UNIVERSIDADE FEDERAL DE SANTA MARIA
CENTRO DE TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Evandro Luis da Rosa Fensterseifer Filho

ABORDAGEM INTELIGENTE PARA ALOCAÇÃO DE TAREFAS EM PROJETOS DE SOFTWARE

Santa Maria, RS
2022

Evandro Luis da Rosa Fensterseifer Filho

ABORDAGEM INTELIGENTE PARA ALOCAÇÃO DE TAREFAS EM PROJETOS DE SOFTWARE

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Santa Maria, como requisito parcial para a obtenção do título de **Mestre em Ciência da Computação**.

Orientadora: Profª Drª. Lisandra Manzoni Fontoura

Santa Maria, RS
2022

Fensterseifer Filho, Evandro Luis da Rosa
ABORDAGEM INTELIGENTE PARA ALOCAÇÃO DE TAREFAS EM PROJETOS DE SOFTWARE / Evandro Luis da Rosa Fensterseifer Filho.- 2022.
76 p.; 30 cm

Orientadora: Lisandra Manzoni Fontoura
Dissertação (mestrado) - Universidade Federal de Santa Maria, Centro de Tecnologia, Programa de Pós-Graduação em Ciência da Computação , RS, 2022

1. Sistemas de recomendação 2. Alocação de tarefas 3. Gerenciamento de projetos I. Fontoura, Lisandra Manzoni II. Título.

Sistema de geração automática de ficha catalográfica da UFSM. Dados fornecidos pelo autor(a). Sob supervisão da Direção da Divisão de Processos Técnicos da Biblioteca Central. Bibliotecária responsável Paula Schoenfeldt Patta CRB 10/1728.

Declaro, EVANDRO LUIS DA ROSA FENSTERSEIFER FILHO, para os devidos fins e sob as penas da lei, que a pesquisa constante neste trabalho de conclusão de curso (Dissertação) foi por mim elaborada e que as informações necessárias objeto de consulta em literatura e outras fontes estão devidamente referenciadas. Declaro, ainda, que este trabalho ou parte dele não foi apresentado anteriormente para obtenção de qualquer outro grau acadêmico, estando ciente de que a inveracidade da presente declaração poderá resultar na anulação da titulação pela Universidade, entre outras consequências legais.

Evandro Luis da Rosa Fensterseifer Filho

ABORDAGEM INTELIGENTE PARA ALOCAÇÃO DE TAREFAS EM PROJETOS DE SOFTWARE

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Santa Maria, como requisito parcial para a obtenção do título de **Mestre em Ciência da Computação**.

Aprovado em 10 de junho de 2022:

Lisandra Fontoura

Lisandra Manzoni Fontoura, Drª. (UFSM)

(Presidenta/Orientadora)

Ana Trindade Winck, Drª. (UFCSPA)

Giana Lucca Kroth

Giana Lucca Kroth, Drª. (UFSM)

Santa Maria, RS
2022

AGRADECIMENTOS

Agradeço a minha família, em especial minha mãe, minha companheira e meu irmão, Leonarda, Lidiane e Guilherme, que me apoiaram nesta etapa. Obrigado pela paciência, carinho, apoio e compreensão. Obrigado por me incentivarem, por acreditarem em mim e estarem ao meu lado, apoiando-me nos momentos de dificuldades e comemorando as conquistas alcançadas.

Agradeço a minha orientadora Lisandra por todas as trocas de ideias, ensinamentos, paciência e conselhos. Assim como, toda a ajuda dispendida antes e durante o desenvolvimento deste trabalho.

Agradeço ao Professor Joaquim Vinicius Carvalho Assunção, mais uma vez, pelo conhecimento, dicas e esclarecimentos, fundamental na conclusão deste trabalho.

Aos participantes desta pesquisa por compartilharem seu tempo durante a validação deste trabalho.

Por fim, agradeço a todos que, direta ou indiretamente contribuíram para a conclusão desta dissertação.

RESUMO

ABORDAGEM INTELIGENTE PARA ALOCAÇÃO DETAREFAS EM PROJETOS DE SOFTWARE

AUTOR: Evandro Luis da Rosa Fensterseifer Filho
ORIENTADORA: Lisandra Manzoni Fontoura

O conhecimento técnico e a personalidade da equipe influenciam em projetos de software, podendo reduzir ou aumentar a qualidade dos produtos gerados e a velocidade de desenvolvimento. Para uma alocação de tarefas bem-sucedida é importante considerar as preferências e o perfil de cada desenvolvedor, maximizando assim sua produtividade. Em projetos com muitos desenvolvedores, a alocação de tarefas pode ser uma tarefa desafiadora podendo ser auxiliada por ferramentas de recomendação. Neste trabalho é proposta uma abordagem inteligente para alocação de tarefas de desenvolvimento de software, adequadas ao perfil do desenvolvedor. Com base na literatura, foram definidos perfis de desenvolvedores necessários em uma equipe de software considerando as competências e os perfis técnicos. Visando avaliar o perfil do desenvolvedor e associar tarefas adequadas a este, esse trabalho utiliza um questionário com questões que visam identificar o perfil do desenvolvedor. A partir das respostas, um Sistema de recomendação foi desenvolvido para alocar tarefas aos desenvolvedores, empregando técnicas desde processamento textual, MultinomialNB e Random Forest. As alocações realizadas são avaliadas pelos desenvolvedores e utilizadas para melhorar o sistema de recomendação. A alocação de tarefas, de acordo com o perfil de cada desenvolvedor, é um catalisador para se obter um melhor desempenho do projeto, melhorando a qualidade dos produtos gerados e diminuindo o esforço do desenvolvimento. As validações mostraram que a abordagem elaborada realiza recomendações condizentes e coerentes de tarefas aos desenvolvedores, de acordo com seu perfil.

Palavras-chave: Sistemas de recomendação. Alocação de tarefas. Gerenciamento de projetos.

ABSTRACT

SMART APPROACH TO TASK ALLOCATION IN SOFTWARE PROJECTS

AUTHOR: Evandro Luis da Rosa Fensterseifer Filho
ADVISOR: Lisandra Manzoni Fontoura

The team's technical knowledge and personality influence software projects, and can reduce or increase the quality of the generated products and the speed of development. For successful task allocation, it is needed to consider the preferences and profile of each developer, thus maximizing their productivity. In projects with many developers, task allocation can be a challenging task and can be aided by recommendation tools. In this work, we propose an intelligent approach for allocating software development tasks appropriate to the developer's profile. Based on the literature, profiles of developers needed in a software team were defined, considering skills and technical profiles. Aiming to evaluate the profile of the developer and associate appropriate tasks with it, this work uses a questionnaire with questions that aim to identify the profile of the developer. From the answers, a recommender system allocates tasks to developers, employing text processing techniques, MultinomialNB, and Random Forest. Developers evaluate allocations, and the system uses them to improve the recommender system. The allocation of tasks, according to the profile of each developer, seeks to improve the performance of the project and the quality of the products developed, as well as to reduce the development effort. The validations showed that the developed approach makes consistent and coherent task recommendations to developers, according to their profile.

Keywords: Recommender systems. Task allocation. Project management.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

BPMN	*Business Process Model and Notation*
CSV	*Comma-separated values*
FFM	*Five Factor Model*
IDF	*Inverse Document Frequency*
MBTI	*Myers-Briggs Type Indicator*
ML	*Machine Learning*
OCEAN	*Openess*, *Conscientiousness*, *Extraversion*, *Agreeableness*, *Neuroticism*
TF	*Term Frequency*
TF-IDF	*Term Frequency Inverse Document Frequency*
UFSM	Universidade Federal de Santa Maria
PC	Peso competência
PT	Peso técnico
VSM	*Vector space model*

SUMÁRIO

1 INTRODUÇÃO

Gerenciar projetos envolve aplicar conhecimentos, habilidades, ferramentas e técnicas para atingir os objetivos de um projeto que são satisfazer os requisitos do cliente, em prazos e custos previamente estabelecidos (INSTITUTE, 2017). Para que o gerenciamento seja efetivo, este deve considerar as pessoas, produto, processo e projeto (PRESSMAN, 2011). As pessoas que trabalham em uma organização são seus maiores ativos. É custoso recrutar e reter boas pessoas cabe aos gerentes de software garantir que a empresa obtenha o melhor retorno possível. Empresas e economias de sucesso alcançam isso, quando as pessoas são respeitadas pelas organizações e lhes são atribuídas responsabilidades que refletem suas habilidades e experiências (SOMMERVILLE, 2011). Para que projetos de software sejam bem sucedidos, os desenvolvedores precisam ser valorizados (PRESSMAN, 2011).

Para o gerenciamento das pessoas, é importante considerar o equilíbrio entre as habilidades técnicas, a experiência e a personalidade na definição de uma equipe (SOMMERVILLE, 2011). O desempenho na execução de uma tarefa é influenciado por alguns fatores, tais como: motivação, personalidade ou por sua combinação (CAPRETZ; VARONA; RAZA, 2015). Vários autores consideram que fatores relacionados à personalidade e interação podem influenciar no desempenho dos desenvolvedores, portanto, devem ser considerados durante a alocação de tarefas (MENDES; MENDES; SALLEH, 2019; SOOMRO *et al*., 2016; WEINBERG, 1971).

Atividades como memorização, observação e exploração de situações para aprender fatos, melhorar habilidades motoras e cognitivas e organizar o conhecimento novo em representações apropriadas podem ser consideradas atividades de aprendizado (FACELI *et al*., 2011). A capacidade de aprendizado está associada a um comportamento inteligente.

Uma forma de desempenhar o aprendizado, consequentemente associado a um comportamento inteligente, é por inteligência artificial e algoritmos de *machine learning* (ML). Dentre uma das várias finalidades que possuem, é o seu emprego na construção de sistemas de recomendação. Dentre as funcionalidades que sistemas de recomendação possuem, cita-se: analisar dados dos usuários e extrair as informações úteis para futuras predições, salientando os itens mais relevantes, assim como tratando o problema da sobrecarga de informações (ROBILLARD; WALKER; ZIMMERMANN, 2010).

Desenvolvedor é um nome genérico que pode ser melhor definido por meio dos papéis: analista de sistemas, *designers*, programadores, testadores e mantenedores (CAPRETZ; VARONA; RAZA, 2015).

1.1 MOTIVAÇÃO

Em muitos projetos, a alocação de tarefas é realizada pelo gerente de projeto com base no papel exercido por cada desenvolvedor e pela quantidade de tarefas adequadas à capacidade de cada membro da equipe. Muitos trabalhos exploram o impacto e a relação que a personalidade do desenvolvedor tem sobre o desenvolvimento de software (CAPRETZ; AHMED, 2010; CAPRETZ; VARONA; RAZA, 2015; MENDES; MENDES; SALLEH, 2019; WANG; LIN; HAJLI, 2019), bem como aspectos relacionados ao perfil e as tarefas em desenvolvimento *crowdsourcing* (MAO *et al*., 2015) e preferências individuais e interesses (WANG; WANG; LI, 2018).

Autores sugerem que a Engenharia de software necessita de técnicas e ferramentas automatizadas, adaptabilidade e escalabilidade para acompanhar o aumento da produtividade do desenvolvedor (BENNACEUR; MEINKE, 2018).

A utilização de técnicas e ferramentas automatizadas auxilia os gerentes a tomar melhores decisões, principalmente quando uma grande quantidade de dados é disponibilizada. Quando se analisa grandes conjuntos de dados informações relevantes podem passar despercebidas, enquanto que, se fossem consideradas possibilitariam um melhor arranjo dos recursos disponíveis para a conclusão do projeto.

1.2 OBJETIVOS

O objetivo deste trabalho é propor uma abordagem inteligente para alocação de tarefas adequadas ao perfil de desenvolvedores. Para isso é proposto um modelo de alocação que considera dados relacionados ao perfil de desenvolvedor e os dados extraídos das tarefas.

A abordagem inteligente desenvolvida emprega processamento textual, desde técnicas como *Vector Space Model* (VSM), *Term Frequency Inverse Document Frequency* (TF-IDF), e algoritmos de aprendizado de máquina, em específico MultinomialNB e *Random Forest*. Estas técnicas são utilizadas para facilitar a análise

textual das tarefas e os algoritmos de aprendizado de máquina para treinar os classificadores, desta forma, recomendando tarefas para os desenvolvedores.

MultinomialNB é uma variante do algoritmo *Naive Bayes*. Ele foi escolhido pois é conhecido por ser simples e eficaz na aplicação de classificação de texto e também por ser uma técnica popular em mineração de dados (ABU SHEIKHA; INKPEN, 2010; MANGMANG; FELISCUZO; MARAVILLAS, 2019). *Random Forest* é uma técnica usada para fins de classificação e de regressão (BREIMAN, 2001; HAN; KAMBER; PEI, 2012). Ele possibilita analisar os atributos/colunas empregados para treinar o classificador. O algoritmo *Random Forest* constrói uma serie de árvores de decisão durante o treino e defini a classe final de acordo com as classes geradas pelas árvores distintas. Para obter melhores resultados do que os resultados de modelos de árvores de decisão individuais. Ele combina os resultados de diferentes modelos de tipo semelhante ou de diferentes tipos (SATAPATHY; ACHARYA; RATH, 2016).

Em vista disso, espera-se realizar alocações condizentes e coerentes de tarefas a desenvolvedores, motivando-os a realizar a tarefa e obtendo um melhor desempenho em relação ao tempo e qualidade.

1.3 ESTRUTURA DO TRABALHO

No Capítulo 2 são apresentados os conceitos necessários para o entendimento deste trabalho. Os trabalhos relacionados são apresentados no Capítulo 3. No Capítulo 4 é apresentada a abordagem desenvolvida para a alocação de tarefas. A validação e os resultados obtidos são discutidos no Capítulo 5. Por fim, no Capítulo 6 é realizada a conclusão do trabalho e descrito os trabalhos futuros.

2 REFERENCIAL TEÓRICO

A alocação de tarefas é uma tarefa complexa de ser executada, pois, requer a análise de muitas variáveis, tanto em relação ao conhecimento técnico como à personalidade do desenvolvedor. Essa atividade faz parte do gerenciamento de projetos de software, portanto, na Subseção 2.1 são descritos conceitos relacionados a essa área visando facilitar a compreensão do trabalho. Na Subseção 2.2 são descritos diferentes perfis de personalidade e uma correlação dentre competências e funções técnicas conforme a literatura. A Subseção 2.3 apresenta alguns algoritmos de processamento de texto, utilizados neste trabalho para preparar os dados textuais para posterior uso pelo sistema de recomendação. Por fim, na Subseção 2.4 aborda os conceitos referentes a sistemas de recomendação, além das técnicas escolhidas para a implementação desta abordagem.

2.1 GERECIAMENTO DE PROJETOS

O gerenciamento de projetos visa entregar um software que satisfaz as necessidades dos clientes, no prazo e orçamento previamente definidos (INSTITUTE, 2017). Segundo o *Project Management Institute* (PMI), a alocação de tarefas é realizada no processo “Desenvolver o cronograma” e tem forte ênfase na disponibilidade do recurso consoante o calendário de trabalho de cada membro da equipe e nos requisitos de recursos, definidos por cada atividade. A prioridade é o nivelamento da carga de atividades atribuídas a cada recurso, não sobrecarregando e nem deixando recursos ociosos.

A ênfase nas pessoas e interação entre elas são características dos métodos ágeis, sendo este um dentre seus princípios (BECK *et al*., 2001). O desenvolvimento de software é uma atividade extremamente criativa e que requer que as pessoas sejam tratadas como indivíduos e não como “unidades de programação substituíveis” (FOWLER, 2005). Assim como, a qualidade de um produto de software depende muito das pessoas, da organização e dos procedimentos utilizados para criá-lo e entregá-lo (FUGGETTA, 2000). Os autores Williams e Cockburn (2003) afirmam que as pessoas são o fator mais relevante do desenvolvimento.

Para a criação de equipes consistentes é necessário o reconhecimento das forças humanas (PRESSMAN, 2011). Portanto, o desenvolvimento de software é uma atividade complexa que exige criatividade, cooperação e esforço da equipe de

desenvolvimento. Em razão de sua natureza complexa, a personalidade dos desenvolvedores - individualidade e identidade - são fatores de suma importância no sucesso (MENDES; MENDES; SALLEH, 2019; SOOMRO *et al*., 2016; WEINBERG, 1971).

Tarefas delegadas a um desenvolvedor considerando sua personalidade, tendem a ter um melhor desempenho (DEMARCO; LISTER, 2013). Um elemento poderoso no desempenho dos objetivos da tarefa é geralmente a motivação (HALL *et al*., 2008).

A definição de perfis técnicos depende das características do projeto e do processo de desenvolvimento (CAPRETZ; VARONA; RAZA, 2015). Trabalhos descritos na literatura caracterizam alguns perfis técnicos exercidos durante o desenvolvimento de software e suas atividades (AQEEL IQBAL; ALDAIHANI; SHAH, 2019; CAPRETZ; AHMED, 2010; CAPRETZ; VARONA; RAZA, 2015; YILMAZ *et al*., 2017), dentre alguns:

- Analista de sistemas — maior ênfase na identificação de componentes de alto nível em um aplicativo do mundo real a partir das necessidades do usuário. Entende quais são os recursos essenciais do sistema e cria um modelo de aplicativo abstrato que atenda a esses requisitos, decompondo-os. Para atingir os objetivos desse papel, o profissional precisa interagir com outros usuários e/ou clientes. Além disso, os analistas de sistemas devem conseguir simpatizar com os problemas dos usuários, de modo a compreender melhor as suas necessidades, em vista disto, as habilidades interpessoais são altamente desejáveis;
- *Designer* — o campo de *design* de software requer a criatividade humana. O *design* de software é um processo exploratório, em que o *designer* procura componentes, tentando uma variedade de esquemas para descobrir o modo mais natural e razoável de refinar uma solução. Embora o *design* de software possa parecer uma tarefa fácil, no *design* de softwares grandes e complexos, a identificação dos principais componentes é um trabalho árduo e demorado. Repetições não são incomuns, pois um bom *design* geralmente leva várias iterações. Os *designers* de software devem ter a capacidade de ver o cenário ao todo, assim como isolar itens relevantes de grandes quantidades de dados vagos e imprecisos; o que requer a intuição de discernir padrões;

- Programador — os programadores devem seguir um processo iterativo de aprimoramento passo a passo. Assim como, atender aos detalhes e manter a forma lógica e analítica de pensamento. Além disso, desempenhar tarefas de programação como determinar os detalhes da lógica do módulo, estabelecer o *layout* do arquivo e programas de codificação. Os programadores que trabalham com as especificações dos projetistas precisam ser lógicos e estar atentos aos detalhes;
- Testador — testar envolve encontrar defeitos no software. As estratégias de teste não são aleatórias nem casuais; pelo contrário, devem ser abordadas de maneira metódica e sistemática. Depois que um defeito é detectado, a depuração pode ser uma atividade frustrante e emocionalmente desafiadora que pode levar os seus encarregados a reestruturar seus pensamentos e decisões. Os testes requerem atenção aos detalhes, além de exercer enorme pressão sobre os seus profissionais para cumprir os prazos e entregar o produto. Assim, precisão e ordem são características altamente desejáveis. O processo de teste exige uma grande quantidade de persistência, especialmente a tarefa de escolher entre uma ampla gama de possibilidades e manter um incrível grau de atenção aos detalhes.

