Universidade Federal de Campina Grande
Centro de Engenharia Elétrica e Informática
Coordenação de Pós-Graduação em Ciência da Computação

Uma Abordagem de Apoio à Decisão para Formação de Múltiplas Equipes em Projetos Ágeis de Software

Antonio Alexandre Moura Costa

Tese submetida à Coordenação do Curso de Pós-Graduação em Ciência da Computação da Universidade Federal de Campina Grande - Campus I como parte dos requisitos necessários para obtenção do grau de Doutor em Ciência da Computação.

Área de Concentração: Ciência da Computação
Linha de Pesquisa: Engenharia de Software

Hyggo Oliveira de Almeida
(Orientador)

Campina Grande, Paraíba, Brasil

“UMA ABORDAGEM DE APOIO À DECISÃO PARA FORMAÇÃO DE MÚLTIPLAS EQUIPES EM PROJETOS ÁGEIS DE SOFTWARE”

ANTONIO ALEXANDRE MOURA COSTA

TESE APROVADA EM 25/02/2019

HYGGO OLIVEIRA DE ALMEIDA, Dr., UFCG
Orientador(a)

JOSEANA MACÊDO FECHINE RÉGIS DE ARAÚJO, Dra., UFCG
Examinador(a)

EVANDRO DE BARROS COSTA, Dr., UFAL
Examinador(a)

MIRKO BARBOSA PERKUSICH, Dr., IFPB
Examinador(a)

UIRA KULESZA, Dr., UFRN
Examinador(a)

CAMPINA GRANDE - PB

C837a Costa, Antonio Alexandre Moura.
Uma abordagem de apoio à decisão para formação de múltiplas equipes em projetos ágeis de software / Antonio Alexandre Moura Costa. – Campina Grande, 2019.
146 f. : il. color.

Tese (Doutorado em Ciência da Computação) – Universidade Federal de Campina Grande, Centro de Engenharia Elétrica e Informática, 2019.
"Orientação: Prof. Dr. Hyggo Oliveira de Almeida".
Referências.

1. Software. 2. Algoritmo Genético. 3. Projetos Ágeis – Formação de Equipes. I. Almeida, Hyggo Oliveira de. II. Título.

CDU 004.42(043)

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECÁRIA SEVERINA SUELI DA SILVA OLIVEIRA CRB-15/225

Resumo

Metodologias ágeis surgiram como uma forma de gestão e desenvolvimento baseadas em uma abordagem incremental para atender às demandas dos clientes e seus projetos. O Scrum é um arcabouço para gerenciamento e desenvolvimento ágil de projetos de software, centrado no conceito de equipe, com o intuito de entregar valor de negócio. Um dos maiores desafios, não só das empresas de software, mas da indústria como um todo, está focado na formação de equipes. Tradicionalmente, a formação de equipe para projetos de software é um processo não automatizado, dependente da ação humana e sujeito a erros. A complexidade desse processo se torna ainda maior quando se considera a distribuição de pessoas, dentro de uma mesma organização, em diversas equipes, com diferentes demandas de competência e experiência, o que é denominado na literatura de Formação de Múltiplas Equipes. O objetivo geral da pesquisa ora descrita, consiste na concepção de uma abordagem de apoio à decisão para formação de múltiplas equipes para projetos ágeis de software, mais especificamente, que seguem o Scrum, a partir da realocação dos recursos humanos disponíveis na empresa. A abordagem proposta utiliza algoritmos genéticos para sugerir equipes, formadas a partir de perfis técnicos gerados durante a execução de projetos de software internos à empresa. Para validação do trabalho, foi gerada uma base de dados históricos a partir de informações reais de 12 projetos de desenvolvimento de software e 52 desenvolvedores distintos. A abordagem foi executada e validada em 13 diferentes cenários criados a partir dos dados da base. Com o auxílio de gestores de projetos ágeis, a abordagem foi avaliada resultando em uma média de 86,4% de Precisão em relação às equipes escolhidas pelos gestores. Além disto, obteve-se uma média de 75% de aceitação em relação às equipes recomendadas.

Abstract

Agile methodologies have emerged as a form of management and development, based on an incremental approach to satisfy customers demands and their projects. Scrum is a framework for management and agile development of software projects, centered on the team concept, aiming to deliver business value. One of the greatest challenges, not only for software companies, but also for industry as a whole, is focused on team formation. Traditionally, software project team formation is a human-based non-automated process, susceptible to errors. The process complexity becomes even greater when we consider the distribution of people, within the same organization, in several teams, with different demands of competence and experience, which is called multiple teams formation problem. The overall objective of this research is to design a decision support approach to form multiple teams for agile software projects, specifically, those following Scrum, from the reallocation of human resources available in the company. The proposed approach uses genetic algorithm to suggest teams, derived from technical profiles generated during the execution of the company software projects. The validation was performed with a historical database generated from real world data from 12 software development projects and 52 different developers. The approach was executed and validated in 13 different scenarios created from the database. Afterward, it was evaluated with the support of four agile project managers and as result it reached an average of 86.4 % of Precision. In addition, an average of 75 % of acceptance was obtained.

Agradecimentos

Agradeço a Deus por me dar saúde e paz para conseguir vencer os desafios que surgem em meu caminho.

À minha esposa Mithylenny Tharsys Lima por todo o amor, compreensão e companheirismo que me forneceu ao longo de todo o caminho trilhado.

Aos meus pais e irmãos por estarem sempre ao meu lado fornecendo todo o apoio e incentivo necessários.

Ao meu orientador, Hyggo Oliveira de Almeida, que foi de fundamental importância para a concretização desse trabalho e de vários outros ao longo dos últimos anos.

Aos amigos do grupo de pesquisa que estiverem sempre dispostos a auxiliar durante a jornada da pós-graduação.

À CAPES pelos investimentos na pós-graduação.

A todos que contribuíram direta ou indiretamente para a conclusão deste trabalho.

Conteúdo

Lista de Símbolos

AG - *Algoritmos Genéticos*

API - *Application Programming Interface*

Embedded - *Laboratório de Sistemas Embarcados e Computação Pervasiva*

ES - *Engenharia de Software*

FIPA - *Finnish Information Processing Association*

ISE - *Intelligent Software Engineering*

PB - *Product Backlog*

PO - *Product Owner*

UFCG - *Universidade Federal de Campina Grande*

US - *User Story*

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

As metodologias ágeis surgiram como uma forma de gestão e desenvolvimento de software, baseadas em uma abordagem incremental para planejamento e execução [17, 61]. Desde o surgimento do Manifesto Ágil em 2001 [14], os métodos ágeis vêm ganhando cada vez mais adoção na indústria de software. Um estudo exploratório de larga escala [81], realizado em 2011, na Finlândia, com organizações (educacionais, governamentais e empresas) associadas à FIPA - *Finnish Information Processing Association*[1], apontou que 55% das organizações estudadas utilizavam métodos ágeis. Mais recentemente, em 2014, outro estudo [91] reforçou os acentuados índices de uso de métodos ágeis na industria. Os métodos ágeis vêm atraindo atenção também da comunidade científica. Um estudo de 2012 [27] verificou que entre os anos de 2001 e 2010 foram publicados 1.551 artigos científicos sobre desenvolvimento ágil de software, em 63 países. Uma revisão sistemática [48], publicada em 2017, encontrou outras 28 revisões sistemáticas que focam em 10 diferentes temas de pesquisa na área de desenvolvimento ágil de software, tais como adoção, práticas, aspectos humanos e sociais, usabilidade, agilidade organizacional, etc.

A forte adesão aos métodos ágeis nas últimas décadas se justifica pela crescente necessidade do mercado em atender as demandas dos clientes e seus projetos, de maneira mais rápida, dinâmica, flexível e com foco em maior produtividade [47]. Neste sentido, o Scrum [83], arcabouço para gerenciamento e desenvolvimento ágil de projetos de software, destaca-se por ser um dos métodos ágeis mais utilizados [64]. A *Scrum Alliance*[2], fundada

[1] http://www.ifip.org/minutes/GA99/finland_ga99.htm

[2] https://www.scrumalliance.org/

em 2001 pelos criadores do Scrum, Jeff Sutherland e Ken Schwaber, é uma das principais organizações que representa o arcabouço internacionalmente. Periodicamente, a organização libera relatórios com estatísticas de utilização do Scrum. Na edição de 2017, *State of Scrum Report* [4], realizou-se uma pesquisa com mais de 2.000 membros, na qual 89% dos participantes reportaram que utilizam o Scrum em suas organizações, estando presente em pelo menos 76 países.

O Scrum é caracterizado, dentre outras coisas, por ser um método colaborativo centrado na equipe, com o intuito de entregar valor de negócio [4]. De acordo com o Guia do Scrum [84], a equipe deve ser auto-organizada, isto é, ela deve possuir autoridade e responsabilidade em relação a diversos aspectos de seu trabalho, tais como, planejamento, agendamento, distribuição das tarefas entre os membros, tomada de decisões e consequências econômicas [46, 68]. Outro aspecto importante é a multidisciplinaridade que se caracteriza pela capacidade dos membros da equipe estarem aptos a realizar funções distintas e possuírem os requisitos necessários para se atingir o objetivo.

Na Engenharia de Software (ES), a formação de equipes apresenta-se como um desafio para as empresas e vem sendo alvo de diversos trabalhos ao longo dos últimos anos [7, 34, 50, 51, 56, 74]. Equipes são formadas por pessoas com diferentes interesses e experiências profissionais. Essas pessoas, por sua vez, representam o maior patrimônio de uma organização de software, pois constituem o capital intelectual da empresa [88]. Consequentemente, a formação inadequada de equipes configura entre as razões para baixo desempenho e maus resultados no desenvolvimento de projetos de software [101].

Tradicionalmente, o processo de formação de equipes para projetos de software caracteriza-se por ser não automatizado e dependente da ação humana, ocorrendo a partir da coleta de informações que auxiliam na definição dos perfis dos candidatos. Esses perfis são analisados e a escolha dos membros é realizada visando uma formação que possa maximizar o sucesso do projeto. Nesse processo, diversos atributos são considerados para a tomada de decisão, tais como competências técnicas, características interpessoais, experiências profissionais, personalidades, interações sociais etc. Todavia, sem um suporte automático, não há como garantir a análise de todas as possíveis combinações dos perfis, a fim de escolher a melhor equipe. Por exemplo, para formar uma equipe de cinco pessoas, dentre 20 candidatos, há um total de 1.860.480 combinações possíveis, destacando ainda que cada perfil é

composto por uma série de atributos distintos. Diante de tal complexidade, é humanamente impossível avaliar todas as possibilidades, restando apenas realizar tal escolha com base na experiência e intuição do recrutador, o que pode resultar em uma má formação de equipe.

A complexidade inerente ao processo de formação de equipes se torna ainda maior quando se considera a distribuição de pessoas, dentro de uma mesma organização, em diversas equipes, com diferentes demandas de conhecimentos e habilidades, o que é denominado na literatura de Formação de Múltiplas Equipes [44]. No contexto de ES, este problema pode ser entendido como o problema para alocar múltiplos desenvolvedores, caracterizados por uma ou várias competências, para equipes ou projetos que demandem pessoas ou competências específicas.

O processo de alocação e realocação inevitavelmente resulta em novas formações de equipes de projetos e, comumente, os gestores de cada projeto são responsáveis pela escolha dos membros de suas respectivas equipes, possuindo autonomia para definir critérios e regras durante o processo. Entretanto, como o processo de formação de múltiplas equipes impacta na empresa como um todo, há a necessidade de um esforço e consenso coletivo, a fim de que não haja discordância em relação à configuração final das equipes de cada projeto, uma vez que os profissionais representam os recursos humanos da empresa e não de um projeto em particular.

A formação de múltiplas equipes não é uma tarefa trivial, pois existe uma série de fatores que influenciam na escolha final. Geralmente, as empresas não dispõem de ferramentas para automatizar o processo, o que o torna longo, exaustivo e mais suscetível ao erro. Neste caso, para otimizar a realocação de múltiplos desenvolvedores entre múltiplos projetos seria preciso que cada gestor, além de conhecer os perfis dos membros de sua equipe, conhecesse também os perfis de todos os outros desenvolvedores da empresa, o que na prática torna-se inviável em empresas de médio e grande portes, geralmente compostas por dezenas de projetos e centenas de desenvolvedores. Outro fator impactante é a escassez de informações confiáveis. Muitas vezes a única fonte de informação sobre um desenvolvedor é o currículo, o qual nem sempre reflete fielmente o seu perfil, além de poder conter informações desatualizadas, não padronizadas e duvidosas. Há ainda a possibilidade de conflitos de interesses, uma vez que cada gestor é responsável pelo resultado de seu projeto, o que faz com que cada um almeje os melhores profissionais em suas equipes, podendo gerar assim uma disputa na

escolha final. A presença desses fatores contribui significativamente para aumentar o risco de uma má alocação dos recursos, podendo resultar a médio e longo prazo em prejuízo para a empresa.

As abordagens para formação de equipes encontradas na literatura podem ser classificadas de acordo com os tipos de atributos utilizados. Há trabalhos [31, 37, 40, 71] que focam na utilização de *soft skills*, que são atributos comportamentais e sociais de um indivíduo, tais como, atitude, liderança, criatividade, motivação, positividade etc. Estes atributos são mais difíceis de ensinar e mensurar, pois correspondem, muitas vezes, a capacidades inatas do ser humano. Nesta linha, alguns trabalhos apresentam soluções direcionadas aos atributos de natureza psicológica, utilizando ferramentas específicas que se baseiam em questionários para tentar identificar comportamentos (traços de personalidade) dos candidatos [40, 71]. As *soft skills* estão fora do escopo deste trabalho.

Neste trabalho, o foco está na utilização de *hard skills*, que são atributos que podem ser aprendidos, geralmente, em sala de aula, com livros e apostilas ou até mesmo no ambiente de trabalho. Em geral, são facilmente quantificados e usualmente avaliados durante processos seletivos, sendo possível sua comparação com os atributos de outros candidatos. Tais atributos são predominantemente técnicos, típicos de campos da engenharia e tecnologia, como por exemplo domínios sobre linguagens de programação, ferramentas, certificados de tecnologia e experiência em plataformas específicas.

Pesquisas baseadas em *hard skills* para a formação de equipes [7, 8, 13, 30, 34, 35, 50, 51, 74, 94] utilizam, geralmente, técnicas de busca e otimização tais como Algoritmos de *Backtracking* [13], Programação Dinâmica [30] e Algoritmo Genético (AG) [94]. Particularmente, há um campo recente na ES conhecido como *Search Based Software Engineering* (SBSE) [45] que utiliza estas técnicas de busca e otimização, para fornecer soluções automatizadas ou semi-automatizadas para problemas complexos da ES. Em especial, constatou-se mediante mapeamento sistemático, que AG é a técnica mais utilizada para formação de equipes. Ag surge como opção interesse para o problema me questão, pois é capaz de fornecer soluções ótimas (ou aproximadamente ótimas), de maneira eficiente, mesmo diante de amplos espaços de soluções [5]. Outras característica que difere AG de outras técnicas, é a capacidade de personalização mediante utilização das funções de *fitness*, possibilitando encontrar soluções ótimas de acordo com diferentes objetivos e restrições. Por exemplo, é

possível formar equipes visando maximizar um conjunto específico competências e minimizar outro, sem ultrapassar um determinado limite financeiro.

É neste contexto de formação de múltiplas equipes para projetos ágeis de software, que utilizam o arcabouço Scrum, que se insere este trabalho. Mais especificamente, propõe-se uma abordagem baseada em AG e *hard skills* para prover suporte aos gestores de projetos na tomada de decisão sobre a formação de múltiplas equipes.

1.1 Problemática

Durante o processo de formação de múltiplas equipes, para cada indivíduo, diversos atributos devem ser avaliados sob diferentes demandas, visando maximizar a formação global. Neste sentido, uma das principais lacunas presentes nas soluções propostas na literatura é a modelagem da competência técnica do indivíduo. Neste trabalho, a definição de competência segue o modelo CHA [11]: *Conhecimento*, que envolve o saber teórico; *Habilidade*, que envolve o saber fazer; e *Atitude*, relacionada à ação, o querer fazer. De forma alinhada com as *hard skills*, o foco deste trabalho reside no conhecimento e na habilidade.

Em geral, os trabalhos adotam níveis superficiais de modelagem, construindo perfis a partir de atributos de alta granularidade, ou seja, utilizam características que representam os conhecimentos e habilidades de forma abstrata. Tsegn et al. [99] utilizam em sua solução atributos como, por exemplo, “técnicas de rede”, “design de software”, “teste de hardware”, dentre outras. Domingo et al. [52] propuseram uma abordagem na qual os perfis são compostos por atributos como “fundamentos de programação”, “qualidade de software” e “sistemas operacionais”. Silva et al. [30] utilizam atributos como “conhecimento de domínio”, “técnicas de negociação” e “gerenciamento de risco”. Este tipo de especificação em alto nível dificulta a formação de equipes capazes de satisfazer adequadamente demandas que envolvam elementos específicos. Por exemplo, o ato de desenvolver um software envolve a escrita, teste e manutenção do mesmo, que são tarefas que dependem dos conhecimentos e habilidades em linguagens de programação, interfaces de programação de aplicativos (API - *Application Programming Interface*), ferramentas etc. Ao se utilizar um atributo do tipo “experiência em desenvolvimento”, não são consideradas demandas específicas de conhecimento e habilidade que são importantes para a formação de equipes. Uma equipe

de projeto de software para dispositivo portátil deve possuir habilidades e conhecimentos diferentes de uma equipe para software web. Dependendo da plataforma para dispositivo portátil, competências diferentes são necessárias devido às diferenças de linguagem, modelo de desenvolvimento e até mecanismos de testes e garantia de qualidade.

Outra lacuna encontrada nas pesquisas da literatura diz respeito à subjetividade e confiabilidade das informações utilizadas para formação das equipes. Na maioria dos trabalhos, realiza-se a quantificação dos atributos por meio dos gerentes de projetos, grupos de especialistas ou pelos próprios desenvolvedores. Gray et al. [43] enviaram questionários para coletar as informações necessárias para criação dos perfis. Domingo et al. [52] utilizaram um método de avaliação conhecido como *feedback* 360º, em que cada indivíduo é responsável por quantificar tanto seus atributos quanto os dos seus colegas. Palacios et al. [20] propuseram uma solução em que cada gerente de projeto é encarregado pela atribuição dos valores dos atributos dos membros de suas respectivas equipes. A prática de se autoavaliar ou avaliar terceiros resulta em uma visão parcial ou irreal sobre o indivíduo alvo, podendo resultar na formação de equipes incapazes de atender as demandas estabelecidas. Estas técnicas são importantes como suporte complementar, mas não são suficientes para atingir o nível de eficácia e de granularidade de identificação de competência, sem viés, pra demandas específicas.

Diante do exposto, surge a seguinte questão: como formar equipes capazes de atender demandas específicas de projetos de software, a partir da criação de perfis técnicos com base em habilidades e conhecimentos de baixa granularidade, reduzindo o viés associado à geração das competências, de forma a otimizar a formação global?

1.2 Objetivos

O objetivo geral da pesquisa é a concepção de uma abordagem de suporte à tomada de decisão dos gestores durante o processo de formação de múltiplas equipes para projetos ágeis de software baseados em Scrum, considerando as *hard skills* dos indivíduos.

A abordagem tem como base a análise do histórico de projetos de uma organização e dos indivíduos que participaram de tais projetos, criando um mecanismo automatizado de formação de perfil de competência, com base em habilidades e conhecimentos específicos e de baixa granularidade. Em seguida, utiliza-se AG para propor a formação de múltiplas

equipes com base no perfil dos indivíduos, buscando o melhor resultado global dentro da organização.

A validação e a avaliação do trabalho foram realizadas mediante um estudo experimental, com dados reais de projetos de desenvolvimento de software conduzidos no Laboratório de Sistemas Embarcados e Computação Pervasiva [3] (Embedded), o qual faz parte do Centro de Engenharia Elétrica e Informática da Universidade Federal de Campina Grande (UFCG), em Campina Grande, Paraíba.

O objetivo principal da pesquisa pode ser definido como a composição dos seguintes objetivos específicos:

- Definir um modelo de especificação de tarefas de baixa granularidade que viabilize a criação de perfis técnicos capazes de refletir as competências adquiridas durante o desenvolvimento de software;
- criar uma abordagem de suporte à decisão com AG que possibilite a formação de múltiplas equipes para projetos ágeis baseados em Scrum, de modo a atender às demandas específicas de cada projeto e alcançar uma formação global otimizada;
- validar e avaliar a abordagem proposta em projetos reais de desenvolvimento de software baseados em Scrum.

1.3 Metodologia

Este trabalho está inserido no contexto do grupo ISE (*Intelligent Software Engineering Group*) [4], uma iniciativa do Núcleo de Pesquisa, Desenvolvimento e Inovação em Tecnologia da Informação, Comunicação e Automação (VIRTUS) [5] da UFCG. No ISE, todas as pesquisas são baseadas em problemas reais identificados no VIRTUS, que possui uma equipe de mais de 200 profissionais e executa, por ano, cerca de 40 projetos de pesquisa e desenvolvimento para clientes industriais, predominantemente na área de software. Uma vez identificado o problema específico do VIRTUS na formação de múltiplas equipes, buscou-se na literatura a generalização do problema e também eventuais soluções existentes, através da

[3] http://www.embeddedlab.org/

[4] http://isegroup.org

[5] http://virtus.ufcg.edu.br

realização de um mapeamento sistemático. Com motivação na ausência de abordagens na literatura para a solução do problema geral de formação automatizada de múltiplas equipes com base em informações de baixa granularidade de competência, definiram-se o problema e os objetivos deste trabalho.

Após a definição do problema e dos objetivos, foram definidas as questões de pesquisa e suas respectivas hipóteses, as quais são apresentadas a seguir.

- QP_1: É possível utilizar dados históricos para construir perfis técnicos com base em artefatos produzidos durante o processo de desenvolvimento de software?
  - H_{1-1}: Os conhecimentos técnicos de um desenvolvedor podem ser modelados com a atribuição de *tags* aos artefatos produzidos durante o processo de desenvolvimento de software.
  - H_{1-2}: As habilidades técnicas de um desenvolvedor podem ser modeladas a partir da padronização de artefatos produzidos durante o processo de desenvolvimento de software.
- QP_2: É possível formar equipes utilizando Algoritmos Genéticos, buscando alcançar uma formação global com base na modelagem dos conhecimentos e habilidades dos desenvolvedores?
  - H_{2-1}: Os conhecimentos técnicos de um desenvolvedor podem ser quantificados e utilizados como parâmetro da função de *fitness* de Algoritmos Genéticos para formar equipes.
  - H_{2-2}: As habilidades técnicas de um desenvolvedor podem ser quantificadas e utilizadas como parâmetro da função de *fitness* de Algoritmos Genéticos para formar equipes.
  - H_{2-3}: É possível utilizar diferentes funções de *fitness* para formar múltiplas equipes, buscando alcançar uma formação global otimizada.
- QP_3: A abordagem proposta consegue dar suporte à tomada de decisão, no sentido de auxiliar os gestores na escolha de suas equipes de acordo com as demandas definidas?
  - H_{3-1}: A abordagem proposta consegue sugerir equipes capazes de atender as demandas definidas para os projetos.

Esta pesquisa segue o ciclo de engenharia proposto por Wieringa et al. [104], o qual compreende a investigação do problema, design da solução, implementação da solução, validação e avaliação. Neste sentido, a pesquisa iniciou-se com a verificação do problema no VIRTUS e confirmação na literatura mediante mapeamento sistemático seguindo as diretrizes do método *Snowballing*, com foco na identificação dos principais tipos de atributos e técnicas utilizados para formação de equipes no contexto de desenvolvimento de software. Como resultado, constatou-se lacunas em relação à construção dos perfis dos indivíduos. Desta forma, definiu-se um modelo de especificação de tarefas baseado na atribuição de *tags*, com o intuito de prover uma forma de criar perfis com base em conhecimentos e habilidades de baixa granularidade, coletados durante o processo de desenvolvimento de software. O modelo foi aplicado em 12 projetos ágeis baseados em Scrum, dos quais participaram 52 desenvolvedores. Em seguida, foi gerada uma base de dados históricos para possibilitar operacionalização da abordagem, a qual foi implementada utilizando-se Algoritmo Genético. Para validação, foram definidos 13 cenários criados com dados históricos, com o intuito de encontrar a melhor configuração de parâmetros para compor a abordagem proposta, bem como verificar a precisão dos resultados obtidos. Por fim, seis novos cenários foram definidos e a abordagem foi avaliada, em termos de precisão e utilidade, por quatro gestores com experiência em projetos ágeis.

Do ponto de vista cronológico, as seguintes atividades foram desenvolvidas ao longo do trabalho, respaldando a metodologia:

1. Investigação das principais abordagens para formação de equipes no contexto de desenvolvimento de software, encontradas na literatura (mapeamento sistemático);
2. Definição e análise da viabilidade de utilização de Algoritmos Genéticos como base para a abordagem proposta;
3. Definição de um modelo de especificação de tarefas de baixa granularidade de modo que possam ser coletados dados, durante o processo de desenvolvimento de software, que possibilitem a criação de perfis que reflitam as competências técnicas dos desenvolvedores;
4. Construção de uma base de dados a partir de informações reais provenientes de projetos de desenvolvimento de software, a fim de possibilitar a operacionalização da abordagem proposta;

5. Concepção da abordagem de suporte à tomada de decisão utilizando algoritmo genético;
6. Implementação da abordagem proposta como uma ferramenta de software;
7. Validação e avaliação da abordagem proposta em projetos reais de software no Laboratório Embedded;
8. Redação deste documento de Tese.

1.4 Contribuições e Relevância

As principais contribuições alcançadas neste trabalho foram:

- Definição de um modelo de baixa granularidade para criação de perfis técnicos com base em conhecimentos e habilidades adquiridos durante o processo de desenvolvimento de software;
- Criação de uma base de dados históricos com informações reais de projetos, desenvolvedores e tecnologias utilizadas no nível de tarefas técnicas oriundas do arcabouço Scrum;
- Criação de uma abordagem para otimizar o processo de formação de múltiplas equipes, dando suporte aos gestores, no intuito de fornecer maior confiabilidade na tomada de decisão em relação à formação de suas equipes.

Do ponto de vista de relevância, o trabalho contribuiu com o avanço no estado da arte, no sentido de oferecer mecanismos para modelar as informações geradas nos ambientes internos às empresas, possibilitando assim a criação de perfis que possam refletir mais fielmente as competências técnicas adquiridas pelos profissionais.

A base de dados resultante da coleta de informações provenientes de projetos de desenvolvimento reais também é uma contribuição do trabalho e possibilitará a condução de pesquisas futuras na área de ES.

Além disto, acredita-se que a abordagem proposta irá beneficiar especialmente médias e grandes empresas, tornando o processo de formação de múltiplas equipes mais rápido e menos suscetível ao erro, proporcionado maior confiabilidade na decisão final dos gestores em relação à formação de equipes.

Por fim, espera-se que a abordagem proposta possa contribuir com as pesquisas do ISE, que tem por objetivo avançar no estado da arte de produtividade em Engenharia de Software através do uso de técnicas inteligentes.

