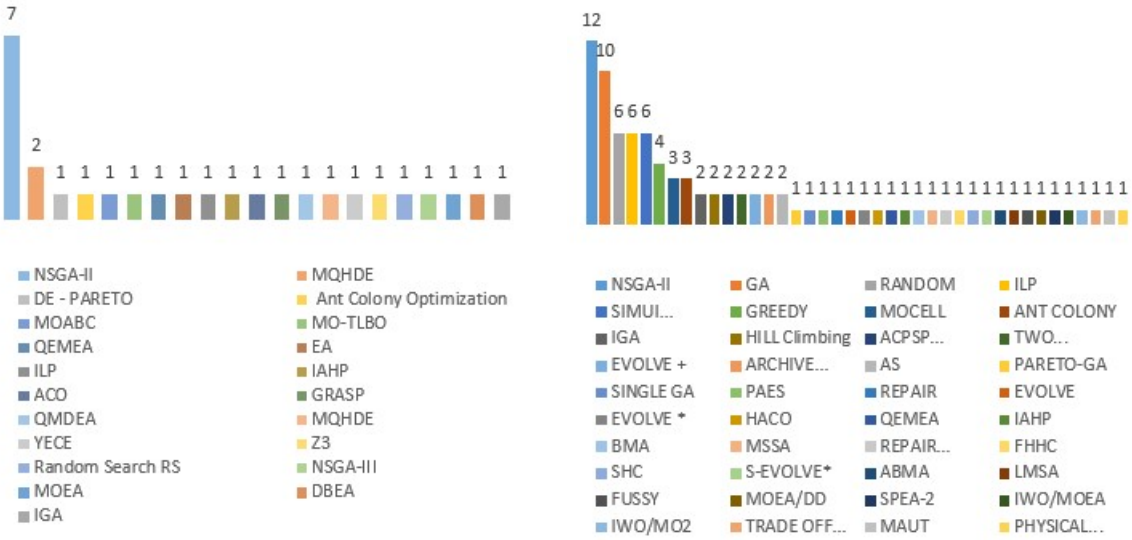
Figura 13: Distribuição de estudos por Paises



Fonte: Autoria própria

Pode-se verificar na Figura 14(a) os resultados obtidos nesse MS que respondem a questão (Q1). O gráfico mostra no seu eixo X a distribuição das técnicas, enquanto o eixo Y representa o número de vezes que essas técnicas foram abordadas nos trabalhos. O algoritmo que mais aparece nos estudos, de acordo com os dados obtidos é o NSGA-II, sendo um total de 7 estudos.

Um fato semelhante ao comparar os resultados obtidos neste MS, Figura 14(a), com os resultados apresentados na RS de Pitangueira, Maciel e Barros (2015), Figura 14(b), foi a presença do algoritmo NSGA-II em primeiro lugar. Na RS de Pitangueira, essa técnica foi identificada em 12 trabalhos, mostrando assim a sua eficiência na aplicação de problemas relacionados à priorização de requisitos. Como foi apresentado na subseção 2.2.1, essa técnica mostra-se mais eficiente por tratar-se da utilização de uma função multi-objetiva, podendo assim abordar vários critérios ao mesmo tempo no momento de priorizar os requisitos. Muitos autores descrevem essa técnica como a solução mais rápida e eficaz para uma pesquisa em espaços grandes e complexos, por consequência essa técnica torna-se a mais estudada.



(a) Técnicas avaliadas neste MS

(b) Técnicas avaliadas na RS de Pitangueira, Maciel e Barros (2015)

Fonte: Adaptada de Pitangueira, Maciel e Barros (2015)

4 PROPOSTA

Este capítulo apresenta abordagem a ser desenvolvida em relação a priorização de requisitos baseada em busca, seguindo o objetivo geral proposto. O capítulo está dividido da seguinte maneira. Seção 4.1 descreve a proposta a ser realizada neste trabalho, sendo uma abordagem para priorização de requisitos baseada em busca. A descrição da abordagem proposta está apresentado na subseção 4.1.1, no qual são descrito as funções de *fitness* propostas para esse trabalho.

4.1 UMA ABORDAGEM PARA PRIORIZAÇÃO DE REQUISITOS BASEADA EM BUSCA

Segundo Gilb e Maier (2005) o conceito de prioridade é o direito relativo que um requisito tem de utilizar recursos limitados ou escassos, sendo que tais recursos podem ser compreendidos como tempo, esforço humano, recursos financeiros, espaço ou qualquer outro tipo.

A falta de padronização ou diculdade na seleção de uma métrica para priorizar requisitos durante o processo faz com que as empresas optem pelo mais conveniente para o produto, cliente ou momento, assim, induzindo as equipes a realizar incorretamente a priorização na maioria dos casos (MACHADO, 2017).

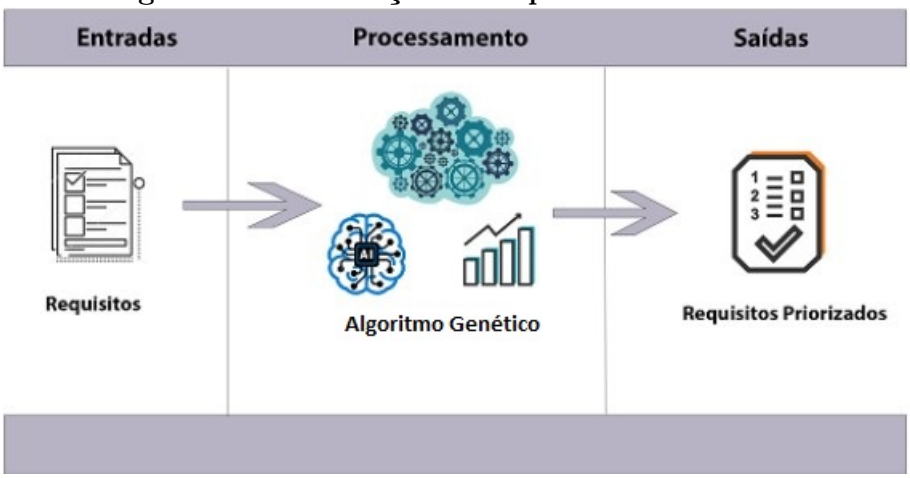
Mesmo com a utilização das metodologias ágeis ainda são encontrados problemas durante o processo de desenvolvimento de um software. A incorreta priorização de requisitos prejudica o software e seus usuários, entregar funcionalidades que não serão utilizadas e não agregarão valor podem causar prejuízos e insatisfação ao produto. Tendo em vista tal circunstância, surge a necessidade de explorar alternativas para sanar tais problemas identificados na área de ER.

Dessa forma, a utilização de métodos de priorização de requisitos ganham o seu espaço, auxiliando nas tomadas de decisões. Assim, a proposta deste trabalho emprega os conceitos de priorização de requisitos de software baseado em busca, que consiste em

solucionar problemas de priorização utilizando técnicas baseadas em busca guiada por uma determinada função de avaliação projetada para a solução do problema proposto.

Conforme apresentado na Figura 14, a abordagem proposta inicia a partir de um conjunto de requisitos não ordenados obtidos de uma base de dados. Esses requisitos servirão de entrada para a técnica de busca, a qual utiliza regras e estratégias provenientes do AG para identificar os melhores requisitos a serem priorizados. A partir do processamento o algoritmo realizará a priorização dos requisitos, levando em consideração as características de cada função de avaliação proposta.

Figura 14: Priorização de requisitos usando AG



Fonte: Autoria própria

Para que a priorização seja bem sucedida o conjunto de requisitos passam por uma função de *fitness* para avaliar a qualidade dos mesmos obtendo assim um conjunto de requisitos aptos a serem priorizados, caso os mesmos sejam definidos como o melhor conjunto. Para isso, foram propostas 4 funções de *fitness* para a priorização, baseadas em técnicas tradicionais e novas perspectivas: 1) Matriz GUT-AG (GUT-AG) ; 2) *Theme screening-AG* (TS-AG) ; 3) Tempo *versus* Satisfação (TvS-AG); 4) Menor Tempo (MT-AG). Além das funções propostas são utilizadas técnicas tradicionais de priorização (Matriz GUT e *Theme Screening*), a fim de comparar suas eficiências.

Vale ressaltar que as funções que estão sendo propostas neste estudo basearam-se nas informações coletadas do *Survey* realizado para esse estudo. O *Survey* foi aplicado para um público restrito de profissionais, sendo eles gerentes de projetos e pessoas que trabalham com a análise de requisitos. Ao todo participaram 15 profissionais os quais responderam um questionário e com base nas respostas pode-se obter as informações necessárias para a condução dos experimentos. Para mais detalhes o *Survey* pode ser

consultado no Apêndice A.

Com os resultados obtidos no *Survey* pode se perceber que 78,6% dos participantes possuem um conhecimento médio em relação a ER. Além disso, ficou evidente que mais da metade deles (53,3%) não utiliza nenhuma técnica de priorização, indicando o uso da seleção informal dos requisitos, esses resultados demonstram a necessidade da utilização de técnicas automatizadas de priorização.

Outro ponto a se destacar é a importância aplicada as dependências dos requisitos, uma vez que, não tratadas elas podem impedir a execução do requisito. De acordo com os dados obtidos 40% considera o nível de importância alto para esse quesito e 26,7% considera extremamente alto. Outra questão que vale destacar é a importância atribuída aos prazos de entrega, no qual 53,3% dos participantes indicaram que esse fator é de suma importância.

Com tudo, as informações obtidas serviram de indicadores para a construção da abordagem de priorização, alguns pontos como a importância dos prazos de entrega e a consideração das dependências dos requisitos foram tratadas na abordagem e serão apresentados nos próximos capítulos.

4.1.1 FUNÇÕES DE *FITNESS*: GUT-AG, TS-AG, TVS-AG E MT-AG

- ***Matriz GUT-AG (GUT-AG)***

A função GUT-AG é uma função derivada da técnica tradicional de priorização Matriz GUT, essa técnica consiste em quatro diferentes critérios de avaliação, sendo eles:

I. Gravidade (G): indica o impacto que o problema gerará nos envolvidos.

II. Urgência (U): mostra a urgência do problema em ser resolvido.

III. Tempo (T): apresenta a probabilidade (ou o potencial) que o problema tem de crescer com o passar do tempo.

IV. Dependência (Dep): indica a quantidade de dependências que o requisito possui.

Para essa função de *fitness* assume se um conjunto de n requisitos, no qual $R = \{r_1, r_2, ..., r_n\}$, esse conjunto está contido em um outro conjunto maior de requisitos que representa a base de dados ($R_i \subseteq R$), a função GUT-AG consiste no produto dos critérios da técnica GUT tradicional, isto é, gravidade(G), urgência(U), e tendência(T).