Cada programador possui uma personalidade, deste modo a sua interação é um fator fundamental e as características de cada membro que compõem a equipe do projeto (SOOMRO *et al*., 2016). Sendo assim, uma equipe deve ter equilíbrio entre habilidades técnicas, a experiência e as personalidades de seus membros, sendo esta uma atividade crítica para o gerenciamento (SOMMERVILLE, 2011).

2.2 PERFIS DE PERSONALIDADE

Nesta Subseção é introduzido alguns modelos utilizados na área da Psicologia, retratando-os de forma sucinta. Na Subseção 2.3.1 é descrito sobre o Modelo *Five Fator Model* (FFM). Na Subseção seguinte é dado uma visão geral sobre o modelo de *Myers Briggs Type Indicator* (MBTI). Por fim, na Subseção 2.3.3 é apresentada uma análise sobre os perfis técnicos, anteriormente discutidos, e as competências necessárias para o seu exercício.

2.2.1 Modelo Five Factor Model

Dentre os modelos presentes mais citados na literatura para descrever perfis de personalidade, destaca-se (SOOMRO *et al*., 2016): o Modelo *Five Fator Model* (COSTA; MCCRAE JR, 1992; GOLDBERG, 1993) ou Modelo dos cinco fatores, também conhecido por FFM ou OCEAN; acrônimo para o*peness*, *conscientiousness*, *extraversion*, *agreeableness* e *neuroticism*.

O modelo FFM tem se destacado em trabalhos que buscam entender a influência da personalidade sobre a equipe, dentre alguns: Feldt *et al*. (2010); Jia, Zhang e Zhang (2015); Yilmaz *et al*. (2017); e Aqeel Iqbal, Aldaihani e Shah (2019).

O modelo vem sendo desenvolvido por vários psicólogos ao longo dos anos. Entre um dos inventários utilizados para medir os construtos da personalidade está o NEO-*Personality Inventory* (NEO-PI-R) (COSTA; MCCRAE JR, 1992). O qual é desenvolvido por Costa e McCrae (1992) e utilizado para mensurar cinco dimensões da personalidade. Este modelo divide-se em (FURNHAM; MOUTAFI; CRUMP, 2003):

Abertura à experiência (*openess*): representa a tendência de se envolver em atividades intelectuais e novas experiências. Normalmente altos valores denotam um indivíduo receptivo a aventuras, artístico e criativo;

Conscienciosidade (*conscientiousness*): está associada à persistência, autodisciplina e necessidade de realização. Algumas caraterísticas relacionadas a estes indivíduos são a organização e a autodireção;

Extroversão (*extraversion*): refere-se à alta atividade, sociabilidade e a tendência de experimentar emoções positivas. O indivíduo característico pode ser considerado extrovertido, entusiasta e sociável;

Simpatia/Amabilidade (*agreeableness*): refere-se a um comportamento amigável, atencioso e modesto. Dentre alguns valores destes indivíduos denota-se gentileza e empatia;

Neuroticismo ou Estabilidade emocional (*neuroticism*): pode ser descrito como a experiência de emoções negativas, principalmente ansiedade, depressão e raiva.

2.2.2 Modelo Myers Briggs Type Indicator

O Modelo MBTI (MYERS, 1998), também é conhecido por *Myers Briggs Type Indicator*. Entre alguns trabalhos que tem utilizado este modelo: Yilmaz e O'Connor (2012); Soomro *et al*. (2016); Capretz, Varona, Raza (2015); e Capretz e Ahmed (2010).

O modelo MBTI, é baseado na teoria do psicólogo Carl Jung, que propõe que as pessoas podem ser classificadas em tipos psicológicos, conforme as características pessoais, hábitos, preferências e iniciativas. Jung classifica os indivíduos em 4 pares opostos de acordo com sua atitude em relação à energia, atenção, decisão e vivência. Os quatro grupos opostos são (JUNG, 2011):

Energia: como uma pessoa é energizada, podendo ser: extrovertido ou introvertido — altos valores para extrovertido denotam que o indivíduo é extrovertido, gosta de conversar e normalmente é bem comunicativo. Por outro lado, o introvertido é mais quieto, reservado, tendem a responder a uma conversa a iniciá-la;

Atenção: como uma pessoa dedica atenção, variando de sensitivo e intuitivo — duas formas alternativas de perceber informações. A sensação envolve o recebimento de informações diretamente através dos sentidos, enquanto a intuição envolve a descoberta de possibilidades que podem não ser imediatamente óbvias a partir de dados sensoriais. Novos problemas não costumam ser apreciados por indivíduos sensitivos, por outro lado, pessoas intuitivas gostam de resolver novos problemas e costumam desgostar de executar tarefas triviais;

Decisão: como uma pessoa toma suas decisões, podendo ser emotivo (sentimental) ou racional (pensador) — são duas formas alternativas de julgar informações. O pensamento envolve a análise lógica da informação em termos dos princípios estritos de causa e efeito, e o sentimento envolve a identificação do valor emocional vinculado a objetos ou eventos. O pensador envolve os indivíduos que confiam primariamente em seu próprio raciocínio; são assertivos e orientados a princípios. Enquanto seu oposto, os indivíduos sentimentais são subjetivos, empáticos e tendem a considerar todas as pessoas envolvidas em um determinado contexto;

Vivência: estilo de vida que a pessoa adota, variando de perceptivo (filosófico) a julgador — julgar e perceber são dois processos pelos quais percebemos e agimos conforme a informação. Perceber se preocupa em receber informações diretamente

sem avaliação, mais espontâneo. Já no outro extremo, julgar se preocupa em organizar e processar informações (organizado e metódico).

2.2.3 Funções técnicas e suas competências

As funções técnicas mencionadas na Subseção 2.1, têm sido estudadas sua relação com os aspectos da personalidade atividades (AQEEL IQBAL; ALDAIHANI; SHAH, 2019; CAPRETZ; AHMED, 2010; CAPRETZ; VARONA; RAZA, 2015; YILMAZ *et al*., 2017).

Na Tabela 1 pode ser visualizado um mapeamento dos perfis técnicos, exercido durante o desenvolvimento de software, e das personalidades propostas pelo modelo MBTI (CAPRETZ; AHMED, 2010; CAPRETZ; VARONA; RAZA, 2015). Já na Tabela 2 pode ser visualizado o mapeamento dos perfis técnicos utilizando-se o modelo FFM (AQEEL IQBAL; ALDAIHANI; SHAH, 2019; YILMAZ *et al*., 2017).

Tabela 1 – Mapeamento do perfil técnico segundo o modelo MBTI.

	Trabalhos	
Perfis técnicos	(CAPRETZ; AHMED, 2010)	(CAPRETZ; VARONA; RAZA, 2015)
Analista de sistemas	Extrovertido Sentimental	Extrovertido Sensitivo Sentimental Perceptivo
Designer	Intuitivo Racional	Introvertido Sensitivo Racional Julgador
Programador	Introvertido Sensitivo Racional	Extrovertido Sensitivo Sentimental Perceptivo
Testador	Sensitivo Julgador	Extrovertido Sensitivo Racional/Sentimental Julgador

Fonte: elaborado pelo autor.

Na Tabela 2 adotou-se porcentagem na primeira coluna, ao lado dos traços do modelo FFM, já que o trabalho desenvolvido pelos autores analisa um conjunto amostral e sugere não os traços concernentes ao perfil, mas uma porcentagem total de $X\%$ participantes de um determinado perfil técnico detentor de tal traço da personalidade.

Tabela 2 – Mapeamento do perfil técnico segundo o modelo FFM.

	Trabalhos	
Perfis técnicos	(YILMAZ et al., 2017)	(AQEEL IQBAL; ALDAIHANI; SHAH, 2019)
Analista de sistemas	Extroversão (58%) Abertura à experiência (51%) Conscienciosidade (43%) Simpatia/Amabilidade (62%) Neuroticismo (26%)	Extroversão Abertura à experiência Conscienciosidade
Designer	-	Extroversão Abertura à experiência Conscienciosidade Simpatia/Amabilidade
Programador	Extroversão (41%) Abertura à experiência (68%) Conscienciosidade (35%) Simpatia/Amabilidade (25%) Neuroticismo (23%)	Extroversão Abertura à experiência Conscienciosidade Simpatia/Amabilidade
Testador	Extroversão (79%) Abertura à experiência (35%) Conscienciosidade (37%) Simpatia/Amabilidade (42%) Neuroticismo (28%)	Extroversão Abertura à experiência Conscienciosidade

Fonte: elaborado pelo autor.

Um modelo de conversão entre os modelos psicológicos FFM e MBTI é utilizado neste trabalho, visando padronizar o modelo utilizado para análise. Este modelo é proposto por Furnham, Moutafi e Crump (2003), como pode ser visualizado na Tabela 3.

Tabela 3 – Correlação entre modelos MBTI e FFM.

	FFM				
	Abertura à experiência	Conscienciosidade	Extroversão	Simpatia	Neuroticismo
MBTI	Intuição	Julgador	Extroversão	Sentimental	-

Fonte: adaptado de (FURNHAM; MOUTAFI; CRUMP, 2003).

Vários trabalhos da literatura de Engenharia de software e de aspectos humanos no desenvolvimento de software (CAPRETZ; AHMED, 2010; CAPRETZ; VARONA; RAZA, 2015; WAZLAWICK, 2013) definem como funções dos analistas de sistemas: identificar componentes de alto nível; traduzir os requisitos do cliente em resumos de projeto, altamente detalhados; compreender quais são os recursos essenciais do sistema; possuem grande domínio sobre habilidades interpessoais; e trabalham com estreita colaboração com desenvolvedores e *stakeholders*.

Os trabalhos analisados afirmam que o *designer* possui ênfase em atividades como (CAPRETZ; AHMED, 2010; CAPRETZ; VARONA; RAZA, 2015; WAZLAWICK, 2013): capacidade de criar cenários, *storyboards*, arquitetura de informação, recursos e interfaces; prototipar a experiência do usuário e ideias de *design*; criar um projeto de arquitetura com as especificações necessárias para o hardware, software e dados; desenvolver, documentar e revisar os procedimentos do projeto do sistema; e possuir uma capacidade analítica e de resolução de problemas.

Atividades comumente exercidas por programadores são (CAPRETZ; AHMED, 2010; CAPRETZ; VARONA; RAZA, 2015; WAZLAWICK, 2013): atender aos detalhes e manter a forma lógica e analítica de pensamento; ser lógico e estar atento aos detalhes; desempenhar tarefas de programação, como determinar os detalhes da lógica do módulo, estabelecer o *layout* do arquivo e programas de codificação; e ter capacidade analítica e de resolução de problemas.

O testador deve (CAPRETZ; AHMED, 2010; CAPRETZ; VARONA; RAZA, 2015; WAZLAWICK, 2013): utilizar de abordagens metódicas e sistemáticas; reunir requisitos de teste e produzir especificações de testes; gerenciar os defeitos, identificar, registrar, rastrear e verificar os problemas; analisar e avaliar, documentar e comunicar o progresso dos testes para os *stakeholders*; e priorizar atenção aos detalhes.

A criatividade é uma competência associada aos perfis de analista e de *designer*. As atividades como a análise de requisitos (PRESSMAN, 2011; SOMMERVILLE, 2011) visa a conclusão de um modelo, tais como: baseados em cenários; de dados; orientados a classes; orientados a fluxos; ou comportamentais. A análise é desempenhada por engenheiros de software, analistas ou modeladores (PRESSMAN, 2011). As atividades do analista e do *designer* estão bem próximas, pois o analista deve criar um modelo abstrato da aplicação, no qual são atendidas as necessidades do usuário, e o designer deve modelar o sistema proposto como também avaliar e validar a interface (CAPRETZ; VARONA; RAZA, 2015).

A competência organização, em especifico atividades de teste. As estratégias de teste não são aleatórias nem casuais. Elas devem ser abordadas de forma metódica e sistemática (CAPRETZ; AHMED, 2010). Em vista disto, percebe-se que indivíduos organizados possuem um melhor rendimento nestas tarefas. Trabalhos da literatura (CAPRETZ; VARONA; RAZA, 2015; YILMAZ *et al*., 2017; AQEEL IQBAL; ALDAIHANI; SHAH, 2019) têm indicado que testadores são mais organizados do que os demais perfis analisados.

A comunicação é uma competência associada aos perfis: analista (CAPRETZ; AHMED, 2010; CAPRETZ; VARONA; RAZA, 2015; YILMAZ *et al*., 2017; AQEEL IQBAL; ALDAIHANI; SHAH, 2019), programador (CAPRETZ; VARONA; RAZA, 2015; YILMAZ *et al*., 2017) e testador (CAPRETZ; VARONA; RAZA, 2015; YILMAZ *et al*., 2017; AQEEL IQBAL; ALDAIHANI; SHAH, 2019).

Os analistas de sistemas devem ser capazes de ter empatia com os problemas dos usuários para entender completamente suas necessidades, portanto, as habilidades interpessoais são altamente desejáveis (CAPRETZ; AHMED, 2010). Assim como, os testadores, nos ambientes socialmente interativos com uma alta tendência a ser cooperativo, apresentaram uma presença maior de traços sentimentais, no caso emotivo (agradabilidade, altruísta). Os indivíduos sentimentais respondem às situações dependendo de seus sentimentos sobre aquela situação e

muitas vezes querem um trabalho que auxilie às pessoas (CAPRETZ; VARONA; RAZA, 2015). Os perfis de analista (CAPRETZ; AHMED, 2010; CAPRETZ; VARONA; RAZA, 2015; YILMAZ *et al*., 2017) e testador (CAPRETZ; VARONA; RAZA, 2015; YILMAZ *et al*., 2017) possuem a competência sentimental, no caso avaliado a emotividade.

Tarefas como teste são abordadas de forma sistemática (CAPRETZ; AHMED, 2010). Além do que, os indivíduos organizados possuem melhor rendimento nestas tarefas. Segundo a literatura (CAPRETZ; VARONA; RAZA, 2015; YILMAZ *et al*., 2017; AQEEL IQBAL; ALDAIHANI; SHAH, 2019) testadores são mais organizados do que demais perfis analisados. As atividades como refatoração e padrões de codificação denotam traços de competências relacionadas à razão, pois é necessário conhecimento lógico, um pensamento analítico, prestar atenção aos detalhes e capacidade de resolução de problemas. Competências estas que estão associadas aos perfis de programador e *designer* (CAPRETZ; AHMED, 2010; CAPRETZ; VARONA; RAZA, 2015; WAZLAWICK, 2013).

A disciplina é uma característica de organização, segundo Michaelis (2015) sua definição é a "obediência às normas convenientes para o bom andamento dos trabalhos". Esta por sua vez é uma característica da competência de organização e que também pertence ao perfil de testador (CAPRETZ; AHMED, 2010; CAPRETZ; VARONA; RAZA, 2015; YILMAZ *et al*., 2017; AQEEL IQBAL; ALDAIHANI; SHAH, 2019).

O projeto do *design* de software é um processo exploratório. Busca-se os componentes experimentando uma variedade de esquemas para descobrir uma maneira natural e razoável de refinar uma solução. Repetições não são incomuns, já que um bom *design* geralmente requer várias iterações. Além disso, o número de iterações também depende da percepção e experiência do *designer* (CAPRETZ; AHMED, 2010). O programador deve conhecer profundamente a linguagem e o ambiente de programação (WAZLAWICK, 2013). Logo, pressupõe-se que para conhecer profundamente uma linguagem, esta será realizada repetidamente, consolidando o conhecimento adquirido. A competência racionalista está associada aos perfis de *designer* (CAPRETZ; AHMED, 2010; CAPRETZ; VARONA; RAZA, 2015; AQEEL IQBAL; ALDAIHANI; SHAH, 2019) e programador (CAPRETZ; AHMED, 2010; AQEEL IQBAL; ALDAIHANI; SHAH, 2019).

A criatividade está associada aos perfis de analista (YILMAZ *et al*., 2017; AQEEL IQBAL; ALDAIHANI; SHAH, 2019) e *designer* (CAPRETZ; AHMED, 2010; AQEEL IQBAL; ALDAIHANI; SHAH, 2019). O *designer* é criativo, imaginativo e inovador (CAPRETZ; AHMED, 2010), pois exploram/pesquisam por componentes e formas de se refinar uma solução. Enquanto que o analista é aberto a experiência, inovador, flexível e curioso (YILMAZ *et al*., 2017).

A competência comunicação está associada aos perfis de analista, programador e testador. Os quais requerem que o indivíduo trabalhe em equipe, comunique-se sendo claro e assertivo em suas discussões (YILMAZ *et al*., 2017; CAPRETZ; VARONA; RAZA, 2015).

Habilidades de resolução de problemas é uma competência inerente a indivíduos racionais (CAPRETZ; AHMED, 2010). Demais trabalhos da literatura associam os perfis *designer* e programador a competência racionalista (CAPRETZ; AHMED, 2010; CAPRETZ; VARONA; RAZA, 2015; AQEEL IQBAL; ALDAIHANI; SHAH, 2019).

A competência agradabilidade e empatia são valores associados a emoção (YILMAZ *et al*., 2017). Esta competência está associada aos perfis de (CAPRETZ; AHMED, 2010; CAPRETZ; VARONA; RAZA, 2015; YILMAZ *et al*., 2017) e testador (CAPRETZ; VARONA; RAZA, 2015; YILMAZ *et al*., 2017).