1.5 Estrutura do Documento

O restante do documento está organizado da seguinte forma:

- No Capítulo 2, é apresentada a fundamentação teórica da pesquisa, abordando conceitos necessários ao entendimento do restante do documento;
- No Capítulo 3, é apresentado um mapeamento sistemático centrado no tema de formação de equipes no contexto de desenvolvimento de software;
- No Capítulo 4, é apresentada a abordagem proposta de apoio à tomada de decisão para formação de múltiplas equipes em processos ágeis;
- No Capítulo 5, é apresentada a validação e avaliação da abordagem proposta mediante um estudo experimental com dados reais de projetos de desenvolvimento de software;
- No Capítulo 6, são apresentadas as considerações finais do trabalho.

Capítulo 2

Fundamentação Teórica

2.1 Desenvolvimento Ágil de Software

Os métodos ágeis [17, 61] surgiram como uma reação ao desenvolvimento tradicional de software, com o objetivo de diminuir a sobrecarga inerente aos métodos tradicionais e tentar atender à demanda do mercado por mais produtividade, flexibilidade e qualidade nos projetos.

O desenvolvimento ágil consiste em uma maneira mais simples e eficiente de construir software, buscando maximizar a satisfação do cliente, com foco nos resultados e na colaboratividade de todos os interessados. Este processo ganhou notoriedade em 2001 com base nos preceitos do Manifesto Ágil [14], o qual foi elaborado por diversos profissionais experientes da indústria de software, de acordo com os seguintes valores:

- *Indivíduos e interações* mais que processos e ferramentas;
- *Software em funcionamento* mais que documentação abrangente;
- *Colaboração com o cliente* mais que negociação de contratos;
- *Respostas a mudanças* mais que seguir um plano.

A partir do Manifesto Ágil, os métodos ágeis ganharam destaque na indústria de software, dentre eles, os mais comuns são Kanban [1], *eXtreme Programming* [58] e Scrum [84]. De maneira geral, os métodos ágeis são fundamentados em doze princípios contidos no Manisfesto Ágil, descritos a seguir.

1. *“Nossa maior prioridade é satisfazer o cliente, através da entrega adiantada e contínua de software de valor.”*
2. *“Aceitar mudanças de requisitos, mesmo no fim do desenvolvimento. Processos ágeis se adéquam a mudanças, para que o cliente possa tirar vantagens competitivas.”*
3. *“Entregar software funcionando com frequência, na escala de semanas até meses, com preferência aos períodos mais curtos.”*
4. *“Pessoas relacionadas à negócios e desenvolvedores devem trabalhar em conjunto e diariamente, durante todo o curso do projeto.”*
5. *“Construir projetos ao redor de indivíduos motivados. Dando a eles o ambiente e suporte necessário, e confiar que farão seu trabalho.”*
6. *“O Método mais eficiente e eficaz de transmitir informações para, e por dentro de uma equipe de desenvolvimento, é através de uma conversa cara a cara.”*
7. *“Software funcional é a medida primária de progresso.”*
8. *“Processos ágeis promovem um ambiente sustentável. Os patrocinadores, desenvolvedores e usuários, devem ser capazes de manter indefinidamente, passos constantes.”*
9. *“Contínua atenção à excelência técnica e bom design, aumenta a agilidade.”*
10. *“Simplicidade: a arte de maximizar a quantidade de trabalho que não precisou ser feito.”*
11. *“As melhores arquiteturas, requisitos e designs emergem de equipes auto-organizáveis.”*
12. *"Em intervalos regulares, a equipe reflete em como ficar mais efetivo, então, se ajustam e otimizam seu comportamento de acordo.”*

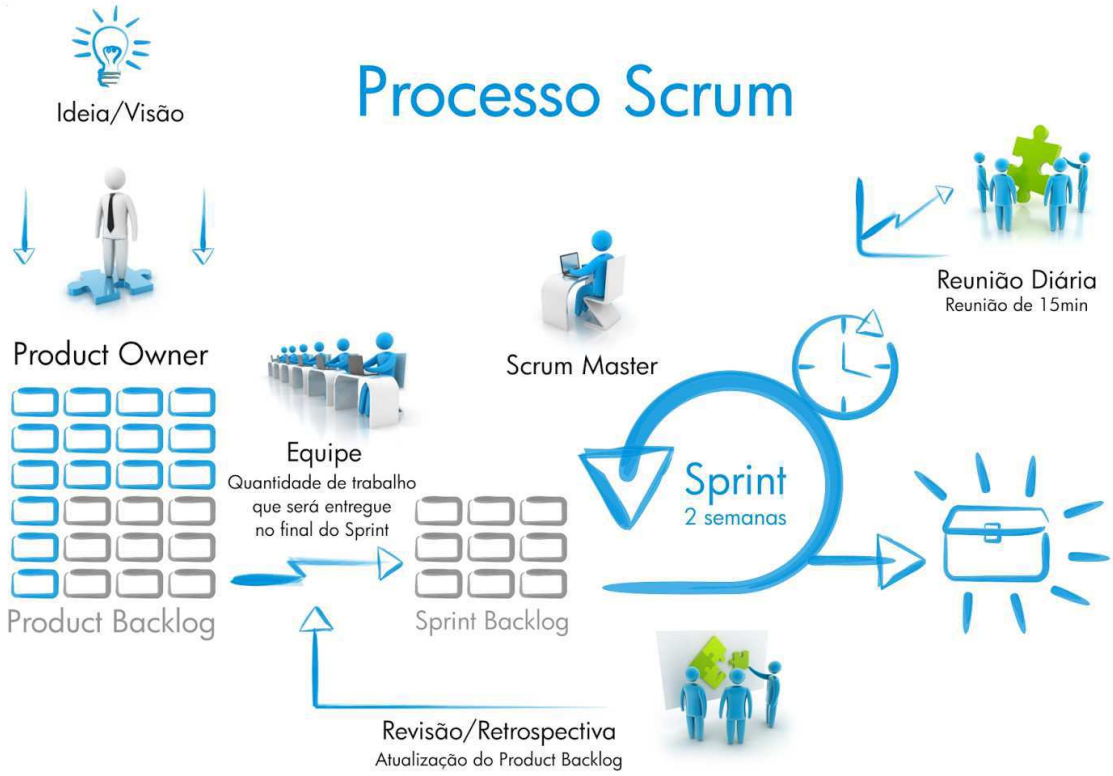
2.2 Scrum

O Scrum é um arcabouço criado por Ken Schwaber e Jeff Sutherland, no início da década de 90, para manter e desenvolver produtos complexos. O arcabouço engloba um conjunto de valores, princípios e práticas que fornecem a base para um processo iterativo e incremental para gerenciamento de projetos e desenvolvimento ágil de software.

2.2.1 Visão Geral

A partir da Figura 2.1 é possível observar uma visão geral da execução da metodologia Scrum. O *Product Owner* (PO) elabora o *Product Backlog* (PB) e a cada *Sprint* prioriza um conjunto de Itens de *Backlog* a serem desenvolvidos. A equipe por sua vez, quebra o *Sprint Backlog* em tarefas técnicas que irão guiar os membros para alcançar o objetivo da *Sprint*, nas próximas duas a quatro semanas. Durante a *Sprint*, são realizadas reuniões diárias de no máximo 15 minutos, a fim de disseminar o estado atual das tarefas definidas. Ao final da *Sprint* um Incremento do produto é entregue e as novas funcionalidades são apresentadas e avaliadas na *Sprint Review*. Posteriormente, todo o processo é apreciado com o objetivo de introduzir ações que venham a melhorá-lo.

Figura 2.1: Visão geral do Scrum.



Fonte: imagem disponível em [70].

2.2.2 Artefatos do Scrum

Os artefatos do Scrum correspondem aos elementos, geralmente de natureza textual ou gráfica, que representam trabalho ou valor para o fornecimento de transparência e oportunidades para inspeção e adaptação. Os principais artefatos do Scrum são: *Product Backlog*, *Sprint Backlog* e Incremento. Existem outros artefatos, que apesar de não estarem definidos no Guia do Scrum [84], são comumente utilizados pela grande maioria das equipes Scrum, tais como Gráfico de *Burndown*, Quadro de tarefas e *User Stories* (US).

Product Backlog

É uma lista ordenada com os requisitos necessários para desenvolver e entregar o produto final após a execução do projeto. Geralmente, o PB é dividido em Itens de *Backlog* que podem ser representados por US, a fim de facilitar a compreensão do cliente. Originalmente, o Guia do Scrum não especifica um padrão para escrita de requisitos, porém é bastante comum a definição em forma de US que consiste em uma descrição de uma necessidade de um usuário do produto [18].

Sprint Backlog

Representa o conjunto de requisitos necessários para cumprir o objetivo da *Sprint*, o qual corresponde à entrega de uma parte do produto. Durante a elaboração do *Sprint Backlog*, os Itens de *Backlog* são decompostos em tarefas técnicas a serem realizadas dentro da *Sprint*.

Incremento

O Incremento representa a soma de todos os Itens de *Backlog* desenvolvidos durante a *Sprint*, juntamente com o valor dos Incrementos de *Sprints* anteriores [84]. Dizer que um Incremento está "pronto" significa que ele está em condições de ser utilizado e de acordo com as definições da equipe.

2.2.3 Equipe Scrum

O Scrum é uma metodologia ágil que é centrada na equipe, na qual seus membros trabalham colaborativamente para atingir um objetivo em comum. O arcabouço promove uma interação

efetiva entre os membros da equipe para que eles possam entregar valor de negócio. A Equipe Scrum é composta por três papéis fundamentais: *Product Owner*, Equipe (ou Time) de Desenvolvimento e *Scrum Master*.

Product Owner

O PO é o responsável por fazer a ponte entre a área de negócios e a Equipe de Desenvolvimento. O PO é uma peça chave de um projeto ágil, uma vez que ele é o responsável por elaborar, gerenciar e manter o PB, ou seja, é ele quem decide quais recursos e funcionalidades serão construídos e em qual ordem devem ser feitos.

Equipe de Desenvolvimento

São os profissionais responsáveis por transformar Itens de *Backlog* em incremento de software potencialmente utilizável. Na Equipe de Desenvolvimento não existe uma divisão funcional através de papéis como analista, programador, testador, etc. Todos os membros devem possuir conhecimentos e habilidades específicas e generalistas simultaneamente, formando assim uma equipe multidisciplinar. Além disto, a equipe deve ser auto-organizável, ou seja, os membros precisam ser estruturados e autorizados para organizar e gerenciar seu próprio trabalho. De acordo com o Guia do Scrum, a Equipe de Desenvolvimento deve ser composta por 3 a 9 integrantes, sem levar em consideração o PO e *Scrum Master*. A formação de múltiplas equipes a que se propõe este trabalho se refere à Equipe de Desenvolvimento.

Scrum Master

O *Scrum Master* atua como um facilitador e potencializador do trabalho da equipe. Ele é o responsável por assegurar que a equipe respeite e siga os valores e as práticas do Scrum. Além disto, cabe também ao *Scrum Master* remover todos e quaisquer impedimentos que possam interferir no objetivo da equipe.

2.2.4 Eventos do Scrum

Os eventos do Scrum são utilizados para estabelecer uma rotina e minimizar a necessidade de reuniões não definidas pelo arcabouço. Todos os eventos são *time-boxed*, ou seja, todos

têm uma duração máxima predefinida, o que garante uma quantidade de tempo adequada para planejamento, sem prejuízo ao processo. Além disto, os eventos do Scrum proporcionam transparência e inspeção criteriosa. Os principais eventos do Scrum são: *Sprint*, *Sprint Planning*, *Daily Scrum*, *Sprint Review* e *Sprint Retrospective*.

Sprint

No Scrum, o desenvolvimento do produto é dividido em etapas, chamadas de *Sprints*. A *Sprint* é um evento *time-boxed* com duração de duas a quatro semanas e é considerada o coração do Scrum. Ao fim de uma *Sprint*, uma versão incremental de software potencialmente utilizável é criada.

Sprint Planning

O *Sprint Planning* é um evento de duração máxima de 8 horas, no qual devem estar presentes o PO, o *Scrum Master* e a Equipe de Desenvolvimento, bem como, quaisquer interessados que possam estar representando a gerência ou o cliente. Nesta fase, são abordados dois pontos principais: i) o que será entregue? ii) como será realizado o trabalho?

Durante o evento, o PO define as funcionalidade (Itens de *Backlog*) prioritárias para serem executadas na *Sprint*, esclarecendo quaisquer dúvidas da equipe. Em seguida, elabora-se o *Sprint Backlog*, no qual as funcionalidades previamente definidas são quebradas em tarefas técnicas que guiarão os esforços para atingir o objetivo da *Sprint*.

O objetivo da *Sprint* também é definido durante esse evento, fornecendo orientações ao Time de Desenvolvimento ao longo da implementação do Incremento. Este objetivo proporciona flexibilidade à equipe para priorizar determinadas funcionalidades.

Daily Meeting

É uma reunião diária de no máximo 15 minutos, realizada durante toda a *Sprint*, geralmente no mesmo lugar e na mesma hora do dia e com a participação de toda a Equipe de Desenvolvimento. Ela tem como objetivo compartilhar o conhecimento acerca do estado atual do projeto, mediante três perguntas básicas:

1. O que você fez ontem que ajudou a Equipe de Desenvolvimento a alcançar os objetivos da *Sprint*?
2. O que você fará hoje para ajudar a Equipe de Desenvolvimento a alcançar os objetivos da *Sprint*?
3. Há algum impedimento que prejudique a Equipe de Desenvolvimento a alcançar os objetivos da *Sprint*?

Sprint Review

Este evento tem duração de até 4 horas e é realizado ao final de cada *Sprint* com o intuito de avaliar o projeto em relação aos objetivos da *Sprint*. Normalmente, este evento assume um formato de demonstração de novas funcionalidades para o PO. Desta forma, ele poderá checar e avaliar aquilo que foi considerado como pronto.

Sprint Retrospective

É um evento de duração máxima de 3 horas e ocorre ao final da *Sprint*. Realiza-se uma reunião, na qual o objetivo maior é a garantia de qualidade, mas não em relação ao produto em si, mas sim ao processo. Ela serve para identificar o que deu certo e errado e que ações tomar para melhorar.

2.3 Algoritmos Genéticos

Os métodos de busca tradicionais são otimizados para meios em que os dados armazenados possuem características bem definidas, restringindo o espaço de busca e, consequentemente, auxiliando no desempenho destes métodos. Por exemplo, realizar uma busca por determinado usuário em uma base de dados. Em contrapartida, existem problemas que são aparentemente simples, mas que possuem tamanha complexidade que não admitem solução em tempo viável mediante uso de técnicas tradicionais. Tais problemas são conhecidos como NP-difícil [105]. Por exemplo, o problema da mochila e a Torre de Hanoi. Nestes casos, em que o espaço de soluções é grande ou até mesmo infinito, Algoritmo Genético surge como uma alternativa.

Algoritmo Genético [5] faz parte da Computação Evolucionária [54], subárea da Inteligência Artificial, e consiste em uma técnica de busca e otimização, inspirada na Teoria de Evolução das Espécies de Charles Darwin [23], a qual defende que os organismos mais adaptados ao meio têm mais chances de sobrevivências, produzindo um número maior de descendentes.

2.3.1 Terminologia

A terminologia dos AG (Figura 2.2) é uma referência à Biologia Evolutiva e os principais termos são detalhados a seguir:

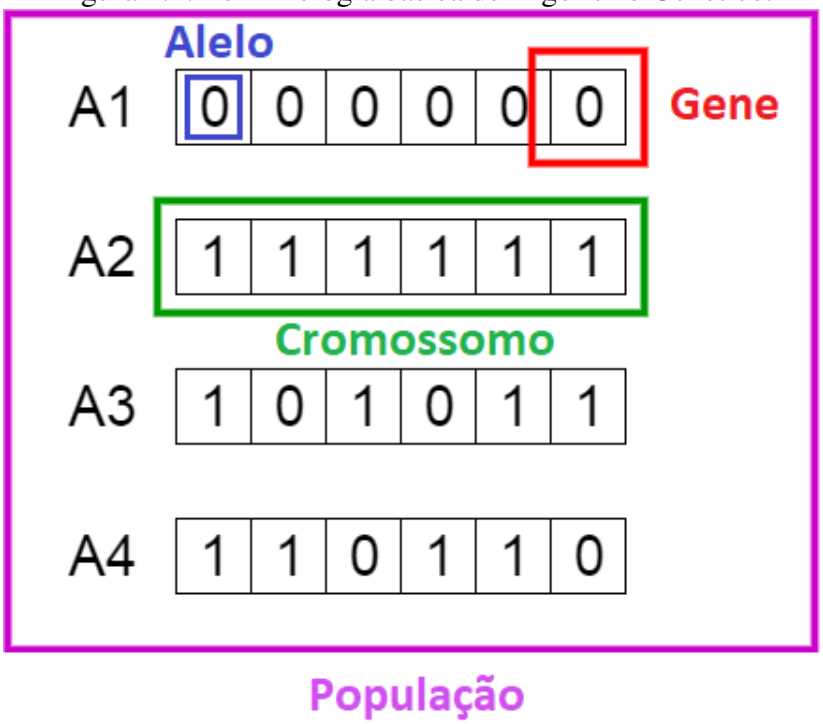
- **População:** é um subconjunto de todas as possíveis soluções para um determinado problema;
- **Cromossomo (ou indivíduo):** representa uma possível solução para um determinado problema;
- **Gene:** é uma posição do elemento de um cromossomo;
- **Alelo:** é o valor de um gene em um determinado cromossomo;
- **Genótipo:** representa o conjunto de possíveis soluções (população) no espaço computacional, de modo que elas possam ser manipuladas por um sistema computacional;
- **Fenótipo:** representa o conjunto de possíveis soluções (população) no espaço real, de modo que elas possam ser compreendidas em situações do mundo real.

2.3.2 Tipos de Representação

Uma das decisões que deve ser tomada ao se implementar um AG diz respeito ao tipo de representação da solução. A seguir, são detalhadas algumas representações utilizadas em AG.

- **Representação Binária:** uma das mais simples, porém mais usadas em AG é a binária (Figura 2.3(a)). Para alguns problemas, o espaço de soluções é formado por variáveis de decisão booleanas, fazendo com que o genótipo corresponda a vetor de bits (0 e 1). Por exemplo, o problema da mochila (binária) [89], que consiste em dada uma mochila com uma capacidade limitada, e alguns itens que possuem pesos e valores, escolher

Figura 2.2: Terminologia básica de Algoritmo Genético.



Fonte: imagem adaptada a Internet.

aqueles cujo peso somado não exceda a capacidade máxima da mochila e a soma dos valores seja a maior possível.

- **Representação Inteira:** algumas vezes não é possível limitar o espaço de soluções a valores binários "sim" ou "não", portanto, utilizam-se valores discretos (Figura 2.3(c)). Por exemplo, encontrar um conjunto de valores inteiros que satisfaçam uma determinada equação polinomial.
- **Representação Real:** é indicada para problemas em que se deseja definir genes utilizando valores contínuos ao invés de discretos (Figura 2.3(b)).
- **Representação de Permutação:** é usada, geralmente, em problemas nos quais a solução é representada pela ordem dos elementos (Figura 2.3(d)), não podendo haver repetição entre eles. Um exemplo clássico é o problema do Caixeiro Viajante [77], em que um vendedor precisa viajar por todas as cidades, visitando cada cidade uma única vez e regressando à cidade inicial, na menor distância possível.

Figura 2.3: Tipos de representações em AG.

Fonte: imagem adaptada de [100].

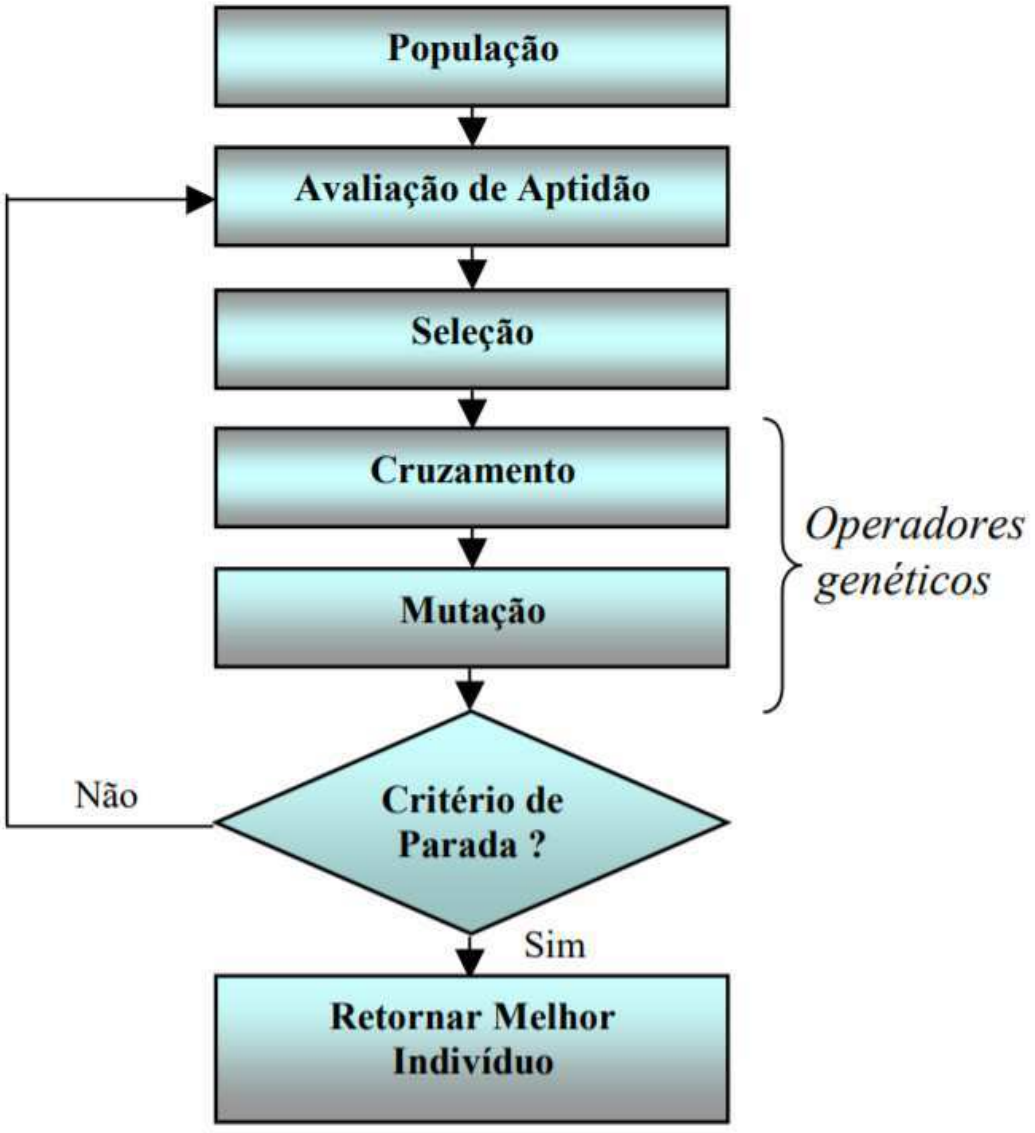
2.3.3 Estrutura

A estrutura básica de um AG obedece a um ciclo de funcionamento que pode ser observado na Figura 2.4. Na fase inicial, uma população é gerada a partir de um conjunto aleatório de cromossomos, em que cada um corresponde a uma possível solução para o problema. Em seguida, os cromossomos são avaliados e recebem, individualmente, determinados valores de aptidão que representam suas respectivas capacidades de adaptação ao ambiente. Um subconjunto mais apto da população é mantido e o restante é descartado (seleção). Os indivíduos mais adaptados podem sofrer alterações a partir de operadores genéticos, tais como cruzamento e mutação, fornecendo descendentes para a próxima geração. Este ciclo é repetido até que uma solução satisfatória seja encontrada. Nas seções subsequentes são detalhados os componentes do ciclo de um AG.

2.3.3.1 População Inicial

A execução de um AG ocorre a partir da geração de sua população inicial [5]. Em geral, a população é gerada de maneira aleatória, com o intuito de aumentar sua diversidade genética e tentar garantir um maior alcance do espaço de busca. Outra forma de gerar a população

Figura 2.4: Estrutura básica de um algoritmo genético.



Fonte: imagem disponível em [9].

inicial é partir de uma função heurística, a qual busca fornecer uma simplificação ou aproximação para reduzir ou limitar o espaço de busca. Entretanto, ao criar a população de forma determinística, esta pode acabar convergindo prematuramente, ou seja, é possível que em um curto período de tempo a população possua indivíduos muito semelhantes, resultando em pouca diversidade e dificultando assim a escolha de uma solução ótima para o problema.

2.3.3.2 Função de Fitness

A função de *fitness* pode ser considerada o principal mecanismo de um AG, uma vez que ela representa o único elo entre o AG e problema proposto. Esta função tem como objetivo fornecer uma nota individual (valor de *fitness*) para cada cromossomo de acordo com os parâmetros estabelecidos para o problema. Esta nota é posteriormente usada na seleção dos indivíduos mais aptos.

É de fundamental importância que a função de *fitness* tenha máxima representatividade para que seja possível diferenciar corretamente as boas soluções das más. Quando uma função é pouco representativa em sua avaliação, uma solução ótima pode ser descartada durante a execução do algoritmo. Por exemplo, ao considerar o clássico problema do Caixeiro Viajante [77] em que um vendedor precisa viajar por todas as cidades, visitando cada cidade uma única vez e regressando à cidade inicial, na menor distância possível. Uma função de *fitness* poderia ser dada pelo inverso da soma das distâncias de um determinado conjunto de cidades, porém se a esta função não for incorporada um mecanismo para penalizar soluções que contenham cidade "repetidas", as soluções encontradas serão pouco representativas para o problema em questão.

2.3.3.3 Seleção

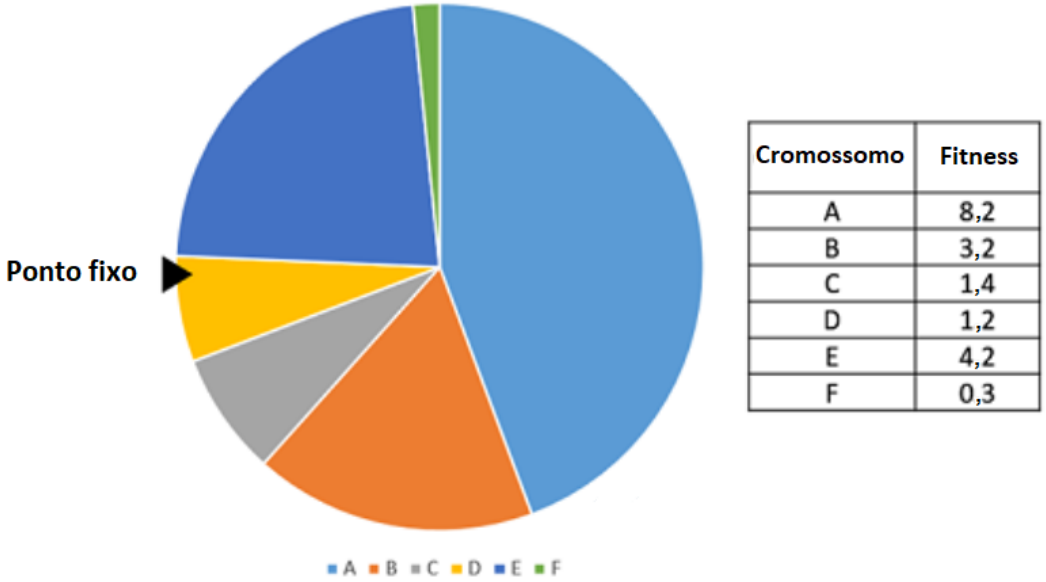
Após o cálculo de aptidão de todos os cromossomos da população, realiza-se o processo de seleção [5], no qual os indivíduos (pais) com maior valor de aptidão possuirão maior probabilidade de gerar mais descendentes. A seleção não é baseada somente nos indivíduos mais aptos, pois há chances de um indivíduo com valor de aptidão menor possuir características genéticas favoráveis à geração de um cromossomo que venha a representar uma melhor solução para o problema. Desta forma, os indivíduos menos aptos também poderão gerar descendentes, porém isto acontece em menor escala. Existem diversos métodos de seleção,

dentre eles, têm-se: Roleta, Amostragem Estocástica Universal, Seleção por Rank e Torneio.