Esses critérios são multiplicados em seguida o valor obtido é penalizado com o valor das dependências do conjunto de requisitos (Dep), tendo assim o valor de *fitness*, A sua formulação é apresentada na seguinte forma:

$$Maximize \sum_{i=1}^{n} (G_i.U_i.T_i) - Dep$$

(3)

Por fim, o objetivo dessa função é encontrar o conjunto de requisitos que maximize os critérios GUT com menor dependências de requisitos, ou seja, quanto maior o número de dependências um requisito possui, pior é o grupo de requisitos priorizados.

- ***Theme screening*-AG (TS-AG)**

A função TS-AG proposta, é derivada da função *Theme screening*, ela utiliza os princípios dessa técnica para realizar o cálculo da função de *fitness*. Sua formulação é dada da seguinte maneira.

$$Maximize \sum_{i=1}^{n} (Pr_i) - Dep \tag{4}$$

O conjunto de n requisitos é avaliado pela função no qual é somado o valor de sua prioridades (Pr) em seguida o valor é subtraído pela quantidade de dependências (Dep) de cada r_i obtendo então o valor de *fitness*. Sendo assim, o objetivo da função é maximizar o valor de prioridade obtido por meio das variáveis da técnica *Theme screening* para encontrar o melhor conjunto de requisitos a ser priorizado.

- ***Tempo versus Satisfação (TvS - AG)***

A função elaborada para essa abordagem, busca maximizar a satisfação dos cliente selecionando n requisitos que possuem um menor nível de satisfação e um maior tempo de espera na fila de desenvolvimento, isto é, o requisito é importante para o cliente, porém por alguma razão técnica pode ter sido julgado como pouco relevante, assim, podendo gerar uma insatisfação a ele. Dessa forma, a função TvS - AG atua por meio de um cálculo

de grandezas inversamente proporcionais. A função irá avaliar os conjuntos que possuem o maior tempo de espera (Tp), maior prioridade (Pr), maior importância do cliente (I) e menor Satisfação (Sat).

Esse cálculo é obtido pela somatória dos valores Tp e Pr, em sequência esse valor resultante é elevado pelo valor de I do cliente, o resultado obtido e posteriormente dividido pelo valor Sat. O valor da somatória dos critérios e subtraído pela quantidade de dependências, Dep, de cada r_i resultando assim no valor da função de *fitness* 5:

$$Maximize \sum_{i=1}^{n} \frac{(Tp_i + Pr_i)^{I_i}}{Sat} - Dep \tag{5}$$

- ***Menor Tempo (MT - AG)***

Para essa função de *fitness* são selecionados n requisitos que tenham menor tempo de execução (Tr), maior importância do cliente (I), maior prioridade do requisito (Pr) e não ultrapassem o total de horas trabalhadas pela equipe de desenvolvimento (H).

Para o cálculo da função são somados os critérios Pr e I de cada r_i em seguida o valor obtido e dividido pelo valor de Tr, resultando no valor da função. O valor obtido pelo cálculo passa por uma avaliação de restrição, caso o valor de Tr do conjunto de requisitos seja superior ao valor de H, o mesmo é penalizado definindo que o conjunto de requisitos não é apto para a priorização. Essa penalização faz com que o indivíduo dificilmente passe para um próxima geração avaliado como excelente indivíduo, porém ainda sim, pode ter chance de cruzamento.

$$Maximize \sum_{i=1}^{n} \frac{Pr_i + I_i}{Tr_i} \tag{6}$$

Tal que, o resultado (*fvalue*) da função de *fitness* deve ser menor que H, ou seja $fvalue \leq H$. Computando a função utilizando o conceito de grandezas inversamente proporcionais o objetivo da função é maximizar os resultados a fim de encontrar o melhor conjunto de requisitos, em outras palavras, quanto menor o valor de Tr maior será o valor de *fvalue* e mais apto será o individuo.

4.2 FERRAMENTA COMPUTACIONAL E AMBIENTE

Para a construção do algoritmo, foi utilizado a linguagem de programação Python. O Python é uma linguagem de programação de alto nível, interpretada e orientada abjetos sendo muito eficiente e simples, mas eficaz.

Além disso, o Python possui uma grande quantidade de bibliotecas auxiliares ("Toolboxes") que otimizam o tempo gasto para realizar tarefas, oferecendo diversas funções já definidas, poupando o tempo de criá-las. Uma das bibliotecas utilizadas na construção do AG foi a biblioteca DEAP (FORTIN et al., 2012), essa biblioteca nos forneceram todos os componentes e exemplos necessários para a construção do algoritmo. Para a execução deste trabalho, utilizou-se um computador com sistema operacional Windows 10, processador: Intel(R). Core(TM) I3-7020u CPU 2.30GHz, Memória(RAM) 4,00 GB, Tipo de Sistema: Sistema Operacional de 64 bits, com o processador com base de 64x.

4.2.1 PARÂMETROS UTILIZADOS NO AG

Para seu funcionamento, o AG necessita da configuração de alguns paramentos, como mostra a Figura 15. Esses parâmetros são representados como:

Figura 15: Parâmetros de configuração do AG

```
# Numero de Requisitos a ser priorizados
numrequisitos = 5
# Populacao Total
populacao = 20
# Probabilidade De Um Individuo Sofrer Mutacao
probmut = 0.5  # 0.1
# Probabilidade De Dois Individuos Cruzarem
probcross = 0.5 # 0.3
# Quantidade maxima de Geracoes
numgeracoes = 200  # 300
# Melhor resultado possivel da funcao de avaliacao
resulfunc = numrequisitos * 125
```

Fonte: Autoria Própria

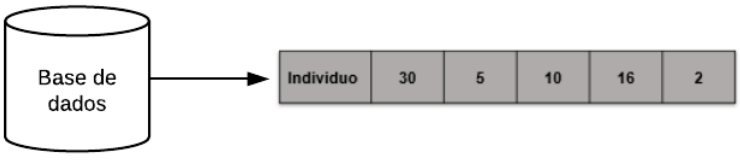
- **numrequisitos:** trata-se do número de requisitos mais prioritários, que se quer encontrar.
- **qtderequisitos:** indica a quantidade de requisitos presente na base de dados.

- **populacao:** representa a quantidade de indivíduos presentes em uma geração.
- **probmut:** define a probabilidade de um indivíduo sofrer mutação, em porcentagem.
- **probcross:** define a probabilidade de um indivíduo cruzar com outro indivíduo, em porcentagem.
- **numgeracoes:** indica o limite de gerações que o algoritmo pode gerar.
- **resultfunc:** representa o resultado da função de avaliação, multiplicando o número de requisitos com a pontuação máxima de prioridade que um requisito pode ter.

Para a condução dos experimentos foram utilizadas duas configurações, as quais estão representadas na Tabela, e serão apresentadas mais adiante na condução dos experimentos.

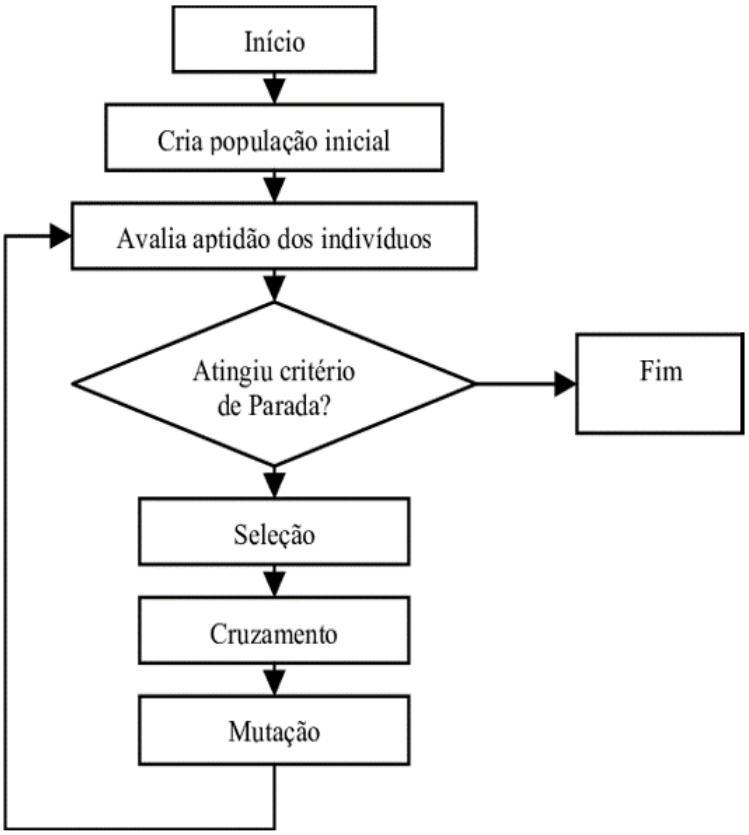
Para representar o individuo em nosso algoritmo são selecionado cinco requisitos da base de dados de modo aleatório, cada requisito esta identificado pelo seu código de identificação, Figura 16, após selecionado o indivíduo, o mesmo se submete a avaliação da função fitness, caso o individuo seja considerado um bom candidato ele é armazenado sendo considerado uma solução, no entanto, caso ele não seja apto o mesmo passa pelos operadores genético de seleção, cruzamento e mutação, gerando assim um novo individuo que passa pelo mesmo processo de avaliação, este ciclo é repetido descartando os indivíduos menos aptos e buscando os que possuem maior aptidão até se chegar ao critério de parada. A Figura 17 representar o diagrama funcional do AG.

Figura 16: Exemplo de individuo proveniente da base de dados



Fonte: Autoria Própria

Figura 17: Diagrama funcional do AG



Fonte: Autoria Própria

5 AVALIAÇÃO EXPERIMENTAL

No capítulo anterior foi proposto uma abordagem para a priorização de requisitos utilizando AG para a seleção do melhor conjunto de requisitos dentre todos em uma base de dados. Para validar a abordagem, neste capítulo é apresentado a avaliação experimental realizada comparando as técnicas propostas, onde são apresentadas as diferentes funções *fitness*. Este estudo avalia a eficácia da abordagem proposta para a priorização de requisitos. Os resultados foram comparados e avaliados, mostrando quais técnicas se sobressaíram.

5.1 PLANEJAMENTO

O presente trabalho tem como objetivo analisar os experimentos realizados utilizando as quatro funções de *fitness* (GUT-AG, TS-AG, TvS-AG e MT-AG) as quais foram denominadas de F_1,F_2,F_3,F_4 em relação às técnicas tradicionais de priorização. É importante ressaltar que somente as funções GUT-AG, TS-AG podem ser comparadas em relação as técnicas tradicionais, uma vez que elas foram baseadas na matriz GUT e *Theme screening*. Assim, o propósito desse estudo experimental é analisar os resultados alcançados por cada uma das funções de *fitness* verificando qual função é a mais eficiente. Para isso, são consideradas as seguintes Questões de Pesquisas (*QPs*):

QPs1. Quão eficiente é a abordagem proposta para priorização de requisitos utilizando algoritmo genético?