A organização é uma competência associada ao perfil de testador (CAPRETZ; AHMED, 2010; CAPRETZ; VARONA; RAZA, 2015; YILMAZ *et al*., 2017; AQEEL IQBAL; ALDAIHANI; SHAH, 2019).

A competência sensorial envolve o recebimento de informações diretamente por meio dos sentidos (JUNG, 2011). Assim como, associada aos perfis de programador (CAPRETZ; AHMED, 2010; CAPRETZ; VARONA; RAZA, 2015; AQEEL IQBAL; ALDAIHANI; SHAH, 2019) e testador (CAPRETZ; AHMED, 2010; CAPRETZ; VARONA; RAZA, 2015). Esta competência é empregada por meio de testes unitários, realizados pelo próprio programador, durante o desenvolvimento do software. E pelos testes que o testador realiza para garantir o funcionamento adequado do sistema e a correção de *bugs* e possíveis brechas de segurança.

2.3 PROCESSAMENTO DE TEXTO

O processamento de texto é uma etapa importante quando se manipula grandes quantidades de dados textuais. Ao lidar com documentos com coleções muito grandes de palavras pode ser interessante reduzir o conjunto de palavras-chave representativas (BAEZA-YATES; RIBEIRO-NETO, 2013). Isto pode ser feito eliminando-se as *stopwords* — como artigos e preposições — e aplicando-se *stemming* — reduzindo as palavras distintas a sua raiz gramatical. Além de, identificar os grupos de substantivos — eliminando adjetivos, advérbios e verbos. Estas operações reduzem a complexidade e modificam a representação do texto inteiro para um conjunto de termos de indexação, consequentemente reduzindo o custo computacional (BAEZA-YATES; RIBEIRO-NETO, 2013). Um exemplo contendo um conjunto de dados textuais originais e processados pode ser visualizado na Tabela 4.

Um termo também utilizado neste contexto é *bag-of-words*. Este é um conjunto que contém todas as palavras de um documento, com múltiplas ocorrências de uma palavra aparecendo inúmeras vezes (Witten *et al*., 2017).

Este trabalho emprega técnicas como VSM e TF-IDF, que correspondem aos algoritmos CountVectorizer[1] e TfidfTransformer[2]. VSM é um modelo IR algébrico. Ele possui uma representação de um *corpus* (conjunto de textos) em um formato de matriz de n termos versus d documentos (ABDELLATIF; CAPRETZ; HO, 2019). TF-IDF é uma medida estatística para se definir pesos para palavras dentro de um documento (DEVI; SAHARIA, 2020; PEROVŠEK *et al*., 2016). Para esta medida, o peso do termo para um determinado documento é alto, se ele aparece com alta frequência neste documento e, ao mesmo tempo, o termo é raro e tem baixa frequência dentro do *corpus* (ABDELLATIF; CAPRETZ; HO, 2019).

[1] https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer
[2] https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfTransformer

Tabela 4 – Exemplo conjunto textual original e processado.

Título original	*NaN values in reward*
Título processamento	*nan values in reward*
Labels originais	*bug, needs-info*
Labels processadas	*bug needs info*
Descrição original	*Hi, I'm training my model but at some point, a strange thing happens. The reward returned by PPO sometimes has 'nan' value. It occurs in different places (not necessarily on specific epoch) and after a few steps, it can get back to correct float values and then once again return 'nan' in a future. It doesn't apply only to _Mean Reward_, the _cumulative_reward_ shown in tensorboard is broken at this epochs as well. I'm attaching part of an output from training process. Maybe someone knows why it's happening. Thanks > Mean Reward: -82.99281905388821 > Saved Model > Mean Reward: -48.54445689334988 > Saved Model > Mean Reward: -37.32348710034999 > Saved Model > D:\ProgramData\Miniconda3\envs\unity\lib\site-packages\numpy\core\fromnumeric.py:2909: RuntimeWarning: Mean of empty slice. > out=out, **kwargs) > D:\ProgramData\Miniconda3\envs\unity\lib\site-packages\numpy\core_methods.py:80: RuntimeWarning: invalid value encountered in double_scalars > ret = ret.dtype.type(ret / rcount) > Mean Reward: nan > Saved Model > Mean Reward: nan > Saved Model > Mean Reward: nan > Saved Model*
Descrição processado	*hi training model point strange thing happens reward returned ppo sometimes nan value occurs different places necessarily specific epoch steps get back correct float values return nan future apply mean reward cumulative reward shown tensorboard broken epochs well attaching part output training process maybe someone knows happening thanks mean reward saved model mean reward saved model mean reward saved model*

Fonte: elaborado pelo autor.

O algoritmo CountVectorizer converte uma coleção de documentos textuais para uma matriz de *tokens* (PEDREGOSA *et al*., 2011), ou seja, para cada palavra

no conjunto é criada uma coluna na matriz, cada linha representa um documento textual e a associação de uma linha e coluna representa a quantidade de vezes que uma determinada palavra ocorre em um determinado documento.

Já o algoritmo TfidfTransformer transforma uma matriz de tokens em uma representação normalizada TF ou TF-IDF (PEDREGOSA *et al*., 2011). A utilização da técnica TF-IDF visa reduzir o impacto dos *tokens* que ocorrem com muita frequência em um determinado *corpus* e que são, portanto, empiricamente menos informativos do que os recursos que ocorrem em uma pequena fração do *corpus* de treinamento.

Para melhor compreensão dos algoritmos, é fornecido um exemplo do processo adotado neste trabalho. Considere o seguinte trecho, extraído de A Arte da Guerra de Sun Tzu, para análise dos algoritmos:

> "*If you know the enemy and know yourself, you need not fear the result of a hundred battles. If you know yourself but not the enemy, for every victory gained you will also suffer a defeat. If you know neither the enemy nor yourself, you will succumb in every battle.*"

A partir do trecho destacado é delimitado que cada ponto final caracteriza um documento, para fins explicativos e demonstrativos dos algoritmos CountVectorizer e TfidfTransformer. Os algoritmos utilizados no trabalho são disponibilizados pela biblioteca Sklearn[3] — uma biblioteca de ML e de código aberto para a linguagem de programação Python.

Primeiramente aplica-se o algoritmo CountVectorizer. O qual retorna a matriz exibida na Tabela 5, correspondendo às colunas Vocabulário, D_1, D_2 e D_3. As colunas D_1, D_2 e D_3 correspondem aos documentos 1, 2 e 3, anteriormente delimitados. As Equações 2.6.1, 2.6.2 e 2.6.3 são as equações referentes a Frequência do termo (TF – *Term frequency*), Frequência inversa de documento (IDF – *Inverse document frequency*) e a Frequência do termo inversa do documento (TF-IDF) (BAEZA-YATES; RIBEIRO-NETO, 2013).

$$TF_i = 1 + \log_2 \sum_{j=1}^{N} f_{i,j} \tag{2.6.1}$$

$$IDF_i = \log_2 \left(\frac{N}{n_i}\right) \tag{2.6.2}$$

[3] https://scikit-learn.org/

$$TF - IDF_i = TF_i * IDF_i \quad (2.6.3)$$

$$IDF_i = \log_{\mathcal{E}}\left(\frac{N+1}{n_i+1}\right) + 1 \quad (2.6.4)$$

Na Equação 2.6.2 é utilizado o valor 2 (dois) como base logarítmica (BAEZA-YATES; RIBEIRO-NETO, 2013). Todavia, será utilizado como base logarítmica a constante $\mathcal{E}$, mais conhecida como número de Euler (equivalente a 2,71828), visto que o algoritmo implementado na biblioteca Sklearn utiliza esta base numérica.

A Equação 2.6.4, implementada na classe TfidfTransformer, foi utilizada para cálculo do IDF. A variável N (maiúscula), usada nas Equações 2.6.1, 2.6.2 e 2.6.4, corresponde ao número de documentos. A variável i corresponde aos termos do vocabulário. Já a variável n_i (Equação 2.6.2 e 2.6.4) é correspondente ao número de documentos em que um determinado termo ocorre.

O algoritmo TfidfTransformer recebe como entrada as colunas obtidas pelo algoritmo CountVectorizer e têm como resultado a coluna TF-IDF calculada por meio da aplicação da Equação 2.6.3, como pode ser visualizado na Tabela 5.

Tabela 5 – Resultado obtido dos algoritmos CountVectorizer e TfidfTransformer.

i	Vocabulário	D_1	D_2	D_3	TF	n_i	IDF	TF-IDF
1	also	0	1	0	1	1	1,6931471	1,6931471
2	and	1	0	0	1	1	1,6931471	1,6931471
3	battle	0	0	1	1	1	1,6931471	1,6931471
4	battles	1	0	0	1	1	1,6931471	1,6931471
5	but	0	1	0	1	1	1,6931471	1,6931471
6	defeat	0	1	0	1	1	1,6931471	1,6931471
7	enemy	1	1	1	2,5849625	3	1	2,5849625
8	every	0	1	1	2	2	1,2876820	2,5753641
9	fear	1	0	0	1	1	1,6931471	1,6931471
10	for	0	1	0	1	1	1,6931471	1,6931471
11	gained	0	1	0	1	1	1,6931471	1,6931471
12	hundred	1	0	0	1	1	1,6931471	1,6931471
13	if	1	1	1	2,5849625	3	1	2,5849625
14	in	0	0	1	1	1	1,6931471	1,6931471
15	know	2	1	1	3	3	1	3
16	need	1	0	0	1	1	1,6931471	1,6931471
17	neither	0	0	1	1	1	1,6931471	1,6931471
18	nor	0	0	1	1	1	1,6931471	1,6931471
19	not	1	1	0	2	2	1,2876820	2,5753641
20	of	1	0	0	1	1	1,6931471	1,6931471
21	result	1	0	0	1	1	1,6931471	1,6931471
22	succumb	0	0	1	1	1	1,6931471	1,6931471
23	suffer	0	1	0	1	1	1,6931471	1,6931471
24	the	2	1	1	3	3	1	3
25	victory	0	1	0	1	1	1,6931471	1,6931471
26	will	0	1	1	2	2	1,2876820	2,5753641
27	you	2	2	2	3,5849625	3	1	3,5849625
28	yourself	1	1	1	2,5849625	3	1	2,5849625

Fonte: elaborado pelo autor.

2.4 SISTEMAS DE RECOMENDAÇÃO

Os sistemas de recomendação têm como objetivo analisar os dados e extrair informações úteis para futuras predições, diminuindo a sobrecarga de informações, e mostrando os itens mais interessantes e relevantes (ROBILLARD; WALKER; ZIMMERMANN, 2010). Eles são uma combinação de ferramentas de software e técnicas que fornecem sugestões de itens que podem ser usados pelo usuário (RICCI; ROKACH; SHAPIRA, 2011).

Sistemas modernos de recomendação são baseados em ML, que permite que os computadores abordem problemas envolvendo o conhecimento do mundo real e tomem decisões que parecem subjetivas (GOODFELLOW; BENGIO; COURVILLE, 2016). Tais decisões podem ser aprendidas por um algoritmo com dados suficientes (HAN; KAMBER; PEI, 2012; MITCHELL, 1997). Os modelos, gerados pela execução de ML, podem ser preditivos realizando previsões futuras ou descritivas, obtendo conhecimento dos dados, ou ambos (ALPAYDIN, 2010; HAN; KAMBER; PEI, 2012).

Os algoritmos de ML se dividem em: aprendizado supervisionado, aprendizado não supervisionado, aprendizado por reforço e aprendizado semi-supervisionado. Este trabalho utiliza dois algoritmos, MultinomialNB e *Random Forest*, classificados como aprendizado supervisionado. Esta classificação pode ser expressa como se existisse um “supervisor externo”, ou seja, a saída desejada para cada entrada já é conhecida antes de sua execução (FACELI *et al*., 2011).

2.4.1 Classificador MultinomialNB

O algoritmo MultinomialNB é um classificador Bayesiano. Classificadores Bayesianos utilizam probabilidade para refletir graus de certeza de estados de conhecimento (GOODFELLOW; BENGIO; COURVILLE, 2016). O conjunto de dados é observado diretamente, portanto, não é aleatório. Ele é um classificador estatístico, o qual pode prever a probabilidade de associação de classe (HAN; KAMBER; PEI, 2012).

Os cálculos probabilísticos desempenhados pelo classificador Bayesiano, têm como base a teoria da probabilidade condicional, proposta por Thomas Bayes. O teorema de Bayes parte do pressuposto que caso duas variáveis X e Y sejam independentes, então $P(Y|X) = P(Y)$; onde X é a representação de todos os dados e

Y a hipótese a ser verificada. As probabilidades condicionais $P(Y|X)$ e $P(X|Y)$ podem ser expressas em termos uma da outra utilizando uma fórmula conhecida por teorema de Bayes (TAN et al., 2009), veja Equação 2.4.1.

$$P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)} \tag{2.4.1}$$

Em seu teorema, $P(Y)$ representa a probabilidade de Y, já $P(X|Y)$ representa a probabilidade de X tendo ocorrido Y. Na diversidade de classificadores Bayesianos o MultinomialNB implementa o algoritmo *Naive Bayes* para dados distribuídos multinomialmente e é uma de suas variantes, utilizadas na classificação textual (MANNING; RAGHAVAN; SCHÜTZE, 2008).

Em um determinado contexto, em que $n_1, n_2, \dots, n_k$ é o número de vezes que a palavra i ocorre no documento e $P_{1,}P_2, \dots, P_K$ é a probabilidade de obter a palavra i ao fazer uma amostragem de todos os documentos da categoria H. Suponha que a probabilidade seja independente do contexto e da posição da palavra no documento. Essas suposições levam a uma distribuição multinomial para probabilidades de documentos. Para esta distribuição, a probabilidade de um documento E dada a sua classe H – a regra de Bayes para se calcular esta probabilidade $P(E|H)$, segundo Witten *et al*. (2017) pode ser visualizada na Equação 2.4.2.

$$P(E|H) = N! \times \prod_{i=1}^{d} P_i^{n_i}/n_i! \tag{2.4.2}$$

Onde $N = n_1 + n_2 + \cdots + n_k$ é o número de palavras no documento. A razão para os fatoriais é dar conta do fato de que a ordenação das ocorrências de cada palavra é irrelevante de acordo com o modelo *bag-of-words*. P_i é estimado calculando a frequência relativa da palavra i no texto de todos os documentos de treinamento pertencentes à categoria H (Witten *et al*., 2017).

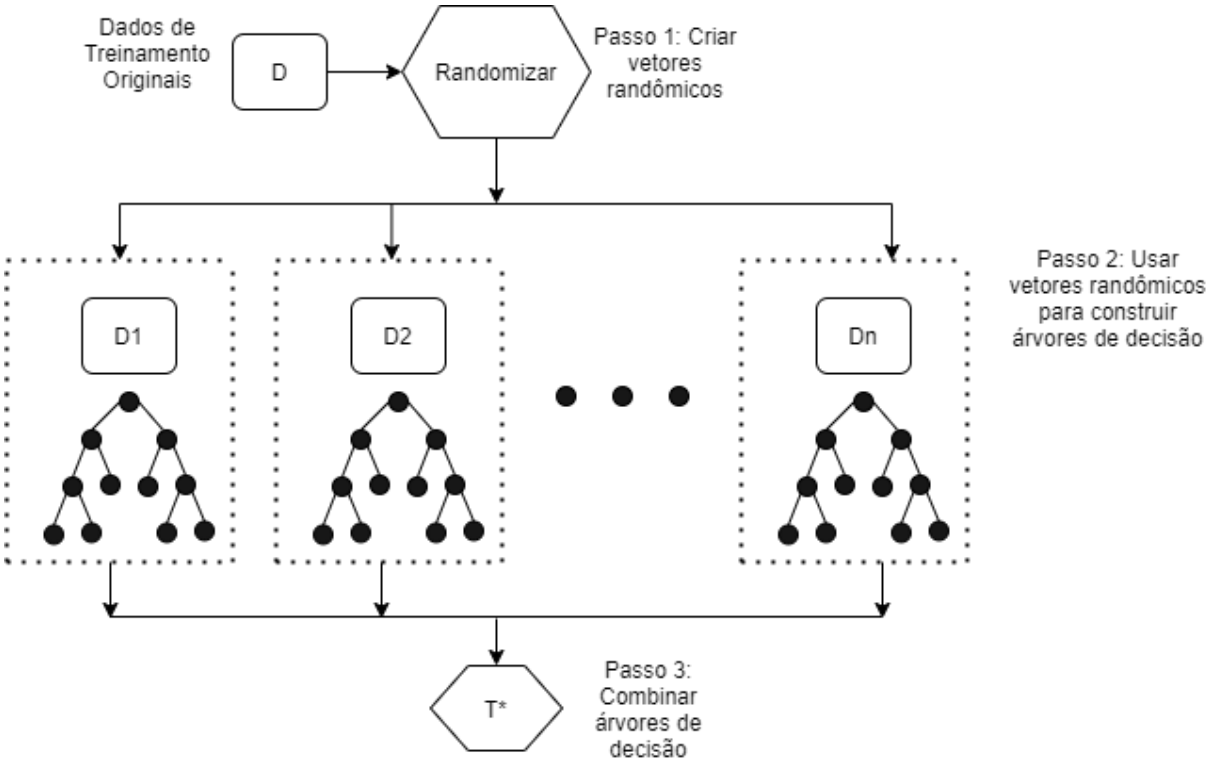
2.4.2 Classificador Random Forest

O classificador *Random Forest* combina previsões feitas por múltiplas árvores de decisão, onde cada árvore é gerada baseada nos valores de um conjunto independente de vetores aleatórios (TAN *et al*., 2009). Os vetores aleatórios são gerados a partir de uma distribuição de probabilidade fixa, podendo ser incorporado durante o processo de desenvolvimento da árvore, isto é, as árvores de decisões são geradas utilizando estes vetores aleatórios, sendo estes uma seleção aleatória

dos atributos, em cada nó para determinar a divisão (HAN; KAMBER; PEI, 2012; TAN *et al*., 2009).

Random Forest é comparável em acurácia ao algoritmo AdaBoost, assim também como é mais robusto a erros e *outliers* (HAN; KAMBER; PEI, 2012; SUTTON; BARTO, 2018). A exemplificação do funcionamento do algoritmo pode ser visualizada na Figura 1.