Roleta

É um método que seleciona os indivíduos de acordo com seus valores de *fitness*. Na Figura 2.5 é apresentada uma ilustração do funcionamento deste método. Cada indivíduo é representado na roleta, em um local proporcional ao seu valor de *fitness*. Desta forma, indivíduos com maiores aptidões têm maiores probabilidades de serem selecionados ao "girar" a roleta. Em contrapartida, aos indivíduos menos aptos são atribuídas porções menores da roleta, consequentemente, apresentando menos chances de serem escolhidos.

Figura 2.5: Método de seleção: Roleta.



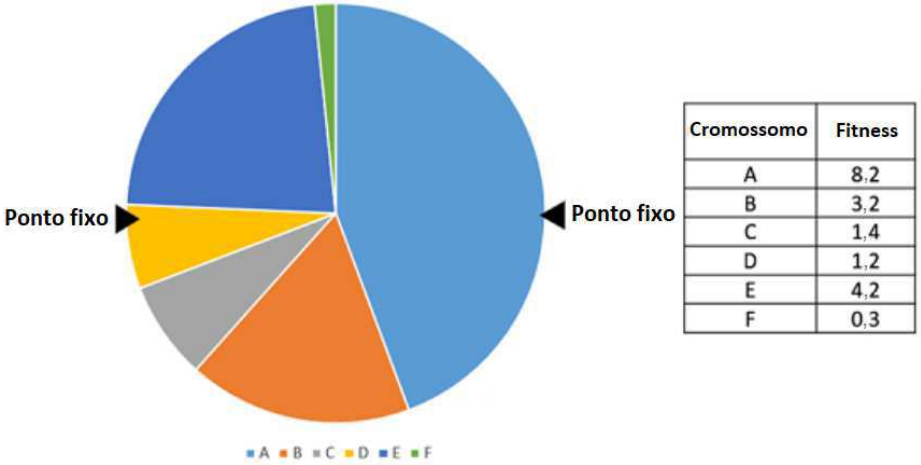
Cromossomo	Fitness
A	8,2
B	3,2
C	1,4
D	1,2
E	4,2
F	0,3

Fonte: imagem adaptada de [100].

Amostragem Estocástica Universal

Este método é semelhante ao da Roleta (Figura 2.6), com a exceção de que são definidos múltiplos pontos de escolha, ou seja, em um único "giro" vários pais são selecionados. A vantagem deste método é que há mais chances de que os indivíduos de maior *fitness* sejam escolhidos ao menos uma vez.

Figura 2.6: Método de seleção: Amostragem Estocástica Universal.



Cromossomo	Fitness
A	8,2
B	3,2
C	1,4
D	1,2
E	4,2
F	0,3

Fonte: imagem adaptada de [100].

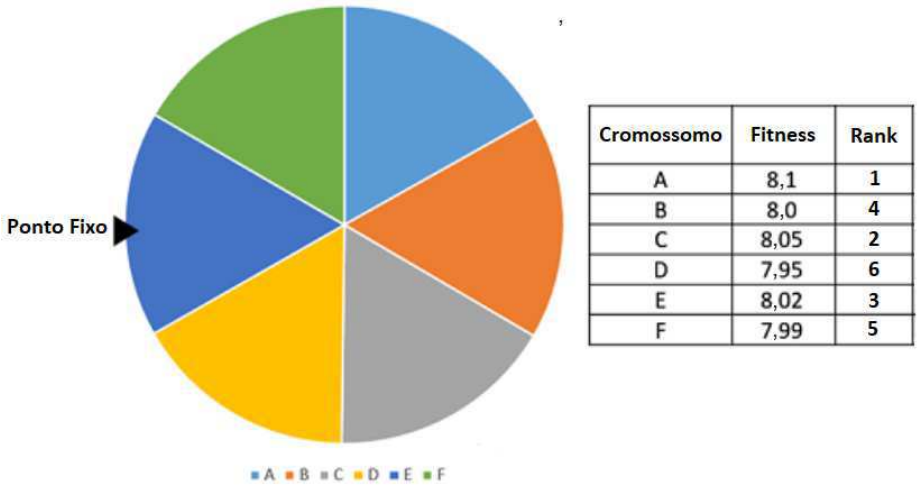
Seleção por Rank

É um método de seleção indicado para quando os indivíduos da população possuem valores de *fitness* muito próximos. Nestes casos, tanto os indivíduos mais aptos quanto os menos aptos teriam, aproximadamente, as mesmas probabilidades de serem escolhidos, como observado na Figura 2.7. Tal situação levaria o AG a fazer seleções pouco eficientes. Diante disto, os indivíduos são ordenados de acordo com seus valores de *fitness* e a seleção é realizada com base nesta estrutura. Quanto mais bem posicionado no *ranking*, maior a chance de um individuo ser escolhido. Uma das vantagens deste método é que ele pode ser usado quando se tem valores de *fitness* negativos, ao contrário dos métodos Roleta e Amostragem Estocástica Universal.

Torneio

Este método (Figura 2.8) seleciona aleatoriamente um número n de indivíduos que irão competir entre si. Aquele com maior valor de aptidão é selecionado e os demais são retornados para a população original. O processo é repetido até que a população temporária esteja completa. Este método é um dos mais populares na literatura e também funciona com valores de *fitness* negativos.

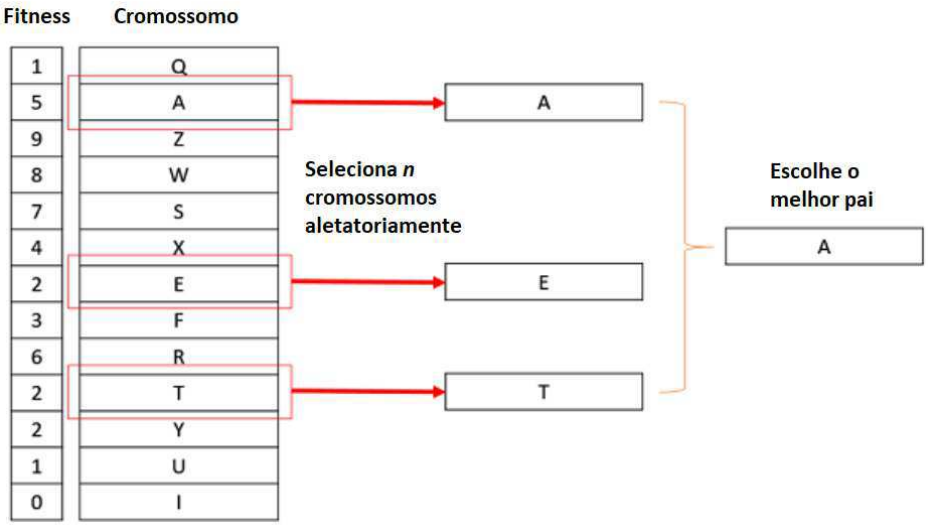
Figura 2.7: Método de seleção: Seleção por Rank.



Cromossomo	Fitness	Rank
A	8,1	1
B	8,0	4
C	8,05	2
D	7,95	6
E	8,02	3
F	7,99	5

Fonte: imagem adaptada de [100].

Figura 2.8: Método de seleção: Torneio.



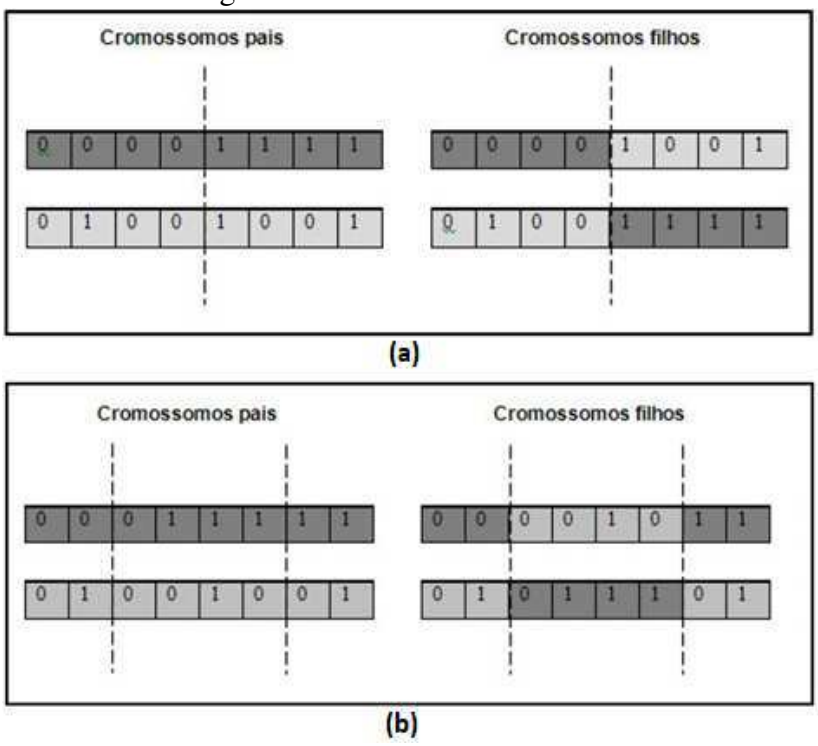
Fonte: imagem adaptada de [100].

2.3.3.4 Cruzamento (*Crossover*)

Também conhecido como *crossover*, o cruzamento [5] é um operador genético que consiste na combinação de características genéticas, mediante a troca de trechos equivalentes de dois cromossomos pais. O objetivo principal do cruzamento é propagar as características positivas dos indivíduos mais aptos da população, originando novos indivíduos. Os tipos de cruzamento mais comuns são:

- **Ponto Único:** escolhe-se um ponto de corte aleatório para troca de informações entre os indivíduos pais, dando origem a dois novos indivíduos (Figura 2.9(a)).
- **Ponto Duplo:** escolhem-se dois pontos de corte aleatórios para troca de informações de forma intercalada entre os indivíduos pais, dando origem a dois novos indivíduos (Figura 2.9(b)).
- **Multi-ponto:** escolhem múltiplos pontos de cortes aleatório para troca de informações entre os indivíduos pais.

Figura 2.9: Pontos de cruzamento.



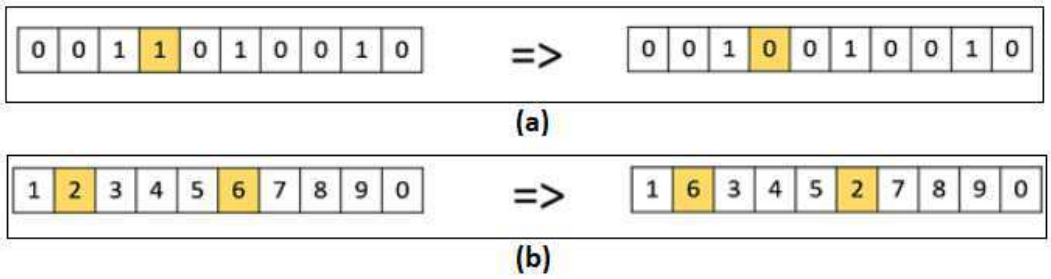
Fonte: adaptada da Internet.

2.3.3.5 Mutação

Outro operador genético corresponde à mutação [5], o qual consiste na modificação aleatória de determinadas características genéticas de indivíduos de uma população. O principal propósito da mutação é realizar a manutenção da diversidade genética da população, uma vez que isto possibilitará a obtenção de características genéticas que não existiam ou eram encontrados em menor frequência. A aplicação da mutação deve ocorrer em pequenas porcentagens (taxa de mutação), pois caso contrário os indivíduos gerados apresentarão pouca semelhança com seus pais. Os tipos de mutação mais comuns são:

- **Mutação por Flip:** um ou mais genes são escolhidos e seus valores são "virados", conforme pode ser observado na Figura 2.10(a). Este tipo de mutação é usado para representações binárias.
- **Mutação Aleatória:** dentre os possíveis valores que um gene pode assumir, um valor é sorteado e atribuído a um gene que sofrerá mutação.
- **Mutação por Troca:** também conhecida por *Swap Mutation*, neste tipo de mutação pares de genes são sorteados e em seguida são trocados dentro do cromossomo, conforme pode ser observado na Figura 2.10(b). Este tipo de mutação é utilizado em representações de permutação.

Figura 2.10: Tipos de mutação.



Fonte: imagem adaptada de [100].

2.3.3.6 Critério de Parada

Após a realização do processo evolutivo são realizados testes para verificar se a execução do AG deve ser concluída ou não. Estes testes correspondem a uma condição de parada previamente estabelecida. Os critérios de parada mais comuns são:

- Parar quando o AG atingir um determinado número de gerações;
- Parar quando obtiver um valor ótimo da função de aptidão, se este for conhecido;
- Quando não houver melhoramento significativo no cromossomo de maior valor de aptidão durante um determinado número de gerações.

2.3.4 Aplicações

Search Based Software Engineering é um campo recente da Engenharia de *Software* que utilizada técnicas de busca e otimização, dentre elas Algoritmo Genético, para fornecer soluções automatizadas ou semi-automatizadas para problemas complexos da ES. Este campo de otimização visa, de maneira geral, selecionar os melhores elementos a partir de algum conjunto de opções disponíveis. Esta seleção é realizada de acordo determinados critérios (função objetivo) e métodos de busca.

De acordo com Colanzi et al. [19], os algoritmos de busca e otimização mais utilizados na SBSE provém de dois grupos principais. O primeiro é composto por técnicas clássicas, tais como *Branch and Bound* e Programação Linear. Neste grupo os algoritmos são determinísticos, ou seja, produzem sempre o mesmo resultado diante de uma determinada entrada. O segundo grupo, inclui os meta-heurísticos, os quais são mais comumente utilizados na SBSE, devido a natureza dos problemas da ES, que representam situações do mundo real, e geralmente estão relacionados a objetivos que não podem ser caracterizados por equações lineares e nem são tratáveis por métodos determinísticos. AG está inserido neste último grupo.

Na SBSE, os AG podem ser utilizados para resolução de problemas, como por exemplo, de formação de equipes [21], de geração de testes [66], de estimativa de esforço [86], de estimativa de custo de software [60], etc. Além da área de ES, os AG são aplicados em problema tradicionais de otimização, tais como o problema de roteamento de veículos [69], que consiste em atender um conjunto de consumidores mediante uma frota de veículos que partem de um ou mais pontos, chamados de depósitos; e o problema de agendamento de tarefas em sistemas computacionais heterogêneos [109]. Na análise de imagens também é possível observar o uso de AG para reconhecimento facial [96], que é um método biométrico para identificar pessoas a partir de seus rostos; e na encriptação de imagens [75], uma vez que com o advento das multimídias, as imagens têm-se tornado um recurso vital para extração

de informações, tornando a segurança um problema a ser considerado. Além das aplicações supracitadas, tem-se o uso de AG nas áreas de Saúde para reconhecimento de doenças como Parkinson [10] e Alzheimer [53], em Aprendizado de Máquina [103], na Robótica [102], em *Design* de Aeronaves [87], dentre outras.

2.4 Considerações Finais do Capítulo

Foram apresentados conceitos relacionados ao arcabouço Scrum e a Algoritmo Genético, ambos fundamentais paro o entendimento da presente pesquisa. O aprofundamento acerca do Scrum, irá auxiliar a compreensão de como ocorre a coleta e estruturação dos dados, durante o processo de desenvolvimento de software, para criação dos perfis de desenvolvedores e projetos. A fundamentação sobre AG, por sua vez, irá ajudar a entender como a abordagem proposta utiliza as informações destes perfis para sugerir equipes, de modo a alcançar uma formação global otimizada.

Capítulo 3

Mapeamento Sistemático

Neste capítulo, é apresentado um mapeamento sistemático centrado no tema de formação de equipes para desenvolvimento de software. O processo de mapeamento foi conduzido com o auxílio de pesquisadores do grupo ISE da UFCG. O principal objetivo deste mapeamento foi identificar e analisar os principais trabalhos relacionados na literatura, com foco nas técnicas e atributos utilizados para formação de equipes.

3.1 Método de Mapeamento

O mapeamento sistemático foi realizado de acordo com as diretrizes do método *Snowballing* [106]. Neste procedimento, o primeiro passo é a definição de um conjunto inicial de artigos, conhecido como *Start Set*. De acordo com Wohlin [106], um *Start Set* representativo pode ser obtido com auxílio de uma ferramenta de busca, como por exemplo o Google Acadêmico[1]. Segundo Wohlin [106], o uso de uma ferramenta de busca corresponde a uma alternativa para se evitar vieses provenientes de publicadores específicos, uma vez que os artigos retornados não são atrelados a nenhuma base específica.

Uma vez definido o *Start Set*, incluindo apenas artigos que farão parte da análise final, inicia-se a primeira iteração, a qual é composta pelos processos de *Backward* e *Forward Snowballing*. No *Backward*, são extraídas as referências dos artigos do *Start Set*, em seguida, obedecendo os critérios de seleção pré-definidos (Seção 3.4) os artigos são escolhidos. O processo de *Forward* ocorre de maneira análoga ao anterior, entretanto ao invés das referên-

[1] https://scholar.google.com.br/

cias, as citações são extraídas. Os artigos incluídos a partir destes dois processos formarão o conjunto final de artigos da iteração atual e consequentemente representarão o ponto de partida para a iteração seguinte. A cada nova iteração os passos previamente descritos são repetidos até que nenhum novo artigo seja incluído e o conjunto final de artigos da iteração seja vazio.

3.2 Questões de Pesquisa

Conforme mencionado anteriormente, o principal objetivo deste mapeamento é identificar e analisar os principais trabalhos na literatura com base nas técnicas e atributos utilizados para formação de equipes. Portanto, foram formuladas as seguintes questões de pesquisa:

- $QP01$: Quais os principais fatores utilizados para a formação de equipes no contexto de desenvolvimento de software?
- $QP02$: Quais as principais técnicas utilizadas para a formação de equipes no contexto de desenvolvimento de software?

3.3 Termos de Busca

Os termos de busca foram obtidos dos resumos de artigos relacionados ao tema, previamente conhecidos pelo presente pesquisador. Internacionalmente, o termo mais usual entre os trabalhos que abordam o tema de formação de equipes é *Team Formation*, porém ao analisar artigos relacionados, foram encontrados termos como *Team Allocation*, *Team Selection*, *Team Composition*, *Team Configuration* e *Human Resouce Allocation*. A fim de minimizar o risco de não encontrar artigos relevantes, optou-se por mesclar o termo *Human Resource* com os demais, resultando em *Human Resource Selection*, *Human Resource Formation*, *Human Resource Configuration* e *Human Resource Composition*.

Conforme orientações presentes em [106] e detalhadas na Seção 3.1, utilizou-se o Google Acadêmico como ferramenta de busca, realizando as pesquisas dos termos apenas nos títulos dos artigos e com filtro para o ano de 2016, ano anterior à data de início deste mapeamento. A restrição do ano não oferece prejuízo aos resultados, pois artigos anteriores, em geral, são

encontrados durante o processo de *Backward*. De maneira análoga, acontece com os artigos de anos posteriores durante o processo de *Forward*.

3.4 Critérios de Seleção

Os critérios de seleção para inclusão e exclusão dos artigos foram divididos em três fases: critérios gerais de exclusão, critérios básicos e critérios avançados.

Critérios Gerais de Exclusão

Com o intuito de eliminar imediatamente artigos irrelevantes, decidiu-se descartar:

- Artigos publicados antes do século 21 (2001).
- Artigos não publicados em Inglês.
- Artigos duplicados ou já analisados em iterações anteriores.
- Revisões Sistemáticas.
- Artigos não completos (*Short Papers*) e Resumos Expandidos.
- Relatórios Técnicos, dissertações, teses e livros.
- Artigos não publicados em Jornais, Revistas, Conferências ou *Workshops*.
- Artigos que não apresentam algum método, abordagem, procedimento, modelo, ferramenta, diretrizes ou critérios que explicitamente forneçam meios para formar equipes de desenvolvimento de software.

Critérios Básicos

Nesta fase, pares de revisores são escolhidos aleatoriamente e ficam responsáveis por ler e analisar os títulos e resumos de um determinado subconjunto de artigos, remanescentes da fase anterior. Cada artigo é avaliado e classificado de acordo com o procedimento definido em Ali et al. [2] como:

- Relevante: artigos relacionados com o tema formação de equipes para desenvolvimento de software;
- Irrelevante: artigos não relacionados com o tema formação de equipes para desenvolvimento de software;

- Incerto: as informações disponíveis nos títulos e resumos são insuficientes ou inconclusivas para classificar o artigo como relevante ou irrelevante.

Ali et al. [2] sugerem seis níveis de concordância entre os revisores, como é possível observar na Figura 3.1. As categorias A ou B significam que pelo menos um dos revisores avaliou o artigo como relevante e consequentemente este artigo é incluído na próxima fase. A categoria B indica que um revisor classificou o artigo como Incerto, porém o artigo será incluído, almejando-se minimizar os riscos de se descartar um estudo significativo. Esta decisão não tem impacto negativo, pois se o artigo não for relevante será descartado na fase seguinte.

Figura 3.1: Nível de concordância.

		Revisor 2		
		Relevante	Incerto	Irrelevante
Revisor 1	Relevante	A	B	D
	Incerto	B	C	E
	Irrelevante	D	E	F

Fonte: imagem adaptada de [2].

A categoria C significa que nenhum dos revisores tem certeza da classificação e é necessária uma avaliação mais detalhada. Neste caso, ambos os revisores, independentemente, fazem uma leitura seletiva do artigo, começando pela seção de Introdução, não sendo suficiente, partindo para a Conclusão e, permanecendo a incerteza, usam-se as palavras chave do artigo para verificar a relação com o tema do mapeamento sistemático.

As categorias D e E indicam que os revisores estão em discordância e precisam discutir entre si quais razões os levaram àquela classificação. Se não houver um consenso, um terceiro revisor é chamado para dar um parecer final.

Por fim, a categoria F significa que os dois revisores concordam que o artigo não é relevante e não deve ser incluído.

Critérios Avançados

Nesta fase, os artigos são avaliados de maneira aprofundada, realizando-se uma leitura completa. Novamente, dois revisores (*Data Extractor* e *Data Checker*) são escolhidos aleato-

riamente e ficam responsáveis pela avaliação de um determinado subconjunto de artigos, remanescentes da fase anterior. O *Data Extractor* é encarregado da avaliação de qualidade do artigo (Seção 3.5) e da extração dos dados do mesmo (Seção 3.6). Em seguida, o *Data Checker* revisa o resultado, a fim de confirmar a corretude dos dados, seguindo as diretrizes sugeridas em Brereton et al. [15] e Staples e Niazi [90].

3.5 Avaliação de Qualidade

A fim de determinar a qualidade dos artigos encontrados, foram utilizados 11 critérios, estabelecidos em Dyba e Dygsoyr [29]. Neste caso, após a leitura do artigo, o *Data extractor* avalia cada um dos 11 itens utilizando uma escala booleana (0 ou 1). Por fim, o *Data Checker* revisa as respostas e, em caso de divergência, os dois revisores discutem e tentam chegar a um consenso. Os critérios avaliados podem ser observados a seguir:

1. O artigo representa uma pesquisa científica?
2. Os objetivos da pesquisa são definidos claramente?
3. Há uma descrição adequada do contexto no qual a pesquisa foi realizada?
4. O *design* da pesquisa foi apropriado para abordar seus objetivos?
5. A estratégia de recrutamento foi adequada aos objetivos da pesquisa?
6. Houve um grupo de controle com o qual comparar os resultados?
7. Os dados foram coletados de maneira a abordar o problema de pesquisa?
8. A análise dos dados foi suficientemente rigorosa?
9. Houve análise das ameaças à validade da pesquisa?
10. Há uma definição clara dos resultados obtidos?
11. O valor do estudo é para pesquisa ou prática?

3.6 Extração dos Dados

A extração dos dados consiste na obtenção de informações gerais dos artigos. Os itens *Tipo de questão de pesquisa*, *Tipo de resultado da pesquisa* e *Tipo de validação da pesquisa* são classificações fornecidas em [85].

1. Título do artigo

2. Autores
3. Ano de publicação
4. Local de publicação
5. Tipo de contexto (ágil ou não)
6. Propósito do artigo
7. Tipo de questão de pesquisa (Tabela 3.1)
8. Tipo de resultado da pesquisa (Tabela 3.2)
9. Tipo de validação da pesquisa (Tabela 3.3)
10. Técnica usada para formação de equipes
11. Atributos usados para formação de equipes

Tabela 3.1: Tipos de questão de pesquisa.

Tipo	**Exemplo**
Método ou meio de desenvolvimento	Como podemos fazer/criar/modificar/evoluir (ou automatizar) X?- Qual é a melhor maneira de fazer/criar/modificar/evoluir o X?
Método para análise ou avaliação	Como posso avaliar a qualidade/corretude de X?- Como faço para escolher entre X e Y?
Design, avaliação ou análise de uma instância particular	Quão bom é Y? O que é a propriedade X do artefato / método Y?- Qual o melhor design, implementação, manutenção ou adaptação para a aplicação X?- Como X se comporta em relação a Y?- Qual é o estado atual de X/prática de Y?
Generalização ou caracterização	Dado X, o que Y (necessariamente) será?- O que exatamente queremos dizer com X? Quais são suas características importantes?- O que é um bom modelo formal/empírico para o X?- Quais são as variedades do X, como elas estão relacionadas?
Estudo de viabilidade ou exploração	X existe mesmo, e se assim for, como é?- É possível obter X de alguma forma?

3.7 Resultados

Na Tabela 3.4, são apresentadas estatísticas relacionadas à formação do *Start Set*. Foram realizadas 10 buscas independentes no Google Acadêmico, as quais retornaram 140 trabalhos, resultando em sete artigos relevantes, após a aplicação dos critérios de seleção (Seção 3.4). Observa-se que os termos *Team Formation*, *Team Composition* e *Team Selection* se destacam na quantidade de artigos retornados, sendo também os únicos termos a culminarem

Tabela 3.2: Tipos de resultado da pesquisa.

Tipo	Descrição
Método ou técnica	Novo ou aperfeiçoado jeito de fazer uma tarefa, tais como design, implementação, manutenção, medição, avaliação, seleção a partir de alternativas; inclui técnicas para implementação, representação, gerenciamento, representação, e análise; a técnica deve ser operacional, não conselhos ou diretrizes, mas sim um procedimento.
Modelo descritivo ou qualitativo	Estrutura ou taxonomia para a área da problemática; estilo arquitetônico, arcabouço ou padrão de projeto; análise informal de domínio, *checklist* bem fundamentado, orientação para a integração de outros resultados, observações interessantes bem organizadas.
Modelo empírico	Modelo preditivo empírico baseado em dados observados.
Modelo analítico	Modelo estrutural que permite análise formal ou manipulação automática.
Ferramenta ou notação	Ferramenta implementada que incorpora uma técnica; linguagem formal para suportar uma técnica ou modelo (deve ter cálculo, semântica ou outra base para computação ou inferência)
Solução específica, protótipo, resposta ou julgamento	Solução que fornece aplicação dos princípios da ES, podendo ser design, protótipo ou implementação completa; análise cuidadosa de um sistema ou seu desenvolvimento, resultado de uma análise específica, avaliação ou comparação.
Relatório	Observações interessantes, regras práticas, mas não suficientemente gerais ou sistemáticas para chegar ao nível de um modelo descritivo.