Para responder a *QPs1*, foram analisadas as 4 funções de *fitness*, as quais foram executadas 10 vezes cada e calculado a média (μF_i) das execuções da melhor configuração, com o intuito de reduzir as chances dos resultados serem encontrados ao acaso. Para esta questão as seguintes hipóteses foram definidas:

- $\boldsymbol{H1_0}$: Não existe diferença na abordagem em relação aos valores de *fitness* das

funções F_1, F_2, F_3, F_4.

$$H1_0 : \mu F_1 = \mu F_2 = \mu F_3 = \mu F_4 \tag{7}$$

- $\boldsymbol{H1_1}$: Existe diferença na abordagem em relação aos valores de *fitness* das funções F_1, F_2, F_3, F_4.

$$H1_1 : \mu F_1 \neq \mu F_2 \neq \mu F_3 \neq \mu F_4 \tag{8}$$

QPs2. Quão efetivo é a abordagem proposta para priorização de requisitos utilizando algoritmo genético em relação ao tempo.

A eficácia refere-se ao tempo médio μT_i a partir das 10 execuções que se leva para gerar os melhores grupos de requisitos. O tempo é computado para cada função de *fitness* utilizando as duas configurações apresentadas na Tabela 7, e inclui todo o tempo computacional usado para gerar os grupos de requisitos. A execução dos experimentos foi realizado em uma máquina Core I3 7ª Geração, 4 GB de memória, 1TB de HD, com o sistema operacional Windows 10 e o tempo foi calculado pela Expressão 9.

$$T = Time(Fim)\text{-}Time(inicio) \tag{9}$$

Onde $time(inicio)$ refere-se ao tempo de início da execução do algoritmo e $time(fim)$ o tempo final da execução do algoritmo. O tempo é gravado usando uma função nativa da linguagem de programação *Python* denominada *time* a qual captura o tempo de início e fim de processamento computacional o que nos permite realizar o cálculo da expressão citada acima. Para esta questão as seguintes hipóteses foram definidas:

- $\boldsymbol{H2_0}$: Não existe diferença de tempo na abordagem de priorização de requisito nas funções de *fitness* F_1,F_2, F_3, F_4.

$$H1_0 : \mu T_1 = \mu T_2 = \mu T_3 = \mu T_4 \tag{10}$$

- $\boldsymbol{H2_1}$: Existe diferença de tempo na abordagem de priorização de requisito nas funções de *fitness* F_1,F_2, F_3, F_4.

$$H1_1 : \mu T_1 \neq \mu T_2 \neq \mu T_3 \neq \mu T_4 \tag{11}$$

5.2 DEFINIÇÃO DOS OBJETIVOS

O objetivo do estudo é analisar as variáveis da abordagem proposta no que diz a respeito ao seu tempo e eficácia em comparação entre elas e com as técnicas tradicionais. Os objetivos específicos são classificados como, (Basili; Weiss, 1984):

- **Objetivo de estudo:** o objetivo é a abordagem proposta neste trabalho.
- **Propósito:** o propósito deste experimento é avaliar a abordagem proposta considerando a qualidade dos grupos de requisitos gerados comparando sua eficiência em relação às técnicas tradicionais de priorização.
- **Perspectiva:** este experimento é executado do ponto de vista de um pesquisador.
- **Qualitativo**: o principal foco da investigação é a eficácia da abordagem em encontrar os melhores requisitos para a priorização.
- **Contexto:** o experimento foi realizado utilizando uma base de requisitos extraídos de tarefas realizadas em uma empresa de software, localizada na cidade de Quedas do Iguaçu-PR.

5.3 DEFINIÇÕES DAS VARIÁVEIS

Para os experimentos foram comparados seis variáveis independentes: (i) função de *fitness* F_1; (ii) função de *fitness* F_2; (iii) função de *fitness* F_3; (iv) função de *fitness* F_4; (v) GUT por ranqueamento; (vi) TS por ranqueamento. As variáveis dependentes coletadas a partir das funções *fitness* e abordagens são os conjuntos de requisitos gerados e as métricas que deles podem se extrair: o melhor grupo de requisitos, o valor de *fitness* dos melhores requisitos e tempo de geração.

5.4 SELEÇÃO DOS SUJEITOS

Para os experimentos foram utilizadas duas bases de dados contendo requisitos de software para a priorização. A primeira base contém 41 requisitos e a segunda base contém 213 requisitos, totalizando 254 requisitos, como mostra o exemplo da Tabela 5. A base contém os seguintes campos:

- **Requisitos:** representa a identificação dos requisitos dentro da base.

- **Tarefa:** representa a tarefa a qual os requisitos pertencem.
- **Tempo:** exibe o tempo estimado para o requisito ser contemplado.
- **Prioridade:** representa-se o nível de prioridade do requisito, esse valor vai de 1 a 5, onde 1 representa uma prioridade baixa e 5 uma alta prioridade.
- **Premium:** representa a importância do cliente. Caso o cliente tem um alto valor agregado para empresa o mesmo recebe o título de cliente Premium, esse campo possui dois valores 0 e 1, onde o 0 indica que o cliente não é Premium e 1 indica que ele é Premium.
- **Dependência:** representa a quantidade de dependências de outros requisitos que impedem esse requisito de ser contemplado.
- **Satisfação:** mostra o nível de satisfação do cliente em relação aos serviços da empresa, o seu valor e apresentado 1 a 5, onde o valor 1 representa uma baixa satisfação e o valor 5 uma alta satisfação.

Tabela 5: Base de requisitos

REQUISITO	TAREFA	TEMPO	PRIORIDADE	PREMIUM	GRAU DE DEPENDÊNCIA	SATISFAÇÃO
1	81146	14:50:26	3	1	0	4.5
2	82187	00:11:30	3	0	0	4.5
3	82187	00:23:05	3	0	1	4.5

Fonte: Autoria própria

E importante ressaltar que a base utilizada para os experimentos é referente a requisitos de manutenção e evolução de software, alguns exemplo podem ser vistos na Tabela 6.

Tabela 6: Exemplo de requisitos de evolução e manutenção

REQUISITO	DESCRIÇÃO
6	Criar nova permissão na tela de cadastro » permissões de supervisores
7	Adicionar campo “valor moeda” na tela de cotação de moedas;
18	Inserir uma validação no patch para que ao importar os xml em massa
12	adicionar filtro de dias na guia “histórico preços”
20	Otimizar relatório de vendas melhorando o tempo de consulta

Fonte: Autoria Própria

5.5 CONDUÇÃO

O procedimento realizado durante a execução do experimento consiste nas seguintes etapas: 1) geração dos grupos de requisitos, utilizando as funções propostas (F_1,F_2,F_3 e F_4), além do uso de técnicas tradicionais; 2) coleta dos resultados gerados pelas funções *fitness*, na etapa anterior; 3) comparação dos resultados das abordagens propostas com AG e técnicas tradicionais utilizando o simples ranqueamento. Antes desse experimento foram realizados testes preliminares para identificar uma parametrização adequada para o algortimo genético, após esses testes foram definidos duas configurações que poderiam alcançar bons resultados e elas são apresentadas na Tabela 7. Onde P é o tamanho da população, MR, taxa de mutação, G é o número de gerações e por fim TC, taxa de cruzamento.

Tabela 7: Configurações de experimentos

Funções *Fitness*	Configurações	Parâmetros			
		P	MR	G	TC
F_1, F_2, F_3, F_4	Cf_1	10	0.3	100	0.3
	Cf_2	20	0.5	200	0.5

Fonte: Autoria própria

As etapa do experimento podem ser descritas como:

1. **Seleção dos indivíduos:** foram selecionados 254 requisitos extraídos de uma base real, provida de uma empresa de software, tal que esse conjunto de requisitos é representado como, R = $\{r_1, r_2, r_3, .., r_n\}$.

2. **Geração dos grupos de requisitos:** o conjunto de requisitos (C_i), o qual $C_i \subseteq R$ é gerado por meio do AG, onde o C_i representa os melhores requisitos a serem priorizados no momento. Para esse experimento foi definido $i = 5$ para melhor análise dos requisitos priorizados e o tamanho da base de dados. No entanto a abordagem é escalável para quaisquer quantidade.

3. **Avaliação de aptidão do conjunto de requisito:** cada conjunto C_i gerado passa por uma avaliação de *fitness* , onde são definidos os melhores resultados . Essa etapa está dividida em 4 sub-etapas:

   (a) ***fitness* F_1 :** cada indivíduo do conjunto possui seu valor de gravidade (G), urgência (U) e tendência (T) somados, o valor obtido pela função de *fitness*

F_1 é penalizado com o valor total das dependências do requisito (Dep), caso o mesmo possua.

(b) ***fitness* F_2:**cada indivíduo do conjunto tem apenas seu valor de prioridade (Pr) somados, o valor obtido pela função *fitness* F_2 é penalizado com o valor total das dependências de requisitos (Dep), caso o mesmo possua.

(c) ***fitness* F_3:** cada indivíduo do conjunto tem seu valor de Tempo (Tp) somado com o valor da prioridade (Pr). O valor dessa soma é elevado pela importância do cliente (I) ao qual esse requisito pertence. Em seguida esse valor é dividido pela satisfação do cliente. O valor obtido pela função *fitness* F_3 é penalizado com o valor total das dependências dos requisitos (Dep), caso o conjunto de indivíduos possua.

(d) ***fitness* F_4:** cada indivíduo do conjunto tem seu valor de prioridade (Pr) somado com a importância do cliente (I) ao qual esse requisito pertence, o resultado dessa soma é dividido pelo tempo estimado de entrega do requisito (Tr). O valor resultante passa por uma avaliação onde o Tr deve ser menor que o total de horas trabalhadas pela equipe de desenvolvimento (H) caso o valor seja superior, o mesmo sofre uma penalização em seu valor.

4. **Cálculo do tempo da abordagem:** o tempo total (em segundos) para a abordagem encontrar o melhor conjunto de requisitos a ser priorizado.

5. **Comparação entre as abordagens utilizando AG e as técnicas tradicionais:** O valor de *fitness* do grupo de requisito de cada abordagem foi comparado, a fim de verificar qual abordagem é mais eficaz.