Figura 1 – Funcionamento de *Random Forest.*



Fonte: adaptado de (TAN *et al*., 2009).

2.4.3 Métricas de Avaliação do Desempenho

A avaliação do desempenho de um modelo de classificação é baseada na contagem de registros de testes previstos correta e incorretamente (TAN *et al*., 2009). Estas contagens são normalmente tabuladas em uma matriz denominada matriz de confusão, veja um exemplo na Tabela 6.

Tabela 6 – Exemplo matriz de confusão.

		Classe predita	
		Positivo	Negativo
Classe atual	Positivo	TP	FN
	Negativo	FP	TN

Fonte: adaptado de (HAN; KAMBER; PEI, 2012).

A diagonal principal indica a quantidade de elementos preditos corretamente. As medidas de avaliação do classificador incluem acurácia, sensibilidade (*recall*), especificidade, precisão e F_1 (HAN; KAMBER; PEI, 2012). Os autores Han, Kamber e Pei (2012) e Tan *et al*. (2009) definem a seguinte terminologia para definição dessas medidas de avaliação:

- Verdadeiro positivo (TP) — corresponde ao número de exemplos positivos preditos corretamente pelo modelo de classificação;
- Verdadeiro negativo (TN) — número de exemplos negativos preditos corretamente pelo modelo;
- Falso positivo (FP) — número de exemplos negativos preditos erroneamente como positivos pelo modelo de classificação;
- Falso negativo (FN) — número de exemplos positivos preditos erroneamente como negativos.

A determinação das métricas de avaliação é realizada por meio das seguintes equações (HAN; KAMBER; PEI, 2012):

$$acurácia = \frac{TP + TN}{TP + FN + FP + TN} \tag{2.5.1}$$

$$recall = \frac{TP}{TP + FN} \tag{2.5.2}$$

$$precisão = \frac{TP}{TP + FP} \tag{2.5.3}$$

$$f1 = \frac{2 * precisão * recall}{precisão * recall} \tag{2.5.4}$$

A acurácia também é referenciada como taxa de reconhecimento geral do classificador (Equação 2.5.1), isto é, reflete o quão bem o classificador reconhece os exemplos de teste das várias classes (HAN; KAMBER; PEI, 2012).

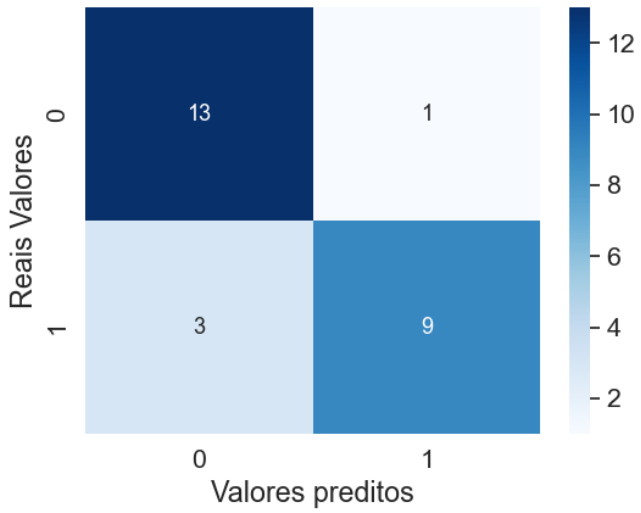
Na Equação 2.5.2 é retratada a sensitividade ou *recall*. Ela mede a integridade, ou seja, mensura a porcentagem de exemplos positivos classificados como tal (HAN; KAMBER; PEI, 2012). Classificadores com *recall* alto têm poucos exemplos positivos mal classificados (TAN *et al*., 2009).

A precisão é uma medida de exatidão, ou seja, qual a porcentagem de exemplos rotulados como positivos que realmente são (HAN; KAMBER; PEI, 2012), veja Equação 2.5.3.

Uma medida alternativa para o uso da precisão e *recall*, seria combiná-los em uma única métrica (HAN; KAMBER; PEI, 2012; TAN *et al*., 2009). Esta é a proposta da medida F (também conhecida como F_1 *score* ou *F-score*). A medida F é uma média harmônica entre a precisão e *recall* (TAN *et al*., 2009); neste caso é atribuído pesos iguais para a precisão e *recall*.

Em um exemplo prático, visualize a Figura 2, é exibido uma matriz de confusão. Os eixos X e Y da matriz correspondem aos valores preditos e aos valores reais, respectivamente. No exemplo da Figura 2 tem-se treze elementos preditos pertencentes a classe 0 (TP) e nove da classe 1 (TN). Os demais valores preditos, foram classificados erroneamente (FP e FN).

Figura 2 - Exemplo prático de uma matriz de confusão.



Fonte: elaborado pelo autor.

3 TRABALHOS RELACIONADOS

Este capítulo apresenta os trabalhos relacionados a esta proposta, sendo estes: Aslam e Ijaz (2018); Yilmaz *et al*. (2017); Otero *et al*. (2010); Farhangian *et al*. (2015); e Aqeel Iqbal, Aldaihan, Shah (2019).

O trabalho de Aslam e Ijaz (2018) trabalha com a alocação de funções aos membros de uma equipe, preencher todos os papéis, para que, logo após, sejam alocadas as tarefas a cada membro. O desenvolvimento das tarefas ocorre de forma distribuída (*distributed agile software* — DAS). O seu trabalho não utiliza diretamente um perfil, mas considera as habilidades e experiências do desenvolvedor, como habilidades de comunicação, concentração, responsabilidade, compreensibilidade, capacidade de aprendizagem, entre outros. Neste sentido, é utilizado como dados as capacidades e experiências de cada indivíduo; esta estrutura é tratada como uma matriz de n desenvolvedores e m capacidades. Não é empregado algoritmos de ML. Em seu trabalho é denotado maior ênfase no método quantitativo desenvolvido em seu trabalho, que aloca tarefas aos membros da equipe que melhor atendem aos requisitos da tarefa.

Otero *et al*. (2010) propõem uma metodologia que por meio da “desejabilidade” é realizada a alocação de desenvolvedores. A desejabilidade que os autores sugerem em seu trabalho, é utilizada para calcular o índice quantitativo de cada *skill* dos desenvolvedores e ao final calcular o índice geral de cada indivíduo. Estes índices são utilizados para realizar um *ranking*, a partir de uma lista de candidatos, e alocar as tarefas. A alocação proposta por Otero *et al*. (2010) sugere um planejamento de alocação de recursos para um projeto de software baseado em múltiplos critérios. Em sua validação, os autores utilizam um conjunto de dados sintéticos para desenvolver os testes de sua proposta, diferindo dos demais trabalhos analisados, com relação ao conjunto de dados.

O trabalho de Farhangian *et al*. (2015), desenvolve uma abordagem para selecionar membros da equipe com base em suas personalidades e habilidades. O perfil do desenvolvedor é avaliado por meio de seu desempenho, sendo este de uma equipe (fórmula proposta pelo trabalho dos autores). Ela é uma equação elaborada para calcular o desempenho da equipe, por meio dos índices de composição da personalidade e competência. A composição da personalidade é formada pela: personalidade, papel, criatividade, urgência, sociabilidade, etc. Essas competências

são habilidades/traços presentes no perfil psicológico – avaliadas pelos autores utilizando o modelo MBTI.

Aqeel Iqbal, Aldaihan, Shah (2019) realizaram um estudo, mapeando as dimensões do modelo FFM e das funções técnicas exercidas durante as atividades de desenvolvimento de software. Este trabalho não realiza alocação de tarefas, mas somente avalia os perfis técnicos e o modelo FFM. Os autores definem requisitos para cada função técnica exercida, sendo as funções: analista de sistemas, *designer*, testador e mantenedor. Os requisitos de cada função são comparados com sete competências propostas, sendo estas: comunicabilidade; relacionamento interpessoal; habilidades analíticas e de resolução de problemas; adaptável e aberto para mudanças; habilidades organizacionais; trabalha em equipe; e trabalha de forma independente. Com isto, realiza-se um mapeamento destas competências e dos requisitos necessários para cada função.

O trabalho de Yilmaz *et al*. (2017) iniciou seu estudo com base em uma abordagem de criação de perfil, dinamicamente, isto é, as perguntas realizadas para se avaliar o perfil, mudam consoante as respostas do participante. Sua pesquisa se baseou no modelo de avaliação FFM. O objetivo de seu estudo foi revelar os traços de personalidade dos profissionais de software com o intuito de explorar as estruturas das equipes de software eficazes. Os autores exploraram a importância da personalidade dos desenvolvedores em uma equipe, através de uma abordagem empírica. A sua abordagem avalia os traços de personalidade dos profissionais de software, através de um *survey*.

Na Tabela 7 é apresentada uma comparação dentre esta proposta e os trabalhos relacionados. Este trabalho utiliza competências inerentes aos modelos psicológicos MBTI e FFM, contudo não os modelos psicológicos. Estas competências são associadas com os perfis técnicos. Nos trabalhos de Aqeel Iqbal, Aldaihan, Shah (2019) e Yilmaz *et al*. (2017) o modelo FFM é utilizado. No trabalho de Farhangian *et al*. (2015) é empregado o modelo MBTI. Em vista da diversidade de trabalhos utilizando diferentes modelos psicológicos, um modelo de conversão entre os modelos psicológicos é empregado. Este modelo é proposto por Furnham, Moutafi e Crump (2003). Com isto, é padronizado os modelos utilizados pelos demais trabalhos para um único, desta forma, facilitando a análise que os trabalhos desempenharam com relação ao perfil e modelo psicológico.

Tabela 7 – Comparativo dentre os trabalhos relacionados.

	Trabalhos relacionados					
	(ASLAM; IJAZ, 2018)	(OTERO et al., 2010)	(FARHANGIAN et al., 2015)	(AQEEL IQBAL; ALDAIHANI; SHAH, 2019)	(YILMAZ et al., 2017)	Nossa proposta
Dados técnicos	X	X	X			X
Competências	X		X	X	X	X
Baseia-se no modelo psicológico			MBTI	FFM	FFM	
Modelo	Modelo matemático	Abordagem proposta desejabilidade	Modelo matemático	Modelo teórico	Modelo empírico	Abordagem RecomTask
Alocação de tarefas	Sim (equipe)	Sim (individual)	Sim (equipe)	Não	Não	Sim (individual)

Fonte: elaborado pelo autor.

A proposta deste trabalho é recomendar tarefas para serem cumpridas pelos desenvolvedores, da mesma forma que demais trabalhos (ASLAM; IJAZ, 2018; FARHANGIAN *et al*., 2015; OTERO *et al*., 2010). Com relação à análise dos dados históricos utilizados pelos trabalhos relacionados, apenas o trabalho de Aslam e Ijaz (2018) utilizou dados históricos. Em Farhangian *et al*. (2015) a alocação de tarefas ocorre segundo os dados disponíveis, como os dados técnicos e as competências. Já em Otero *et al*. (2010) apenas os dados técnicos são utilizados. Por outro lado, os trabalhos de Yilmaz *et al*. (2017) e Aqeel Iqbal, Aldaihani e Shah (2019) não trabalham com a alocação de tarefas, mas sim analisando as competências para os membros da equipe, isto é, utilizando ainda o perfil técnico de cada membro para realizar as suas análises. No trabalho desenvolvido por Otero *et al*.(2010) é

considerado as habilidades técnicas dos membros da equipe para alocar tarefas e não as suas competências.

Este trabalho elabora uma abordagem para a alocação de tarefas. Para isto, são empregadas técnicas de ML, mineração de dados e processamento de texto. A alocação é feita por meio dos algoritmos *Random Forest* e MultinomialNB. Esta proposta tem como base a utilização dos dados dos desenvolvedores e das tarefas, para isto, é empregado um questionário, por meio do qual, é realizado o mapeamento do perfil e as competências dos desenvolvedores; buscando quantificar as informações a respeito de cada sujeito. As competências utilizadas consistem de um conjunto inerente aos modelos psicológicos, veja a Tabela 8.

Tabela 8 - Relação das dimensões do modelo MBTI e as competências propostas.

Dimensões do modelo MBTI	**Competências**
Extroversão	comunicativo
Introversão	reservado
Intuição	criatividade
Sensitivo	sensorial
Racional	racionalista
Sentimental	emotivo
Percepção	espontaneidade
Julgador	organizado

Fonte: elaborado pelo autor.

As competências propostas por este trabalho são: comunicativo, reservado, criatividade, sensorial, racionalista, emotivo, espontaneidade e organizado. A competência comunicativo está relacionada à dimensão extroversão, reservado à introversão, criatividade a intuição, sensorial a sensitivo, racionalista a racional, emotivo a sentimental, espontaneidade à percepção e, finalmente, organizado a julgador.

A proposta deste trabalho para com a relação as competências e os perfis de desenvolvedor, pode ser visualizada na Tabela 9. Nesta adaptou-se a proposta dos trabalhos analisados, para a realizada neste trabalho (Tabela 8) com o auxílio do modelo de conversão entre os modelos psicológicos (FURNHAM; MOUTAFI; CRUMP, 2003).

Tabela 9 – Competências propostas quanto a este trabalho e trabalhos relacionados.

	Perfis de desenvolvedor			
Trabalhos	analista de sistemas	designer	programador	testador
(CAPRETZ; AHMED, 2010)	comunicativo emotivo	criatividade racionalista	reservado sensorial racionalista	sensorial organizado
(CAPRETZ; VARONA; RAZA, 2015)	comunicativo emotivo sensorial espontaneidade	reservado racionalista sensorial organizado	comunicativo emotivo sensorial espontaneidade	comunicativo racionalista/emotivo sensorial organizado
(YILMAZ et al., 2017)	comunicativo emotivo criatividade	-	comunicativo criatividade organizado	comunicativo emotivo organizado
(AQEEL IQBAL; ALDAIHANI; SHAH, 2019)	comunicativo criatividade organizado	criatividade racionalista	reservado sensorial racionalista	comunicativo criatividade organizado
Nossa proposta	comunicativo criatividade emotivo	racionalista criatividade	comunicativo sensorial racionalista	comunicativo sensorial emotivo organizado

Fonte: elaborado pelo autor.

Como pode ser visualizado na Tabela 9, o conjunto final de competências utilizadas compreende: comunicativo, criatividade, sensorial, racionalista, emotivo e organizado. Visto que, as competências reservado e espontaneidade não foram associadas por mais da metade da associação perfil e trabalho.

Os dados das tarefas são extraídos de repositórios de projetos de desenvolvimento de software, servindo para validar esta proposta. Os perfis técnicos são os papéis descritos na literatura de Engenharia de software que são os mesmos que os perfis de desenvolvedor. A diferença é que a quantificação do perfil do desenvolvedor, segundo essa proposta, considera as competências desejáveis para o perfil. Para cada perfil, é possível definir as competências do desenvolvedor de acordo com trabalhos descritos na literatura (AQEEL IQBAL; ALDAIHANI; SHAH,

2019; CAPRETZ; AHMED, 2010; CAPRETZ; VARONA; RAZA, 2015; YILMAZ *et al*., 2017): analista de sistemas, *designer*, programador e testador.

4 RECOMTASK - ABORDAGEM PARA RECOMENDAÇÃO DE TAREFAS

A abordagem para recomendação de tarefas (RecomTask) atribui tarefas adequadas ao perfil de desenvolvedor. O perfil de desenvolvedor é composto pelo perfil técnico e pelas suas competências necessárias. A alocação de tarefas desenvolvida divide-se em dois momentos principais: (1) na classificação das tarefas, em qual o perfil de desenvolvedor mais adequado, de acordo com o processamento textual e o resultado obtido do classificador MultinomialNB 1; (2) na alocação das tarefas, a qual emprega os classificadores MultinomialNB 2 e *Random Forest*. Alocando as tarefas para os indivíduos, segundo o perfil de desenvolvedor destes e o resultado obtido do classificador MultinomialNB 1.

Esta seção foi organizada em quatro subseções, sendo estas: a elaboração do questionário (Subseção 4.1); a preparação dos dados (Subseção 4.2); alocação de tarefas (Subseção 4.3); e avaliação das alocações (Subseção 4.4).

4.1 ELABORAÇÃO DO QUESTIONÁRIO

O questionário é composto por duas partes: questões técnicas e questões sobre competências e personalidade. A primeira parte é composta de um conjunto de perguntas elaboradas a partir dos trabalhos Capretz, Varona, Raza (2015) e Yilmaz *et al*. (2017), veja Tabela 10.

Tabela 10 – Primeira parte do questionário.

Pergunta	**Resposta esperada**
E-mail	*e-mail* do participante
Está de acordo com a pesquisa?	Participante responde se está de acordo ou não.
Qual é o seu cargo na organização em que trabalha? Caso o seu cargo não esteja na lista, selecione aquele que mais se aproxima.	Analista, *designer*, programador ou testador.
Há quanto tempo você exerce está função?	Tempo exercido pelo participante, a função.

Fonte: elaborado pelo autor.

A pergunta “Qual é o seu cargo na organização em que trabalha?” visa identificar qual o cargo que o indivíduo exerce ou o mais próximo ao conjunto de papéis sugeridos. Com isto, é avaliado/comparado ao “perfil” indicado pela proposta elaborada. A pergunta seguinte “Há quanto tempo você exerce está função?” visa identificar o tempo. Com isto pode-se identificar as situações em que um indivíduo exerça um cargo, mas que possua outro “perfil” dominante, contrário ao informado inicialmente. Assim também, notar demais atividades que este sujeito esteja apto ou que possa receber atividades correlatas.

Na segunda parte é avaliada as competências e a personalidade. A resposta de cada pergunta é compreendida em um valor dentro de uma escala (1-5). As competências empregadas consistem no conjunto exibido na Tabela 8.

As descrições técnicas foram extraídas de trabalhos da literatura de Engenharia de software (BOURQUE; FAIRLEY, 2014; PRESSMAN, 2011; SOMMERVILLE, 2011; WAZLAWICK, 2013) e de aspectos humanos no desenvolvimento de software (CAPRETZ; AHMED, 2010; CAPRETZ; VARONA; RAZA, 2015; YILMAZ *et al*., 2017; AQEEL IQBAL; ALDAIHANI; SHAH, 2019), segundo a fundamentação teórica realizada na Subseção 2.2.3.