Tabela 3.3: Tipos de validação da pesquisa.

Tipo	Descrição
Análise	Realização de uma verificação criteriosa, com o auxílio de métricas e/ou métodos específicos.
Avaliação	É baseada em critérios previamente definidos ou realizada por especialistas.
Experiência	Os resultados foram utilizados em exemplos reais por outros além do autor, e forneceram evidências da corretude/utilidade/eficiência.
Exemplo	Baseia-se em exemplos que descrevem como o problema pode ser resolvido no mundo real.
Persuasão	É puramente persuasiva e raramente suficiente para uma pesquisa científica.
Afirmação	Nenhum tipo de validação é apresentada.

Tabela 3.4: Estatísticas do Start Set.

Ferramenta de Busca	Termos de Busca	Art. Retornados	Art. Relevantes
Google Acadêmico	Team Allocation	6	0
Google Acadêmico	Team Formation	55	3
Google Acadêmico	Team Selection	20	2
Google Acadêmico	Team Configuration	1	0
Google Acadêmico	Team Composition	35	2
Google Acadêmico	Human Resource Allocation	16	0
Google Acadêmico	Human Resource Selection	6	0
Google Acadêmico	Human Resource Formation	1	0
Google Acadêmico	Human Resource Configuration	0	0
Google Acadêmico	Human Resource Composition	0	0
	Total	**140**	**7**

em artigos relevantes. O termo *Human Resource Allocation* que aparece em quarto (artigos retornados), embora não tenha apresentado resultado relevante, é bastante utilizado em trabalhos que focam na formação de equipes em domínios como saúde, educação e economia.

Na Tabela 3.5, é possível observar as estatísticas referentes às iterações. 2.404 trabalhos foram submetidos aos critérios gerais de exclusão (fase 1), visando descartar de imediato trabalhos irrelevantes. Posteriormente, 862 trabalhos (referências e citações) foram avaliados utilizando-se os critérios básicos (fase 2), ou seja, tiveram seus títulos e resumos analisados. Por fim, 97 trabalhos foram verificados mediante os critérios avançados, os quais demandam uma leitura completa, resultando assim em 44 artigos relevantes.

Tabela 3.5: Estatísticas das iterações.

Iteração	Referências	Citações	Após a Fase 1	Após a Fase 2	Após a Fase 3
1	189	4	98	17	11
2	370	476	580	30	18
3	473	295	135	48	13
4	444	61	37	2	2
5	50	42	12	0	0
Total	**1526**	**878**	**862**	**97**	**44**

Ao todo, foram analisados 2.544 trabalhos, dos quais 51 artigos foram considerados relevantes, culminando em 40 estudos identificados (Tabelas 3.6 e 3.7). Os artigos *A1 a A7*

são resultantes do *Start Set*, artigos *A8 a A18* da iteração 1, artigos *A19 a A36* da iteração 2, artigos *A37 a A49* da iteração 3 e artigo *A50 e A51* da iteração 4. Na iteração 5 não foram encontrados artigos relevantes, atingindo-se assim o critério de parada do *Snowballing*.

Os artigos encontrados foram caracterizados em estudos, pois alguns trabalhos são resultados de pesquisas segmentadas ao longo dos anos, ou seja, uma solução é publicada, em seguida é melhorada (ou mais detalhada) gradativamente e novas publicações vão surgindo, porém a essência do trabalho continua a mesma. Ao não considerar os estudos, algumas informações podem se tornar enviesadas, como por exemplo a identificação das técnicas utilizadas, uma vez que vários artigos de um mesmo estudo acentuariam o número de vezes que uma técnica foi utilizada. O processo de identificação do estudo consistiu na análise e comparação do conjunto de autores, do propósito da pesquisa, das técnicas utilizadas e da linha de pesquisa adotada. Para representação de um estudo foi considerado o artigo publicado mais recentemente (ano) e, em caso de empate, optou-se por aquele que obteve maior pontuação na avaliação de qualidade. No Apêndice A, é apresentada uma lista com informações acerca do ano, local de publicação, questão de pesquisa, resultado, validação, atributo, técnica e pontuação de qualidade de cada estudo.

Na Figura 3.2, tem-se a quantidade de publicações entre os anos de 2001 e 2018. Neste caso, foi utilizado o número de artigos e não de estudos, pois ao se considerar apenas os estudos, artigos de pesquisas que haviam sido iniciadas em anos anteriores não apareceriam, dando uma falsa impressão de menor interesse dos pesquisadores acerca do tema em questão.

Figura 3.2: Número de artigos X anos.

Fonte: próprio autor.

Tabela 3.6: Conjunto final de artigos encontrados - parte 1.

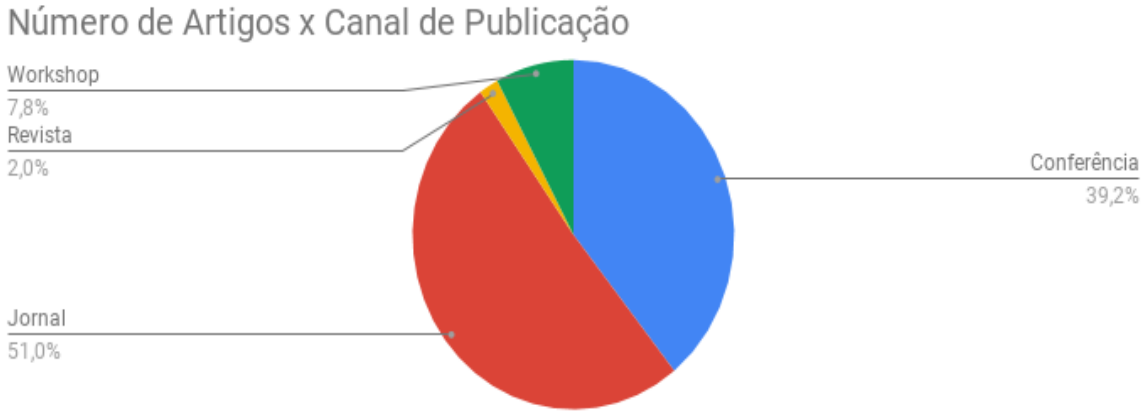
Artigo	Estudo	Título
A01	E01	Team Member Selection In Agile [51]
A02	E02	Resolving team selection in agile development using NSGA-II algorithm [8]
A03	E03	Skill-based Team Formation in Software Ecosystems [82]
A04	E04	Applying Fuzzy Technique In Software Team Formation Based On Belbin Team Role [71]
A05	E04	Towards a balanced software team formation based on Belbin team role using fuzzy technique [72]
A06	E05	A rule-based model for software development team composition: Team leader role with personality types and gender classification [40]
A07	E05	Balancing The Personality Of Programmer: Software Development Team Composition [39]
A08	E06	A hybrid approach to solve the agile team allocation problem [16]
A09	E07	Capacitated team formation problem on social networks [59]
A10	E08	The use of search-based optimization techniques to schedule and staff software projects: An approach and an empirical study [25]
A11	E09	Novel approach to multi-functional project team formation [99]
A12	E10	Who should work with whom? Building effective software project teams [42]
A13	E05	A rule-based approach for discovering effective software team composition [79]
A14	E05	Discovering personality types and diversity based on software team roles [41]
A15	E11	Experiences in software engineering courses using psychometrics with RAMSET [63]
A16	E12	Software developer selection: A holistic approach for an eclectic decision [67]
A17	E05	A Set of Rules for Constructing Gender-based Personality types' Composition for Software Programmer [38]
A18	E13	Exploitation of social semantic technology for software development team configuration [76]
A19	E14	Staffing a software project: A constraint satisfaction and optimization-based approach [13]
A20	E14	Staffing a software project: A constraint satisfaction approach [12]
A21	E15	Human resource selection for software development projects using Taguchi‘s parameter design [98]
A22	E16	Forming Grouped Teams with Efficient Collaboration in Social Networks [49]
A23	E17	A multi-objective genetic algorithm for intelligent software project scheduling and team staffing [94]
A24	E18	A novel prototype tool for intelligent software project scheduling and staffing enhanced with personality factors [95]
A25	E19	Cooperative co-evolutionary optimization of software project staff assignments and job scheduling [80]

Tabela 3.7: Conjunto final de artigos encontrados - parte 2.

Artigo	Estudo	Título
A26	E20	A fuzzy-genetic decision support system for project team formation [92]
A27	E21	A quantitative assessment on team building criteria for software project teams [32]
A28	E21	An empirical study on the use of team building criteria in software projects [22]
A29	E22	Formal model for assigning human resources to teams in software projects [6]
A30	E23	Forming successful extreme programming teams [43]
A31	E05	Making programmer suitable for team-leader: Software team composition based on personality types [36]
A32	E24	Managing Software Engineering Student Teams Using Pellerin's 4-D System [28]
A33	E25	Personalities And Software Development Team Performance, A Psycholinguistic Study [31]
A34	E26	Supporting agile team composition: A prototype tool for identifying personality (In) compatibilities [57]
A35	E11	Using MatLab's fuzzy logic toolbox to create an application for RAMSET in software engineering courses [62]
A36	E27	Resyster: A hybrid recommender system for scrum team roles based on fuzzy and rough sets [20]
A37	E18	A Multi-objective Genetic Algorithm for Software Development Team Staffing Based on Personality Types [93]
A38	E28	A multi-criteria approach for team recommendation [7]
A39	E29	A multi-objective genetic algorithm for software personnel staffing for HCIM solutions [52]
A40	E30	Decision model for allocating human resources in information system projects [30]
A41	E31	Minimal cost stable workforce allocation in presence of ties [35]
A42	E32	Scatter search for trainees to software project requirements stable allocation [34]
A43	E33	Skill-based framework for optimal software project selection and resource allocation [110]
A44	E34	Staffing open collaborative projects based on the degree of acquaintance [3]
A45	E35	A multi-criteria decision making approach for resource allocation in software engineering [73]
A46	E36	An ontology-based approach with which to assign human resources to software projects [74]
A47	E37	Binary fuzzy goal programming for effective utilization of IT professionals [50]
A48	E38	Measuring social networks when forming information system project teams [56]
A49	E05	Finding an effective classification technique to develop a software team composition model [37]
A50	E39	Value-based multiple software projects scheduling with genetic algorithm [108]
A51	E40	Recommending people in developers' collaboration network [97]

Na Figura 3.3, é apresentada a distribuição dos artigos de acordo com os canais de publicação. Novamente, optou-se pelo número de artigos, objetivando demonstrar uma visão realística dos principais canais alvos de publicação.

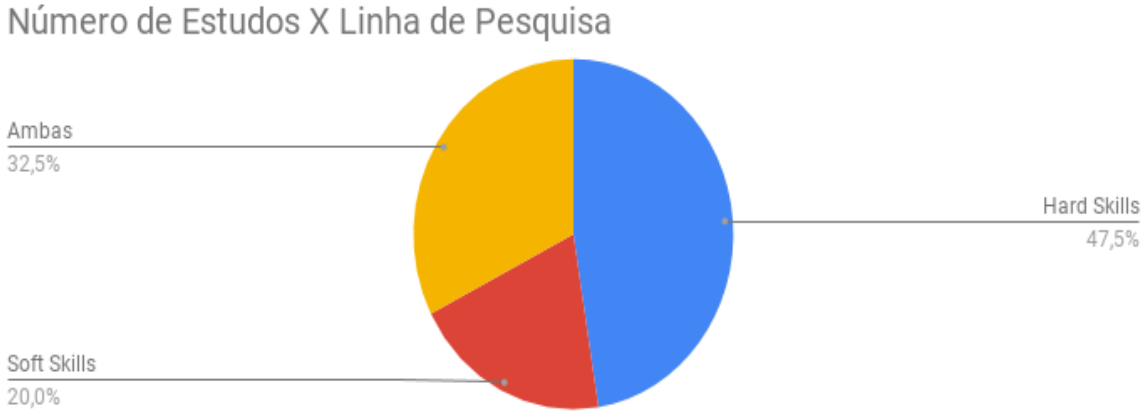
Figura 3.3: Número de artigos X canal de publicação.



Fonte: próprio autor.

Na Figura 3.4 são apresentados os resultados em relação à linha de pesquisa adotada pelos estudos, podendo ser com foco em *hard skills*, *soft skills* ou ambas.

Figura 3.4: Número de estudos X linha de pesquisa.



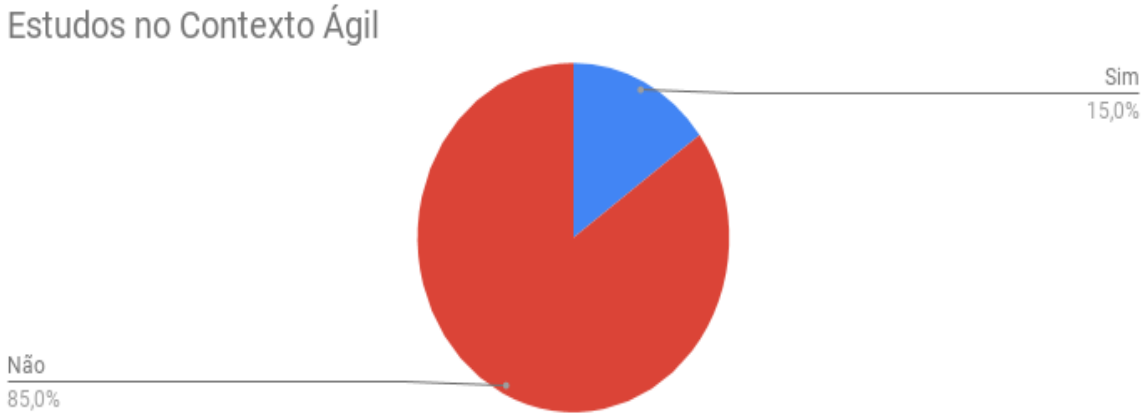
Fonte: próprio autor.

Na Figura 3.5, é apresentada a divisão dos estudos no que diz respeito ao tipo de contexto em que a solução foi proposta ou validada, podendo pertencer ou não ao contexto ágil.

Na Figuras 3.6, 3.7 e 3.8, têm-se, respectivamente, os resultados em relação aos tipos de *Questão de Pesquisa*, *Resultado da Pesquisa* e *Validação da Pesquisa*, os quais foram

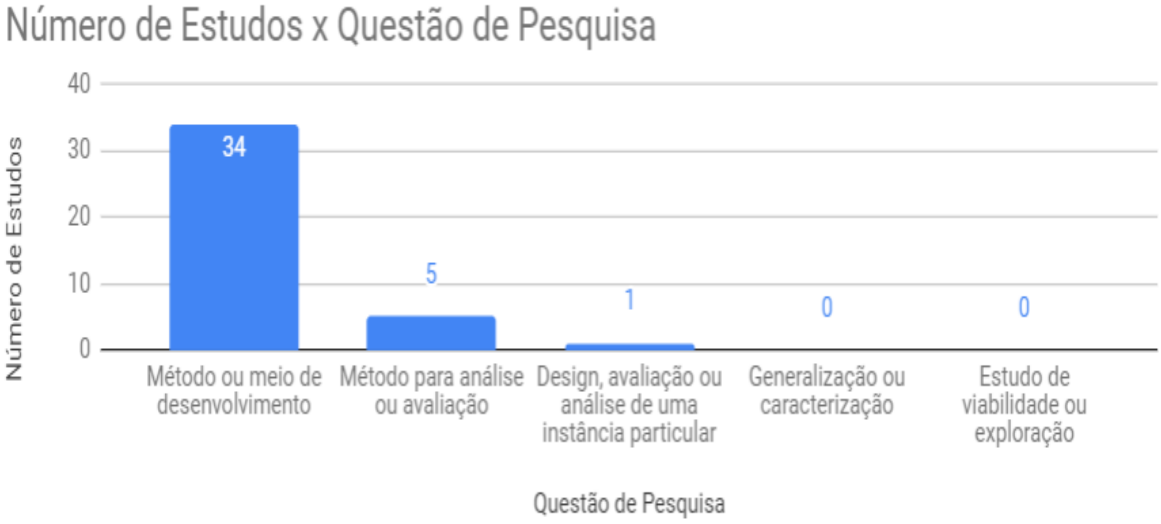
definidos de acordo com Shaw [85].

Figura 3.5: Número de estudos X contexto ágil.



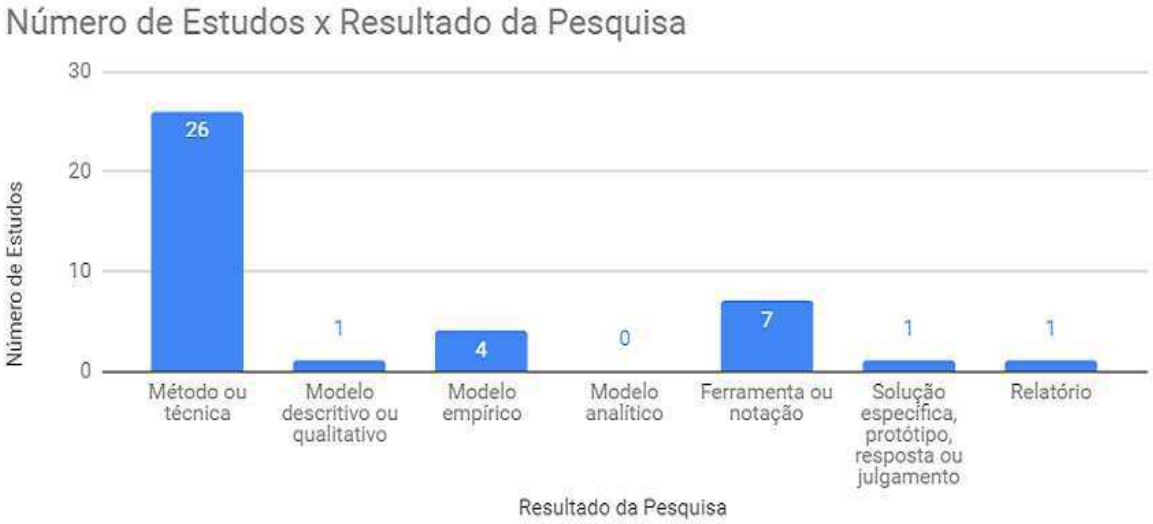
Fonte: próprio autor.

Figura 3.6: Número de estudos X questão de pesquisa.



Fonte: próprio autor.

Figura 3.7: Número de estudos X resultado da pesquisa.



Fonte: próprio autor.

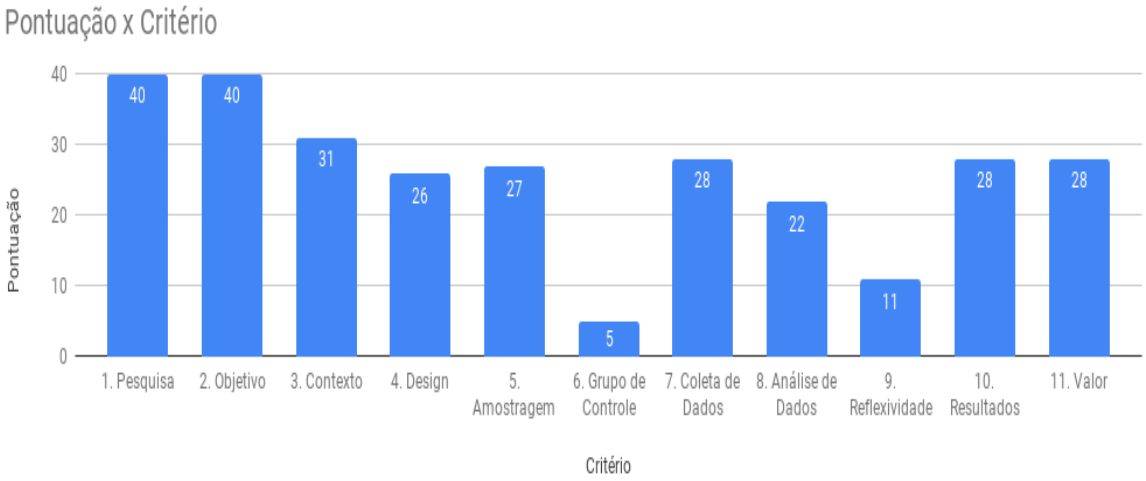
Figura 3.8: Número de estudos X validação da pesquisa.

Fonte: próprio autor.

Na Figura 3.9, tem-se a quantidade de estudos de acordo com os critérios de qualidade estabelecidos em [29].

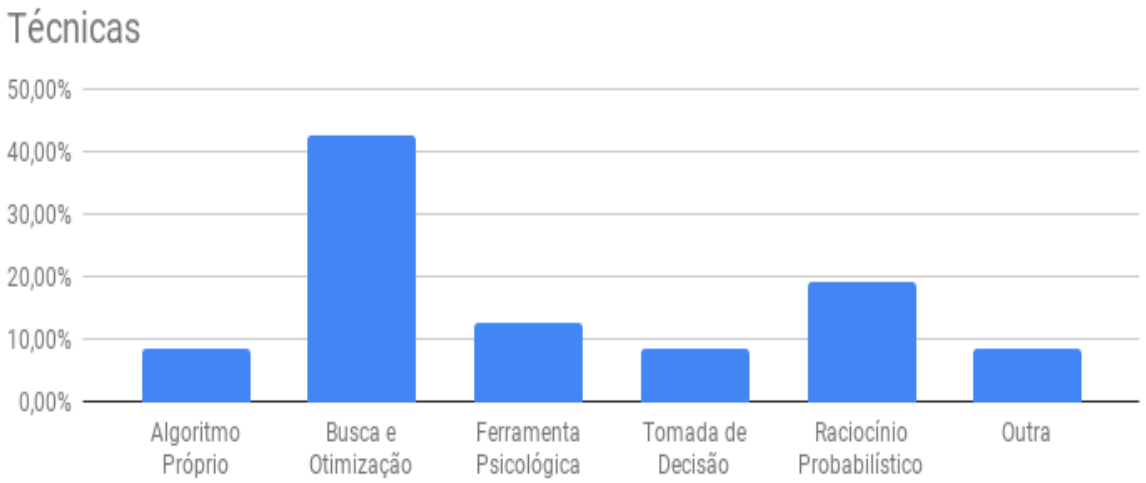
Na Figura 3.10, são apresentadas as categorias das técnicas presentes nos estudos avaliados.

Figura 3.9: Número de estudos X pontuação de qualidade.



Fonte: próprio autor.

Figura 3.10: Técnicas utilizadas nos estudos.



Fonte: próprio autor.

3.8 Discussão

De acordo com os resultados obtidos (Figura 3.2), percebe-se que houve aumento no número de artigos publicados ao longo dos anos, porém de forma não linear. Nota-se que ocorrem alternâncias entre crescimentos e decaimentos, atingindo o pico de publicações em 2016. Ao analisar as publicações da última década (2008-2018), verifica-se um aumento significativo no número total de artigos em relação aos anos anteriores, o que indica ampliação de interesse acerca do tema em questão. Por outro lado, observa-se um decaimento acentuado nos dois últimos anos, mas que se justifica devido ao presente mapeamento ter sido iniciado em meados de 2017 e finalizado em meados de 2018, portanto, artigos publicados nesse período possuíam poucas ou nenhuma citação, impossibilitando o retorno de prováveis artigos relevantes.

Dentre os canais de publicação levantados (Figura 3.3), *Periódico* é o principal alvo dos artigos (51%), uma vez que é o meio que, geralmente, proporciona maior visibilidade e credibilidade. Entretanto, considerando todos os canais, notou-se que a distribuição dos locais de publicação é heterogênea, pois não há repetição de locais, com exceção do *European Journal of Operational Research* e do *International Journal of Project Management*, os quais foram alvos de publicação duas vezes cada um.

Ao analisar os gráficos das Figuras 3.6 e 3.7, verifica-se que a maioria dos estudos busca responder questões sobre *Método ou meio de desenvolvimento* (85%), tendo como resultado *Método ou técnica* (65%). O processo de formação de equipe está diretamente relacionado ao desenvolvimento de software, uma vez que os desenvolvedores constituem recurso fundamental para tal processo, pois são um dos principais responsáveis pelo resultado final, justificando assim o foco dos estudos.

O gráfico da Figura 3.6 indica os principais tipos de validação, com destaque para *Análise* (47,5%) e *Avaliação* (32,5%). Em geral, estudos que utilizam *Análise* como forma de validar o resultado da pesquisa, baseiam-se em dados adquiridos a partir de repositórios específicos. Por exemplo, alguns estudos extraíram dados de repositórios públicos para montar suas respectivas bases de projetos de software, as quais tiveram parte de seus dados utilizados para treinamento e a outra parte para teste das soluções propostas. Na *Análise* não há participação de especialistas, uma vez que a validação é feita com auxílio de métricas, como

Precisão (do inglês, *Precision*) e Revocação (do inglês, *Recall*) [24], e/ou métodos específicos, como testes estatísticos. No caso da *Avaliação*, a validação conta com a presença de especialistas, geralmente, gestores familiarizados com os projetos de software, que avaliam as equipes sugeridas, de acordo com determinados critérios.

Apesar do crescimento da adoção dos métodos ágeis nos últimos anos, nota-se que a minoria (15%) dos estudos ($E01$, $E02$, $E06$, $E23$, $E26$ e E27) propôs soluções para formação de equipes ágeis ou validou nesse contexto (Figura 3.5). Dentre eles, apenas dois especificaram para qual método ágil a solução é direcionada, sendo um deles *eXtreme Programming* ($E23$) e o outro Scrum ($E27$).

O gráfico da Figura 3.9 fornece a pontuação geral dos estudos em relação aos critérios de qualidade (Seção 3.5). Observa-se que dois pontos são pouco explorados pelos estudos: *Grupo de Controle* e *Reflexividade*. Nos estudos que utilizaram grupos de controle, foram conduzidos experimentos nos quais equipes eram formadas com métodos tradicionais das empresas e com as soluções propostas. Nestes casos, o principal objetivo é verificar quais equipes demonstram ser mais eficientes ao longo de um determinado período de tempo e assim avaliar a solução proposta. Uma das limitações de tal prática consiste na baixa receptividade das empresas em relação a adoção/teste de soluções experimentais em ambiente real, uma vez que isto impacta diretamente no sucesso ou fracasso de um projeto, cujos resultados são inspecionados frequentemente por seus clientes. Em relação à *Reflexividade*, percebe-se que a maioria dos trabalhos não faz autocrítica, ou seja, não analisa as ameaças à validade da pesquisa. Nos demais critérios de qualidade, os estudos apresentam uma média de 32,00 pontos, representado mais de 80% da pontuação máxima. Entretanto, alguns estudos obtiveram pontuações significativamente baixas, com apenas dois pontos ($E01$ e $E35$) em *Pesquisa* e *Objetivos*, e três pontos ($E02$ e $E04$) em *Pesquisa*, *Objetivos* e *Design*.