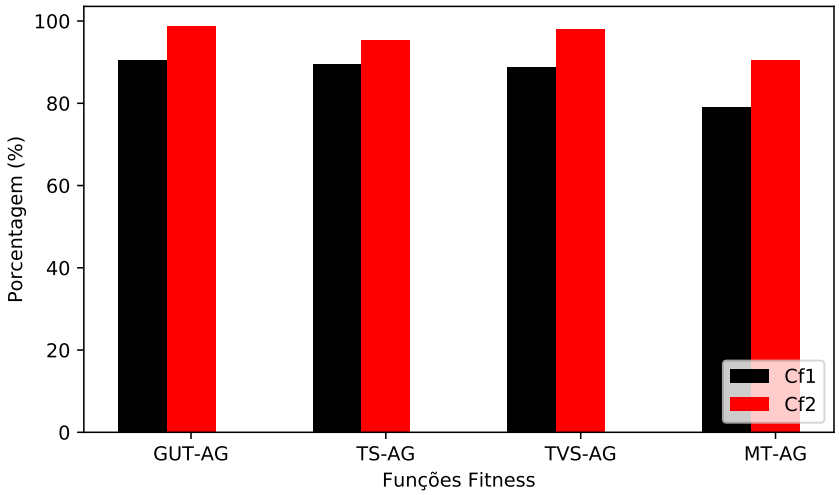
5.6 ANÁLISE DOS DADOS

Esta seção descreve a análise descritiva dos dados coletados durante a execução dos experimentos. A primeira análise a ser realizada está relacionada a eficácia da abordagem.

- **Melhor função de *fitness (QP1)*:**

Essa questão considera a eficácia da abordagem proposta referente à geração dos conjuntos de requisitos priorizados, a Figura 18 mostra o valor de *fitness* médio alcançado pelas funções (F_1, F_2, F_3, F_4) em cada uma das configurações (Cf_1, Cf_2) executadas no experimento.

Figura 18: Valor médio de *fitness* alcançado em cada função



Fonte: Autoria própria

Vale destacar que a abordagem passou por uma sequência de 10 execuções, onde cada resultado obtido foi armazenado e em seguida, suas médias aritméticas foram calculadas para posteriormente os dados serem analisados.

O eixo X na Figura 18 representa as funções de *fitness* propostas usando as configurações definidas, e o eixo Y apresenta o valor de *fitness* em porcentagem que cada função alcançou, indo de 0 a 100. Em geral, as eficiências se mostram similares nas funções F_1, F_2, F_3. No entanto é possível perceber que a função F_4 se mostrou um pouco abaixo da média. Mais detalhes podem ser vistos na Tabela 8, o qual apresenta os valores de *fitness* obtidos em cada configuração. Isso pode ter ocorrido devido ela necessitar combinar mais variáveis para obter um melhor valor.

Tabela 8: Valor de *fitness* médio alcançado por cada função proposta

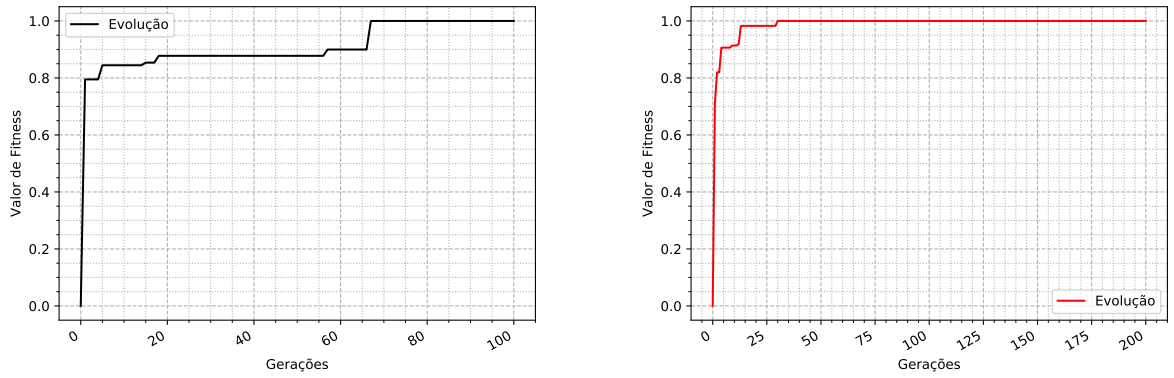
Função *Fitness*	Escore de *Fitness* (%)	
	Cf_1	Cf_2
F_1	90,51%	98,63%
F_2	89,56%	95,40%
F_3	88,81%	97,99%
F_4	79.14%	90.47%

Fonte: Autoria própria

Para uma análise mais detalhada, a seguir é apresentado o resultado obtido por cada função. A primeira abordagem com a função *fitness* F_1 é apresentada na Figura

19, no gráfico pode se avaliar a evolução dos valores de *fitness* com o passar da gerações. No eixo X do gráficos temos as gerações do AG, já no eixo Y obtemos a evolução dos valores de *fitness*. Vale ressaltar que o gráfico de evolução foi construído a partir do valor médio de *fitness* encontrados ao longo das gerações. É possível notar também, que para Cf_1, em torno da geração 67 foi encontrada a solução ótima, enquanto que para Cf_2 em torno da geração 30 foi identificada a solução ótima e convergiu mais rapidamente. Porém necessitou de mais 10 indivíduos dentro da população e consequentemente mais recursos computacionais.

Figura 19: Comparação de resultados da função *fitness* F_1



(a) Função de *fitness* F_1 usando a configuração Cf_1 (b) Função de *fitness* F_1 usando a configuração Cf_2

Fonte: Autoria própria

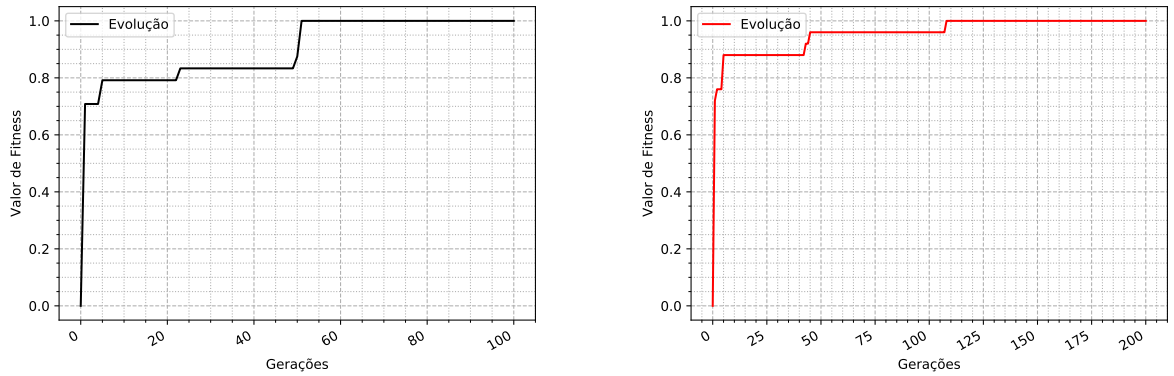
Para abordagem utilizando a função de *fitness* F_2, verificou-se que seguem a mesma tendência, porém com uma evolução mais lenta a partir de *fitness* ≈0.7, como mostra a Figura 20. No eixo X do gráficos temos as gerações do AG, e o eixo Y corresponde aos valores de *fitness*.

A terceira e quarta abordagem, F_3, F_4, respectivamente segue a mesma apresentação gráficas demonstrada anteriormente e podem ser verificadas nas Figuras 21 e 22.

Os melhores resultados obtidos nas configurações apresentadas estão dispostas na Tabela 9. É possível observar que o melhor conjunto de requisitos encontrado na função F_1 foi gerada na 30º geração utilizando a Cf_2, já as funções F_2 e F_3 obtiveram o melhor valor de *fitness* na 51º e 53º gerações, ambas usando a Cf_1, a função F_4 foi a que obteve o resultado mais discrepante, alcançando seu melhor valor de *fitness* na 75º geração.

Para o estudo comparativo foram analisados os conjuntos de requisitos obtidos nas funções F_1 e F_2, em sequência esses conjuntos foram comparados com as técnicas

Figura 20: Comparação de resultados da função *fitness* F_2



(a) Função de *fitness* F_2 usando a configuração Cf_1 (b) Função de *fitness* F_2 usando a configuração Cf_2

Fonte: Autoria própria

Tabela 9: Geração de melhor resultado *Fitness*

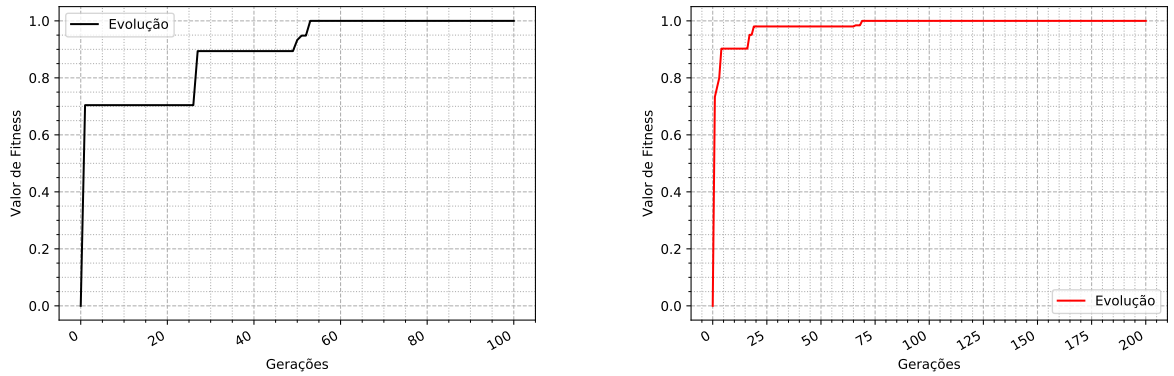
Função *fitness*	Número da geração de melhor resultado *Fitness*	
	Cf_1	Cf_2
F_1	67	30
F_2	51	108
F_3	53	69
F_4	75	137

Fonte: Autoria própria

tradicionais de priorização Matriz Gut e *Theme screening*, que nessa seção foram denominadas como *GUT* e *TS*. As Tabelas 10, 11, 12 e 13 sumarizam os resultados dos requisitos priorizados por cada função. A primeira coluna apresenta a identificação do requisito por meio de um código único, em seguida, na segunda coluna identifica-se as dependências do requisito. Logo após na terceira, quarta e quinta colunas são representados os valores da Gravidade, Urgência e Tendência dos requisitos, respectivamente.