A segunda parte do questionário é apresentada na Tabela 11. Este conjunto compreende vinte e uma questões. O qual é realizado uma divisão para melhor distinção entre o conjunto de perguntas utilizado para avaliar o perfil técnico e as competências. Nas perguntas de um a oito é avaliado o perfil técnico e deve ser respondido usando a seguinte escala: (1) atividade inadequada ao meu perfil; (2) em caso de necessidade poderia executar a atividade; (3) neutro em relação à atividade; (4) atividade parcialmente adequada ao perfil; e (5) atividade totalmente adequada ao perfil. Já as competências são avaliadas nas perguntas nove a vinte e um e usando a seguinte escala: (1) não possuo a habilidade; (2) pouca habilidade; (3) habilidade mediana; (4) habilidade acima da média; e (5) altamente proficiente.

Tabela 11 – Questionário perfil.

Pergunta	Descrição
P1	Definir o escopo e as funcionalidades do sistema, por meio da interação com o cliente/usuário.
P2	Possuir boas habilidades de comunicação e interpessoais para interagir com os clientes/usuários e os desenvolvedores.
P3	Avaliar um cenário completo, mas também isolar itens relevantes de grandes quantidades de dados vagos e imprecisos, produzindo assim um modelo que dê suporte às atividades do programador.
P4	Processo exploratório e a busca por componentes, avaliando uma variedade de esquemas para identificar o modo mais natural e razoável de refinar uma solução.
P5	Ser lógico e atento aos detalhes, buscando cumprir com as especificações dos projetistas.
P6	Facilidade em utilizar padrões de codificação, estruturas de controle, estruturas de dados, IDEs e um repositório para o armazenamento do projeto.
P7	Executar testes, registrar os resultados e investiga-los são algumas tarefas exercidas durante e após o desenvolvimento. A definição e implementação de planos e estratégias de teste de maneira sistemática, auxilia a correção e mitigação de erros e *bugs*.
P8	Garantir que o progresso, as metodologias e as ferramentas de teste sejam aplicadas adequadamente e que os critérios de entrada/saída dos testes estejam de acordo com os casos de teste. Deste modo, nota-se a importância da identificação e mitigação dos riscos técnicos e de negócios no desenvolvimento e execução das estratégias adotadas para o teste.
P9	Criar cenários, *storyboards*, estruturas de informação, recursos e interfaces criativamente.
P10	A organização para os testes deve seguir uma abordagem sistemática para identificar os erros por meio de um processo manual ou automatizado de testes.
P11	Transmitir informações de forma clara e objetiva para seus colegas e

	subordinados.
P12	Simpatizar com os problemas dos *stakeholders*, compreendendo as necessidades para o produto, é uma habilidade interpessoal altamente desejável.
P13	Executar tarefas tais como: testes, refatoração, integração contínua e padrões de codificação.
P14	Trabalhar de forma independente e disciplinada.
P15	Trabalhar repetidamente em uma atividade faz com que seja fixado e consolidado o conhecimento.
P16	Você é alguém inovador, possui capacidade de apresentar novas soluções criativas?
P17	Interagir com outras pessoas, ser assertivo e ativo em suas atividades.
P18	Compreender, articular, resolver problemas complexos e tomar decisões sensatas com base nas informações disponíveis são habilidades analíticas e de resolução de problemas.
P19	Ser agradável, atencioso ou gentil são alguns valores apreciados e costumam denotar um indivíduo simpático.
P20	Fornecer, manter e atualizar a documentação dos sistemas, refletindo novos aplicativos ou melhorias para os aplicativos existentes denotam um perfil de um indivíduo organizado.
P21	Formular casos de teste para testar o software em desenvolvimento, garantindo que a funcionalidade de um programa corresponda aos requisitos de negócio. Assim como, assegurar que os padrões de programação sejam seguidos são práticas que contribuem para a conclusão do desenvolvimento de um software.

Fonte: elaborado pelo autor.

Cada pergunta avalia um perfil técnico ou competência. O seu mapeamento pode ser visualizado na Tabela 12. Com relação às perguntas utilizadas para se avaliar as competências, sua descrição foi elaborada associando uma única competência por pergunta. Há somente uma única exceção na pergunta 13, avalia-se três competências. Todas estas, estão relacionadas à descrição da pergunta e

aos elementos que denotam a presença destas competências: organizado, racionalista e sensorial.

Tabela 12 - Relação dos atributos e as questões que os avalia.

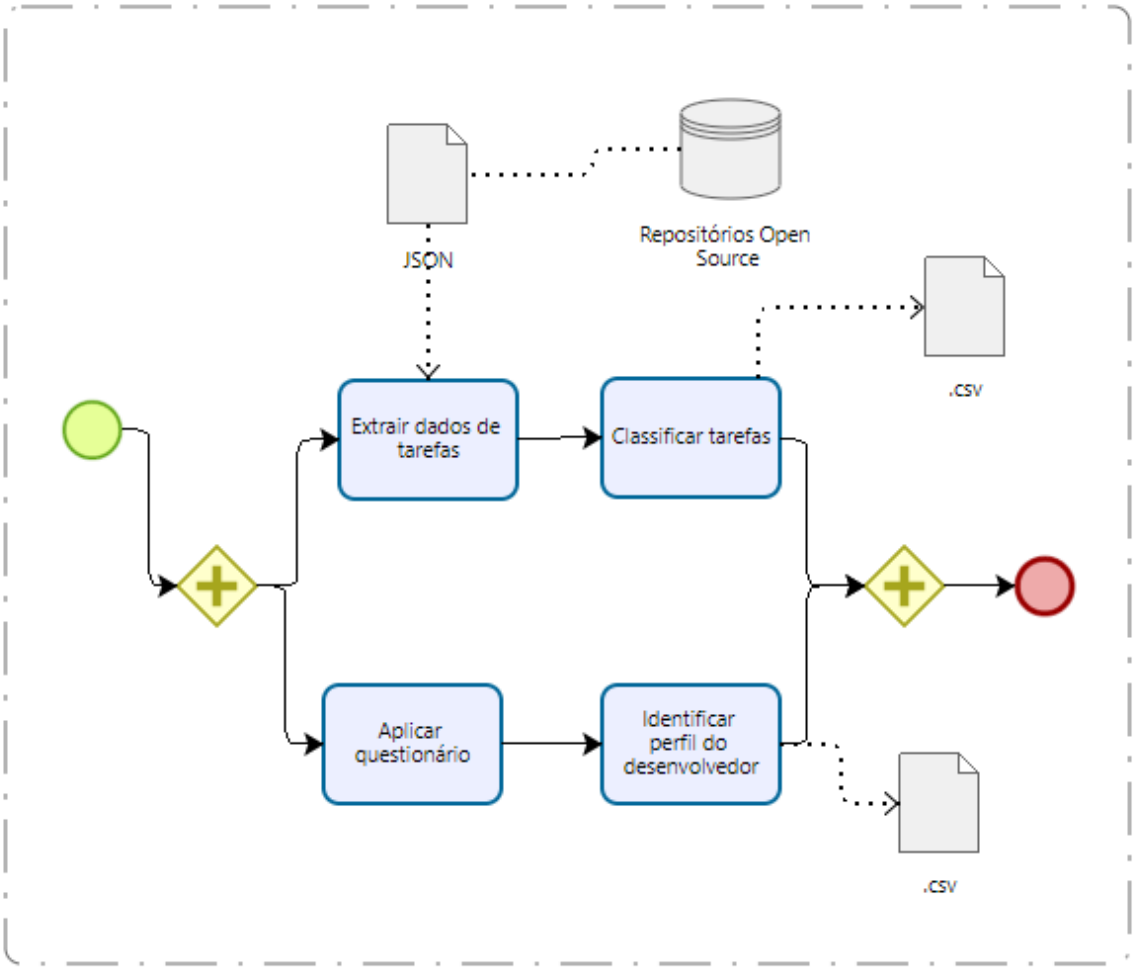
Questões/Atributos	analista de sistemas	designer	programador	testador	comunicativo	criatividade	sensorial	racionalista	emotivo	organizado
P1	X									
P2	X									
P3		X								
P4		X								
P5			X							
P6			X							
P7				X						
P8				X						
P9						X				
P10										X
P11					X					
P12									X	
P13							X	X		X
P14										X
P15								X		
P16						X				
P17					X					
P18								X		
P19									X	
P20										X
P21							X			

Fonte: elaborado pelo autor.

4.2 PREPARAÇÃO DOS DADOS

A preparação de dados consiste na obtenção dos dados necessários ao algoritmo de recomendação de tarefas, sendo estes os dados referentes às tarefas e o perfil de desenvolvedor. As atividades que compõem esta etapa podem ser visualizadas na Figura 3.

Figura 3 - Diagrama do processo “Preparação dos dados".



Fonte: elaborado pelo autor.

Na atividade “Extrair dados de tarefas” foram extraídos dados relativos a tarefas de desenvolvimento de softwares. Na Tabela 13 podem ser visualizados os atributos dos dados obtidos nesta atividade. Esses são atributos são utilizados para definir tarefas, chamadas *issues*, em repositórios de código.

Tabela 13 - Dados extraídos das *issues*.

Nome da coluna	**Descrição**
language	Linguagens de programação utilizadas
issue_title	Título da *issue*/tarefa
type	O tipo da atividade: *Change request, Iteration, Phase, Task, User story*
labels	Rótulos/palavras-chaves
status	Estados possíveis da atividade: *Approved, Closed, Developed, In progress, In testing, New, Rejected, Scheduled, To be scheduled*
description	Descrição da tarefa
assignee	A quem foi atribuída a tarefa
time_spent	Tempo gasto para concluir a tarefa

Fonte: elaborado pelo autor.

A atividade “Classificar tarefas” teve como objetivo atribuir rótulos para as tarefas armazenadas em formato textual. Para tanto, utilizou-se os dados extraídos na atividade anterior e adicionou-se as colunas *classification* e *language_text_description*. A descrição destas colunas pode ser visualizada na Tabela 14. Convencionou-se atribuir nulo para os campos não disponíveis.

Tabela 14 - Colunas adicionadas.

Nome	**Descrição**
classification	Classificação da tarefa no seguinte conjunto: *system analysts*, *designer*, *programmer* e *tester*.
language_text_description	Armazena o idioma que a tarefa foi escrita.

Fonte: elaborado pelo autor.

A coluna *classification* recebeu valores categóricos, como analista de sistemas, *designer*, programador ou testador, para indicar o perfil de desenvolvedor apropriado para a tarefa. Esse rótulo também é usado para treinar os algoritmos de classificação e recomendação, para isto, foi atribuído rótulos para 271 tarefas, segundo o perfil do desenvolvedor mais adequado. Essa classificação considerou as

informações referentes à tarefa (título, *labels* e descrição) e a literatura de Engenharia de Software (BOURQUE; FAIRLEY, 2014; PRESSMAN, 2011; SOMMERVILLE, 2011; WAZLAWICK, 2013).

Na coluna *language_text_description* é armazenado o idioma da tarefa, visto que, não havia informação prévia a respeito. Utilizaram-se as bibliotecas textblob[4] e googletrans[5] da linguagem de programação Python para analisar o idioma; usando os dados das colunas título, *labels* e a descrição da tarefa para identificar o idioma de cada tarefa. Este dado serve como parâmetro no processo de eliminação de *stopwords* e *stemming*, realizado para classificar as tarefas.

Para mapear o perfil do desenvolvedor foram definidas duas atividades: “Aplicar o questionário” e “Identificar o perfil do desenvolvedor”.

Visando identificar o perfil de cada desenvolvedor, na atividade “Aplicar questionário”, os desenvolvedores devem responder um conjunto de questões que têm como finalidade identificar o perfil técnico e de suas competências.

O questionário elaborado por esta proposta é descrito na Subseção 4.1 e pode ser visualizado na Tabela 11.

Como um exemplo, na Tabela 15, são exibidas as questões usadas para avaliar o perfil de analista de sistemas. Com as questões P1 e P2 se visa avaliar as atividades práticas e concernentes exercidas pelo profissional responsável pelo cargo de analista de sistemas. As questões P9, P11, P12, P16, P17 e P19 avaliam as competências: comunicativo, criatividade e emotivo.

[4] https://textblob.readthedocs.io/en/dev/
[5] https://pypi.org/project/googletrans/

Tabela 15 – Lista de questões usadas na análise do perfil analista de sistemas.

Questão	Descrição	Perfil técnico	Competência
P1	Definir o escopo e as funcionalidades do sistema, por meio da interação com o cliente/usuário.	analista de sistemas	
P2	Possuir boas habilidades de comunicação e interpessoais para interagir com os clientes/usuários e os desenvolvedores.	analista de sistemas	
P9	Criar cenários, *storyboards*, estruturas de informação, recursos e interfaces criativamente		criatividade
P11	Transmitir informações de forma clara e objetiva para seus colegas e subordinados.		comunicativo
P12	Simpatizar com os problemas dos stakeholders, compreendendo as necessidades para o produto, é uma habilidade interpessoal altamente desejável.		emotivo
P16	Você é alguém inovador, possui capacidade de apresentar novas soluções criativas?		criatividade
P17	Interagir com outras pessoas, ser assertivo e ativo em suas atividades.		comunicativo
P19	Ser agradável, atencioso ou gentil são alguns valores apreciados e costumam denotar um indivíduo simpático.		emotivo

Fonte: elaborado pelo autor.

Na atividade “Identificar perfil do desenvolvedor”, as respostas do questionário são utilizadas para mapear o perfil de desenvolvedor por meio da aplicação de equações matemáticas elaboradas por esta proposta, fundamentando-se na relação atributo e questão exibida na Tabela 12.

Nas equações elaboradas é realizado um mapeamento de cada um dos perfis e das competências do indivíduo. As equações possuem pesos. O valor é informado pelo gerente de projetos, sendo um peso relacionado às competências e outro ao perfil técnico, por exemplo, caso o peso do perfil técnico seja 60%, as competências terão um peso de 40% consequentemente.

As equações elaboradas por este trabalho baseiam-se em médias ponderadas, veja o intervalo de Equações 4.1.1 a 4.1.10. Elas são utilizadas para calcular os valores de cada competência e perfil de desenvolvedor.

$$analista = \left(\frac{P1 + P2}{2} * PT\right) + \left(\frac{comunicativo + criatividade + emotivo}{3}\right) \quad (4.1.1)$$

$$designer = \left(\frac{P3 + P4}{2} * PT\right) + \left(\frac{criatividade + racionalista}{2}\right) \quad (4.1.2)$$

$$programador = \left(\frac{P5 + P6}{2} * PT\right) + \left(\frac{comunicaitvo + sensorial + racionalista}{3}\right) \quad (4.1.3)$$

$$testador = \left(\frac{P7 + P8}{2} * PT\right) + \left(\frac{comunicativo + sensorial + emotivo + \ organizado}{4}\right) \quad (4.1.4)$$

$$comunicativo = \left(\frac{P11 + P17}{2}\right) * PC \quad (4.1.5)$$

$$criatividade = \left(\frac{P9 + P16}{2}\right) * PC \quad (4.1.6)$$

$$sensorial = \left(\frac{P13 + P21}{2}\right) * PC \quad (4.1.7)$$

$$racionalista = \left(\frac{P13 + P15 + P18}{3}\right) * PC \quad (4.1.8)$$

$$emotivo = \left(\frac{P12 + P19}{2}\right) * PC \quad (4.1.9)$$

$$organizado = \left(\frac{P10 + P13 + P14 + P20}{4}\right) * PC \quad (4.1.10)$$

Por exemplo, na Equação 4.1.1 pode ser visualizada a equação utilizada para calcular o perfil de desenvolvedor relacionado ao analista de sistemas. A primeira parte da equação quantifica o perfil técnico. A segunda parte da equação avalia as

competências desejáveis a um analista de sistemas, no caso em especifico as competências comunicativo, criatividade e emotivo. Para cada competência existe um conjunto de questões para sua avaliação, conforme mostrado nas Equações 4.1.5, 4.1.6 e 4.1.9. O cálculo final é uma média ponderada, com a atribuição dos pesos para o perfil técnico (PT) e para as competências (PC).

Os pesos definidos devem ter um valor total de 1, ou seja, se somado as variáveis “PT” (peso técnico) e “PC” (peso competência) da fórmula, deverá ser obtido um valor total de 1 (um).

Caso o sujeito tenha respondido, os seguintes valores 4, 5, 3, 4, 5, 3, 4 e 5 para P1, P2, P9, P11, P12, P16, P17 e P19 e o gerente tenha definido os pesos 0,6 e 0,4 para “PT” e “PC”, respectivamente, ao aplicar-se a Equação 4.1.1 para calcular o perfil de desenvolvedor analista de sistemas, obtém-se o valor 4,03; veja os passos tomados nas Equações 4.1.11 a 4.1.14. Da mesma forma, é realizado para os demais perfis técnicos.

Quanto maior o valor numérico obtido em uma determinada equação, maior a probabilidade do desenvolvedor (sujeito) possuir o perfil ou a competência que está sendo avaliada.

$$analista = \left(\frac{P1 + P2}{2} * PT\right) + \frac{\left(\left(\frac{P11 + P17}{2}\right) * PC\right) + \left(\left(\frac{P9 + P16}{2}\right) * PC\right) + \left(\left(\frac{P12 + P19}{2}\right) * PC\right)}{3} \quad (4.1.11)$$

$$analista = \left(\frac{4 + 5}{2} * 0{,}6\right) + \frac{\left(\left(\left(\frac{4 + 4}{2}\right) * 0{,}4\right) + \left(\frac{3 + 3}{2} * 0{,}4\right) + \left(\frac{5 + 5}{2} * 0{,}4\right)\right)}{3} \quad (4.1.12)$$

$$analista = (2{,}7) + \frac{(1{,}6) + (1{,}2) + (2)}{3} \quad (4.1.13)$$

$$analista = 4{,}033 \quad (4.1.14)$$

Sendo concluídas as atividades exibidas na Figura 3, os dados referentes às tarefas e aos desenvolvedores estão delineados e prontos para serem utilizados na

fase seguinte, isto é, o conjunto de dados com os dados das tarefas e o conjunto de dados com os dados dos desenvolvedores. O conjunto de dados dos desenvolvedores é composto pelas colunas exibidas na Tabela 16.

Tabela 16 – Conjunto de dados do CSV de desenvolvedores.