No que diz respeito à questão de pesquisa $QP01$ (Figura 3.4), a maioria dos estudos (47,5%) ($E01$, $E02$, $E08$, $E09$, $E12$, $E15$, $E17$, $E19$, $E20$, $E28$, $E31$, $E32$, $E33$, $E34$, $E35$, $E36$, $E37$, $E39$ e $E40$) utiliza somente *hard skills* como base para a formação de equipes. Observa-se que os atributos utilizados são predominantemente técnicos, tais como experiências em programação, design, teste, modelagem, banco de dados, redes, dentre outros. Um ponto comum à maioria destes estudos é o alto nível de abstração utilizado para especificação dos atributos. Por exemplo, ao quantificar a experiência em banco de dados,

as soluções propostas não especificam em quais bancos de dados um determinado indivíduo tem experiência. Desta forma, não é possível saber exatamente qual a extensão da experiência do candidato. Alguns estudos apresentam menor granularidade, mas, em geral, se restringem à especificar a linguagem de programação. Isto ocorre devido ao custo de se atingir maior granularidade, pois muitas vezes a coleta dos dados é realizada mediante aplicação de questionários ou a partir da visão dos gestores. No primeiro caso, a aplicação de longos questionários poderia resultar em um número reduzido de respostas com baixa confiabilidade. No segundo caso, tem-se a subjetividade da visão do gestor associada ao baixo nível de detalhes das informações, uma vez que dificilmente um gestor consegue detalhar o que todos os membros de suas equipes fizeram.

Em relação aos estudos que utilizam apenas *soft skills* (20%) ($E04$, $E05$, $E07$, $E10$, $E11$, $E24$, $E25$ e $E26$), basicamente dois tipos de atributos são usados: personalidade e interação social. A personalidade está relacionada a características como introversão, extroversão, sensitividade, intuição, julgamento, percepção, pensamento, sentimento, agregação, coordenação, articulação, dentre outras. Estudos que utilizam estes atributos defendem que a incompatibilidade entre membros pode comprometer a efetividade da equipe. Da mesma forma, uma equipe formada por indivíduos com alto grau de compatibilidade podem funcionar muito bem como um todo. Portanto, em alguns trabalhos os esforços são direcionados para determinar a combinação ideal dos tipos de personalidades na equipe, objetivando-se potencializar os benefícios em termos de produtividade, custos e satisfação das partes interessadas. Estudos que utilizam aspectos sociais para formar equipes exploram a ligação entre os indivíduos a partir de relações colaborativas. Por exemplo, o *Github* é uma plataforma de hospedagem de código que possibilita o controle de versão de projetos de software. Esta plataforma armazena um grande número de projetos e provê recursos como troca de mensagens, fluxos de atividades e relações de amizade, que são características encontradas em redes sociais. Diante disto, é possível estabelecer conexões sociais que auxiliam no processo de formação de equipes. Por exemplo, no estudo $E07$ foram utilizados *logs* de milhares de projetos hospedados no *Github* e calculou-se a correlação entre a natureza social da equipe e o número de *commits* do projeto. Como resultado, foram encontradas evidências de que equipes com mais conexões na rede são mais ativas (em termos de número de *commits*).

No caso dos estudos que utilizam tanto *hard skills* quanto *soft skills* (32,5%) ($E03$, $E06$,

$E13$, $E14$, $E16$, $E18$, $E21$, $E22$, $E23$, $E27$, $E29$, $E30$ e $E38$), observou-se que além dos atributos previamente descritos, tem-se a utilização de características como liderança, proatividade, disciplina, criatividade, motivação e dedicação, as quais geralmente não são mensuradas por ferramentas específicas. Em geral, os gestores das equipes são os responsáveis pela mensuração destes atributos, o que acarreta em um alto grau de subjetividade, uma vez que esta percepção pode variar de gestor para gestor.

Em relação à questão de pesquisa $QP2$ (Figura 3.10), realizou-se um levantamento das principais técnicas utilizadas nos estudos, classificando-as em seis categorias. Vale salientar que alguns trabalhos utilizam mais de uma técnica em suas soluções. 42,55% das técnicas pertencem à categoria *Busca e Otimização* e provêm dos estudos $E02$, $E03$, $E05$, $E06$, $E08$, $E14$, $E16$, $E17$, $E18$, $E19$, $E20$, $E28$, $E29$, $E30$, $E32$, $E33$, $E35$, $E39$ e $E40$. Esta categoria se destaca devido à maioria dos estudos abordarem a formação de equipes como um problema de otimização. Dentre as principais técnicas estão Algoritmos Genéticos (e suas derivações), Programação Dinâmica e Algoritmo de *Backtracking*. Particularmente, AG representam o conjunto de técnicas mais adotado, estando presente em 25% de todos os estudos. Uma das razões para tal é a capacidade dos AG de encontrarem soluções ótimas ou aproximadamente ótimas em grandes espaços de busca, com eficiência e rapidez. *Raciocínio Probabilístico* é a segunda categoria mais recorrente (19,15%) e reúne técnicas provenientes dos estudos $E04$, $E06$, $E09$, $E12$, $E20$, $E27$ e $E37$, que se baseiam em probabilidade subjetiva, a qual é gerada a partir de crenças ou opiniões expressas como probabilidade. Lógica *Fuzzy* [55] e Redes *Bayesianas* [33] se destacam nesta categoria, sendo aplicadas para tratar a incerteza associada à geração dos perfis por parte dos gestores. Na categoria *Ferramenta Psicológica* (12,77%) destacam-se os métodos *Belbin Team Role* e *Myers-Briggs Type Indicator* como ferramentas que auxiliam diretamente na formação das equipes nos estudos $E11$, $E22$, $E24$, $E25$ e $E26$. *Jung theory of personality* também foi utilizada porém apenas em um estudo. Na categoria *Tomada de Decisão* (8,51%) tem-se as técnicas *Grey decision theory*, *Analytical Hierarchy Process* e Ontologia originadas dos estudos $E09$, $E12$, $E13$ e $E36$. Em geral, estudos que utilizam estas técnicas não fornecem equipes automaticamente, mas sim diretrizes que auxiliam na tomada de decisão para escolha dos membros da equipe. Na categoria *Algoritmo Próprio* (8,51%), encontram-se técnicas baseadas em equações e/ou modelos matemáticos propostos pelos autores dos estudos $E01$, $E07$, $E31$ e $E34$. Na catego-

ria *Outra* (8,51%), estão técnicas que não se enquadram nas categorias previamente listadas tais como, questionários simples, *K-means* e *Taguchi's parameter design*, provenientes dos estudos $E10$, $E15$, $E21$, $E23$ e $E38$.

3.9 Trabalhos Relacionados

Dentre os 40 estudos encontrados, foram selecionados os que tratam explicitamente do problema de Formação de Múltiplas Equipes, no contexto de Engenharia de Software, o qual consiste em alocar múltiplos desenvolvedores para múltiplos projetos, com o intuito de formar equipes de acordo com as competências e demandas apresentadas.

No estudo $E09$ é apresentada uma abordagem que utiliza agrupamento *Fuzzy* para reunir atributos que compõem o perfil de um projeto. Tal perfil é representando por uma matriz m x n, na qual m são os requisitos do projeto (facilidade de instalação, perspectiva visual, efeito especial, conexão de rede, compatibilidade, conteúdos diferenciais, efeito sonoro e upgrade limitado) e n são as características dos desenvolvedores (técnica de rede, design de software, design de banco de dados, design orientado à visualização, controle de qualidade, design multimídia, *driver* de hardware, design de efeito sonoro, análise de sistema, design de sistema, testes de hardware, embalagem do produto). Para cada elemento da matriz, um grupo de gerentes de projetos denota um valor que representa o grau de associação de um requisito a uma determinada característica. Em seguida, os elementos são ordenados e agrupados utilizando uma técnica chamada *Fuzzy rank order clustering*. O objetivo deste agrupamento é dividir o projeto em módulos (subprojetos) de modo que equipes sejam formadas de acordo com as características de cada grupo. Por fim, os gerentes avaliam os indivíduos em relação às caraterísticas dos desenvolvedores e utilizam uma técnica chamada *Grey theory* para auxiliar na definição das equipes. A solução proposta foi validada mediante apresentação de um exemplo conceitual de como seria seu funcionamento no mundo real.

No estudo $E19$ foi proposto um método para auxiliar no planejamento de projetos, visando diminuir o tempo necessário para finalizá-los, de acordo com duas características específicas: a alocação de indivíduos para equipes e a alocação de equipes para pacotes de trabalho. Pacotes de trabalhos são conjuntos de tarefas agrupadas conforme a necessidade de um projeto. A solução proposta baseia-se em algoritmo de cooperação co-evolutivo, o qual

é originalmente destinado para decompor problemas que envolvem grandes espaços dimensionais em subproblemas menores, os quais podem ser executados por algoritmos evolutivos convencionais. Neste caso, o algoritmo trabalha com dois tipos de solução durante o processo evolutivo. Uma representa a ordem como os pacotes de trabalhos são distribuídos entre as equipes e a outra representa a formação das equipes. Cada pacote de trabalho é composto por dois atributos: esforço estimado em pessoas/dias para completá-lo e a sua dependência com outros pacotes. Estes atributos são utilizados para buscar soluções em que os pacotes possam ser distribuídos entre as equipes de modo que possam ser finalizados o mais rápido possível e sem a necessidade de que uma equipe fique ociosa aguardando a finalização de um pacote dependente. É importante destacar que a solução proposta considera que todos os indivíduos são idênticos, isto é, possuem as mesmas habilidades e conhecimentos. A escolha da solução final é feita com base na combinação dos valores de *fitness* dos dois tipos de solução previamente descritos. A validação do trabalho utilizou dados de quatro projetos reais de software e consistiu na comparação da abordagem proposta com AG e *Random Search*, utilizando como parâmetro o tempo de duração do projetos de acordo com as soluções apresentadas por cada algoritmo. Como resultado, a solução proposta conseguiu superar as demais, encontrando a melhor solução com o menor número de gerações.

No estudo $E20$, propõe-se um sistema de suporte à decisão, o qual emprega Lógica *Fuzzy* e AG. Os perfis dos indivíduos são gerados a partir de atributos técnicos advindos de bases históricas e podem ser, por exemplo, tempo de experiência com uma linguagem de programação, número de projeto em que o desenvolvedor trabalhou, prioridade (baixa, média ou alta) e status de conclusão (antes do prazo, no prazo, ou depois do prazo) destes projetos, etc. Lógica *Fuzzy* é usada para transformar os valores numéricos dos perfis, enquanto que AG é utilizado para otimizar a formação das equipes de acordo com as especificações dos projetos. A solução proposta foi validada com alunos do quarto ano de um curso de tecnologia da informação. Foram formadas equipes de projetos que combinam conhecimentos em C++, Java, algoritmos e computação gráfica. Após um determinado período de tempo, os resultados dos projetos foram comparados com os resultados da turma do ano anterior, no qual os próprios alunos formavam as equipes com base em afinidade e colaboração em projetos de outras disciplinas. Como resultado, foi relatado que houve aumento na nota dos estudantes, diminuição da duração dos projetos e maior qualidade nas entregas dos componentes

dos projetos. Entretanto, não foi apresentado nenhum resultado numérico para comprovar as conclusões.

No estudo $E27$ é proposto um sistema para dar suporte aos gerentes no processo de formação de equipes para projetos *Scrum*. Os gerentes são responsáveis pela quantificação das competências dos desenvolvedores, bem como das demandas dos projetos. Para tanto, eles utilizam uma documentação para auxiliar na quantificação dos atributos que são de natureza comportamental (liderança, motivação, etc), técnica (experiência em resolução de problemas, trabalho em equipe, etc), interpessoal (idade, gênero, nível educacional, etc) e profissional (tempo na empresa, nível de experiência, etc). A solução proposta utiliza Lógica *Fuzzy* e *Rough Set* para diminuir a incerteza inerente aos perfis criados pelos gestores. A solução proposta foi validada com auxílio de quatro gestores, com mais de dois anos de experiência em gestão de projetos Scrum. A validação ocorreu em duas etapas. Na primeira, cada gestor tinha que formar duas equipes, com quatro integrantes cada, para um projeto específico. A primeira equipe tinha que ser formada obedecendo a condição de que ela fosse a melhor, independente da demanda do projeto. A segunda equipe tinha que ser formada obedecendo a condição de que ela fosse a melhor para a demanda do projeto. Na segunda etapa, cada gestor precisava formar duas equipes obedecendo as mesmas condições anteriores, com três integrantes cada, para vários projetos. Em seguida, os gestores tinham que se reunir e chegar a um consenso em relação à solução final. Como resultado a abordagem obteve, no geral, 51,7% Precisão e 51,7% de Revocação.

No estudo $E29$ é proposta uma abordagem híbrida que combina atributos técnicos (fundamentos de programação, organização e arquitetura de computadores, sistemas operacionais, inteligência artificial, design de software, qualidade de software, sistemas distribuídos, gerência da informação, dentre outros) e psicológicos (criatividade, tenacidade, disciplina, empatia, liderança de grupo, planejamento, independência, tomada de decisão, dentre outros) para formar equipes. A abordagem utiliza AG para formar equipes de duas maneiras diferentes: na primeira, o objetivo é sugerir equipes balanceadas, cujos membros possuam os maiores valores possíveis para atributos previamente selecionados por um usuário, porém sem ponderação em relação aos pesos dos mesmos; na segunda, almeja-se formar equipes adaptadas para atender a características específicas, atribuindo-se valores ponderados para os atributos. A validação da solução proposta ($E29$) foi realizada em duas etapas, contando com

a participação de 56 estudantes do último ano de um curso Ciência da Computação e quatro especialistas. Os atributos (técnicos e psicológicos) de cada estudante foram quantificados utilizando-se uma ferramenta de avaliação de desempenho conhecida como *Feedback* 360º, na qual um indivíduo é avaliado por si mesmo e por outros a sua volta, sejam eles superiores, subordinados ou clientes. Neste caso, cada estudante avaliou, com uma nota de 0 a 10, cada um de seus próprios atributos, bem como os de seus colegas. Na primeira etapa da validação, o objetivo consistiu em formar apenas uma equipe com oito integrantes, utilizando as duas maneiras propostas. De forma análoga ocorreu a segunda etapa, porém foram formadas sete equipes com oito membros cada, ou seja, todos os estudantes foram alocados simultaneamente. Em ambas as etapas, os especialistas avaliaram os perfis dos estudantes, formaram suas respectivas equipes e em seguida confrontaram suas soluções entre si, a fim de obter um consenso acerca da solução final, com o intuito de compará-la com a formação fornecida pela abordagem. Como resultado, a abordagem obteve 85,4% de Precisão na primeira etapa da validação e 77,9% na segunda.

No estudo $E30$ é proposto um modelo de decisão baseado em programação dinâmica para formar equipes que possam minimizar a duração de projetos de desenvolvimento de software. O esforço necessário para completar cada projeto é calculado com auxílio da técnica *COCOMO II* [26]. Em seguida, para cada projeto é montado um *ranking* com os indivíduos mais similares, de acordo com fatores intrapessoais (análise, tomada de decisão, independência, inovação, pensamento crítico, persistência e tolerância ao estresse), organizacionais (auto-organização, gerenciamento de risco, conhecimento de domínio, disciplina e orientação), interpessoais (serviço ao cliente, técnicas de negociação, empatia, técnicas sociais e trabalho em equipe) e gerenciamento (suporte ao trabalho, liderança, planejamento e organização). Por fim, o modelo determina o tempo para completar os projetos de acordo com um determinado número de desenvolvedores e fornece a configuração final das equipes para cada projeto. A validação da solução foi realizada com dados simulados de três projetos e 10 desenvolvedores, porém o resultado final não foi submetido a nenhum tipo de avaliação.

No estudo $E37$, a solução proposta visa formar equipes com base em dois objetivos: minimizar custos e maximizar esforços. O custo de um projeto é composto pelo tempo necessário para completá-lo dentro de um orçamento estipulado. O esforço de um indivíduo é representando por um valor de eficiência (0 a 10) que é calculado de acordo com o valor

da hora de trabalho e sua capacidade de realizar determinadas tarefas. Neste estudo, utiliza-se Lógica *Fuzzy* para os valores de custos e esforços. Em seguida, utiliza-se um software específico que tem como base o algoritmo *Branch and bound* para encontrar a configuração final das equipes. A validação da solução foi realizada com dados reais de uma empresa com 28 profissionais de TI e quatro projetos de software. Os resultados obtidos, em relação a custo e esforço, foram comparados com uma abordagem de outro trabalho, indicando que a solução proposta foi melhor.

No estudo $E39$ é apresentada uma abordagem que utiliza AG para formar equipes com base em objetivos e restrições previamente estabelecidos para cada projeto. O objetivo é representado por conjunto de atividades, no qual cada elemento possui uma identificação, um tamanho (linhas de código) e uma ou mais habilidades necessárias, como por exemplo habilidade de codificação em Java por pelo menos três anos. A restrição é formada pelo tempo de duração e orçamento do projeto. O perfil de um desenvolvedor é representado por seu conjunto de habilidades, salário e número de linhas de código que ele consegue codificar por hora. A validação da solução foi realizada com dados simulados de três projetos e 22 desenvolvedores, visando formar equipes capazes de maximizar os objetivos dos projetos, considerando as restrições de cada um.

Ao analisar os trabalhos relacionados, observam-se limitações comuns no que diz respeito à construção dos perfis. Percebe-se que as soluções propostas utilizam atributos abstratos, ou seja, características de alto nível para indicar as competências dos indivíduos durante o processo de formação de equipes, o que dificulta o atendimento de demandas específicas. Outra limitação identificada, corresponde à subjetividade das informações utilizadas nos perfis. Verificou-se que é comum a prática da autoavaliação ou avaliação por terceiros, o que introduz maior viés às informações resultantes.

3.10 Considerações Finais do Capítulo

O presente mapeamento foi conduzido durante um ano, com auxílio de seis pesquisadores (doutorandos) da área de ES, os quais são integrantes do grupo ISE da UFCG. A análise de 2.544 trabalhos resultou em 51 artigos, os quais foram classificados em 40 estudos que propõem soluções para formação de equipes no contexto de desenvolvimento de software.

Constatou-se que o interesse sobre o tema em questão vem aumentando acentuadamente nos últimos anos, com trabalhos que focam principalmente no desenvolvimento de métodos ou técnicas para formação de equipes. Apesar da crescente adoção dos métodos ágeis ao longo dos anos, apenas uma minoria dos estudos apresenta soluções direcionadas para tal contexto, ou seja, soluções para formação de equipes ágeis ainda são pouco exploradas na área. No geral, a qualidade dos trabalhos é considerada satisfatória, ficando abaixo do esperado somente em dois critérios.

Verificou-se que há uma tendência acentuada na utilização de atributos técnicos para formação das equipes, porém o nível de granularidade abordado na maioria dos trabalhos não permite mensurar adequadamente a extensão das competências dos indivíduos. Além disto, percebeu-se que em muitos trabalhos há um significativo grau de subjetividade na geração dos perfis dos indivíduos. Muitas vezes os próprios desenvolvedores e/ou os gestores são responsáveis pela criação dos dados utilizados como entrada para a solução proposta.

Por fim, descobriu-se que a formação de equipes é tratada pela maioria dos trabalhos como um problema de busca e otimização, pois é capaz de fornecer soluções ótimas (ou aproximadas), mesmo diante de amplos espaços de soluções [5]. Além disso, AG destaca-se por sua capacidade de personalização, uma vez que é possível utilizar diversas funções de *fitness* para alcançar objetivos e restrições distintos.

A abordagem proposta (Capítulo 4) difere dos trabalhos encontrados na literatura em dois pontos principais: (i) os perfis dos desenvolvedores são construídos a partir de atributos técnicos de baixa granularidade, coletados durante o desenvolvimento de software, o que reflete mais fielmente as competências técnicas adquiridas; e (ii) a quantificação dos atributos é realizada de forma sistemática, a medida que os desenvolvedores vão implementando novas tarefas, o que torna o processo menos suscetível ao viés inerente à subjetividade da autoavaliação ou avaliação por terceiros.

Capítulo 4

Abordagem Proposta

Neste trabalho, considera-se o cenário no qual uma empresa de software possui dezenas ou centenas de desenvolvedores distribuídos em diferentes equipes multidisciplinares de projetos ágeis, os quais possuem demandas distintas. Considera-se ainda a dinamicidade da entrada e saída de desenvolvedores nas equipes, a finalização de projetos e o início de novos projetos, o que gera a necessidade de realocação interna dos recursos humanos da empresa, resultando assim na alteração do formato das equipes de diversos projetos, inclusive daqueles em execução.

Diante do exposto, a solução proposta neste capítulo consiste em uma abordagem de apoio à decisão para formação de múltiplas equipes para projetos ágeis, a partir da realocação dos recursos humanos disponíveis na empresa, visando obter a melhor distribuição global, considerando as demandas de cada projeto e as competências dos desenvolvedores.

Apesar de utilizar conceitos inerentes a diversos métodos ágeis, pode-se considerar que a abordagem proposta está especializada para a aplicação com o método Scrum. Scrum foi escolhido como método base por ser o método ágil mais utilizado na indústria [64].

A descrição da abordagem se inicia com a apresentação do modelo de especificação de tarefas. Em seguida, apresenta-se a abordagem de apoio à decisão, que tem como base a utilização de Algoritmo Genético para formação de múltiplas equipes.

4.1 Modelo de Especificação de Tarefas Técnicas

O modelo de especificação de tarefas foi desenvolvido para ser aplicado em conjunto com o Scrum, a fim de viabilizar a inserção e modificação de elementos de baixa granularidade, permitindo a criação de perfis técnicos que reflitam os conhecimentos e habilidades adquiridos durante o processo de desenvolvimento de software. Além disto, a aplicação do modelo visa possibilitar a execução de algoritmos para realizar a operacionalização abordagem de apoio à decisão. O modelo proposto compreende a Rotulação e Padronização de tarefas técnicas.

4.1.1 Rotulação de Tarefas

Tags são palavras-chave que auxiliam na classificação de informações, de maneira mais detalhada. Em seu formato original, o Scrum não prevê a atribuição de *tags* às tarefas técnicas, o que dificulta a delimitação dos conhecimentos empregados pelo desenvolvedor durante a implementação das mesmas. Diante disto, definiu-se o processo Rotulação de Tarefas que consiste em atribuir *tags* às tarefas técnicas, permitindo a contextualização destes artefatos de acordo com as tecnologias necessárias às suas implementações.

Conceitualmente, tecnologia envolve um conjunto de instrumentos, métodos e técnicas que visam a resolução de problemas. Neste documento, o termo tecnologia refere-se às linguagens de programação, arcabouços, plataformas, API, bancos de dados etc. Resumidamente, é toda tecnologia que está relacionada à construção de um produto, de um processo ou ao serviço de software.

O processo de Rotulação de Tarefas é realizado na etapa de *Sprint Planning*, na qual, tradicionalmente, as US são decompostas em tarefas técnicas. Estas tarefas representam os requisitos de mais baixo nível do Scrum, alocados para implementação individual, não devendo se estender por mais do que um dia de trabalho. Nesta etapa, propõe-se uma adaptação no Scrum de modo que as tarefas definidas sejam rotuladas com *tags*, isto é, a medida que as US são decompostas, a equipe de desenvolvimento atribui *tags* para cada tarefa resultante, de acordo com os tipos de tecnologias necessárias ao desenvolvimento do artefato. Com isto, ao finalizar uma tarefa, é possível saber quais conhecimentos foram utilizados pelo desenvolvedor e construir seu perfil técnico, gradativamente.

A fim de evitar inconformidades no processo, as *tags* são escolhidas a partir de um repo-

sitório central, o qual é atualizado constantemente. Em caso de inexistência de uma determinada *tag* no repositório, uma solicitação é feita ao suporte da empresa que irá analisar o pedido e incluir ou não a *tag*.

4.1.1.1 Definição das Tecnologias Utilizadas na Rotulação

O desenvolvimento de software engloba diversas tecnologias durante toda a execução de um projeto. Objetivando-se descobrir as principais tecnologias a serem incorporadas ao processo de Rotulação de Tarefas, foi realizada uma pesquisa em vagas de empregos ofertadas para desenvolvedores na rede social LinkedIn[1], a qual consiste em uma rede de negócios lançada no início de 2003 e voltada para aspectos profissionais. Foram analisados 20 perfis (escolhidos aleatoriamente durante o mês de maio de 2018) de vagas lançadas por diversas empresas como Gertec [2], Globo.com[3], Sankyu S.A[4], Sicredi[5], Nuxen[6], dentre outras.

Na Tabela 4.1.1.1 são apresentados os resultados da análise conduzida. Observa-se que a demanda por experiência em linguagens de programação está presente em todas as ofertas de vagas, sendo as exigências mais comuns em *C++*, *HTML*, *Java*, *Javascript*, *PHP*, *Python*, *Ruby* e *Swift*. Das tecnologias dispostas na Tabela 4.1.1.1 foram escolhidas quatro para integrar o processo de Rotulação de Tarefas: linguagem de programação, arcabouço, API e persistência (banco de dados). Esta escolha ocorreu devido a estas tecnologias terem sido as mais solicitadas pelas empresas e por serem consideradas diretamente relacionadas à implementação de uma tarefa técnica. A língua estrangeira foi descartada por não ser considerada uma característica técnica indispensável ao desenvolvimento de software. Arquitetura, ferramentas de versionamento e de automatização de *builds* foram rejeitadas por serem tecnologias a nível de projeto, ou seja, normalmente não variam de tarefa para tarefa. Por fim, optou-se por eliminar a metodologia de desenvolvimento, focando apenas na formação de equipes para projetos baseados em Scrum.

Estabeleceu-se ainda uma composição adicional para o processo de Rotulação que é o tipo de camada, podendo ser *back-end* ou *fron-tend*. Por exemplo, há linguagens de progra-

[1] https://br.linkedin.com/

[2] https://www.gertec.com.br/

[3] https://www.globo.com/

[4] http://www.sankyu.com.br/

[5] https://www.sicredi.com.br/

[6] http://www.nuxen.com.br/

Tabela 4.1: Principais tecnologias solicitadas em vagas de emprego.

Tecnologia	Ocorrência nas vagas ofertadas
Linguagem de programação	100,00%
Arcabouço	75,00%
API	60,00%
Banco de dados	40,00%
Língua estrangeira	40,00%
Arquitetura	40,00%
Ferramentas de versionamento	40,00%
Metodologia de desenvolvimento	40,00%
Ferramentas de automatização de *builds*	25,00%

mação que podem ser usadas tanto na implementação das interfaces com o usuário de um software (*front-end*) quanto na lógica da aplicação (*back-end*), porém de formas diferentes, ou seja, representando tarefas diferentes. Assim, para auxiliar tal diferenciação, utilizou-se também a camada.

Na Figura 4.1 têm-se exemplos de tarefas técnicas especificadas pelo modelo em que *Camada*, *Linguagem de Programação*, *Framework* (arcabouço), *API* e *Persistências* representam os tipos de tecnologias utilizadas durante a implementação. *Tarefa Técnica* representa o descritivo padronizado do artefato, obtido a partir do processo de Padronização de Tarefas que é detalhado a seguir.

Figura 4.1: Exemplos de tarefas técnicas especificadas pelo modelo,

Tarefa Técnica	Camada	Linguagem de Programação	Framework	API	Persistência
Criar metodo para criptografar a senha do admin	Back-end	JavaScript	Node	JsonWebToken	MongoDB
Criar tela de login	Front-end	TypeScript	Angular		
Criar schema de usuario	Back-end	JavaScript	Node	Mongoose	MongoDB

Fonte: próprio autor.