Por meio de uma análise empírica dos requisitos priorizados utilizando as técnicas tradicionais e as propostas, foi possível verificar que as funções F_1 e F_2 (Tabelas 10 e 12) apresentaram resultados diferenciados em comparação os resultados das técnicas *GUT* e *TS* (Tabelas 11 e 13) isso é esperado devido a consideração das dependências de requisitos que as funções F_1 e F_2 compreendem ao selecionar o melhor conjunto de requisitos, diferente das técnicas tradicionais que geram apenas um ranqueamento dos requisitos. Vale lembrar, que para esse experimento foi considerado um conjunto de 5

Figura 21: Comparação de resultados da função *fitness* F_3



(a) Função de *fitness* F_3 usando a configuração Cf_1 (b) Função de *fitness* F_3 usando a configuração Cf_2

Fonte: Autoria própria

requisitos para a priorização, ou seja, a abordagem proposta retorna um conjunto de 5 requisitos que melhor se adapta a função *fitness*

Tabela 10: Requisitos da função F_1

Requisitos	Dep	G	U	T	Total
6	0	5	5	3	75
0	0	5	2	5	50
15	0	3	4	2	28
1	0	3	3	4	36
20	0	4	4	1	16

Tabela 11: Requisitos da Matriz GUT

Requisitos	Dep	G	U	T	Total
30	1	5	5	4	100
6	0	5	5	3	75
5	2	5	5	3	75
0	0	5	2	5	50
1	0	3	3	4	24

Fonte: Autoria própria

Tabela 12: Requisitos da função F_2

Requisito	Prioridade	Grau de dependência
29	5	0
7	5	0
21	5	0
6	5	0
37	5	0

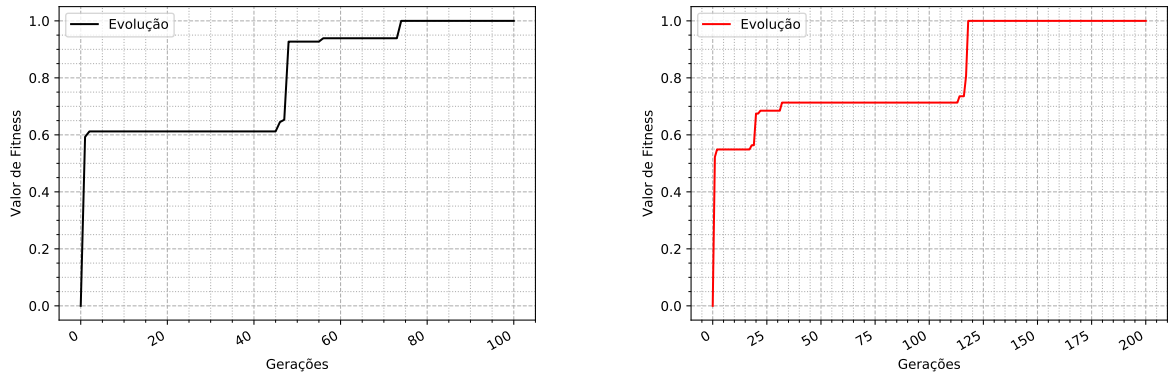
Tabela 13: Requisitos da técnica *Theme screening*

Requisito	Prioridade	Grau de dependência
30	5	1
6	5	0
5	5	2
0	5	0
16	5	0

Fonte: Autoria própria

Para as funções F_3 e F_4, a avaliação dos conjuntos se sucedeu de modo separado, isso foi necessário mediante o fato das funções abordarem propostas diferentes, essas funções foram concebidas considerando características de projetos reais em ambientes ágeis.

Figura 22: Comparação de resultados da função *fitness* F_4



(a) Função de *fitness* F_4 usando a configuração Cf_1 (b) Função de *fitness* F_2 usando a configuração Cf_2

Fonte: Autoria própria

A função F_3 buscava encontrar o melhor conjunto de requisitos levando em consideração o tempo de espera que os requisitos possuíam e a insatisfação dos clientes. Nesse contexto a priorização era focado em priorizar os requisitos que estavam a mais tempo na fila de desenvolvimento e que possuíam os clientes menos satisfeitos, como pode-se perceber na Tabela 14 as informações do melhor conjunto encontrado. Por outro lado a função F_4 buscou encontrar o melhor conjunto de requisitos levando em consideração o menor tempo, esse objetivo foi definido visando encontrar os requisitos que possuíam o menor tempo de desenvolvimento, focando na agilidade em entregar os requisitos, como mostra a Tabela 15.

Tabela 14: Grupo dos melhores requisitos obtidos pela Função F_3

Requisito	Prioridade	Grau de dependência	Premium	satisfação	Tempo
31	3	0	5	5.0	69
3	3	1	3	1.0	43
21	2	0	5	2.0	31
14	5	2	3	1.0	63
32	5	0	3	2.0	120

Fonte: Autoria própria

Para certificar a confiabilidade dos dados obtidos na *QP1* foi realizado teste de hipótese. Como não se espera que as amostras utilizadas tenham distribuição normal, foi utilizado teste não paramétrico para verificar as hipóteses, neste caso foram comparados os resultados produzidos nas abordagens. A hipótese nula $H1_0$ a ser testada trata que

Tabela 15: Grupo dos melhores requisitos obtidos pela Função F_4

Requisito	Horas	Prioridade	Premium	Grau de dependência	Satisfação
131	00:05:02	3	1	0	5.0
130	00:05:41	3	1	0	5.0
129	00:03:23	3	0	0	4.75
67	00:01:00	2	0	0	2.0
46	00:03:00	5	0	2	4.0

Fonte: Autoria própria

as médias das amostras são iguais. Portanto, $H1_0$ somente é rejeitada se o conjunto de requisitos produzir resultados significativos, reforçando a hipótese alternativa $H1_1$ que defende que os dados analisados são estatisticamente diferentes.

Para avaliar a confiabilidade dos dados resultantes na questão *QP1* e mostrar que as médias dos *fitness* são pouco similares, foi aplicado o *Wilcoxon test* com a finalidade de comparar a diferença das funções *fitness* F_1, F_2, F_3, e F_4. O *Wilcoxon test* é um teste estatístico geralmente utilizado para verificar se existem diferenças significativas entre os resultados de duas amostras. Na questão *QP1*, a hipótese nula $H1_0$ para o *Wilcoxon test* indica que não existe diferenças entre os valores de *fitness* nas funções F_1, F_2, F_3, e F_4. Portanto, se o *Wilcoxon test* produzir evidências sobre a diferença entre F_1, F_2, F_3, e F_4, ou seja, rejeitando a $H1_0$, então a hipótese alternativa $H1_1$ que considera diferença nos valores de *fitness* alcançados podem ser assumidos. o resultado do *Wilcoxon test* são apresentados na Tabela 16.

Tabela 16: Resultados do teste de *Wilcoxon* sobre a diferença dos valores *Fitness* obtidos de cada função.

Funções de $fitness$ comparadas	F_1 - F_2	F_3 - F_4	F_1 - F_3	F_1 - F_4	F_2 - F_3	F_4 - F_2
Z	-9,218	-10,165	-10,185	-6,926	-10,175	-9,056
p-value	0.000	0.000	0.000	0.000	0.000	0.000

Fonte: Autoria própria

Conforme pode ser visto na Tabela 16, as funções de *fitness* obtiveram *p-values* inferiores a margem de erro (0,05), então, com nível de confiança de 95%, existem evidências das diferenças entre os valores resultantes das funções de *fitness* F_1, F_2, F_3, e F_4. Portanto, a hipótese alternativa que define a diferença de valores alcançados nas funções *fitness* foi confirmada.

- **Eficiência da abordagem *(QP2)***:

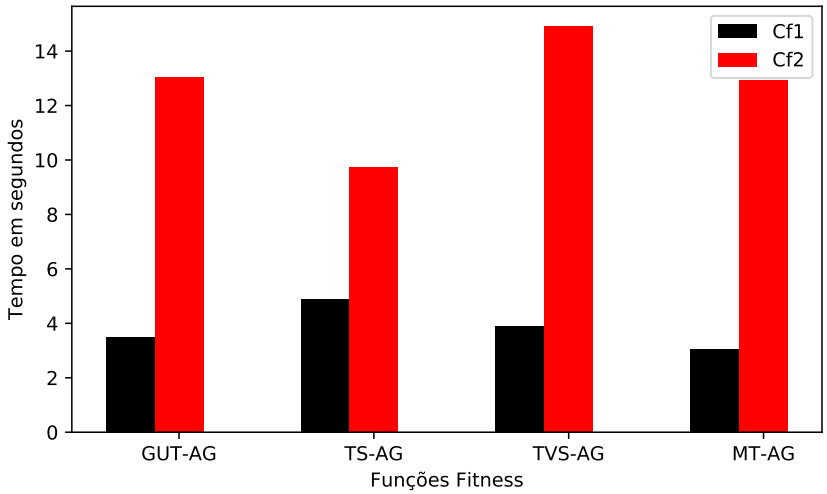
Tabela 17: Tempo médio das geração para cada esquema de *Fitness*

Funções *Fitness*	Tempo Médio	
	Cf_1	Cf_2
F_1	3.47	13.05
F_2	4.87	9.72
F_3	3.91	14.90
F_4	3.06	12.94
Total	15.31	50.61
Média	3.82	12.65

Fonte: Autoria própria

Visando responder à QP2 foi realizada uma análise em relação ao tempo necessário para geração dos conjuntos de requisitos priorizados utilizando AG e as funções de *fitness* propostas. O tempo foi mensurado utilizando as quatro versões de funções *fitness* F_1, F_2, F_3 e F_4. A Tabela 17 apresenta as funções *fitness* na primeira coluna e nas colunas seguintes o tempo para cada configuração(Cf_1 e Cf_2). É possível observar que existe uma notável diferença entre os tempos obtidos, entre uma configuração e outra, isso é devido o fato que a Cf_2 utiliza o dobro de gerações e indivíduos em sua configuração para a geração dos requisitos. Além disso, quanto mais restrições e variáveis para o AG tratar, mais tempo será necessário para encontrar melhores soluções. A distribuição gráfica dos resultados pode ser visto na Figura 23.

Figura 23: Comparação de tempo de cada função *Fitness*



Fonte: Autoria própria

A avaliação de confiabilidade dos dados resultantes na questão *QP2* também se

Tabela 18: Resultados do teste de *Wilcoxon* sobre a diferença de tempo entre as funções *fitness*.

Funções de $fitness$ comparadas	F_1 - F_2	F_3 - F_4	F_1 - F_3	F_1 - F_4	F_2 - F_3	F_4 - F_2
Z	-2,803	-2,803	-2,397	-1,886	-2,803	-2,803
p-value	0,005	0,005	0,017	0,059	0,005	0,005

Fonte: Autoria própria

deu por meio do *Wilcoxon test* com a intuito de comparar a diferença de tempo entre as funções *fitness* F_1, F_2, F_3, e F_4. Na questão *QP2*, a hipótese nula $H2_0$ para o *Wilcoxon test* indica que não existe diferenças entre o tempos de execução entre as funções F_1, F_2, F_3, e F_4. Portanto, se o *Wilcoxon test* produzir evidências sobre a diferença entre F_1, F_2, F_3, e F_4, a $H2_0$ e rejeitada, assumindo-se então a hipótese alternativa $H2_1$ que considera diferença entre o tempo de execução entre as funções *fitness*. o resultado do *Wilcoxon test* são apresentados na tabela 18.