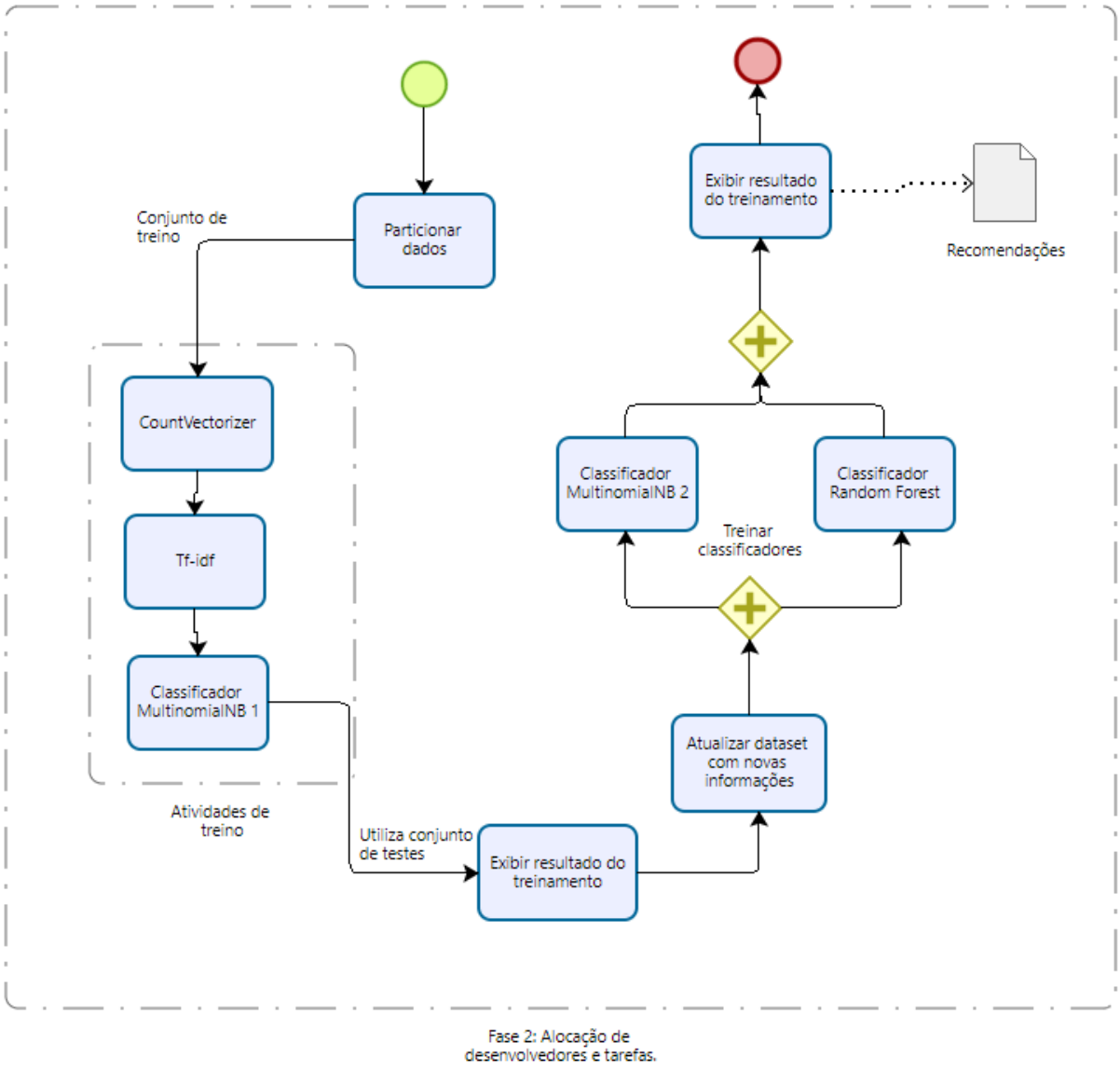
Nome	**Descrição**
User name	Nome do desenvolvedor padronizado para o padrão dev{i}, sendo i o valor numérico no intervalor de 0 a infinito.
Communicative	Valor obtido pela aplicação da equação comunicativo.
Creativity	Valor obtido pela aplicação da equação criatividade.
Sensory	Valor obtido pela aplicação da equação sensorial.
Rationalist	Valor obtido pela aplicação da equação racionalista.
Emotional	Valor obtido pela aplicação da equação emotivo.
Organized	Valor obtido pela aplicação da equação organizado.
System analysts	Valor obtido pela aplicação da equação analista de sistemas.
Designer	Valor obtido pela aplicação da equação *designer*.
Programmer	Valor obtido pela aplicação da equação programador.
Tester	Valor obtido pela aplicação da equação testador.
Predominant_value	Dentre as funções técnicas, armazena o índice de maior valor (0 a 3). Corresponde ao perfil de desenvolvedor predominante.

Fonte: elaborado pelo autor.

4.3 ALOCAÇÃO DE TAREFAS

A etapa relativa à alocação de desenvolvedores e tarefas visa sugerir tarefas aos membros da equipe, segundo o perfil do desenvolvedor e os dados da tarefa, calculados na etapa anterior. A Figura 4 mostra o processo e as atividades envolvidas nesta etapa.

Figura 4 - Diagrama do processo “Alocação de desenvolvedores e tarefas”.



Fonte: elaborado pelo autor.

Na atividade “Particionar dados”, o conjunto de dados classificado manualmente é utilizado. Para isto, é levado em consideração os valores válidos (coluna *classification* diferente de nulo). Esta etapa pode ser visualizada na linha 6 do Algoritmo 1.

Esta amostra — tarefas classificadas — é particionada, em uma razão 70 – 30%, (linha 10 do Algoritmo 1).

Note que na Figura 4, há um conjunto de atividades agrupadas, denominadas “Atividades de treino”. Estas atividades aplicam técnicas de processamento textual sobre o conjunto de dados das tarefas. Este conjunto de dados corresponde a coluna *text_tfidf* (colunas *issue_title*, *labels* e *descript_p* já pré-processadas). Logo

após, é treinado o classificador para identificar o perfil de desenvolvedor mais adequado (atividade “Classificador MultinomialNB 1”, vide subseção 2.4.1). Assim que, o processamento textual realizado pelas atividades “CountVectorizer” e “TF-IDF” (vide Subseção 2.3) é concluído.

A atividade “CountVectorizer” transforma o conjunto de tarefas em uma matriz de n termos por m documentos, sendo n o conjunto total de termos presentes em todos os documentos. Nesta matriz é armazenado o número de vezes que um termo está presente em um dado documento. Esta matriz é utilizada na atividade “TF-IDF”. No qual visa-se reduzir o impacto dos *tokens* que ocorrem com maior frequência. Os dados obtidos do algoritmo TF-IDF são utilizados na atividade “Classificador MultinomialNB 1”. No qual é aplicado o algoritmo MultinomialNB no resultado obtido do algoritmo TF-IDF e sobre os rótulos definidos para as tarefas manualmente. Este processo é realizado visando treinar o primeiro classificador para identificar o perfil de desenvolvedor para as tarefas.

Algoritmo 1 Método execute_training

```
1.  def execute_training(self, dataframe):
2.      self.vocabulary = Utils().gen_vocabulary(dataframe.loc[:, ['issue_title', 'descrip_p',
    'labels']])
3.      self.train = Pipeline([('vect', CountVectorizer(vocabulary=self.vocabulary)),
4.              ('tfidf', TfidfTransformer()),
5.              ('clf', MultinomialNB())])
6.      sample_train, _ = Utils().split_train_test_dataframe(dataframe)
7.
8.      sample_train = self.function_auxiliar_concatenate_text_fields(sample_train)
9.
10.     x_train, x_test, y_train, y_test = train_test_split(list(sample_train.loc[:, 'text_tfidf']),
11.             list(sample_train.loc[:, 'classification']), test_size=0.3, random_state=423)
12.     sample_weight = compute_sample_weight('balanced', y_train)
13.     self.train.fit(x_train, y_train, clf__sample_weight=sample_weight)
14.     self.training_and_test_summary(x_test, y_test)
```

Fonte: elaborado pelo autor.

Na atividade “Exibir resultado do treinamento” exibe-se os resultados obtidos das métricas: acurácia, *recall*, $f1$ e suporte, para cada um dos perfis de desenvolvedor, como também uma matriz de confusão e a acurácia geral do

classificador MultinomialNB aplicado na atividade “Classificador MultinomialNB 1”. Na Seção 5 é exibido os valores obtidos durante esta etapa. Este comportamento é relativo ao método *training_and_test_summary* do Algoritmo 1 (linha 14). Estes valores são relativos ao conjunto de dados particionado inicialmente (atividade “Particionar dados”) que não foi utilizado, de 30% das tarefas rotuladas manualmente.

Durante a atividade “Atualizar *dataset* com novas informações” é compreendida a correção ou adição de novas informações sobre a classificação textual, conforme o possível perfil de desenvolvedor, correspondente à coluna *classification* do conjunto de dados. Neste caso, o conjunto de tarefas que não havia sido rotulado manualmente é utilizado. Com isto, o classificador MultinomialNB treinado na atividade “MultinomialNB 1”, é empregado para identificador qual o perfil de desenvolvedor sugerido para as tarefas ainda não rotuladas.

Após, é realizado o treinamento do segundo classificador para atribuir as tarefas aos desenvolvedores. Esses classificadores (“Classificador MultinomialNB 2” e “Classificador *Random Forest*”) são usados para recomendar tarefas para os indivíduos, de acordo com a maior precisão obtida. O algoritmo MultinomialNB utiliza probabilidade para verificar a relação entre duas variáveis avaliadas, ou seja, a tarefa classificada e o perfil de desenvolvedor de um membro da equipe. O algoritmo *Random Forest* calcula a importância dos atributos utilizados para realizar o treinamento. Os dados utilizados são os perfis calculados, o perfil de desenvolvedor predominante (maior índice), as competências utilizadas para se calcular o perfil do desenvolvedor e a classificação da tarefa resultante do primeiro classificador.

No Algoritmo 2 é exibido a função *execute_training,* utilizada para treinar estes classificadores. Comumente os dados, utilizados nesta etapa para treinar o classificador, correspondem a coluna *User name* — valor a ser predito — e as seguintes colunas são utilizadas como conjunto de dados de entrada: *classification; communicative; creativity; sensory; rationalist; emotional; organized; system analysts; designer; programmer; tester;* e *predominant_value*.

Algoritmo 2 Método execute_training da classe Training_skills

```
1.  def execute_training (self, df_train, stack_values=None, target="User name"):
2.          if target in self.columns:
3.              self.columns.remove(target)
4.
5.          list_targets = np.unique(df_train.loc[:, target])
6.          x_train, x_test, y_train, y_test = train_test_split(df_train.loc[:, self.columns],
7.                      list(df_train.loc[:, target]), test_size=0.3, random_state=423)
8.
9.          for mClassifier in self.vTrain:
10.             mClassifier.fit(x_train, y_train)
11.
12.         self.resume_of_train_and_test_text(list_targets, x_test, y_test)
13.         if stack_values:
14.             self.initialize_stack(stack_values)
15.         else:
16.             self.initialize_stack(self.train.classes_)
```

Fonte: elaborado pelo autor.

Durante o teste do conjunto de dados de tarefas e desenvolvedores utiliza-se a divisão 70-30, como pode ser notado nas linhas 6 e 7. Na linha 12 é chamado o método responsável por exibir o resultado obtido pelos classificadores. Além de, determinar o classificador a ser usado para realizar as recomendações e indicar a importância dos atributos utilizados – obtido pelo algoritmo *Random Forest*.

Neste mesmo método da classe *Training_skills*, há a chamada para outro método na linha 14 e 16, responsável por inicializar a estrutura de pilha para, posteriormente, realizar o controle das recomendações para os desenvolvedores. Nestas linhas é indicado, se as recomendações serão realizadas para um membro de uma equipe ou para uma equipe inteira de desenvolvedores.

No Algoritmo 3 há outro método da classe *Training_skills* - *execute_predict_sample*. Este método está compreendido ainda nas atividades “Classificador MultinomialNB 2” e “Classificador *Random Forest*”. Esse método tem como principal objetivo recomendar tarefas para os desenvolvedores, de acordo com o classificador determinado anteriormente. Como pode ser observado no intervalo das linhas 6 a 25, o conjunto de dados é percorrido realizando-se associações dos desenvolvedores e tarefas, desta forma, verificando quais as combinações que

resultam em maiores probabilidades. Já que, esta proposta visa recomendar a tarefa para um desenvolvedor especifico e não para a função exercida (linha 11 do algoritmo 3). Sendo determinada esta última, a tarefa é atribuída ao desenvolvedor, utilizando a estrutura de pilha anteriormente inicializada no Algoritmo 2, linhas 14 e 16. Para melhor compreensão de como é realizada a recomendação, foi desenvolvido um diagrama *Business Process Model and Notation* (BPMN) para elucidar o processo, veja Figura 5.

Algoritmo 3 Método execute_predict_sample da classe Training_skills

```
1.  def execute_predict_sample(self, df, df_skills):
2.      dfAx = df
3.      dfAx.loc[:, 'key']=1
4.      df_skills.loc[:, 'key'] = 1
5.      dfAx = df.drop(["User name", "System analysts", "Designer", "Programmer", "Tester",
    "Communicative", "Creativity", "Sensory", "Rationalist", "Emotional", "Organized",
    "Predominant_value"], axis=1)
6.      for index, row in dfAx.iterrows():
7.          my_str = "\nTitle issue => {}\nLabels => {}\nDescription => {}\n"
8.          if len(row['descrip_p']) > 1200:
9.              continue
10.
11.         aux = df_skills.merge(pd.DataFrame(data=[row.values], columns=row.index),
    how='outer', on='key').drop('key', axis=1)
12.         data = aux.loc[:,self.columns]
13.         predicted = self.train.predict(np.array(data))
14.         probability = self.train.predict_proba(np.array(data))
15.
16.         predicted, probability = Utils().larger_probability(predicted, probability)
17.         probability_f = max(probability)
18.         output = self.check_stack(predicted)
19.         if output != 3:
20.             self.indexes_update['i_data'].append(index)
21.
22.         my_str = my_str.format(row['issue_title'], row['labels'], row['description'])
23.         c_output = self.check_output(output, my_str, predicted, probability, probability_f)
24.         if c_output:
25.             break
```

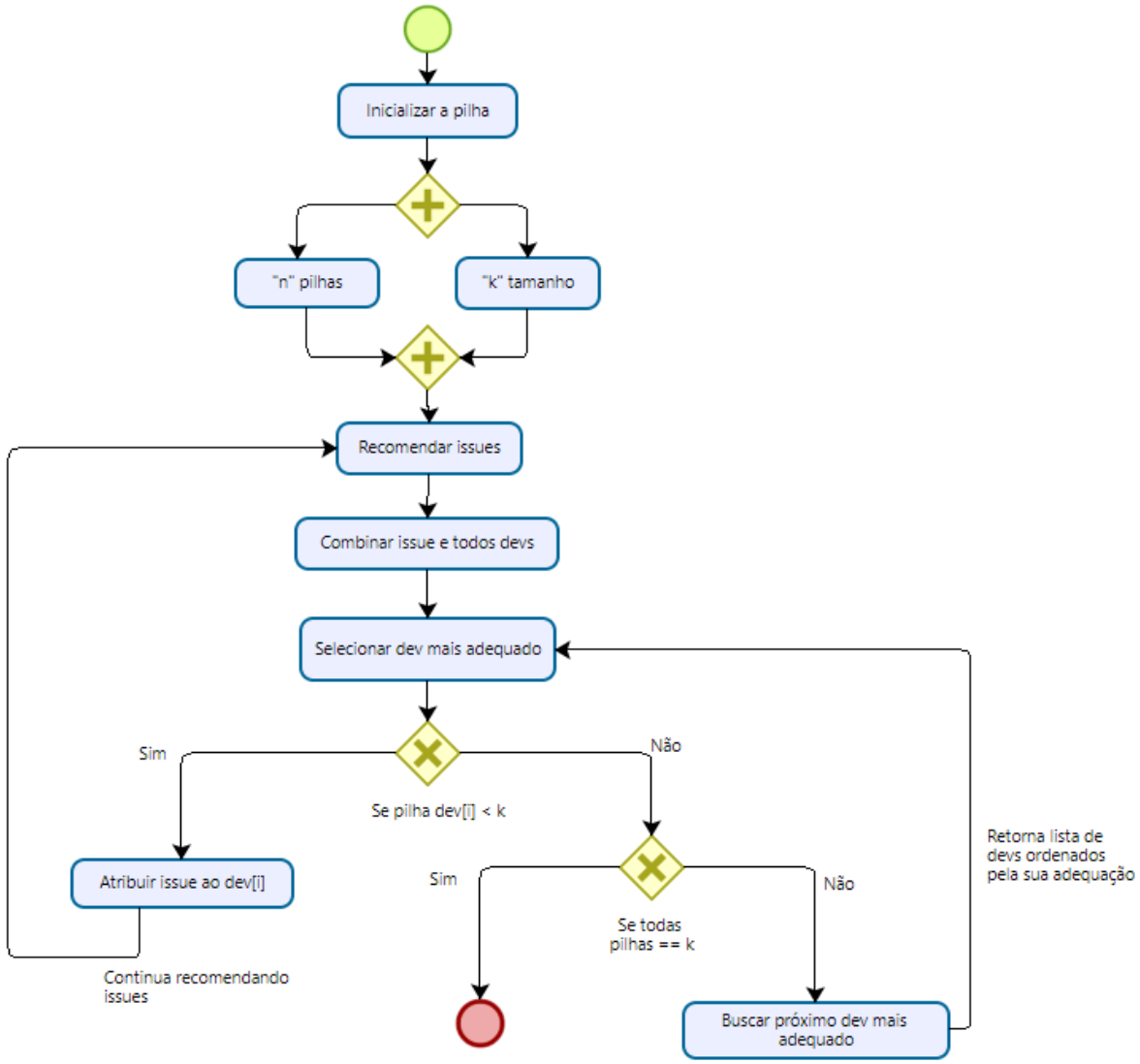
Fonte: elaborado pelo autor.

Devido ao tamanho de algumas descrições de tarefas, que poderiam dificultar a avaliação dos participantes do experimento, decidiu-se limitar o tamanho a 1200 caracteres. Somente as tarefas que cumprem este requisito, são utilizadas para realizar as recomendações, veja linha 8 do Algoritmo 3.