4.1.2 Padronização de Tarefas

No formato original do Scrum, as tarefas técnicas são escritas em linguagem natural. Entretanto, a falta de descritivos textuais estruturados dificulta a construção dos perfis dos desenvolvedores, uma vez que artefatos que possuem um mesmo propósito podem ser escritos

de maneiras diferentes. Por exemplo, um desenvolvedor pode ter em seu histórico de desenvolvimento tarefas técnicas registradas como "Criar tela de login", "Desenvolver interface de login"e "Criar UI de autenticação", as quais, na prática, possuem a mesma finalidade, porém com descritivos distintos. Esta diferenciação dificulta o reconhecimento por parte de algoritmos computacionais, uma vez que as tarefas são tratadas como artefatos diferentes. Com o intuito de mitigar tal limitação, o processo de Padronização de Tarefas visa permitir a utilização de descritivos textuais estruturados de maneira que tarefas técnicas de diferentes projetos, porém com um mesmo propósito, possam ser recuperadas e reusadas para construção dos perfis dos desenvolvedores.

De forma análoga ao processo Rotulação de Tarefas, propõe-se uma adaptação no Scrum, na qual ao decompor as US em tarefas, a equipe de desenvolvimento utilize um descritivo estruturado, ao invés de linguagem natural para descrição do artefato. Esta alteração padroniza a especificação das tarefas, possibilitando saber como os conhecimentos foram aplicados, isto é, quais habilidades foram adquiridas durante o desenvolvimento de software. Por exemplo, ao observar a Figura 4.1, é possível inferir que um desenvolvedor que implementou a tarefa da primeira linha possui conhecimentos ("saber teórico") em *Javascrip*, *Node*, *JsonWebToken* e *MongoDB*, porém sem considerar o descritivo estruturado não é possível identificar de que forma estes conhecimentos foram aplicados, isto é, qual habilidade ("saber fazer") foi adquirida ao finalizar a tarefa. Portanto, ao considerar ambos (descritivo e tecnologias), sabe-se que este desenvolvedor utilizou os conhecimentos nas tecnologias previamente citadas para *Criar método para criptografar senha do admin*.

4.1.2.1 Estruturação dos Descritivos Textuais

Com o intuito de estruturar os descritivos textuais dos artefatos gerados durante o processo de desenvolvimento de software, utilizou-se um modelo estruturado de US e tarefas, o qual foi desenvolvido em conjunto com o grupo ISE da UFCG. Este modelo, inicialmente proposto para US foi posteriormente estendido para tarefas técnicas. O modelo baseia-se na estruturação de US em dois níveis: módulo e operação. Estes níveis têm como objetivo padronizar as funcionalidades representadas pelas US. Na Figura 4.2, é possível observar as categorias (descritivos textuais estruturados) compreendidas pelo modelo de acordo com seus respectivos níveis, em que:

- **Autenticação:** representa operações de autenticação e autorização de usuário;
- **Cadastro:** representa operações de criação, recuperação, atualização e remoção de dados;
- **Gerencial:** representa operações administrativas como geração de relatório, visualização de painéis e notificações em geral.

Figura 4.2: Modelo estruturado de descrição de US.

Módulo	Operação
Autenticação	Fazer login com usuário e senha
	Fazer login com OAuth
	Recuperar senha
	Acessar pela primeira vez o sistema
	Validar permissão do usuário
	Atualizar perfil
	Criar conta
	Remover conta
Cadastro	Recuperação de dados
	Atualizar dados
	Inserir dados
	Remover dados
	Modificar inserção de dados
Gerencial	Visualizar painel
	Exportar relatório em PDF
	Exportar relatório em XLS
	Notificar via e-mail
	Notificar via aplicação

Fonte: próprio autor.

Os três módulos e 18 operações (Figura 4.2) foram obtidos a partir da análise de cinco *backlogs* de projetos reais pertencentes ao domínio de sistemas de informação web, compostos por 118 *US*. As US foram agrupadas de acordo com as funcionalidades que representavam, objetivando-se identificar similaridades para se estabelecer uma nomenclatura comum a cada grupo. Na Figura 4.3 são apresentados exemplos de US mapeadas pelo modelo.

A extensão do modelo para um terceiro nível (tarefa) ocorreu a partir da análise de 336 tarefas técnicas provenientes dos cinco projetos previamente citados. Para cada combinação

Figura 4.3: Exemplos de US mapeadas pelo modelo estruturado de US e tarefas.

User Story	Módulo	Operação
Como administrador, desejo fazer login para ter o primeiro acesso ao sistema	Autenticação	Fazer login com usuário e senha
Como administrador, desejo recuperar minha senha para ter acesso ao sistema novamente	Autenticação	Recuperar senha
Como usuário, quero efetuar o login pelo aplicativo para acessar a aplicação	Autenticação	Validar permissão do usuário
Como usuário, quero poder criar um novo projeto para iniciar as atividades	Cadastro	Inserir dados
Como administrador, desejo criar um contrato para associar as contas criadas	Cadastro	Inserir dados
Como usuário, quero poder editar minhas informações no aplicativo para mudar meus dados pessoais	Cadastro	Atualizar dados
Como administrador, quero ter um sistema de notificaçao para os usários possam ser notificados quando necessário	Gerencial	Notificar via aplicação
Como administrador, quero poder extrair dados dos projetos para gerar relatórios	Gerencial	Exportar relatório em PDF
Como usuário, desejo receber notificações sobre a entrega dos meus livros alugados	Gerencial	Notificar via aplicação

Fonte: próprio autor.

de módulo e operação definiu-se um conjunto de categorias com o intuito de representar tarefas com propósitos similares, seguindo processo análogo ao realizado com as US. Na Figura 4.4, têm-se exemplos de tarefas presentes nos *backlogs* analisados que foram mapeadas pelo modelo. É importante destacar que um conjunto complementar de descritivos foi adicionado para abranger tarefas que não haviam sido identificadas nos *backlogs*, porém são consideradas recorrentes no desenvolvimento de software. Este complemento foi realizado pelos pesquisadores do ISE em conjunto com desenvolvedores de projetos executados no Embedded. Ao todo, foram criados 88 categorias, totalizando 131 combinações de módulo, operação e tarefa. Na Figura 4.5, têm-se exemplos do modelo em três níveis com módulo *Autenticação*, operação *Recuperar senha* e as respectivas *Tarefas* relacionadas. No Apêndice B é apresentada a versão completa do modelo.

A validação do modelo estruturado de US e tarefas foi realizada mediante um estudo empírico com dados de 26 projetos compostos por 530 US e 1.879 tarefas, de quatro empresas de software que trabalham com projetos ágeis baseados em Scrum. Dentre as US coletadas, foram descartadas aquelas que não representam funcionalidades de negócio, ou seja, US relacionadas a configuração de ambiente, implantação de sistema, estudos de tecnologias etc, resultando em 407 US de negócio escritas em linguagem natural. De forma

Figura 4.4: Exemplos de tarefas técnicas mapeadas.

Tarefa Original	Tarefa Mapeada
Cria tela de login	Criar campos Login e Senha
Criar interface de login	Criar campos Login e Senha
Criar UI de autenticação	Criar campos Login e Senha
Criar entidades de usuário	Criar entidade no banco de dados
Criar schema do administrador	Criar entidade no banco de dados
criar tabela de usuario	Criar entidade no banco de dados
Implementar a API do cadastro do dispositivo	Criar método de inserção
Implementar a inserção de dados no dispositivo pelo administrador	Criar método de inserção
Criar método para cadastro dos dados pessoais	Criar método de inserção
Criar tela de alteração de senha padrão	Criar tela para cadastro de nova senha
Cria interface para mudança de senha	Criar tela para cadastro de nova senha
Criar tela de recuperaçao de senha	Criar tela para cadastro de nova senha

Fonte: próprio autor.

Figura 4.5: Exemplo de combinação dos três níveis do modelo estruturado de US e tarefas.

Módulo	Operação	Tarefa
Autenticação	Recuperar senha	Alterar tela de login
Autenticação	Recuperar senha	Validar e-mail e senha informado
Autenticação	Recuperar senha	Criar rota de recuperação de senha
Autenticação	Recuperar senha	Gerar senha padrão
Autenticação	Recuperar senha	Criar serviço de notificação
Autenticação	Recuperar senha	Criar tela para cadastro de nova senha

Fonte: próprio autor.

análoga, descartou-se tarefas não derivadas de US de negócio, resultando em 1.206.

Foram utilizadas duas métricas para avaliar a eficiência do modelo: cobertura e representatividade. Cobertura consiste na razão entre a quantidade total de US mapeadas pelo modelo e a quantidade total de US de negócio (Equação 4.1). Neste caso, quanto mais próximo de 100%, melhor é a cobertura. Representatividade corresponde à quantidade total de operações propostas pelo modelo sobre a quantidade de total de US de negócio (Equação 4.2). Assim, quanto menor o valor, melhor é a representatividade.

$$Cobertura = \frac{Quantidade\ de\ total\ de\ US\ mapeadas\ pelo\ modelo}{Quantidade\ total\ de\ US\ de\ negócio} \times 100 \tag{4.1}$$

$$Representatividade = \frac{Quantidade\ de\ operações\ propostas\ pelo\ modelo}{Quantidade\ total\ de\ US\ de\ negócio} \times 100 \tag{4.2}$$

Como resultado, o modelo obteve 90,17% de cobertura em relação às US de negócio, ou seja, das 407 US de negócio cerca 367 US foram mapeadas pelo modelo. Deste montante, 80,11% foram do módulo *Cadastro*, 10,35% de *Autenticação* e 9,54% de *Gerencial*. No que diz respeito à representatividade, obteve-se um valor igual a 4,9%.

Em relação às tarefas, conseguiu-se uma cobertura de 80,93% (Equação 4.3), ou seja, 976 tarefas associadas a US de negócio foram mapeadas pelo modelo. A representatividade alcançada foi de 13% (Equação 4.4), ou seja, as 131 combinações conseguiram representar as 976 tarefas originais.

$$Cobertura' = \frac{Quantidade\ de\ tarefas\ de\ US\ de\ negócio\ mapeadas\ pelo\ modelo}{Quantidade\ total\ de\ tarefas\ de\ US\ negócio\ mapeadas} \times 100 \tag{4.3}$$

$$Representatividade' = \frac{Quantidade\ de\ descritivos\ de\ tarefas\ do\ modelo}{Quantidade\ total\ de\ US\ de\ negócio} \times 100 \tag{4.4}$$

4.2 Abordagem de Apoio à Decisão

A aplicação do modelo de especificação de tarefas consiste em uma etapa fundamental para operacionalização da solução, pois à medida que este processo vai sendo realizado, uma base

de dados históricos com informações essenciais para a construção dos perfis de desenvolvedores e projetos vai sendo construída.

A operacionalização representa a segunda parte da solução proposta. Nas subseções a seguir são detalhados os passos e os mecanismos necessários para viabilizar a execução do AG para realizar a formação de múltiplas equipes, a partir da realocação interna dos desenvolvedores.

4.2.1 Construção do Perfil do Desenvolvedor

As competências dos desenvolvedores, definidas neste trabalho como seus conhecimentos e suas habilidades, são geradas a partir de uma base de dados contendo todo o histórico de desenvolvimento dos projetos conduzidos na empresa. Conforme explicitado, *tags* são termos que descrevem e permitem a classificação da informação através de palavras-chave. Os conhecimentos de um desenvolvedor são representados por *tags* ponderadas, adquiridas gradativamente ao longo de sua participação em projetos de desenvolvimento de software. A estrutura de uma *tag* ponderada corresponde ao par chave-valor, em que:

- **Chave:** é a palavra-chave que representa a informação da *tag*.
- **Valor:** é o peso que a informação da *tag* possui.

De forma análoga, as habilidades do desenvolvedor são representadas pelas tarefas que ele implementou ao longo de sua participação em projetos.

Projetos baseados em Scrum são planejados com base em *Sprints*, as quais são compostas por US, que por sua vez são quebradas em tarefas técnicas. Com a utilização do modelo proposto (Seção 4.1), estas tarefas são rotuladas e padronizadas, e à medida que vão sendo concluídas pelo desenvolvedor, seu perfil técnico vai sendo automaticamente atualizado, incrementando a pontuação (em uma unidade) de cada *tag* e tarefa (estruturada) presente em seu perfil.

Na Figura 4.6, tem-se um exemplo de perfil de um desenvolvedor. Os valores das *tags* representam o número de vezes que ele implementou uma tarefa rotulada com uma determinada chave (Figura 4.6a). Também registra-se em seu perfil o número de vezes que ele implementou uma determinada tarefa (Figura 4.6b).

Figura 4.6: Exemplo do perfil de um desenvolvedor.

Tag	
Chave	**Valor**
Angular	10
Bootstrap	10
Java	5
JavaScript	8
JsonWebToken	2
MongoDB	12
Mongoose	10
Node	8
TypeScript	10

(a)

Tarefa (mapeada)	Camada	Valor
Adicionar menus no dashboard	Fron-end	3
Alterar menu do sistema	Fron-end	5
Alterar tela de login	Fron-end	12
Consumir API externa	Back-end	4
Criar filtro para listagem	Fron-end	4
Encriptar senha do usuário	Back-end	2
Fazer o roteamento da URL	Back-end	6
Fazer o roteamento da URL	Fron-end	2
Persistir dados banco de dados	Back-end	3
Validar e-mail e senha informado	Back-end	3

(b)

Fonte: próprio autor.

4.2.2 Construção do Perfil do Projeto

Assim como o perfil do desenvolvedor, o perfil do projeto também é formado por tarefas (estruturadas) e *tags* ponderadas. No caso de um projeto alvo, ou seja, projeto para o qual será formada um equipe, seu perfil é originado pelas tarefas técnicas que compõem uma determinada *Sprint*. Em outras palavras, esta abordagem sugere equipes multidisciplinares capazes de atender às demandas de um projeto a nível de tarefa. Na Figura 4.7, é apresentado um exemplo de perfil de um projeto, após aplicação do modelo de especificação de tarefas.

Figura 4.7: Exemplo de perfil de um projeto.

Sprint	User Story	Tarefa	Tarefa_mapeada	Camada	Linguagem	Framework	API	Persistencia
S01	US01	TA01	Criar campos Login e Senha	Front-end	TypeScript	Angular		
S01	US01	TA02	Criar entidade no banco de dados	Back-end	JavaScript	Node	Mongoose	MongoDB
S01	US01	TA03	Persistir no banco os dados de acesso	Back-end	JavaScript	Node	Mongoose	MongoDB
S01	US01	TA04	Criar tela para cadastro de nova senha	Front-end	TypeScript	Angular		
S01	US02	TA01	Criar tela inserção	Front-end	TypeScript	Angular		
S01	US02	TA02	Criar rota de recuperação de senha	Back-end	JavaScript	Node	Mongoose	MongoDB
S01	US02	TA03	Validar permissão do usuário na aplicação	Back-end	JavaScript	Node	Mongoose	MongoDB
S01	US03	TA01	Alterar tela de login	Front-end	HTML/CSS	Angular		
S01	US03	TA02	Criar tela para cadastro de nova senha	Front-end	TypeScript	Angular		
S01	US03	TA03	Criar rota de recuperação de senha	Back-end	JavaScript	Node	Nodemailer	MongoDB

Fonte: próprio autor.

4.2.3 Processo de Formação de Múltiplas Equipes

O processo de formação de múltiplas equipes é realizado com base na determinação das competências técnicas dos desenvolvedores em relação a cada um dos projetos alvos, objetivando-se alcançar a melhor distribuição global. Para calcular estas competências técnicas, os seguintes passos são necessários: i) estruturação do conhecimento; ii) determinação das habilidades; iii) geração da Matriz de Competência Técnica; e iv) realocação interna dos desenvolvedores. Estes passos são detalhados nas subseções seguintes.

4.2.3.1 Estruturação do Conhecimento

Nesta abordagem, vetores de características são utilizados para estruturar o conhecimento do desenvolvedor e o conhecimento necessário para a realização de uma tarefa técnica em um projeto. Um vetor de característica é um vetor de dimensão n formado por valores numéricos que representam um determinado conjunto de características de um objeto. Muitos algoritmos na área de Reconhecimento de Padrões e Aprendizado de Máquina requerem uma representação numérica de objetos, o que permite o processamento e análise dos dados. O processo de Rotulação de Tarefas (Seção 4.1.1) possibilita a atribuição de *tags* que representam as características dos objetos correspondentes aos perfis dos desenvolvedores e projetos. Na Tabela 4.2, tem-se a estrutura padrão dos vetores utilizados. A representação do vetor é dada através das *chaves* das *tags* do objeto e a instanciação ocorre com base nos respectivos *valores* destas *tags*. Todas as instâncias de vetores utilizadas nesta solução apresentam em sua versão final (após a normalização) *valores* no intervalo fechado $[0, 1]$.

Tabela 4.2: Estrutura de um vetor de característica.

Representação do Vetor	$chave_1$	$chave_2$	$chave_3$	...	$chave_n$
Instanciação do Vetor	$valor_1$	$valor_2$	$valor_3$	...	$valor_m$

Geração dos Vetores de Características

Conforme explicitado anteriormente, as equipes são formadas de acordo com *Sprints* específicas de cada projeto alvo. Deste modo, para cada tarefa técnica definida no *Sprint Planning* de um projeto, um vetor de característica é formado. Portanto, se um projeto alvo possui X

tarefas definidas para uma *Sprint*, seu perfil será representado por X vetores de características.

Na solução proposta, por padrão, todas as *tags* que compõem o perfil do projeto têm o mesmo peso, no caso, o valor máximo 1 (um). Portanto, ao criar o vetor de características de uma tarefa, todos os seus índices recebem o valor 1 (um), objetivando assim encontrar os desenvolvedores que possuem maior competência possível para às demandas do projeto. Na Figura 4.8, são apresentados exemplos de vetores de características de tarefa que representariam o perfil de projeto disposto na Figura 4.7.

Figura 4.8: Exemplo de vetores de características de tarefa.

Sprint	User Story	Tarefa	Vetor de Característa da Tarefa				
S01	US01	TA01	Chave	TypeScript		Angular	
			Valor	1		1	
S01	US01	TA02	Chave	JavaScript	Node	Mongoose	MongoDB
			Valor	1	1	1	1
S01	US01	TA03	Chave	JavaScript	Node	Mongoose	MongoDB
			Valor	1	1	1	1
S01	US01	TA04	Chave	TypeScript		Angular	
			Valor	1		1	
S01	US02	TA01	Chave	TypeScript		Angular	
			Valor	1		1	
S01	US02	TA02	Chave	JavaScript	Node	Mongoose	MongoDB
			Valor	1	1	1	1
S01	US02	TA03	Chave	JavaScript	Node	Mongoose	MongoDB
			Valor	1	1	1	1
S01	US03	TA01	Chave	HTML/CSS		Angular	
			Valor	1		1	
S01	US03	TA02	Chave	TypeScript		Angular	
			Valor	1		1	
S01	US03	TA03	Chave	JavaScript	Node	Nodemailer	MongoDB
			Valor	1	1	1	1

Fonte: próprio autor.

Os vetores de características do desenvolvedor são criados a partir da representação dos vetores de características do projeto. Cada projeto alvo possui um número de vetores de características, determinado pela quantidade de tarefas definidas para sua *Sprint*.

A instanciação dos valores dos vetores de características do desenvolvedor é realizada de acordo com os valores das *tags* ponderadas presentes em seu perfil técnico. Por exemplo,

para cada chave de um vetor de características do desenvolvedor, verifica-se se há uma *tag* com chave correspondente ao seu perfil. Em caso positivo, o valor desta *tag* é adicionado na respectiva posição do vetor. Em caso negativo, adiciona-se o valor 0 (zero). Este processo se repete até que todos os índices do vetor tenham sido preenchidos. É importante salientar que inicialmente os valores preenchidos não estão no intervalo fechado $[0, 1]$, portanto, há a necessidade de normalizá-los. Na Figura 4.9, são apresentados exemplos de vetores de características que seriam gerados (antes da normalização) para o desenvolvedor da Figura 4.6.

Figura 4.9: Exemplo de vetores de características do desenvolvedor.

Sprint	User Story	Tarefa	Vetor de Característa do Desenvolvedor				
S01	US01	TA01	Chave	**TypeScript**		**Angular**	
			Valor	10		10	
S01	US01	TA02	Chave	**JavaScript**	**Node**	**Mongoose**	**MongoDB**
			Valor	8	8	10	12
S01	US01	TA03	Chave	**JavaScript**	**Node**	**Mongoose**	**MongoDB**
			Valor	8	8	10	12
S01	US01	TA04	Chave	**TypeScript**		**Angular**	
			Valor	10		10	
S01	US02	TA01	Chave	**TypeScript**		**Angular**	
			Valor	10		10	
S01	US02	TA02	Chave	**JavaScript**	**Node**	**Mongoose**	**MongoDB**
			Valor	8	8	10	12
S01	US02	TA03	Chave	**JavaScript**	**Node**	**Mongoose**	**MongoDB**
			Valor	8	8	10	12
S01	US03	TA01	Chave	**HTML/CSS**		**Angular**	
			Valor	0		10	
S01	US03	TA02	Chave	**TypeScript**		**Angular**	
			Valor	10		10	
S01	US03	TA03	Chave	**JavaScript**	**Node**	**Nodemailer**	**MongoDB**
			Valor	8	8	10	12

Fonte: próprio autor.

4.2.3.2 Normalização dos Vetores

Após a instanciação dos vetores de característica de todos os desenvolvedores, ocorre o processo de normalização. Como as *tags* de cada vetor podem assumir valores em intervalos diferentes, há a necessidade de harmonizar as escalas através da normalização para valores

entre 0 (zero) e 1 (um). Para cada chave presente em um vetor de característica do desenvolvedor, verifica-se qual o maior valor que aquela chave pode assumir, considerando todos os desenvolvedores que a possuem, em seguida o maior valor corresponderá a 1. O menor valor a ser considerado é sempre 0 (zero), representando o caso em que o desenvolvedor não possui uma determinada *tag* em seu perfil. Os demais serão transformados em valores proporcionais dentro deste intervalo. Os valores são normalizados utilizando-se a Equação 4.5, em que $x = (x_1, x_2, x_3, ..., x_n)$ é o valor de cada chave na mesma posição de cada vetor específico do desenvolvedor e z_i é o i^{th} valor normalizado.

$$z_i = \frac{x_i - min(x)}{max(x) - min(x)} \tag{4.5}$$

4.2.3.3 Similaridade entre Vetores

Matematicamente, métricas de distância são utilizadas para medir a distância entre dois pontos no espaço vetorial. Esta abordagem utiliza a Similaridade de *Manhattan* (Equação 4.7) para calcular a similaridade entre dois vetores k-dimensionais, uma vez que esta métrica foi a que proporcionou melhores resultados nos testes realizados (Capítulo 5). Esta métrica é originada a partir da Distância de *Manhattan* (Equação 4.6). De acordo com a Equação 4.6, a distância máxima entre dois vetores varia de acordo com o tamanho dos vetores utilizados. Por exemplo, ao calcular a distância entre dois vetores de tamanho 5 (cinco), o primeiro composto apenas de 0s (zeros) e o segundo apenas de 1s (uns), a distância é 5 (cinco). Analogamente, se os vetores tiverem tamanho 10 (dez), a distância será 10 (dez). Dado que os vetores de características de projeto podem ter diferentes dimensões, optou-se por utilizar a Similaridade de *Manhattan* que fornece valores no intervalo [0,1], independente do tamanho dos vetores.

As similaridades calculadas são utilizadas para determinar o quão semelhantes os conhecimentos dos desenvolvedores são em relação aos conhecimentos necessários em um dado projeto alvo.

$$Distância\ de\ Manhattan = \sum_{i=1}^{k} |u[i] - v[i]| \tag{4.6}$$

$$Similaridade\ de\ Manhattan = 1 - \frac{\sum_{i=1}^{k} |u[i] - v[i]|}{k} \tag{4.7}$$

Na Figura 4.10 é ilustrado um exemplo do cálculo de similaridade entre os vetores de características de tarefas e de desenvolvedores.

Figura 4.10: Exemplo de cálculo de similaridade.

Sprint	User Story	Tarefa	Vetor de Característa da Tarefa				
S01	US01	TA01	Chave	TypeScript		Angular	
			Valor	1		1	
S01	US01	TA02	Chave	JavaScript	Node	Mongoose	MongoDB
			Valor	1	1	1	1

Desenvolvedor	Tarefa	Vetor de Característa do Desenvolvedor					Similaridade
João	S01-US01-TA02	Chave	TypeScript		Angular		0,9
		Valor	1		0,8		
	S01-US01-TA03	Chave	JavaScript	Node	Mongoose	MongoDB	0,25
		Valor	0,1	0,2	0,2	0,5	
Maria	S01-US01-TA02	Chave	TypeScript		Angular		0,5
		Valor	0,5		0,5		
	S01-US01-TA03	Chave	JavaScript	Node	Mongoose	MongoDB	0,95
		Valor	0,9	0,9	1	1	

Fonte: próprio autor.

4.2.3.4 Determinação das Habilidades

As habilidades do desenvolvedor são representadas pelo número de vezes que ele implementou uma determinada tarefa. Desta forma, independente do projeto em que o desenvolvedor trabalhou, as tarefas implementadas podem ser recuperadas e reusadas para compor seu perfil técnico.

Assim como ocorre com os vetores de características, as habilidades dos desenvolvedores também precisam ser normalizadas. Neste caso, a normalização ocorre por tarefa, ou seja, a Equação 4.5 é aplicada para cada categoria de tarefa na base de dados.

Na Figura 4.11 tem-se um exemplo de como seriam as habilidades do desenvolvedor, de acordo com o perfil apresentado na Figura 4.6. É possível perceber que este desenvolvedor possui valores máximos (1) para duas tarefas, ou seja, ele foi o responsável por implementar tal tarefa mais vezes na empresa.

Figura 4.11: Exemplo das habilidades de um desenvolvedor.

Tarefa (mapeada)	Valor (original)	Valor (normalizado)
Adicionar menus no dashboard_Fron-end	3	0,5
Alterar menu do sistema_Fron-end	5	0,5
Alterar tela de login_Fron-end	12	1
Consumir API externa_Back-end	4	0,2
Criar filtro para listagem_Fron-end	4	0,4
Encriptar senha do usuário_Back-end	2	0,2
Fazer o roteamento da URL_Back-end	6	1
Fazer o roteamento da URL_Fron-end	2	0,3
Persistir dados banco de dados_Back-end	3	0,6
Validar e-mail e senha informado_Back-end	3	0,6

Fonte: próprio autor.

4.2.3.5 Matriz de Competência Técnica

A Matriz de Competência Técnica (M_{CT}) é uma estrutura que tem como objetivo reunir todas as competências dos desenvolvedores em relação aos projetos alvos, a fim de viabilizar a execução do AG utilizado para formação de múltiplas equipes.