Conforme pode-se notar na Tabela 18, as funções de *fitness* F_1 e F_4 obtiveram *p-values* acima da a margem de erro (0,05). Portanto, a hipótese nula não é descartada nesse caso, sendo assim é confirmado que não existe diferença de tempo de execução entre as funções de *fitness*.

6 CONSIDERAÇÕES FINAIS

A priorização correta dos requisitos de software é fundamental para o cumprimento das estratégias das empresas deste ramo e também para melhorar o desempenho das equipes de desenvolvimento. Porém, muitas vezes essa atividade acontece de maneira subjetiva, já que a fila de priorização é normalmente determinada por algum especialista que utiliza sua experiência para realizar essa priorização. A ausência deste integrante na equipe ou a ausência de uma metodologia bem definida faz com que os desenvolvedores muitas vezes não tenham bons resultados de ordenação, impactando na qualidade do processo de desenvolvimento utilizado. Visando contribuir com esse problema, esse trabalho apresentou uma abordagem com quatro funções de avaliação diferentes para a priorização automática de requisitos de software, utilizando em sua modelagem os conceitos de AG para definir a priorização. Os resultados do experimento comprovaram a eficácia das abordagens propostos. Com isso espera-se que este trabalho possa contribuir para novos estudos e que possa ser utilizado para a criação de novas técnicas de priorização de requisitos automatizados.

6.1 TRABALHOS FUTUROS

Através desde trabalho é possível implementar novas ideias a fim de contribuir com soluções inovadoras para a área de engenharia de software baseado em busca. Por mais encorajador que seja os resultados, ainda existe a necessidade de tornar mais robusto o algoritmo de priorização de requisitos, visto que nesse trabalho foi utilizado uma abordagem mono-objetivo, nesse sentido um trabalho que que aborde a priorização de requisitos utilizando uma abordagem multi-objetiva possa suprir com mais exatidão as necessidades das empresas.

Outro trabalho futuro seria estender a abordagem, criando uma aplicação gráfica que possa facilitar a manipulação dessa abordagem, uma vez que todo experimento foi conduzido por meios técnicos sendo assim inviável para pessoal que não possuam um

conhecimento nas tecnologias usadas. E por fim, a aplicação dos experimentos em um contexto real, visto que os resultados obtidos nesses experimentos foram apenas em testes realizados de forma acadêmica, sendo assim os resultados obtidos podem não representar oque acontece em situações real.

REFERÊNCIAS

AKKER, M. Van den et al. Determination of the next release of a software product: an approach using integer linear programming. In: **CAiSE Short Paper Proceedings**. [S.l.: s.n.], 2005a.

AKKER, M. Van den et al. Software product release planning through optimization and what-if analysis. **Information and Software Technology**, Elsevier, v. 50, n. 1-2, p. 101–111, 2008.

AKKER, M. Van den et al. Flexible release planning using integer linear programming. **REFSQ'05**, v. 10, 2005b.

AMARAL, A. G.; ELIAS, G. A multi-objective, risk-based approach for selecting software requirements. In: **ICAART (2)**. [S.l.: s.n.], 2018. p. 338–346.

BAGNALL, A.; RAYWARD-SMITH, V.; WHITTLEY, I. The next release problem. **Information and Software Technology**, v. 43, n. 14, p. 883 – 890, 2001. ISSN 0950-5849. Disponível em: <http://www.sciencedirect.com/science/article/pii/S095058490100194X>.

BAKER, P. et al. Search based approaches to component selection and prioritization for the next release problem. In: IEEE. **2006 22nd IEEE International Conference on Software Maintenance**. [S.l.], 2006. p. 176–185.

Basili, V. R.; Weiss, D. M. A methodology for collecting valid software engineering data. **IEEE Transactions on Software Engineering**, SE-10, n. 6, p. 728–738, Nov 1984. ISSN 2326-3881.

BRASIL, M. M. A. et al. A multiobjective optimization approach to the software release planning with undefined number of releases and interdependent requirements. In: SPRINGER. **International Conference on Enterprise Information Systems**. [S.l.], 2012. p. 300–314.

CAI, X.; WEI, O. A hybrid of decomposition and domination based evolutionary algorithm for multi-objective software next release problem. In: IEEE. **2013 10th IEEE International Conference on Control and Automation (ICCA)**. [S.l.], 2013. p. 412–417.

CAI, X.; WEI, O.; HUANG, Z. Evolutionary approaches for multi-objective next release problem. **Computing and Informatics**, v. 31, n. 4, p. 847–875, 2012.

CAMARGO, R. F. D. **Como fazer a Matriz GUT para a resolução de problemas? Conheça a Matriz de Prioridades**. 2018. Disponível em: <https://www.treasy.com.br/blog/matriz-gut/>. Acesso em: 09/07/2019.

CHAVES-GONZÁLEZ, J. M.; PÉREZ-TOLEDANO, M. A. Differential evolution with pareto tournament for the multi-objective next release problem. **Applied Mathematics and Computation**, Elsevier, v. 252, p. 1–13, 2015.

CHAVES-GONZALEZ, J. M.; PEREZ-TOLEDANO, M. A.; NAVASA, A. Software requirement optimization using a multiobjective swarm intelligence evolutionary algorithm. **Knowledge-Based Systems**, Elsevier, v. 83, p. 105–115, 2015a.

CHAVES-GONZÁLEZ, J. M.; PÉREZ-TOLEDANO, M. A.; NAVASA, A. Teaching learning based optimization with pareto tournament for the multiobjective software requirements selection. **Engineering Applications of Artificial Intelligence**, Elsevier, v. 43, p. 89–101, 2015b.

COLARES, F. et al. A new approach to the software release planning. In: IEEE. **2009 XXIII Brazilian Symposium on Software Engineering**. [S.l.], 2009. p. 207–215.

CORDEIRO, A. G. Graduação, **PRIORIZAÇÃO DE REQUISITOS E AVALIAÇÃO DA QUALIDADE DE SOFTWARE SEGUNDO A PERCEPÇÃO DOS USUÁRIOS.** CAMPOS DOS GOYTACAZES, RJ, Brasil: [s.n.], 2010.

Deb, K. et al. A fast and elitist multiobjective genetic algorithm: Nsga-ii. **IEEE Transactions on Evolutionary Computation**, v. 6, n. 2, p. 182–197, April 2002. ISSN 1089-778X.

DELINSKI, J. C. C. L. M. M. Otimização multiobjetivo: uma abordagem conceitual. In: . Ponta Grossa, PR, Brasil: ConBRepro, 2017.

DURILLO, J. J. et al. A study of the multi-objective next release problem. In: IEEE. **2009 1st International Symposium on Search Based Software Engineering**. [S.l.], 2009. p. 49–58.

DURILLO, J. J. et al. A study of the bi-objective next release problem. **Empirical Software Engineering**, Springer, v. 16, n. 1, p. 29–60, 2011.

FINKELSTEIN, A. et al. A search based approach to fairness analysis in requirement assignments to aid negotiation, mediation and decision making. **Requirements engineering**, Springer, v. 14, n. 4, p. 231–245, 2009.

FORTIN, F.-A. et al. DEAP: Evolutionary algorithms made easy. **Journal of Machine Learning Research**, v. 13, p. 2171–2175, jul 2012.

FREITAS, F. G. et al. Software next release planning approach through exact optimization. **Int. J. Comput. Appl**, v. 22, n. 8, p. 1–8, 2011.

GENG, J. et al. Supporting many-objective software requirements decision: An exploratory study on the next release problem. **IEEE Access**, IEEE, v. 6, p. 60547–60558, 2018.

GILB, T.; MAIER, M. W. 11.4.2 managing priorities: A key to systematic decision-making. **INCOSE International Symposium**, v. 15, n. 1, p. 1687–1705, 2005. Disponível em: <https://onlinelibrary.wiley.com/doi/abs/10.1002/j.2334-5837.2005.tb00782.x>.

GREER DES E RUHE, G. Software release planning: uma abordagem evolutiva e iterativa. **Informação e tecnologia de software**, Elsevier, v. 46, n. 4, p. 243–253, 2004.

HARMAN; JONES. Search-based software engineering. **Information and Software Technology**, p. 833–839, 2001.

HARMAN, M. The current state and future of search based software engineering. In: **2007 Future of Software Engineering**. Washington, DC, USA: IEEE Computer Society, 2007. (FOSE '07), p. 342–357. ISBN 0-7695-2829-5. Disponível em: <https://doi.org/10.1109/FOSE.2007.29>.

Harman, M.; Clark, J. Metrics are fitness functions too. In: **10th International Symposium on Software Metrics, 2004. Proceedings.** [S.l.: s.n.], 2004. p. 58–69. ISSN 1530-1435.

HARMAN, M. et al. Search based data sensitivity analysis applied to requirement engineering. In: ACM. **Proceedings of the 11th Annual conference on Genetic and evolutionary computation**. [S.l.], 2009a. p. 1681–1688.

HOLLAND, J. H. **Adaptation in Natural and Artificial Systems**. Ann Arbor, MI: University of Michigan Press, 1975. Second edition, 1992.

HUDAIB, A. et al. Requirements prioritization techniques comparison. **Modern Applied Science**, v. 12, 01 2018.

JIANG, H.; XUAN, J.; REN, Z. Approximate backbone based multilevel algorithm for next release problem. In: ACM. **Proceedings of the 12th annual conference on Genetic and evolutionary computation**. [S.l.], 2010b. p. 1333–1340.

JIANG, H. et al. A hybrid aco algorithm for the next release problem. In: IEEE. **The 2nd International Conference on Software Engineering and Data Mining**. [S.l.], 2010a. p. 166–171.

JOHNSON, D. M. The systems engineer and the software crisis. **SIGSOFT Softw. Eng. Notes**, ACM, New York, NY, USA, v. 21, n. 2, p. 64–73, mar. 1996. ISSN 0163-5948. Disponível em: <http://doi.acm.org/10.1145/227531.227542>.

KIFETEW, F. M. et al. Towards multi-decision-maker requirements prioritisation via multi-objective optimisation. In: **CAiSE-Forum-DC**. [S.l.: s.n.], 2017. p. 137–144.