Quando a tarefa é associada a um desenvolvedor, a estrutura em pilha inicializada anteriormente é utilizada. Veja a Figura 5, i se refere a um dos n membros (*devs*) da equipe. A variável k refere-se ao tamanho da pilha, isto é, quantas tarefas podem ser alocadas a um determinado *dev*.

A recomendação de tarefas visa sempre buscar o *dev* mais adequado a execução da tarefa. No caso do *dev* já estar com a sua capacidade máxima de realização de tarefas, sempre será buscado o próximo mais adequado.

Figura 5 – Processo de criação da pilha e recomendação.



Fonte: elaborado pelo autor.

4.4 AVALIAÇÃO DAS ALOCAÇÕES

A última fase da abordagem consiste em avaliar as alocações realizadas e o armazenamento dos dados de desempenho e de conclusão das tarefas. As atividades propostas para essa etapa podem ser visualizadas na Figura 6.

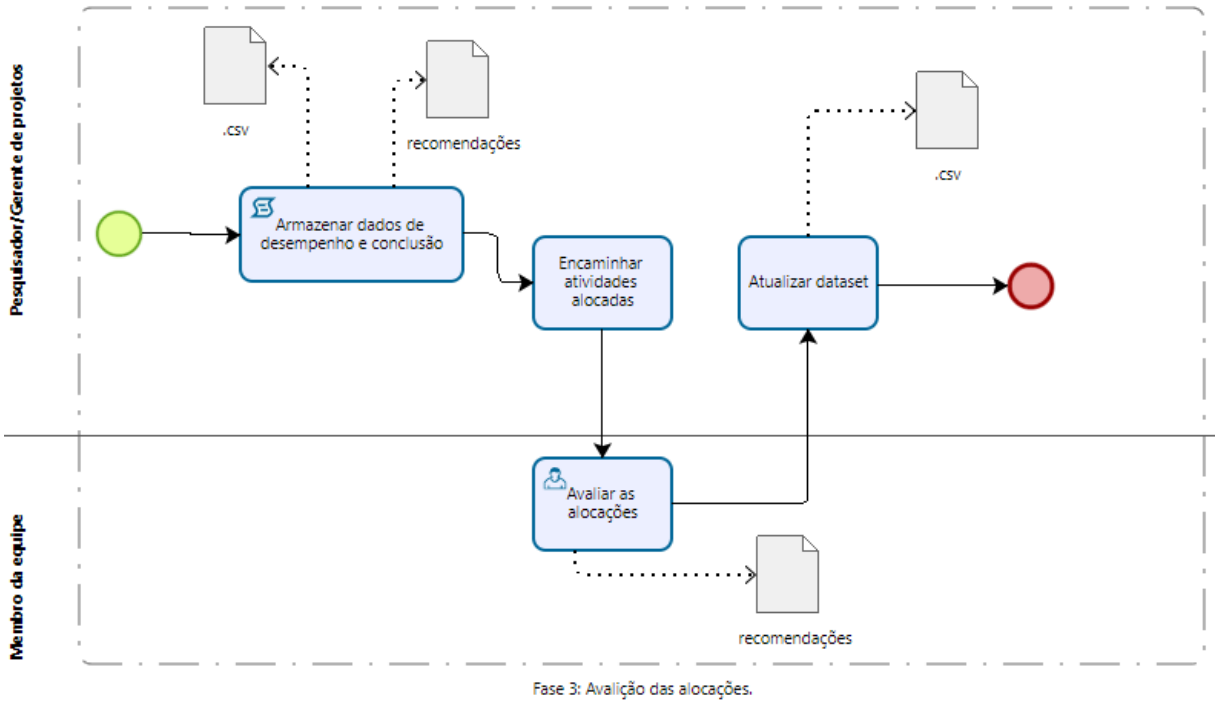
Na atividade, “Armazenar dados de desempenho e conclusão”, são armazenados os dados no conjunto de dados (arquivo de extensão Comma-separated values – CSV), desde as suas informações gerais, campos textuais processados (*descript_p* e *text_idf*), classificações realizadas pelo algoritmo de ML. Além de gerar os documentos de extensão “.txt” com as tarefas alocadas a cada desenvolvedor.

As recomendações tendo sido concluídas, é escrito em um arquivo com extensão “.txt” as tarefas para cada desenvolvedor, por exemplo: “desenvolvedor1.txt”, “desenvolvedor2.txt”, “desenvolvedor3.txt”, etc. Este arquivo é encaminhado posteriormente para os membros da equipe.

Com a conclusão da atividade anterior, são encaminhados os arquivos com as tarefas recomendadas para cada desenvolvedor. Este momento visa acrescentar dados históricos ao conjunto de dados disponível até o momento; esta atividade denomina-se “Encaminhar atividades alocadas”.

No diagrama BPMN da Figura 6, é possível verificar que a atividade “Avaliar as alocações” é executada por cada membro da equipe de desenvolvimento, ou seja, o participante/desenvolvedor recebe as tarefas recomendadas e deve avaliar a relevância da tarefa para seu perfil usando uma escala de 1 a 5 para cada tarefa, com 5 indicando alta prioridade e 1 indicando baixa prioridade.

Figura 6 - Diagrama do processo “Avaliação das alocações”.



Fonte: elaborado pelo autor.

Na atividade “Atualizar *dataset*”, o arquivo com os dados de alocação de tarefas é atualizado para ajustar as informações fornecidas pelo desenvolvedor, isto é, as tarefas que o desenvolvedor atribuiu média a alta prioridade é mantida sua a relação com o desenvolvedor especifico, no conjunto de dados.

5 VALIDAÇÃO DO TRABALHO

A validação seguiu um protocolo definido, sendo aplicada uma validação quantitativa e uma validação qualitativa. Este último seguiu os parâmetros e sugestões indicados pelos trabalhos (KITCHENHAM; PICKARD; PFLEEGER, 1995; WOHLIN; HÖST; HENNINGSSON, 2003). As validações realizadas foram aplicadas em dois grupos, adotou-se os nomes “grupo A” e “grupo B”, tais grupos estão definidos na seção 5.1.

5.1 VALIDAÇÃO QUALITATIVA

A avaliação qualitativa tem como objetivo provar que o perfil do desenvolvedor e suas competências são relevantes para a alocação de tarefas e evidenciar que a abordagem proposta é efetiva para a recomendação de tarefas aos membros de uma equipe de desenvolvimento de software. Para que isto, seguiu-se um protocolo já definido por outros trabalhos (KITCHENHAM; PICKARD; PFLEEGER, 1995; WOHLIN; HÖST; HENNINGSSON, 2003).

Foram selecionados ao todo dez (10) desenvolvedores que foram distribuídos em dois grupos, denominados grupo A e grupo B. O grupo A é composto por membros acadêmicos e profissionais trabalhando na indústria privada. O grupo B é composto por membros acadêmicos e profissionais ligados a projetos desenvolvidos pela universidade. Assim como, os membros do grupo B foram os mesmos que concluíram as tarefas utilizadas durante a análise desta proposta.

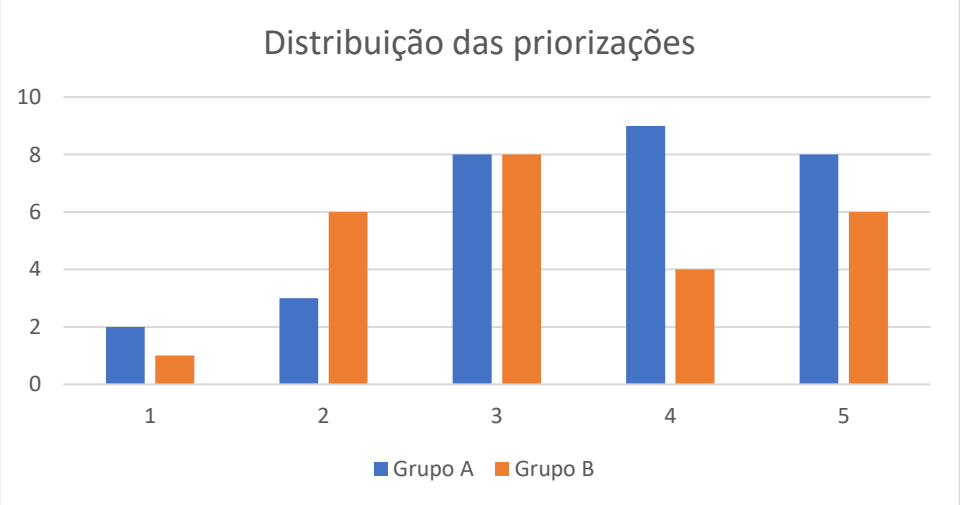
A validação seguiu os seguintes passos: (1) encaminhar o questionário para os participantes e coletar seus dados; (2) realizar a alocação de tarefas e enviar para os participantes; (3) analisar respostas dos participantes e discutir os resultados obtidos. Esses passos são detalhados a seguir.

(1) Encaminhar o questionário para os participantes e coletar seus dados: foi encaminhado para os participantes o questionário por *e-mail* com instruções sobre como seria realizada a validação, destacando que o processo é anônimo e que os desenvolvedores não serão identificados na análise dos resultados. O questionário inclui informações sobre atividades de trabalho, como cargo atual, e perguntas para determinar o perfil e as competências do desenvolvedor (vide subseção 4.1).

(2) Realizar a alocação de tarefas e enviar para os participantes: após o preenchimento do questionário pelos desenvolvedores, atribui-se um conjunto de tarefas a cada um usando o sistema de recomendação. Inicialmente, as tarefas foram atribuídas ao grupo A e, posteriormente, ao grupo B, sem levar em consideração os dados de alocação de um grupo no outro. Foi enviado a lista de tarefas recomendadas para cada participante por e-mail. Ao receber a lista, o participante deveria preencher um questionário de *feedback* indicando se concordava ou não com a alocação. A escala utilizada compreende valores de 1 a 5, onde 5 indica maior prioridade e 1 indica baixa prioridade.

(3) Analisar respostas dos participantes e resultados obtidos: nessa etapa foi analisado o *feedback* fornecido pelos participantes. Na Figura 7 pode ser visualizado o gráfico com a distribuição das respostas fornecidas por cada grupo. A maioria das respostas dos desenvolvedores foi maior que 3, o que indica que o desenvolvedor não discorda da atribuição. Destaca-se o número de respostas 4 e 5 mostrando a concordância do desenvolvedor com a alocação. Verificou-se também a relevância que os atributos sugeridos neste trabalho – perfil do desenvolvedor e competências – têm na alocação de tarefas. Esta análise utiliza índices do algoritmo *Random Forest*. Esses índices podem ser vistos na Figura 8. Como pode ser visto, os valores obtidos são consideráveis para ambos os grupos, sugerindo assim que eles são necessários para realizar alocações de tarefas dentro do conjunto de dados utilizado para treinar os classificadores.

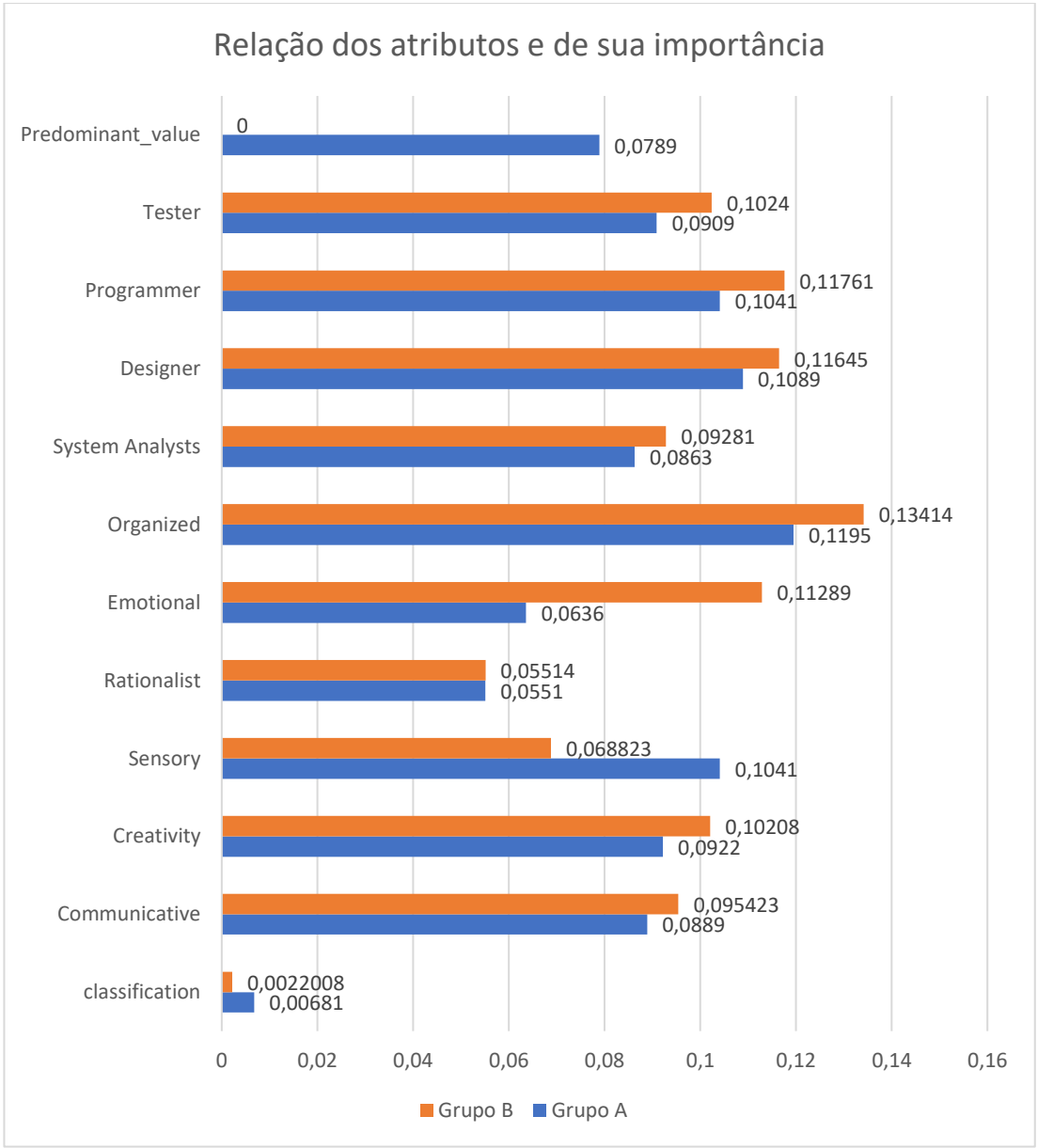
Figura 7 – Distribuição das priorizações quanto aos grupos avaliados.



Fonte: elaborado pelo autor.

Retomando o objetivo deste trabalho, que é desenvolver uma abordagem inteligente para a alocação de tarefas adequadas ao perfil do desenvolvedor. Avaliando os resultados obtidos das validações, atingiu-se os objetivos esperados nesta etapa com a validação e as tarefas recomendadas. Já que 76% das tarefas foram classificadas pelos participantes com média a alta prioridade, quanto a ambos grupos.

Figura 8 – Relação dos atributos e de sua importância para os grupos aplicados.



Fonte: elaborado pelo autor.

Em relação ao conjunto de atributos relacionados ao perfil do desenvolvedor, sugeridos por este trabalho. Se obteve altos valores relacionados a sua importância (algoritmo *Random Forest*) para a classificação/recomendação das tarefas.

Segundo as respostas, em geral, alcançou-se o objetivo esperado nesta etapa com a validação da proposta e das tarefas recomendadas.

5.2 VALIDAÇÃO QUANTITATIVA

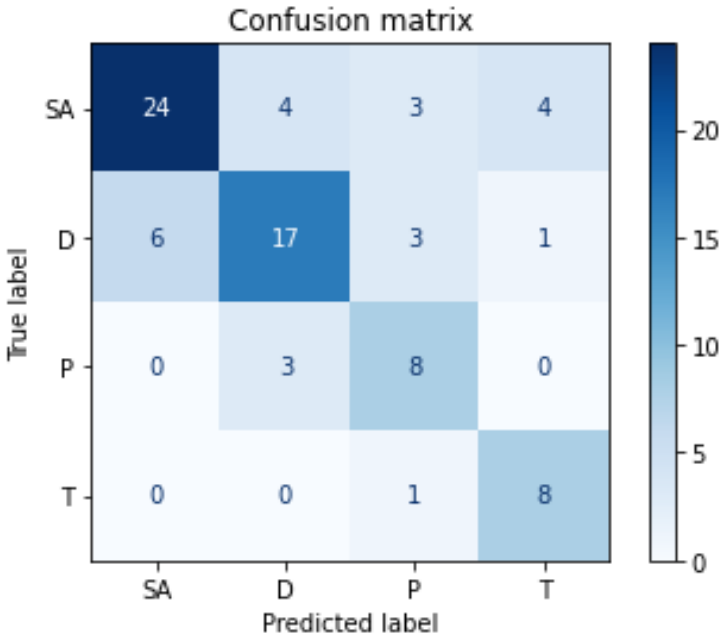
A validação foi realizada utilizando as métricas comuns de ML tais como acurácia, precisão, *recall* e F_1. Para o primeiro classificador são consideradas todas as tarefas. Já na validação do segundo classificador, foi considerado o grupo A e o grupo B.

O objetivo do primeiro classificador é de mapear o perfil de desenvolvedor adequado para cada tarefa, logo, esta validação é realizada somente uma vez, visto que, o conjunto de dados de treino é o mesmo para o grupo A e para o grupo B. O conjunto utilizado consiste em 495 tarefas, sendo que 271 foram classificadas manualmente, de acordo com o título, as *labels* e a descrição da tarefa. No treino obteve-se a seguinte distribuição: 45 tarefas classificadas para o perfil analista de sistemas (AS), 113 para *designer* (D), 90 para programador (P) e 23 para testador (T). Ao se executar o classificador com este conjunto, obtém-se uma acurácia de aproximadamente 73%.

Os resultados relacionados à classificação textual, podem ser visualizados na Figura 9 e na Tabela 17. Na Figura 9 é plotada a matriz de confusão dos elementos preditos versus os elementos originais, eixo X e eixo Y, respectivamente.