M_{CT} é uma matriz de ordem NxM, em que as n linhas representam os desenvolvedores disponíveis para realocação, as m colunas os projetos alvos e os elementos $ct_{i,j}$ correspondem às competências técnicas dos desenvolvedores em relação aos projetos alvos. Cada elemento da matriz M_{CT} é obtido através da Equação 4.8, a qual fornece um cálculo ponderado com base nos conhecimentos e habilidades determinados para cada desenvolvedor. As partes do numerador, separadas pela multiplicação, são acrescidas de uns ($1s$) para evitar que o valor da equação seja zerado caso alguma parte seja igual a zero. Os componentes da Equação 4.8 são descritos a seguir:

$$M_{CT} = \begin{bmatrix} CT_{11} & CT_{12} & CT_{13} & \dots & CT_{1m} \\ CT_{21} & CT_{22} & CT_{23} & \dots & CT_{2m} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ CT_{n1} & CT_{n2} & CT_{n3} & \dots & CT_{nm} \end{bmatrix}$$

$$CT_{i,j} = \frac{\sum_{k=1}^{n}((1 + sim(V_{D_k}, V_{T_k})) \times (1 + I_{(D,T_k))})}{n} \tag{4.8}$$

i representa o i^{th} desenvolvedor e $i = \{1, 2, 3, ..., n\}$;

j representa o j^{th} projeto alvo e $j = \{1, 2, 3, ..., m\}$;

V_{D_k} representa o vetor de características do desenvolvedor em relação a um tarefa k;

V_{T_k} representa o vetor de características de uma tarefa k;

$sim(V_{D_k}, V_{T_k})$ é a similaridade calculada entre um vetor de característica do desenvolvedor e um vetor de características da tarefa;

$I_{(D,T_k)}$ representa o número de vezes que o desenvolvedor implementou a tarefa k;

n é o número de tarefas definidas para a *Sprint* do projeto alvo.

4.2.4 Realocação Interna Utilizando Algoritmo Genético

O processo de formação de múltiplas equipes ocorre através da realocação interna dos desenvolvedores entre os diversos projetos da empresa. A ideia consiste em auxiliar a formação de equipes multidisciplinares capazes de atender às demandas técnicas e de negócio de cada projeto, de forma a alcançar a melhor distribuição global.

As demandas técnicas de um projeto correspondem às competências necessárias para desenvolver um produto, processo ou serviço. As demandas de negócio consistem nas políticas para satisfazer os objetivos ou regras do negócio. De acordo com o Guia do Scrum, a Equipe de Desenvolvimento deve ser multidisciplinar, ou seja, deve possuir todas as competências necessárias para realizar o trabalho sem depender de outros que não fazem parte da equipe. No entanto, há diferentes maneiras de se formar equipes multidisciplinares. Por exemplo, considere uma equipe $E1$, na qual que todos os membros possuem todas as competências necessárias e uma equipe $E2$ em que uma parte dos membros possue um subconjunto das competências necessárias e a outra parte possui um outro subconjunto (complementar) destas competências. Ambos os casos ilustram exemplos de equipes multidisciplinares que atendem o Scrum, apesar de serem geradas de acordo com demandas de negócios diferentes.

O problema de formação de múltiplas equipes é visto como um problema de otimização, uma vez que há a necessidade de formar simultaneamente equipes, a fim de atender variadas demandas, as quais podem ser conflitantes fazendo com que ocorra a disputa por recursos

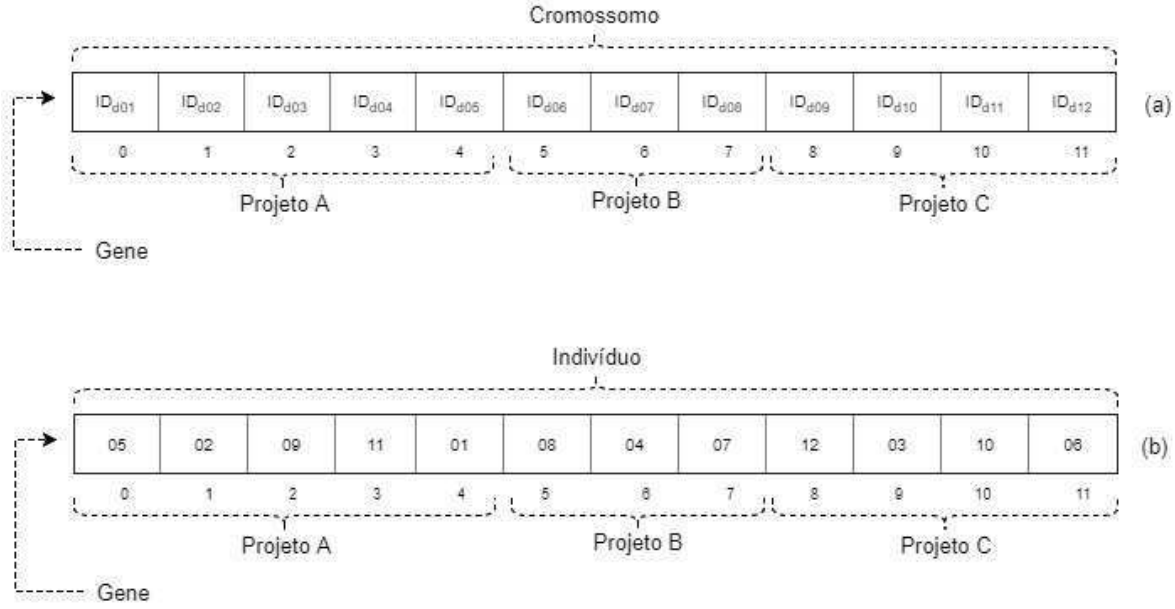
Tabela 4.3: Configuração dos parâmetros do algoritmo Genético.

Parâmetro	**Valor**
Tamanho da população	100
Número de Gerações Estáveis	5
Método de Cruzamento	*Partially Mapped Crossover*
Taxa de Cruzamento	50%
Método de Mutação	*Swap Mutator*
Taxa de Mutação	5%
Método de seleção	Torneio (n = 3)
Seleção dos Sobreviventes	Elitismo (e = 1)

(humanos). Diante disto, AG surge como uma técnica viável para o problema, pois é um método de busca e otimização capaz de fornecer soluções ótimas ou aproximadas, visando atender todas as demandas. Além disso, a capacidade de personalização mediante utilização das funções de *fitness*, permite encontrar soluções de acordo com diferentes objetivos e restrições. Na Tabela 4.3 estão dispostos os parâmetros do AG utilizados nesta abordagem. A configuração foi escolhida com base em testes realizados e descritos no Capítulo 5.

Nesta abordagem, um cromossomo é representado por uma estrutura de permutação, a qual permite a representação da solução a partir da combinação de elementos. Desta forma, os índices sequenciais na estrutura de um cromossomo representam as vagas nas equipes dos projetos da empresa. Cada índice corresponde a um gene, que por sua vez pode assumir um determinado valor correspondente a um identificador de desenvolvedor. Por exemplo, na Figura 4.12 (a), é possível observar como seria a estrutura de um cromossomo para uma empresa com três projetos e 12 desenvolvedores. Os genes dos índices de $0\ a\ 4$ representam as vagas para as equipes de um determinado $projeto\ A$. Analogamente, têm-se os índices de $5\ a\ 7$ para um $projeto\ B$ e de $8\ a\ 11$ para um $projeto\ C$. Cada gene pode assumir um valor inteiro de $1\ a\ 12$, os quais são utilizados como identificadores únicos para os desenvolvedores. Na Figura 4.12 (b), tem-se o respectivo genótipo, o qual representa uma possível solução. Neste caso, tem-se que $Projeto\ A = \{05, 02, 09, 11, 01\}$, ou seja, a equipe deste projeto seria formada pelos desenvolvedores cujos ids correspondem aos números do conjunto. Analogamente, tem-se que $Projeto\ B = \{08, 04, 07\}$ e $Projeto\ C = \{12, 03, 10, 06\}$.

Figura 4.12: Exemplo de estrutura de um cromossomo.



Fonte: próprio autor.

4.2.4.1 Funções de Fitness e Regras de Negócio

A função de *fitness* é um dos principais mecanismos de um AG, pois é através dela que se mede o quão próximo um indivíduo está da solução desejada ou o quão boa é esta solução. Quando esta função é pouco representativa, boas e más soluções acabam tendo pouca diferenciação, o que pode resultar no descarte de uma solução ótima durante a execução do algoritmo.

O processo de formação de múltiplas equipes é caracterizado por um cenário em que cada projeto possui demanda específica. Diante disto, a abordagem proposta dispõe de múltiplas funções de *fitness* que podem ser utilizadas para atender diferentes demandas técnicas e de negócio, as quais são detalhadas a seguir.

Alocação Individual. Esta regra de negócio visa formar equipes multidisciplinares, nas quais todos os membros devem possuir (individualmente) o maior número possível das competências necessárias para atender as demandas técnicas do projeto. A função de *fitness* utilizada para esta regra é representada pela Equação 4.9. Esta função calcula a média dos valores de competência técnica que são calculados para cada desenvolvedor em relação a um dado projeto (Seção 4.2.3.5).

$$f_1(X) = \frac{\sum_{i=1}^{n} CT(D_i, P_i)}{n} \tag{4.9}$$

X representa o cromossomo a ser avaliado;

n representa o número de genes do cromossomo;

i representa a posição do gene na estrutura do cromossomo;

$CT(D_i, P_i)$ representa o valor de competência do desenvolvedor na posição i em relação ao projeto na mesma posição.

Alocação de Especialistas. Esta regra de negócio visa formar equipes multidisciplinares, possibilitando a seleção de membros a partir de demandas técnicas distintas para cada vaga dentro de um mesmo projeto. Por exemplo, para um projeto com uma equipe de tamanho cinco, é possível determinar que uma vaga seja ser preenchida por um especialista com conhecimentos em *TypeScript* (linguagem de programação) e Angular (arcabouço), outra vaga por um especialista com conhecimento em *MySQL* (persistência) e as demais de forma padrão (Alocação Individual). Para tanto, esta regra também utiliza a função de *fitness* $f_1(X)$, porém a abordagem proposta recebe como entrada um conjunto de *tags* ponderadas para cada vaga do projeto. Computacionalmente, a diferença em relação à regra anterior reside na forma como os conhecimentos são estruturados (Seção 4.2.3.1), isto é, durante a geração dos vetores de características, desenvolvedores que possuem as *tags* especificadas para as vagas têm seus conhecimentos (em relação à *tag*) multiplicados pelos pesos das respectivas *tags* de entrada. Por padrão, as *tags* dos vetores de características das tarefas têm valor 1 (um), porém para as vagas personalizadas, estas *tags* têm seu valor também multiplicado pelo peso da *tag* de entrada. Desta forma, desenvolvedores que possuem os conhecimentos solicitados obtêm maior similaridade do que os demais, tendo mais chances de serem escolhidos para compor a equipe.

Alocação Colaborativa. Esta regra de negócio visa formar equipes multidisciplinares, a fim de atender de forma colaborativa as demandas técnicas dos projetos, isto é, a equipe como um todo deve possuir as competências necessárias para realizar o trabalho. A função de *fitness* utilizada para esta regra de negócio é representada pela Equação 4.10. Computacionalmente, ao invés de vetores de características individuais para os desenvolvedores, um único vetor é formado para toda a equipe, a partir da média dos valores de índice dos vetores

individuais. O número de vezes que a tarefa foi implementada também é dado pela média da equipe.

$$f_2(X) = \frac{\sum_{i=1}^{e} CT(E_i, P_i)}{e} \tag{4.10}$$

X representa o cromossomo a ser avaliado;

e representa o número de equipes a serem formadas;

i representa a i^{th} equipe na estrutura do cromossomo;

$CT(E_i, P_i)$ representa o valor de competência da i^{th} equipe em relação ao i^{th} projeto.

Alocação Complementar. Esta regra de negócio é derivada da anterior, com a diferença de que é possível fixar membros para uma equipe. A ideia consiste em selecionar membros para completar equipes pré-existentes, considerando as competências da equipe atual. Esta regra também utiliza a função de *fitness* $f_2(X)$, porém recebe como entrada os desenvolvedores que deverão obrigatoriamente fazer parte da equipe.

Na Figura 4.12, ilustra-se um exemplo de um cenário, no qual os projetos A, B e C são representados na estrutura do cromossomo. Considerando que os três projetos possuem demandas distintas, a abordagem proposta permite que cada gestor escolha uma regra de negócio diferente para formar sua respectiva equipe. Do ponto de vista computacional, diferentes funções de *fitness* são aplicadas a partes específicas de um mesmo cromossomo, fazendo com que o algoritmo tente otimizar a distribuição dos desenvolvedores de forma a satisfazer simultaneamente todas as funções aplicadas.

4.3 Considerações Finais do Capítulo

Neste capítulo, foi descrita uma abordagem de apoio à decisão para formação de múltiplas equipes para projetos ágeis baseados em Scrum.

Originalmente, o Scrum não oferece os mecanismos necessários para operacionalizar a solução, por isso, definiu-se um modelo de especificação de tarefas que compreende processos que propõem algumas adaptações no arcabouço, visando a criação de perfis técnicos a partir de atributos de baixa granularidade. Para tanto, *tags* são utilizadas para contextualizar as tarefas de acordo com as tecnologias associadas, possibilitando a estruturação do conhecimento do desenvolvedor. A padronização dos descritivos textuais permite a recuperação e

reúso das tarefas, mesmo quando originadas de diferentes projetos. A formação de equipes é realizada com auxílio de algoritmo genético, buscando alancar a melhor distribuição global.

Algumas abordagens presentes na literatura utilizam o custo dos desenvolvedores como fator de restrição durante o processo de formação de equipes, com o intuito não ultrapassar orçamentos pré-estabelecidos. O custo financeiro não representa um atributo técnico do indivíduo, por isso, não faz parte do escopo do presente trabalho. Por outro lado, uma vez que a abordagem proposta utiliza Algoritmo Genético para formação de equipes, é possível adicionar restrições às funções de *fitness*, de modo a incorporar os custos durante a execução do algoritmo.

Capítulo 5

Validação e Avaliação da Abordagem

Neste capítulo, são descritos os processos de validação e avaliação da abordagem proposta. Para realização destes processos, foi construída uma base de dados reais, originada a partir da coleta e estruturação de dados históricos armazenados em repositórios de projetos de desenvolvimento de software.

5.1 Montagem da Base de Dados Históricos

Foram coletados e estruturados dados reais de 12 projetos de desenvolvimento de software baseados em Scrum, executados entre os anos de 2015 e 2018. Os projetos, em média, são compostos por oito *Sprints*, 28 US e 124 tarefas técnicas. Cada *Sprint* tem duração média de 15 dias, totalizando dados referentes a quatro meses de execução para cada projeto. Ao todo, a base de dados é composta por 1.496 tarefas, implementadas por 52 desenvolvedores diferentes. É importante destacar que, por questões de confidencialidade, omitem-se os nomes dos projetos e dos desenvolvedores.

Infelizmente, não foi possível reunir um número maior de projetos, uma vez que a base precisa conter informações das tarefas realizadas, bem como, dos desenvolvedores responsáveis. Dessa forma, diversos projetos em potencial foram descartados, pois não dispunham de tais informações. Além disso, para aplicar o modelo de especificação de tarefas é necessária participação de, pelo menos, a maioria da equipe original do projeto para que os dados resultantes sejam fidedignos.

5.1.1 Aplicação do Modelo de Especificação de Tarefas

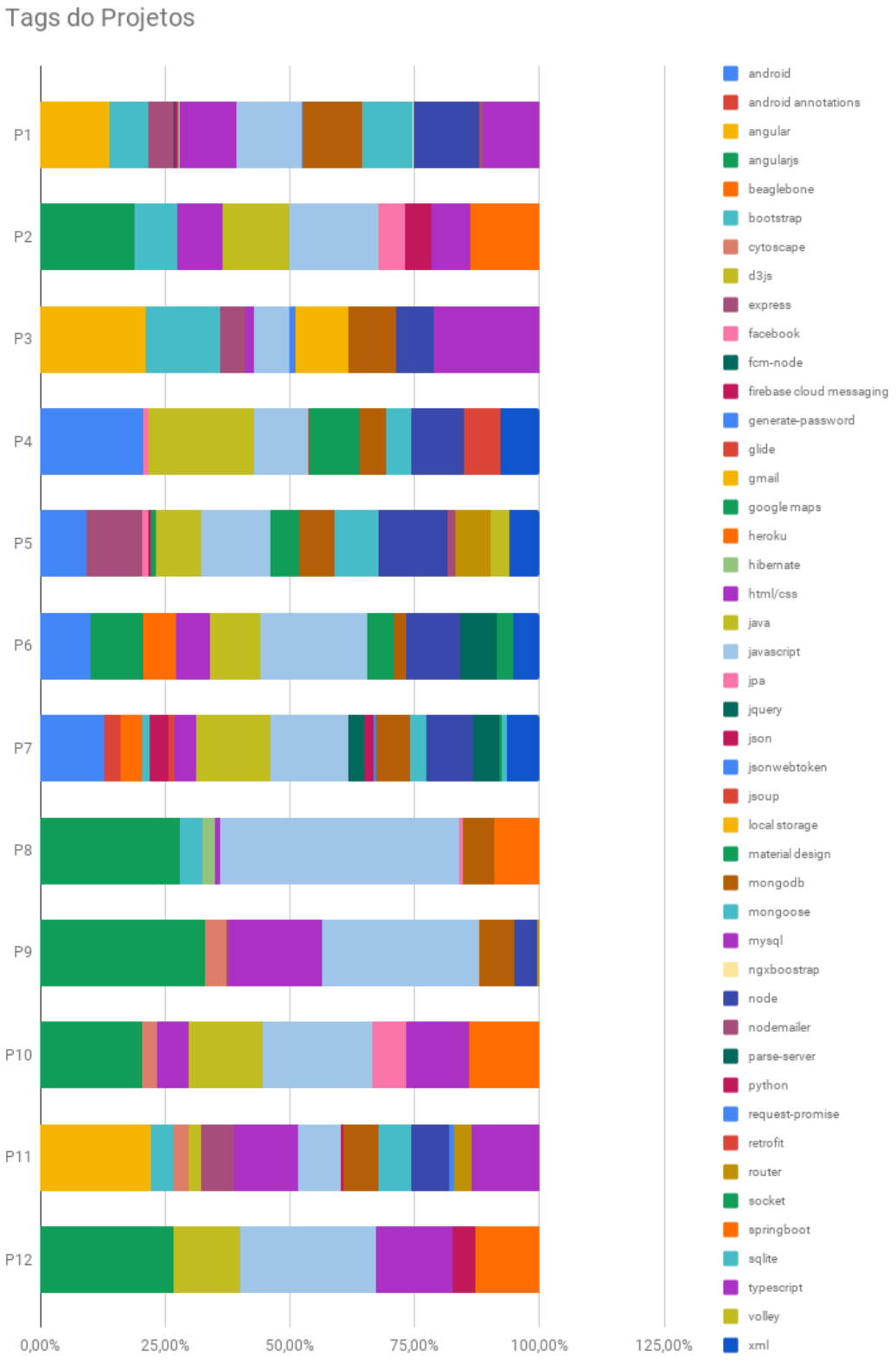
Após a criação da base, foi realizado aplicação do modelo de especificação de tarefas (Capítulo 4, Seção 4.1) que compreende a Rotulação e Padronização de Tarefas. O modelo foi originalmente proposto para ser executado ao longo das execuções das *Sprints*. No entanto, não foi possível tal realização em tempo real, uma vez que a maioria dos projetos já estava finalizada. Assim, os dados foram recuperados de um repositório privado e, posteriormente, estruturados em planilhas eletrônicas para facilitar a utilização.

As etapas de Rotulação e Padronização foram realizadas em ciclos distintos, pois julgou-se que realizá-las simultaneamente causaria esforço excessivo aos participantes, o que poderia afetar a confiabilidade do resultado final. A Rotulação foi realizada pelas equipes de desenvolvimento originais dos respectivos projetos. Entretanto, em alguns casos não foi possível contar com a formação completa, uma vez que membros de projetos mais antigos não estavam disponíveis para a realização do processo. Em cada projeto, pelo menos 70% da equipe original participou desta etapa. É importante destacar que durante o processo de Rotulação os desenvolvedores foram orientados a atribuir novas *tags*, caso não encontrassem opções adequadas no repositório, o que permitiu maior flexibilidade durante o processo. Ao final, as novas *tags* foram checadas, ajustadas e adicionadas ao repositório.

Na Figura 5.1, estão dispostas as *tags* atribuídas em cada projeto. No gráfico, é possível observar os perfis dos projetos, segmentados de acordo com as tecnologias que os compõem. O cálculo da porcentagem é feito com base nas *tags* de tarefas técnicas de cada projeto. Por exemplo, se um projeto possui 10 tarefas, das quais cinco possuem a *tag java* e as outras cinco a *tag mysql*, então seu perfil será representado por 50% *java* e 50% *mysql*. Ao todo, foram utilizadas 45 *tags* distintas (Figura 5.1) para representar as tecnologias associadas às tarefas. A base de dados é composta por projetos da plataforma web e móvel, com predominância de *tags* como *javascript* e *angularjs* que são tipicamente tecnologias voltadas para o desenvolvimento web. Dos 12 projetos coletados, oito são exclusivamente da plataforma web ($P1$, $P2$, $P3$, $P8$, $P9$, $P10$, $P11$ e $P12$), dois são exclusivamente da plataforma móvel (*Android*) (***P4 e P5***) e dois são híbridos web / *móvel (Android)* ($P6$ e $P7$).

A Padronização de Tarefas (Capítulo 4, Seção 4.1.2)) foi realizada em um segundo ciclo, em conjunto com o grupo ISE e desenvolvedores das equipes dos projetos. Na Figura 5.2,

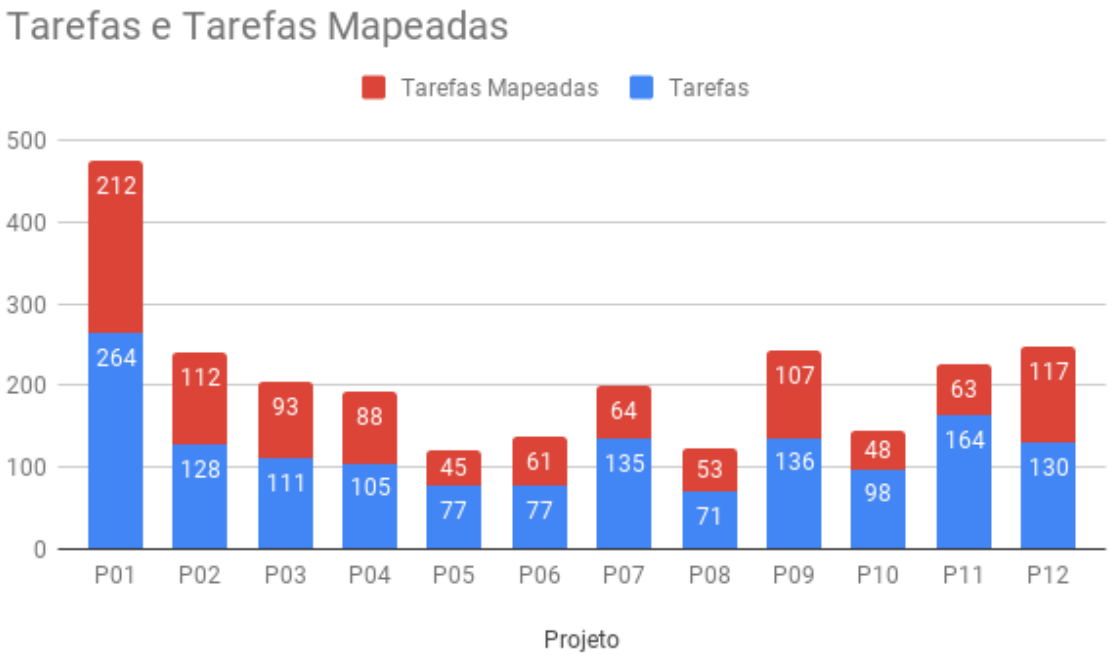
Figura 5.1: Tags dos projetos.



Fonte: próprio autor.

têm-se os perfis de cada projeto, segmentados de acordo com as tarefas que os compõem. A geração do gráfico foi realizada de forma análoga ao gráfico da Figura 5.1. Do total de 1.496 tarefas, foram mapeadas com sucesso 1.063. Desta forma, o modelo se mostrou capaz de mapear em média 70% das tarefas da base de dados. Dos 30% não cobertos, estão tarefas que representam correções de *bugs*, tarefas de teste, tarefas de estudo, dentre outras que originalmente não fazem parte da proposta de padronização e reúso do modelo. Na Figura 5.3 são exibidas as tarefas mapeadas para cada projeto.

Figura 5.2: Quantidade de tarefa originais e tarefas mapeadas pelo modelo.

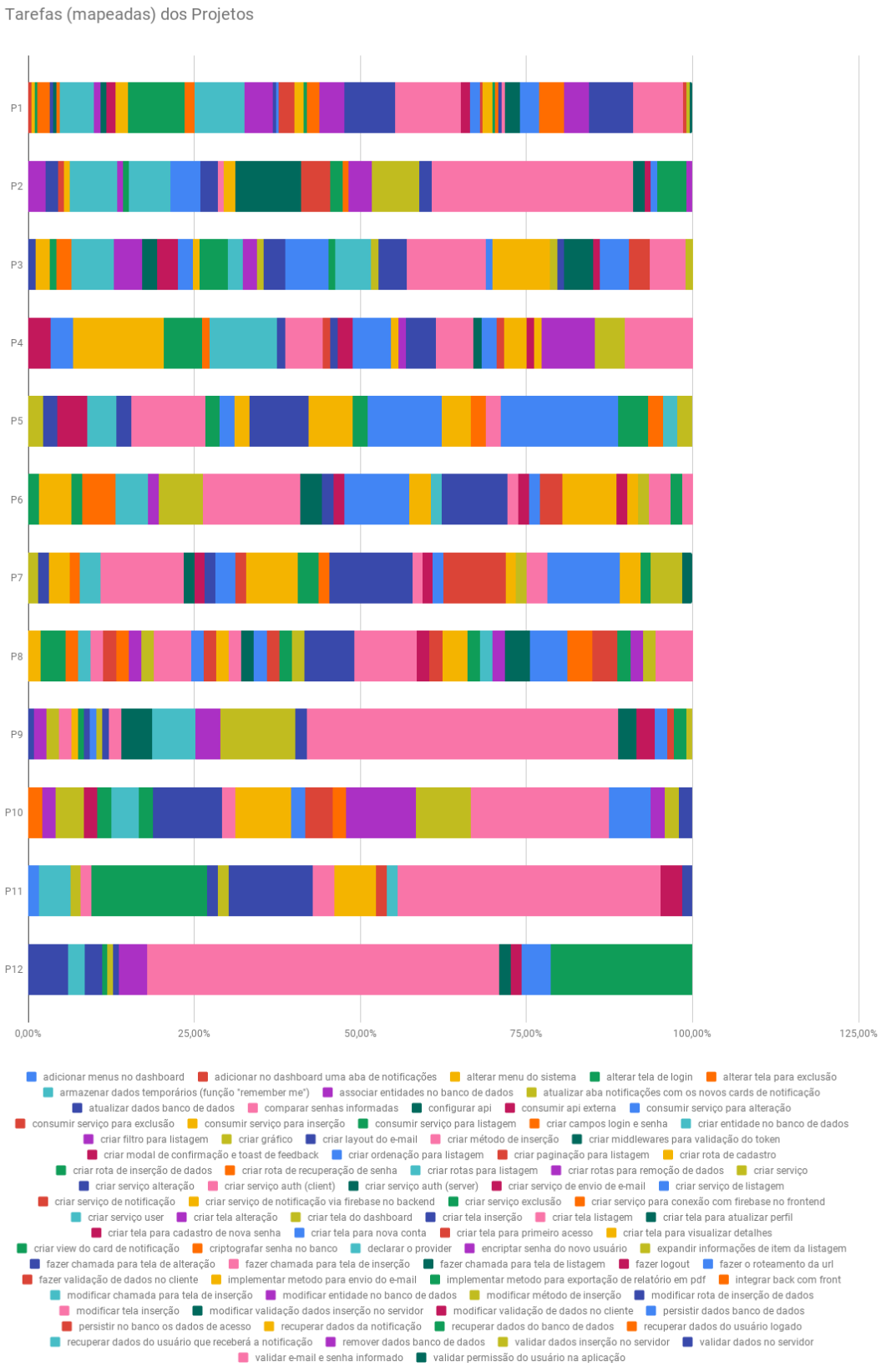


Fonte: próprio autor.