KITCHENHAM, B. et al. Systematic literature reviews in software engineering – a tertiary study. **Information and Software Technology**, v. 52, n. 8, p. 792 – 805, 2010. ISSN 0950-5849. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0950584910000467>.

KOTONYA, G.; SOMMERVILLE, I. **Requirements Engineering: Processes and Techniques**. [S.l.: s.n.], 1998. ISBN John Wiley and Sons.

KUMARI, A. C.; SRINIVAS, K. Comparing the performance of quantum-inspired evolutionary algorithms for the solution of software requirements selection problem. **Information and Software Technology**, Elsevier, v. 76, p. 31–64, 2016.

KUMARI, A. C.; SRINIVAS, K.; GUPTA, M. Software requirements selection using quantum-inspired multi-objective differential evolution algorithm. In: IEEE. **2012 CSI Sixth International Conference on Software Engineering (CONSEG)**. [S.l.], 2012. p. 1–8.

KUMARI, A. C.; SRINIVAS, K.; GUPTA, M. Software requirements optimization using multi-objective quantum-inspired hybrid differential evolution. In: **Evolve-a bridge between probability, set oriented numerics, and evolutionary computation ii**. [S.l.]: Springer, 2013. p. 107–120.

LARMAN, C. **C. Utilizando UML e padrões: uma introduçãoà análise e ao projeto orientado a objetos e ao Processo Unificado.** 2^{th}. ed. [S.l.]: Bookman, 2004. 607 p.

LI, C. et al. Integrated requirement selection and scheduling for the release planning of a software product. In: SPRINGER. **International Working Conference on Requirements Engineering: Foundation for Software Quality**. [S.l.], 2007. p. 93–108.

LI, C. et al. An integrated approach for requirement selection and scheduling in software release planning. **Requirements engineering**, Springer, v. 15, n. 4, p. 375–396, 2010.

LI, X.; XIE, M.; NG, S. H. Multi-objective optimization approaches to software release time determination. **Asia-Pacific Journal of Operational Research**, World Scientific, v. 29, n. 03, p. 1240019, 2012.

LI, Y. et al. Search-based uncertainty-wise requirements prioritization. In: IEEE. **2017 22nd International Conference on Engineering of Complex Computer Systems (ICECCS)**. [S.l.], 2017. p. 80–89.

LIMA, D. C. et al. A fuzzy approach to requirements prioritization. In: SPRINGER. **International Symposium on Search Based Software Engineering**. [S.l.], 2011. p. 64–69.

MACHADO, P. H. D. A. Mestrado, **REDUÇÃO DE DESPERDÍCIOS NO DESENVOLVIMENTO DE SOFTWARE DE GRANDE PORTE POR MEIO DE FERRAMENTAS LEAN**. Pato Branco, PR, Brasil: [s.n.], 2017.

MASSARI, V. L.; VIDAL, A. **Gestão Ágil de Produtos com Agile Think Business Framework: Guia para certificação EXIN Agile Scrum Product Owner**. [S.l.]: Brasport, 2018.

MAUŠA, G. et al. Hill climbing and simulated annealing in large scale next release problem. In: IEEE. **Eurocon 2013**. [S.l.], 2013. p. 452–459.

MITCHELL, M. **An introduction to genetic algorithms**. 2. repr.. ed. New Delhi: Prentice Hall of India, 2002. ISBN 978-81-203-1358-3.

MULLA, N. A new approach to requirement elicitation based on stakeholder recommendation andcollaborative filtering. **International Journal of Software Engineering Applications**, v. 3, p. 51–60, 05 2012.

PAIXÃO, M.; SOUZA, J. A scenario-based robust model for the next release problem. In: ACM. **Proceedings of the 15th annual conference on Genetic and evolutionary computation**. [S.l.], 2013a. p. 1469–1476.

PAIXÃO, M. H. E.; SOUZA, J. T. de. A recoverable robust approach for the next release problem. In: SPRINGER. **International Symposium on Search Based Software Engineering**. [S.l.], 2013b. p. 172–187.

PITANGUEIRA, A. M. Incorporating preferences from multiple stakeholders in software requirements selection an interactive search-based approach. In: IEEE. **2015 IEEE 23rd International Requirements Engineering Conference (RE)**. [S.l.], 2015a. p. 382–387.

PITANGUEIRA, A. M.; MACIEL, R. S. P.; BARROS, M. Software requirements selection and prioritization using sbse approaches. **J. Syst. Softw.**, Elsevier Science Inc., New York, NY, USA, v. 103, n. C, p. 267–280, maio 2015. Disponível em: <http://dx.doi.org/10.1016/j.jss.2014.09.038>.

PITANGUEIRA, A. M. et al. Minimizing the stakeholder dissatisfaction risk in requirement selection for next release planning. **Information and Software Technology**, Elsevier, v. 87, p. 104–118, 2017.

PRESSMAN, R. S. **Software Engineering**. 7^{th}. ed. New York: McGraw Hill, 2011.

Ruhe, G.; Greer, D. Quantitative studies in software release planning under risk and resource constraints. In: **2003 International Symposium on Empirical Software Engineering, 2003. ISESE 2003. Proceedings.** [S.l.: s.n.], 2003. p. 262–270.

RUHE, G. et al. A systematic approach for solving the wicked problem of software release planning. **Soft Computing**, Springer, v. 12, n. 1, p. 95–108, 2008.

SAGRADO, J. D.; ÁGUILA, I. M. D.; ORELLANA, F. J. Ant colony optimization for the next release problem: A comparative study. In: IEEE. **2nd International Symposium on Search Based Software Engineering**. [S.l.], 2010. p. 67–76.

SAGRADO, J. D.; ÁGUILA, I. M. D.; ORELLANA, F. J. Multi-objective ant colony optimization for requirements selection. **Empirical Software Engineering**, Springer, v. 20, n. 3, p. 577–610, 2015.

SALIU, M. O.; RUHE, G. Bi-objective release planning for evolving software systems. In: ACM. **Proceedings of the the 6th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering**. [S.l.], 2007. p. 105–114.

SALIU, O.; RUHE, G. Software release planning for evolving systems. **Innovations in Systems and Software Engineering**, Springer, v. 1, n. 2, p. 189–204, 2005b.

SALIU OMOLADE E RUHE, G. Supporting software release planning decision for evolving systems. In: **29º Workshop Anual de Engenharia de Software da IEEE / NASA**. [S.l.]: IEEE, 2005a. p. 14–26.

SANTOS, G. R. D. Graduação, **Engenharia de Software Baseada em Busca para a Otimização Multiobjetivo de Requisitos Utilizando o Algoritmo NSGA-II.** Boa Vista, RR, Brasil: [s.n.], 2017.

SILVA, C. S. Jarbele Cássia da. Processo de elicitação de requisitos para o desenvolvimento de um Sistema de Controle de Gastos Pessoais. 00 2011. Disponível em: <http://www.lbd.dcc.ufmg.br/colecoes/epiwic/2011/07-WIC-EPI2011.pdf>.

SILVA, T. G. N. D. Mestrado, **PROPOSTA E AVALIAÇÃO DE UM MÉTODO PARA GERAR A POPULAÇÃO INICIAL DE ALGORITMOS GENÉTICOS MULTIOBJETIVOS**. FORTALEZA – CEARÁ, Brasil: [s.n.], 2015.

SOMMERVILLE, I. **Software Engineering**. 9^{th}. ed. Scotland: Pearson Addison-Wesley, 2010. 816 p.

SOMMERVILLE, I. **Software Engineering**. 10^{th}. ed. Scotland: Pearson Addison-Wesley, 2016. 816 p.

SOUZA, F. C. M. **Uma abordagem para geração de dados de teste para o teste de mutação utilizando técnicas baseadas em busca**. Tese (Doutorado) — Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos - SP, Brasil, Agosto 2017.

SOUZA, J. T. de et al. An ant colony optimization approach to the software release planning with dependent requirements. In: SPRINGER. **International symposium on search based software engineering**. [S.l.], 2011. p. 142–157.

SRINIVAS, N.; DEB, K. Muiltiobjective optimization using nondominated sorting in genetic algorithms. **Evolutionary Computation**, v. 2, n. 3, p. 221–248, 1994.

SUREKA, A. Requirements prioritization and next-release problem under non-additive value conditions. In: IEEE. **2014 23rd Australian Software Engineering Conference**. [S.l.], 2014. p. 120–123.

TICONA, W. G. C. Mestrado, **Aplicação de algorimtos genéticos Multiobjetivo para o alinhamento de se sequência biológicas. Dissertação**. São Carlos - SP, Brasil: [s.n.], 2003.

TONELLA, P.; SUSI, A.; PALMA, F. Using interactive ga for requirements prioritization. In: IEEE. **2nd International Symposium on Search Based Software Engineering**. [S.l.], 2010. p. 57–66.

TONELLA, P.; SUSI, A.; PALMA, F. Interactive requirements prioritization using a genetic algorithm. **Information and software technology**, Elsevier, v. 55, n. 1, p. 173–187, 2013.

VEERAPEN, N. et al. An integer linear programming approach to the single and bi-objective next release problem. **Information and Software Technology**, Elsevier, v. 65, p. 1–13, 2015.

XIAO, J. et al. Search based risk mitigation planning in project portfolio management. In: . San Francisco, CA, United states: [s.n.], 2013. p. 146 – 155. Decision supports;Little-JIL;Mitigation planning;Project portfolio management;Resource contention;Risk mitigation;Risk prioritization;Search-based software engineering;. Disponível em: <http://dx.doi.org/10.1145/2486046.2486073>.

XUAN, J. et al. Solving the large scale next release problem with a backbone-based multilevel algorithm. **IEEE Transactions on Software Engineering**, IEEE, v. 38, n. 5, p. 1195–1212, 2012.

ZHANG, Y.; FINKELSTEIN, A.; HARMAN, M. Search based requirements optimisation: Existing work and challenges. In: **Proceedings of the 14th International Conference on Requirements Engineering: Foundation for Software Quality**. Berlin, Heidelberg: Springer-Verlag, 2008. (REFSQ 08), p. 88–94. ISBN 978-3-540-69060-3. Disponível em: <http://dx.doi.org/10.1007/978-3-540-69062-7_8>.

ZHANG, Y.; HARMAN, M. Search based optimization of requirements interaction management. In: IEEE. **2nd international symposium on search based software engineering**. [S.l.], 2010. p. 47–56.

ZHANG, Y. et al. Comparing the performance of metaheuristics for the analysis of multi-stakeholder tradeoffs in requirements optimisation. **Information and software technology**, Elsevier, v. 53, n. 7, p. 761–773, 2011.

ZHANG, Y.; HARMAN, M.; LIM, S. L. Empirical evaluation of search based requirements interaction management. **Information and Software Technology**, Elsevier, v. 55, n. 1, p. 126–152, 2013.