Na amostra testada, foi obtido o valor de 69,5% de exemplos preditos corretamente (verdadeiros positivos), com poucas ocasiões em que foi predito erroneamente cerca de 30,5% das tarefas. Este valor equivale ao total de tarefas classificadas corretamente pela quantia total de tarefas classificadas, que equivale a diagonal principal da matriz de confusão, plotada na Figura 9.

Figura 9 – Matriz de confusão obtida durante o treino dos dados textuais.



Fonte: elaborado pelo autor.

Na Tabela 17 são apresentadas as métricas relacionadas a classificação das tarefas e o perfil de desenvolvedor mais adequado a cada uma, isto é, das tarefas que foram classificadas manualmente. Dentre as funções classificadas a que obteve menor precisão foi a de programador, visto que o valor obtido foi de 53% para uma amostra correspondente de 11 tarefas.

Tabela 17 – Resumo das métricas.

Perfis de desenvolvedor	Precisão	*Recall*	F_1	Suporte
analista de sistemas	0,80	0,69	0,74	35
designer	0,71	0,63	0,67	27
programador	0,53	0,73	0,62	11
testador	0,62	0,89	0,73	9

Fonte: elaborado pelo autor.

Em contrapartida, para analista de sistemas, *designer* e testador, foram obtidas uma alta precisão, respectivamente 80%, 71% e 62%. Todavia, note ainda que o perfil de desenvolvedor *designer* possui um *recall* menor em comparação com os perfis de testador e programador, indicando que alguns elementos podem ser

falsos positivos. Vale lembrar ainda que classificadores com *recall* alto têm poucos exemplos de falso negativos (TAN *et al*., 2009). Considerando as tarefas para analistas, a precisão foi de 80% e *recall* de 69%, obtendo-se um comportamento inverso ao ocorrido com o perfil de testador, no qual foi obtido um baixo valor para a precisão, mas um alto valor para o *recall*.

Outra forma de se avaliar o desempenho é através da métrica F_1, uma medida alternativa para a precisão e *recall*, combinando seus valores em uma única métrica (HAN; KAMBER; PEI, 2012; TAN *et al*., 2009). Analisando os valores obtidos para esta métrica, é possível notar que os resultados seguem a seguinte ordem decrescente: analista de sistemas, testador, *designer* e programador.

Ao realizar o treino foi avaliado conjuntamente as características mais importantes, isto é, as colunas utilizadas pelo algoritmo *Random Forest*. A descrição das colunas utilizadas para treinar e sua importância pode ser visualizada na Figura 8.

O conjunto de atributos utilizados para realizar o processo de treino, que possuem maior relevância para o classificador inclui o seguinte conjunto de atributos: *system analysts*, *designer*, *programmer*, *tester*, *communicative*, *creativiy*, *sensory*, *rationalist*, *emotional*, *organized, classification* e predominant_value. Estas colunas possuem relevância na classificação e recomendação das tarefas, sendo assim corroborando com a sua indicação quando realizar classificações ou recomendações de tarefas para os desenvolvedores.

Tanto para o grupo A quanto para o grupo B utilizou-se o mesmo conjunto de tarefas para o treino dos classificadores. Vale ressaltar que os membros do grupo B fazem parte da equipe da qual as tarefas foram extraídas para realizar a alocação, uma vez que as tarefas são de uma base de dados real. Assim como, o processo de treino do segundo classificador para com o grupo B difere daquele do grupo A, pois se utiliza os dados históricos de desenvolvimento do grupo B.

O procedimento adotado para com o grupo B obteve-se seus perfis. Após este processo, eles foram associados às tarefas que já haviam desenvolvido/resolvido, acrescido das informações obtidas pela quantificação de seus perfis. Com isto, este trabalho obteve a seguinte relação de atributos e sua importância para o grupo B, após realizar o treino do segundo classificador, veja a barra laranja na Figura 8.

Os atributos de maior relevância para o grupo A e para o grupo B são os mesmos. Todavia, com algumas discrepâncias em seus valores finais. Pode se notar que os atributos que detiveram maior diferença entre os grupos foram: *predominant_value*, *emotional* e *sensory*. Além destes atributos, existe pouca variação entre os valores obtidos para os demais atributos. Por fim, ao todo percebe-se que os atributos relacionados ao perfil técnico e as competências possuem impacto quando se realiza a alocação de tarefas.

6 CONCLUSÃO

Neste capítulo são descritas as considerações finais sobre o trabalho, enfatizando as contribuições, limitações e trabalhos futuros. A Seção 6.1 resume os objetivos e resultados deste trabalho. A Seção 6.2 analisa as contribuições advindas da proposta deste trabalho. Na Seção 6.3 é apresentado algumas limitações do trabalho. Por fim, a Seção 6.4 é finalizada com as perspectivas de trabalhos futuros.

6.1 REVISÃO GERAL E RESULTADOS

Neste trabalho foi desenvolvida uma abordagem para a recomendação de tarefas, conforme os dados extraídos das tarefas e o perfil de desenvolvedor. As tarefas foram extraídas de um projeto de desenvolvimento real, e manipuladas por meio de técnicas de processamento de texto, ML e mineração de dados. Para o perfil de desenvolvedor, foi elaborado um conjunto de questões e equações matemáticas, visando quantificar algumas competências inerentes a cada perfil técnico. Estas por sua vez utilizadas quando realizada a recomendação das tarefas. A proposta considera a individualidade do indivíduo, ou seja, as competências do indivíduo definem a tarefa que será recomendada, desta forma, entregando um processo mais próximo à realidade e ao perfil de cada desenvolvedor.

Foram classificadas 271 tarefas do conjunto de tarefas extraído. Essas foram classificadas dentro do intervalo de perfis do desenvolvedor propostos em nossa abordagem. As tarefas foram classificadas para auxiliar os algoritmos a identificar os perfis adequados para as futuras tarefas que seriam recomendadas pela proposta elaborada. Para isto, empregou-se técnicas de processamento textual, desde a conversão de uma coleção de documentos textuais para uma matriz de contagem de *tokens*, como também TF-IDF. A abordagem desenvolvida utiliza dois algoritmos de ML, quando realizada a recomendação das tarefas, priorizando-se assim o algoritmo com a maior acurácia.

A abordagem foi validada com dois grupos de desenvolvedores. Os desenvolvedores avaliaram as tarefas recomendadas, priorizando cada uma delas e fornecendo *feedback* para auxiliar na avaliação da abordagem proposta. Ainda durante a recomendação das tarefas aos desenvolvedores constatou-se que os atributos, como as competências e o perfil técnico, possuem influência sobre a

recomendação de tarefas para desenvolvedores. Assim como, as tarefas recomendadas foram classificadas com alta a média prioridade. A análise destes resultados demonstra que as recomendações são condizentes e coerentes.

6.2 CONTRIBUIÇÕES

Dentre as contribuições pode-se citar:

- Elaboração do perfil do desenvolvedor, tendo em vistas, os estudos anteriores que trabalharam com perfis técnicos e competências;
- Elaboração de um questionário e equações matemáticas para quantificar o perfil do desenvolvedor;
- Concepção de uma abordagem inteligente para a recomendação de tarefas, levando-se em conta os dados das tarefas e o perfil do desenvolvedor;

6.3 LIMITAÇÕES DESTE TRABALHO

Dentro das limitações deste trabalho, está a base de dados utilizada e o número de participantes da validação. O conjunto de dados não é conjunto expressivo, em comparação com trabalhos da literatura

6.4 TRABALHOS FUTUROS

Como trabalho futuro espera-se aprimorar a abordagem desenvolvida, isto é, tornando-a genérica para que o perfil técnico possa ser genérico. Sendo assim, não seria necessária a análise de um conjunto de perfis técnicos, possibilitando a recomendação de tarefas para inúmeras outras profissões técnicas associadas à tecnologia da informação e não apenas ao conjunto sugerido por este trabalho. A elaboração de uma UI para a abordagem desenvolvida, tornando mais acessível a sua futura utilização, visto que, a sua atual execução é por linha de comando. Assim como, a busca e a coleta de mais tarefas para o conjunto de dados, isto é, que possuam um contexto e uma descrição clara do que necessita ser realizado.

REFERÊNCIAS BIBLIOGRÁFICAS

ABDELLATIF, T. M.; CAPRETZ, L. F.; HO, D. Automatic recall of software lessons learned for software project managers. **Information and Software Technology**, v. 115, n. June, p. 44–57, 2019.

ABU SHEIKHA, F.; INKPEN, D. **Automatic classification of documents by formality**. Proceedings of the 6th International Conference on Natural Language Processing and Knowledge Engineering(NLPKE-2010). **Anais**...2010

ALPAYDIN, E. **Introduction to Machine Learning**. Second Edi ed. [s.l.] The MIT Press, 2010. v. 2

AQEEL IQBAL, M.; ALDAIHANI, A. R.; SHAH, A. Big-five personality traits mapped with software development tasks to find most productive software development teams. **International Journal of Innovative Technology and Exploring Engineering**, v. 8, n. 12, p. 965–971, 2019.

ASLAM, W.; IJAZ, F. A Quantitative Framework for Task Allocation in Distributed Agile Software Development. **IEEE Access**, v. 6, p. 15380–15390, 2018.

BAEZA-YATES, R.; RIBEIRO-NETO, B. **Recuperação de Informação: conceitos e tecnologia das máquinas de busca**. 2. Ed. ed. [s.l.] Bookman Editora, 2013.

BECK, K. et al. **Manifesto for Agile Software Development**. Disponível em: <https://agilemanifesto.org/>. Acesso em: 28 out. 2020.

BENNACEUR, A.; MEINKE, K. Machine learning for software analysis: Models, methods, and applications. **Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)**, v. 11026 LNCS, n. 1, p. 3–49, 2018.

BOURQUE, P.; FAIRLEY, R. E. (EDS.). **SWEBOK Guide to the Software Engineering Body of Knowledge**. Version 3. ed. [s.l.] IEEE Computer Society, 2014.

BREIMAN, L. Random Forests. **Machine Learning**, v. 45, n. 1, p. 5–32, 2001.

CAPRETZ, L. F.; AHMED, F. Making sense of software development and personality types. **IT Professional**, v. 12, n. 1, p. 6–13, 2010.

CAPRETZ, L. F.; VARONA, D.; RAZA, A. Influence of personality types in software tasks choices. **Computers in Human Behavior**, v. 52, p. 373–378, 2015.

COSTA, P. T.; MCCRAE JR, R. R. . NEOPI-R professional manual. Odessa: Psychological Assessment Resources. 1992.

DEMARCO, T.; LISTER, T. **Peopleware: Productive Projects and Teams**. [s.l.] Pearson Education, 2013.

DEVI, M. D.; SAHARIA, N. **Learning Adaptable Approach to Classify Sentiment with Incremental Datasets.** Procedia Computer Science. **Anais**...Elsevier B.V., 1 jan. 2020

FACELI, K. et al. **Inteligência artificial: uma abordagem de aprendizado de máquina**. [s.l.] Grupo Gen - LTC, 2011.

FARHANGIAN, M. et al. Agent-based modeling of resource allocation in software projects based on personality and skill. **Communications in Computer and Information Science**, v. 541, n. October, p. 130–146, 2015.

FELDT, R. et al. **Links between the personalities, views and attitudes of software engineersInformation and Software Technology**, 2010.

FOWLER, M. **The New Methodology**. Disponível em: <https://martinfowler.com/articles/newMethodology.html>. Acesso em: 28 out. 2020.

FUGGETTA, A. **Software process: A roadmap**. Proceedings of the Conference on the Future of Software Engineering, ICSE 2000. **Anais**...New York, New York, USA: Association for Computing Machinery, Inc, 1 maio 2000Disponível em: <http://portal.acm.org/citation.cfm?doid=336512.336521>. Acesso em: 28 out. 2020

FURNHAM, A.; MOUTAFI, J.; CRUMP, J. The relationship between the revised neo-personality inventory and the myers-briggs type indicator. **Social Behavior and Personality**, v. 31, n. 6, p. 577–584, 2003.

GOLDBERG, L. R. **The structure of phenotypic personality traits.American Psychologist**USAmerican Psychological Association, , 1993.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep Learning**. [s.l.] MIT Press, 2016.

HALL, T. et al. What do we know about developer motivation? **IEEE Software**, v. 25, n. 4, p. 92–94, jul. 2008.

HAN, J.; KAMBER, M.; PEI, J. **Data Mining Concepts and Techniques**. Third Edit ed. [s.l.] Morgan Kaufmann, 2012.

INSTITUTE, P. M. **A Guide to the Project Management Body of Knowledge**. Sixth Edit ed. [s.l.] Pennsylvania: Project Management Institute, Inc., 2017.

JIA, J.; ZHANG, P.; ZHANG, R. A comparative study of three personality assessment models in software engineering field. **Proceedings of the IEEE International Conference on Software Engineering and Service Sciences, ICSESS**, v. 2015-Novem, p. 7–10, 2015.

JUNG, C. G. **Tipos psicológicos**. [s.l.] Editora Vozes, 2011.

KITCHENHAM, B. A.; PICKARD, L.; PFLEEGER, S. L. Case Studies for Method and Tool Evaluation. **IEEE Softw.**, v. 12, p. 52–62, 1995.

MANGMANG, G. B.; FELISCUZO, L.; MARAVILLAS, E. A. **Descriptive Feedback on Interns' Performance using a text mining approach**. 2019 14th International Joint Symposium on Artificial Intelligence and Natural Language Processing (iSAI-NLP). **Anais**...2019

MANNING, C. D.; RAGHAVAN, P.; SCHÜTZE, H. **Introduction to Information Retrieval**. [s.l.] Cambridge University Press, 2008.

MAO, K. et al. Developer recommendation for crowdsourced software development tasks. **Proceedings - 9th IEEE International Symposium on Service-Oriented System Engineering, IEEE SOSE 2015**, v. 30, p. 347–356, 2015.

MENDES, F. F.; MENDES, E.; SALLEH, N. The relationship between personality and

decision-making: A Systematic literature review. **Information and Software Technology**, v. 111, n. March, p. 50–71, 2019.

MICHAELIS, **Dicionário Brasileiro da Língua Portuguesa**. Editora Melhoramentos Ltda, 2015. Disponível em: <https://michaelis.uol.com.br/moderno-portugues/busca/portugues-brasileiro/disciplina/>. Acesso em: 15/05/2022.

MITCHELL, T. M. **Machine Learning**. [s.l.] McGraw-Hill, 1997.

MYERS, I. B. **MBTI Manual: A Guide to the Development and Use of the Myers-Briggs Type Indicator**. [s.l.] Consulting Psychologists Press, 1998.

OTERO, C. E. et al. A multi-criteria decision making approach for resource allocation in software engineering. **UKSim2010 - UKSim 12th International Conference on Computer Modelling and Simulation**, p. 137–141, 2010.

PEDREGOSA, F. et al. Scikit-learn: Machine Learning in Python. **Journal of Machine Learning Research**, v. 12, n. 85, p. 2825–2830, 2011.

PEROVŠEK, M. et al. TextFlows: A visual programming platform for text mining and natural language processing. **Science of Computer Programming**, v. 121, p. 128–152, 2016.

PRESSMAN, R. S. **Software Engineering: a Practitioner's Approach, 7th Edition**. [s.l: s.n.].

RICCI, F.; ROKACH, L.; SHAPIRA, B. Introduction to Recommender Systems Handbook. In: RICCI, F. et al. (Eds.). . **Recommender Systems Handbook**. Boston, MA: Springer US, 2011. p. 1–35.

ROBILLARD, M.; WALKER, R.; ZIMMERMANN, T. Recommendation systems for software engineering. **IEEE Software**, v. 27, n. 4, p. 80–86, jul. 2010.

RUSSELL, S.; NORVIG, P. **Inteligência artificial: Tradução da 3a Edição**. 3ª Edição ed. [s.l.] Elsevier Brasil, 2014.

SATAPATHY, S. M.; ACHARYA, B. P.; RATH, S. K. Early stage software effort estimation using random forest technique based on use case points. **IET Software**,

v. 10, n. 1, p. 10–17, 2016.

SOMMERVILLE, I. **Engenharia de Software**. 9. Ed. ed. [s.l.] São Paulo: Pearson Prentice Hall, 2011.

SOOMRO, A. B. et al. The effect of software engineers' personality traits on team climate and performance: A Systematic Literature Review. **Information and Software Technology**, v. 73, n. October 2017, p. 52–65, 2016.

SUTTON, R. S.; BARTO, A. G. **Reinforcement Learning: An Introduction**. Second ed. [s.l.] The MIT Press, 2018.

TAN, P. N. et al. **Introdução ao datamining: mineração de dados**. [s.l.] Ciencia Moderna, 2009.

WANG, J.; WANG, Z.; LI, J. A recommendation method for social collaboration tasks based on personal social preferences. **IEEE Access**, v. 6, p. 45206–45216, 2018.

WANG, X.; LIN, X.; HAJLI, N. Understanding Software Engineers' Skill Development in Software Development. **Journal of Computer Information Systems**, v. 00, n. 00, p. 1–10, 2019.

WAZLAWICK, R. **Engenharia de Software: Conceitos e Práticas**. 1ª Ed. ed. [s.l.] Elsevier Brasil, 2013.

WEINBERG, G. M. **The Psychology of Computer Programming**. [s.l.] Van Nostrand Reinhold, 1971.

WILLIAMS, L.; COCKBURN, A. **Agile software development: It's about feedback and changeComputer**, jun. 2003.

WITTEN, I. H. et al. **Data Mining Practical Machine Learning Tools and Techniques**. Fourth Edition ed. [s.l.] Elsevier, 2017.

WOHLIN, C.; HÖST, M.; HENNINGSSON, K. Empirical Research Methods in Software Engineering. In: CONRADI, R.; WANG, A. I. (Eds.). . **Empirical Methods and Studies in Software Engineering: Experiences from ESERNET**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003. p. 7–23.

YILMAZ, M. et al. An examination of personality traits and how they impact on software development teams. **Information and Software Technology**, v. 86, p. 101–122, 2017.

YILMAZ, M.; O'CONNOR, R. V. Towards the understanding and classification of the personality traits of software development practitioners: Situational context cards approach. **Proceedings - 38th EUROMICRO Conference on Software Engineering and Advanced Applications, SEAA 2012**, p. 400–405, 2012.