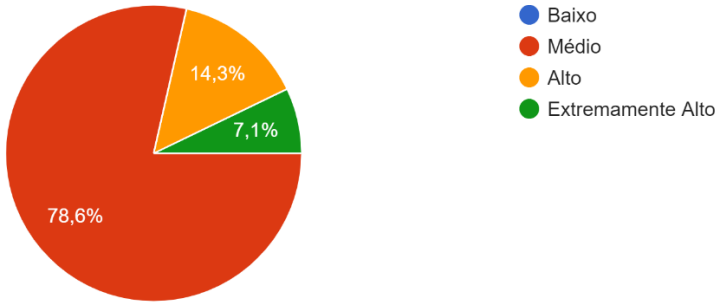
ZHANG, Y.; HARMAN, M.; MANSOURI, S. A. The multi-objective next release problem. In: . New York, NY, USA: ACM, 2007. (GECCO 07), p. 1129–1137. Disponível em: <http://doi.acm.org/10.1145/1276958.1277179>.

APÊNDICE A – SURVEY UTILIZADO NO EXPERIMENTO

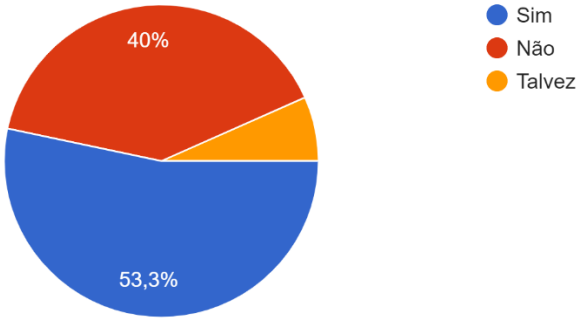
Survey - Priorização de Requisitos de Software

Esta pesquisa tem como objetivo identificar quais aspectos são levados em consideração no momento de priorização de requisitos de software. As informações colhidas não serão identificadas e serão usadas como base de pesquisa para uma proposta de técnica de priorização de requisitos automatizada.

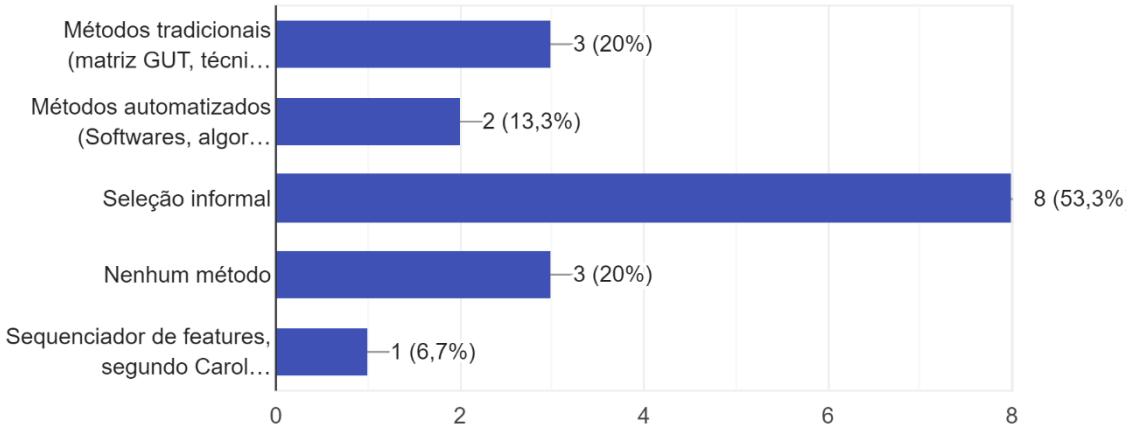
1 - Qual seu nível de conhecimento em Engenharia de Requisitos?



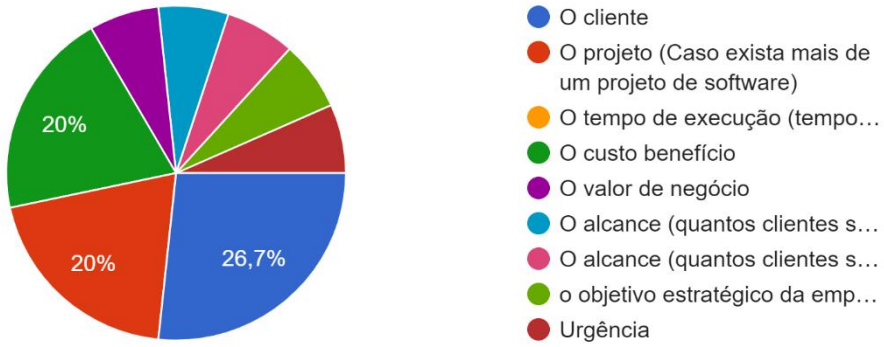
2 - Em algum momento de sua carreira profissional já utilizou algum método ou ferramenta de priorização de requisitos ?



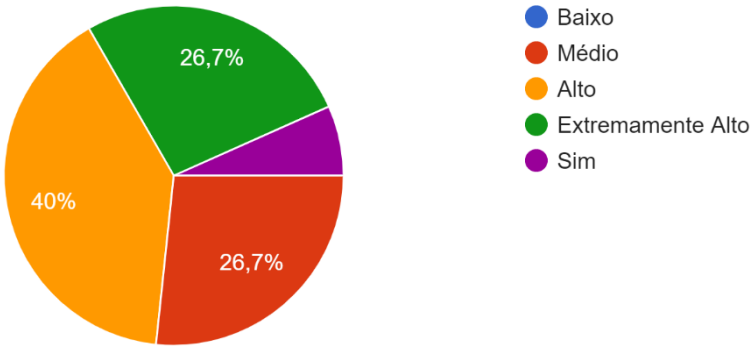
3 - Quais métodos de priorização de requisitos você utiliza ou já utilizou?



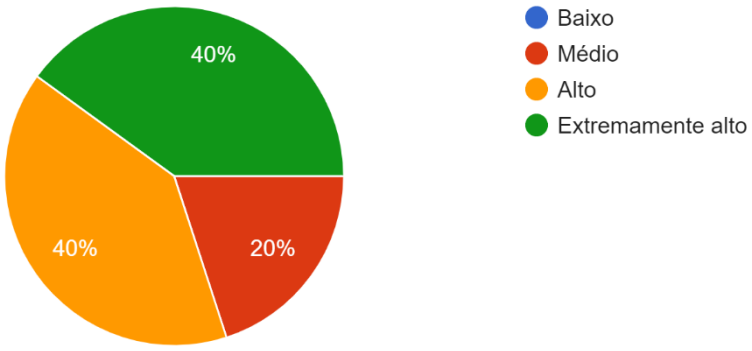
4 - O que é mais relevante no momento da priorização de requisitos?



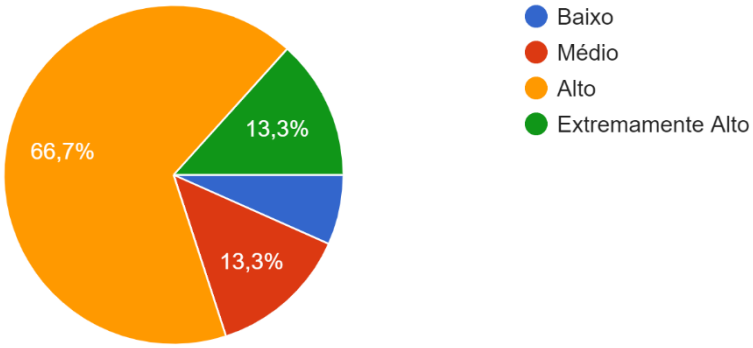
5 - Ao realizar a priorização de requisitos deve ser levado em consideração as dependências entre os requisitos. Qual nível de importância você dá para essa situação?



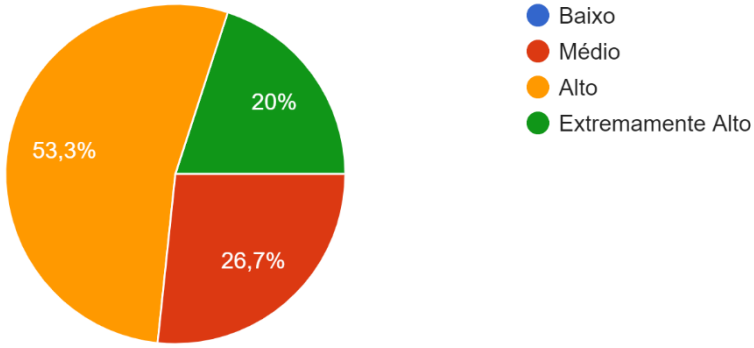
6 - Ao priorizar os requisitos deve se levar em consideração os riscos que os mesmos trarão ao projeto. Indique qual é o nível de importância desse fato ao tomar a decisão de priorizar os requisitos.



7 - Ao priorizar os requisitos deve se levar em consideração o custo benefício que os mesmos irão trazer ao projeto. Indique qual é o nível de importância desse fato ao tomar a decisão de priorizar os requisitos.



8 - Ao priorizar os requisitos deve se levar em consideração os prazos de entrega. Indique qual é o nível de importância desse fato ao tomar a decisão de priorizar um requisito.



9 - Que tipo de medidas ou informações você acha importante para priorizar requisitos?

R: Tempo e complexidade para implementação, benefícios que trará para o sistema e para o cliente.

R: Tempo de entrega, quanto gera de resultado em empoderamento para o cliente e o quanto consigo usar genericamente com os clientes

R; Saber a quantia de valor que vai trazer aos clientes ou em que nível vão possibilitar o início da utilização por parte dos clientes piloto do projeto e não deixar o sistema engessado ou prejudicar outros processos.

R: Avaliar o andamento do atual do software para saber qual o próximo passo e/ou etapa a ser construída, levando em consideração se à algum requisito que precisa de outro pré requisito para sua continuação, por exemplo, exclusão de um usuário, antes de realizar este requisito é necessário avaliar se o já foi ou se tem alguma forma de incluir usuários para poder exclui-los.

R: Saber se aquele é realmente o problema certo para ser resolvido. Entender se está entregando valor para o cliente de fato. Dependências entre as tarefas, pegando em uma ordem correta.

R: Trabalhei apenas com projetos de software, as medidas a serem tomadas que eu levo em consideração é o tamanho da rotina necessária X tamanho da equipe X tempo disponível.

R: Deadlines, alocação do time (considerando perfis e fit com tarefas), nível de relação com os clientes (apesar de ser um ponto delicado e controverso, às vezes o relacionamento com o cliente está demandando maior atenção, demonstração de cuidado e empatia, que é percebido no atendimento ou mesmo posicionamento sobre uma demanda).