5.2 Validação da Abordagem

O processo de validação consistiu na execução da abordagem proposta, em diferentes cenários criados com os dados da base montada (após aplicação do modelo de especificação de tarefas), possuindo dois objetivos principais: (i) a determinação da melhor configuração dos parâmetros para operacionalização da abordagem e (ii) a verificação das regras de negócio e funções de *fitness* propostas.

Figura 5.3: Tarefas mapeadas dos projetos.



Fonte: próprio autor.

A determinação da melhor configuração dos parâmetros da abordagem reside na escolha da métrica de distância, utilizada para o cálculo de similaridade entre vetores (Capítulo 4, Seção 4.2.3.3) e na escolha dos parâmetros do Algoritmo Genético da abordagem. Em relação à métrica de distância, foram testadas as distâncias Euclidiana, de *Manhattan*, de *Canberra* e de *Chebyshev*. As duas primeiras são notoriamente as métricas mais utilizadas na literatura, apresentando resultados significativos nos trabalhos utilizados. As duas últimas, foram utilizadas de forma complementar, com o intuito de experimentar opções menos usuais. As implementações das métricas são provenientes da biblioteca *Apache Commons Mathematics*[1] que é uma ferramenta formada por componentes matemáticos e estatísticos, escrita em linguagem Java. Em relação ao AG, utilizou-se a implementação fornecida pela biblioteca *Jenetics*[2] que é uma ferramenta avançada para implementação de AGs, Algoritmos Evolutivos e Programação Genética, escrita em Java. É importante salientar que o desempenho dos AGs depende diretamente da escolha de seus parâmetros, entretanto, não há uma configuração padrão que consiga fornecer o melhor resultado independente do domínio do problema. Na literatura é possível encontrar recomendações de valores que servem como pontos de partida, porém a configuração ideal apenas é determinada mediante experimentação. Na Tabela 5.1 estão os fatores e os respectivos níveis escolhidos para experimentação.

A abordagem proposta utiliza uma codificação de permutação para a qual são definidos métodos de cruzamento (*Partially Mapped Crossover*) e mutação (*Swap Mutator*) específicos, portanto, não sendo possível experimentar outras opções. No que diz respeito às taxas de mutação e cruzamento, optou-se por partir de valores que são referências na literatura. Por exemplo, em geral recomenda-se que a taxa de mutação seja baixa, em torno de 5%, pois um valor alto aumenta a probabilidade de se procurar por mais áreas no espaço de busca, no entanto, impede a população de convergir para a solução ótima. No caso da taxa de cruzamento, geralmente opta-se por um valor em torno de 70%, a fim de garantir maior diversidade para a população.

A escolha dos métodos de seleção ocorreu em razão de serem os principais métodos utilizados na literatura. Optou-se também por experimentar a abordagem com e sem Elitismo, sendo foi mantida somente a melhor solução em cada geração para evitar convergência pre-

[1] http://commons.apache.org/proper/commons-math/

[2] http://jenetics.io/

Tabela 5.1: Fatores e níveis utilizados.

Fator	**Níveis**			
Métrica de Distância	Canberra	Chebyshev	Euclideana	Manhattan
Tamanho da População	100	500	1000	
Método de Cruzamento	Partially Mapped Crossover			
Taxa de Cruzamento	50%	70%	90%	
Método de Mutação	Swap Mutator			
Taxa de Mutação	5%	10%	20%	
Método de Seleção	Classificação Linear	Roleta	Seleção Estocástica Universal	Torneio
Seleção dos Sobreviventes	Com Elitismo	Sem Elitismo		
Gerações Estáveis	5	10		

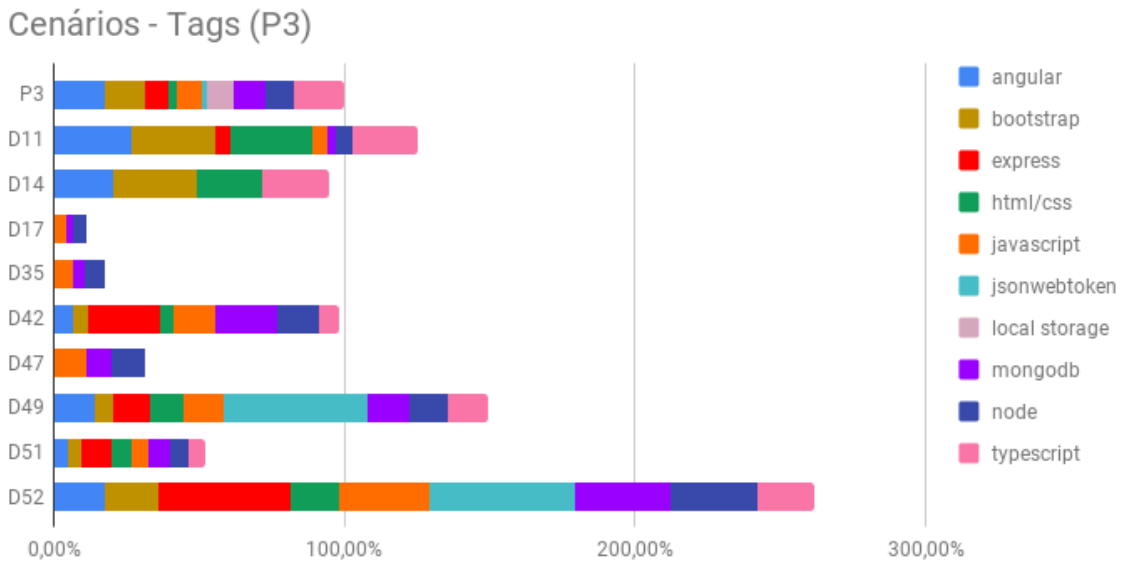
matura. Foram testadas populações de tamanho 100, 500 e 1000, pois em testes preliminares (informais) percebeu-se que populações com tamanhos próximos de 500 forneciam bons resultados, portanto, utilizou-se tal valor, bem como valores extremos inferiores (100) e superiores (1000). Por fim, o critério de parada foi o número de gerações estáveis, ou seja, o número máximo de gerações em que não há evolução em relação ao melhor valor de *fitness* previamente registrado.

5.2.1 Cenários de Validação

Foram criados 13 cenários com dados reais de projetos e desenvolvedores, com o intuito de determinar a melhor configuração dos parâmetros para operacionalização da abordagem e verificação das regras de negócio e funções de *fitness* propostas. Neste sentindo, foram selecionados quatro projetos ($P1$,$P3$,$P4$ E $P5$) da base, sendo dois deles ($P1$ e $P3$) da plataforma web e os demais ($P4$ e $P5$) da plataforma móvel. No projeto $P1$ trabalharam os desenvolvedores $D11$, $D49$, $D14$, $D42$, $D51$, $D52$ e no projeto $P4$ os desenvolvedores $D17$, $D35$, $D47$. Os dados de $P1$ e $P4$ foram utilizados para gerar os perfis dos desenvolvedores

e os dados de $P3$ e $P5$ para gerar os perfis dos projetos alvos. É importante destacar que ao definir um projeto como alvo, os dados deste projeto não são utilizados para gerar perfis de desenvolvedores, para evitar que a abordagem utilize os dados de teste para treinamento. Por exemplo, se um desenvolvedor trabalhou em dois projetos, PA e PB, e PB for definido como alvo em algum cenário, o perfil deste desenvolvedor será gerado apenas com os dados de PA, sendo toda sua participação em PB ignorada.

Figura 5.4: Tags do projeto P3 e desenvolvedores.

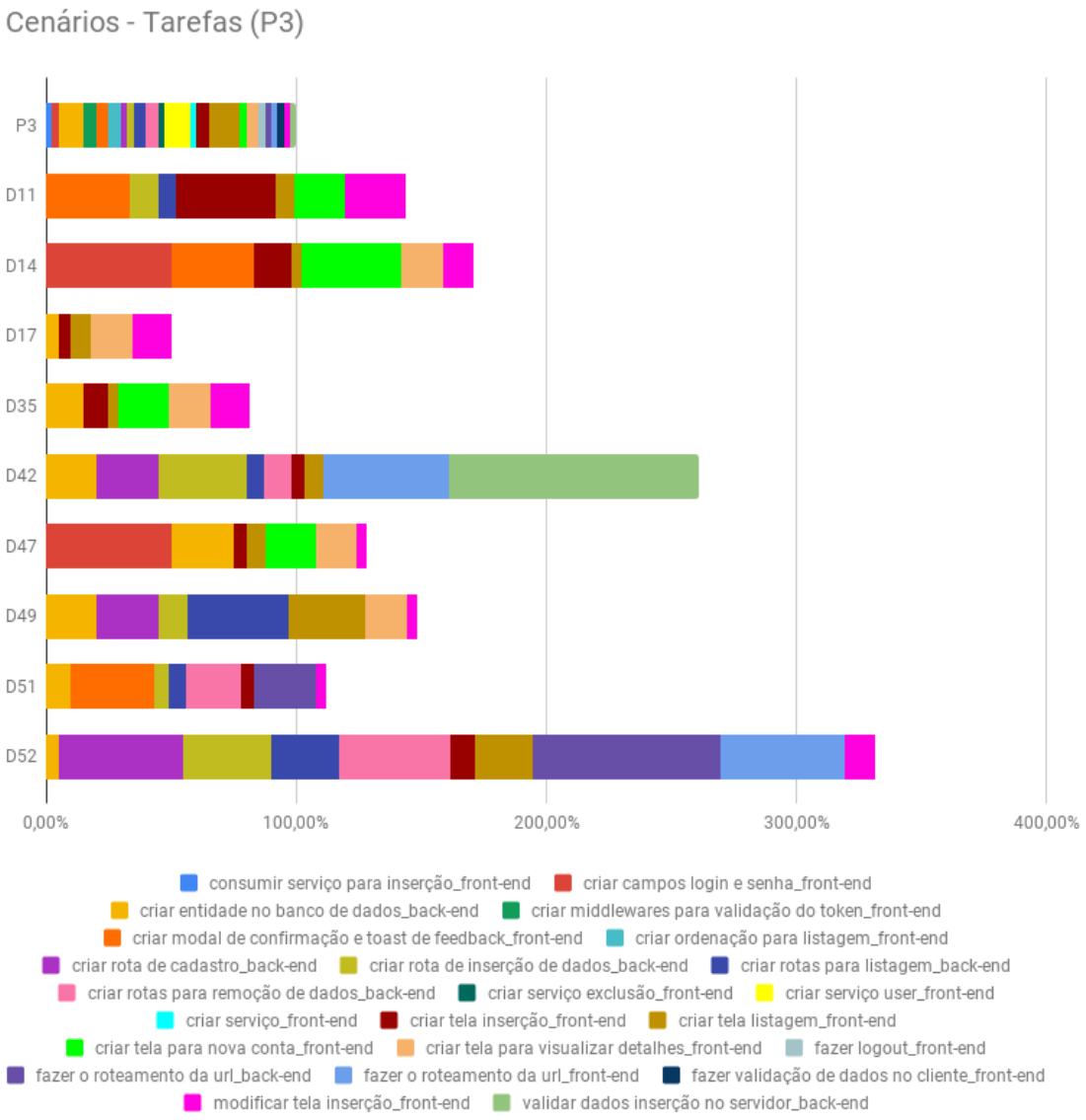


Fonte: próprio autor.

Nas Figuras 5.4, 5.5, 5.6 e 5.7, são apresentados os perfis dos projetos alvos e dos desenvolvedores disponíveis para alocação, de acordo com seus respectivos conhecimentos (*tags*) e habilidades (tarefas).

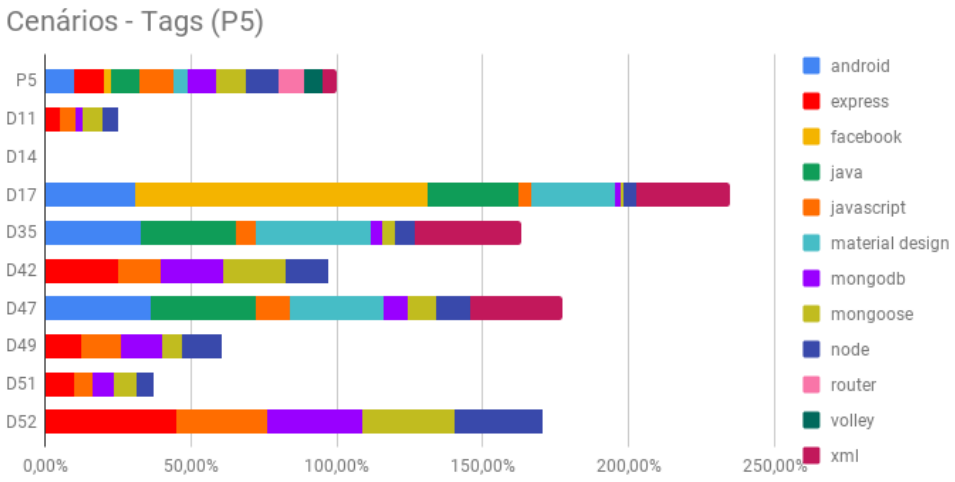
Em todos os cenários experimentados os dados utilizados para treinamento são dos projetos $P1$ e $P4$, ou seja, as tarefas destes projetos originaram os perfis dos desenvolvedores disponíveis para alocação ($D11$, $D49$, $D14$, $D42$, $D51$, $D52$, $D17$, $D35$, $D47$). As diferenças entre os cenários consistem principalmente nas regras de negócio utilizadas, nos projetos alvos, nos tamanhos das equipes a serem formadas e nas saídas esperadas. Para os cenários 01V, 02V, 03V, 04V e 05V a abordagem foi executada utilizando-se a regra de negócio *Alocação Individual* e sua respectiva função de *fitness*; para os cenários 05V, 06V e 07V utilizou-se a regra *Alocação de Especialistas* e sua respectiva função de *fitness*; para os ce-

Figura 5.5: Tarefas do projeto P3 e desenvolvedores.



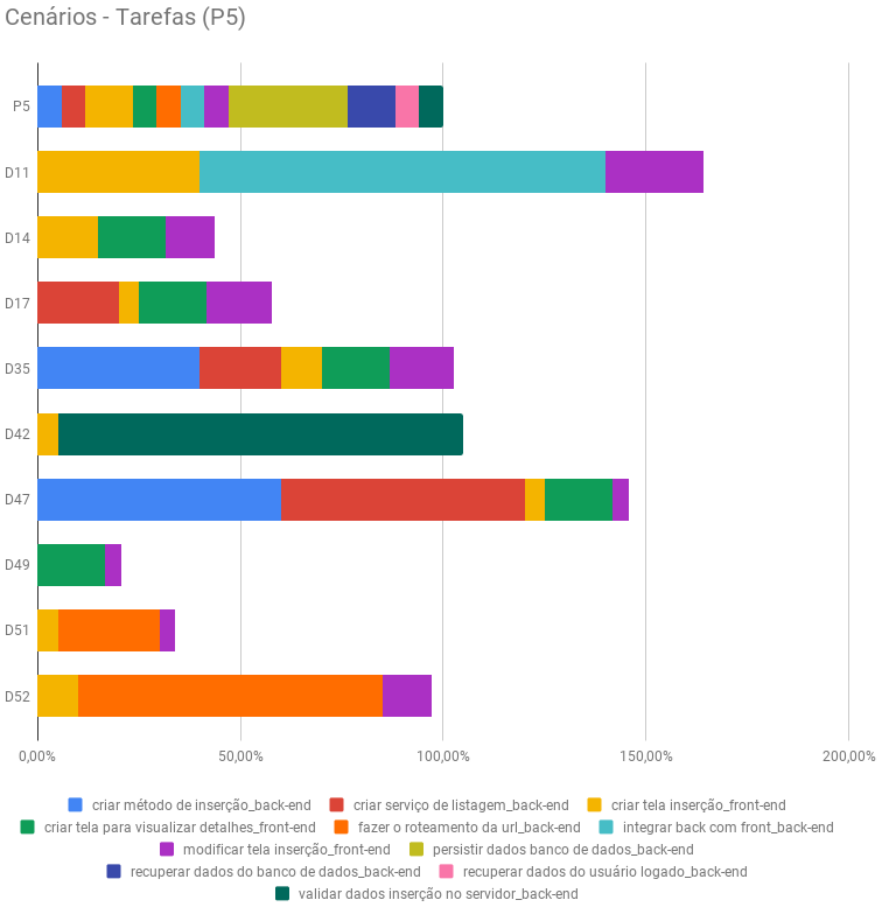
Fonte: próprio autor.

Figura 5.6: Tags do projeto P5 e desenvolvedores.



Fonte: próprio autor.

Figura 5.7: Tarefas do projeto P5 e desenvolvedores.



Fonte: próprio autor.

nários 08V, 09V e 10V aplicou-se a regra *Alocação Colaborativa* e sua respectiva função de *fitness*; e por fim, para os cenários 11V, 12V e 13V foi utilizada a regra *Alocação Complementar* e sua respectiva função de *fitness*. As características de cada cenário são detalhadas nas subseções seguintes.

5.2.1.1 Cenário 01V - Alocação Individual

Neste cenário, foram definidos ambos os projetos $P3$ e $P5$ como alvos, com equipes de tamanhos seis e três, respectivamente. Este cenário representa uma situação em que o gestor deseja formar equipes cujos membros possuam (individualmente) o maior número possível das competências necessárias em cada projeto alvo. Deste modo, a abordagem proposta foi executada com a regra de negócio *Alocação Individual*.

Os projetos $P1$ e $P3$ são ambos da plataforma web e possuem demandas semelhantes, portanto, a saída esperada para este cenário é que os desenvolvedores que trabalharam em $P1$ ($D11$, $D49$, $D14$, $D42$, $D51$, $D52$) sejam selecionados para formar a equipe de $P3$. De maneira análoga, espera-se que os desenvolvedores que trabalharam em $P4$ ($D17$, $D35$, $D47$) sejam alocados para o projeto $P5$.

5.2.1.2 Cenário 02V - Alocação Individual

Este cenário é semelhante ao anterior, porém o tamanho da equipe do projeto alvo $P3$ foi reduzido, a fim de que sejam alocados apenas os três melhores desenvolvedores e os três restantes, que também participaram de $P1$, não sejam direcionados a nenhuma equipe. Ao analisar o gráfico da Figura 5.4, percebe-se que $D11$, $D49$ e $D52$ possuem (individualmente) mais *tags* e em maior porcentagem, relacionadas ao projeto alvo.

Em relação às tarefas (Figura 5.4), $D14$, $D42$ e $D52$ se sobressaem em porcentagem total, porém $D14$ e $D42$ implementaram tarefas de menor relevância. Por exemplo, $D42$ realizou muitas tarefas de *Fazer o roteamento da url (back-end)* e *Validar dados inserção no servidor (back-end)*, as quais têm demanda reduzida para o projeto alvo. Por outro lado, $D11$ e $D49$ implementaram tarefas mais relevantes para $P3$, sendo assim, ao considerar o conjunto total das competências (conhecimentos e habilidades), os mesmos têm vantagem sobre $D14$ e $D42$. Diante disto, a saída esperada para este cenário é uma equipe formada por $D11$, $D49$ e $D52$ para $P3$ e $D17$, $D35$, $D47$ para $P5$.

5.2.1.3 Cenário 03V - Alocação Individual

Neste cenário apenas o projeto $P3$ foi definido como alvo, buscando formar uma equipe com seis integrantes. O objetivo deste cenário é verificar se a abordagem consegue alocar apenas os desenvolvedores que trabalharam no projeto $P1$, por isso, a saída esperada é $D11$, $D14$, $D42$, $D49$, $D51$ e $D52$.

5.2.1.4 Cenário 04V - Alocação Individual

O projeto $P5$, com uma equipe de tamanho três, foi definido como alvo neste cenário, porém a equipe esperada não é a que trabalhou em $P4$. Ao observar as Figuras 5.6 e 5.7, nota-se que parte dos desenvolvedores possuem conhecimentos e habilidades em comum. Ao analisar o gráfico (Figura 5.6), percebe-se que embora não tenha trabalhado em um projeto móvel $D52$ possui vasto conhecimento sobre diversas tecnologias demandadas por $P5$, chegando a ter porcentagens totais semelhantes a $D17$, $D35$, $D47$. No caso de $D17$, apesar de possuir o maior conhecimento sobre Facebook, tal tecnologia possui demanda reduzida para $P5$. Ao verificar o gráfico da Figura 5.7, nota-se que $D52$ supera $D17$, sendo assim, no geral $D52$ é uma melhor opção. Portanto, a saída esperada para este cenário é uma equipe formada por $D47$, $D35$ e $D52$.

5.2.1.5 Cenário 05V - Alocação Individual e de Especialistas

Neste cenário, foram definidos ambos os projetos $P3$ e $P5$ como alvos, com equipes de três integrantes para cada um. Para $P3$ aplicou-se a regra *Alocação Individual*. Para $P5$, utilizou-se a regra *Alocação de Especialista*, a qual representa a situação em que o gestor deseja formar equipes especialistas, ou seja, selecionar membros com ênfase em conhecimentos específicos. Neste caso, as tarefas têm peso zero e estabeleceu-se peso de cinco vezes maior para cada uma das *tags Javascript*, *Node*, *MongoDB* e *Express*, pois elas representam tecnologias em comum aos projetos alvos. Este peso (cinco) foi escolhido por representar, em média, o número de *tags* de uma tarefa na base, portanto, ao dar ênfase a uma tecnologia ela terá peso superior às demais (somadas). Neste cenário, espera-se que para o projeto $P3$ sejam alocados os desenvolvedores $D11$, $D14$ e $D52$, pois eles possuem conhecimentos mais diversificados e em maior quantidade. Para $P5$, espera-se que $D42$, $D49$ e $D47$ sejam

escolhidos, pois durante o processo de formação das equipes seus conhecimentos nas *tags* especificadas serão cinco vezes mais valorizados.

5.2.1.6 Cenário 06V - Alocação de Especialistas

Neste cenário, $P5$ é o projeto alvo, com equipe de tamanho três e tarefas com peso zero. Aplicou-se a regra *Alocação de Especialista* com o peso de cinco vezes nas mesmas *tags* do cenário 05V. De maneira análoga ao que ocorre no cenário anterior (05V), apesar de os desenvolvedores $D17$, $D35$, $D47$ já terem trabalhado em projeto móvel ($P1$), não possuem grandes conhecimentos nas tecnologias enfatizadas, portanto, ao analisar os gráficos da Figura 5.6 percebe-se que $D52$, $D42$ e $D49$ são melhores opções uma vez que seus conhecimentos acerca destas tecnologias serão incrementados durante o processo de formação da equipe. Assim, a saída esperada são os desenvolvedores $D52$, $D42$ e $D49$.

5.2.1.7 Cenário 07V - Alocação de Especialistas

$P3$ foi definido como projeto alvo, com equipe de tamanho três e tarefas com peso zero. Neste cenário, as *tags Angular*, *Bootstrap*, *HTML/CSS* e *Typescript*, que representam tecnologias tipicamente de *fron-end*, foram enfatizadas com peso cinco vezes maior que o padrão. De acordo com a análise do gráfico (Figura 5.4), ao dar ênfase a estas tecnologias espera-se que os desenvolvedores $D11$, $D14$ e $D52$ formem a equipe do projeto alvo.

5.2.1.8 Cenário 08V - Alocação Colaborativa

Neste cenário, $P3$ é o projeto alvo, buscando-se uma equipe com três integrantes. Utilizou-se a regra *Alocação Colaborativa* para formar equipes que, como um todo, possam maximizar as competências necessárias ao projeto alvo. Na prática, tenta-se garantir que, juntos, os conhecimentos e habilidades dos membros possam atender a maior demanda possível de $P3$.

Ao analisar o gráfico da Figura 5.4, percebe-se que $D11$, $D14$, e $D52$ formam a melhor combinação de conhecimentos para o cenário. Por exemplo, para as *tags Angular*, *Bootstrap*, *HTML/CSS* e *Typescript* os três desenvolvedores fornecem, em média, os maiores valores. Para outras *tags*, $D52$ compensa a falta de conhecimento dos outros dois. $D49$ se destaca em porcentagem total, mas é o melhor apenas em *JsonWebToken* (tecnologia com baixa

demanda para $P3$) e por isso ele tem menor vantagem na *Alocação Colaborativa*. Assim, se fosse considerada apenas a combinação dos conhecimentos, $D11$, $D14$, e $D52$ seriam a melhor opção para o projeto.

De acordo com o gráfico da Figura 5.5, verifica-se que $D52$ é a primeira escolha para a equipe, pois ao combinar suas tarefas com as de outros membros as habilidades das equipes aumentarão. Entretanto, a escolha para preenchimento das vagas remanescentes não é tão clara, uma vez há pelo menos quatro desenvolvedores que se destacam em relação às tarefas. Por exemplo, $D11$ se sobressai em *Criar modal de confirmação e toast de feedback (front-end)*, *Criar tela inserção (front-end)* e *Modificar tela inserção (front-end)*; $D14$ se destaca em *Criar campos login e senha (front-end)*, *Criar modal de confirmação e toast de feedback (front-end)*, *Criar modal de confirmação e toast de feedback (front-end)* e *Criar tela para visualizar detalhes (front-end)*; $D42$ evidencia-se em *Criar entidade no banco de dados (back-end)*, *Criar rota de inserção de dados (back-end)*, *Fazer o roteamento da url (fron-end)* e *Validar dados inserção no servidor (back-end)*; $D49$ se destaca em *Criar entidade no banco de dados (back-end)*, *Criar rotas para listagem (back-end)*, *Criar tela listagem (front-end)* e *Criar tela para visualizar detalhes (front-end)*. Entretanto, $D11$ e $D49$ se sobressaem em tarefas que são mais demandadas por $P3$, e por isso, se fosse considerada apenas a combinação das habilidades, as melhores opções seriam $D11$, $D49$ e $D52$.

Ao se considerar ambos os conhecimentos e habilidades, a saída esperada para este cenário é dada por $D11$, $D49$ e $D52$, pois, no geral, $D49$ aumenta mais as competências da equipe do que $D14$.

5.2.1.9 Cenário 09V - Alocação Colaborativa

Este cenário tem as mesmas características do cenário 08V, com exceção de que as tarefas têm peso zero. Esta condição foi experimentada para verificar se a saída esperada é condizente com a análise feita em relação ao cenário anterior, quando considerados apenas os conhecimentos dos desenvolvedores. Portanto, a saída esperada é que a equipe seja formada pelos desenvolvedores $D11$, $D14$ e $D52$.

5.2.1.10 Cenário 10V - Alocação Colaborativa

Este cenário tem como projeto alvo $P5$, com equipe de tamanho três. Ao analisar os gráficos das Figuras 5.6 e 5.7, percebe-se que os desenvolvedores $D17$, $D35$, $D47$ representam uma melhor combinação de conhecimentos e habilidades necessários ao projeto alvo.

5.2.1.11 Cenário 11V - Alocação Complementar

Este cenário tem as mesmas características do cenário 08V, porém fixando-se os desenvolvedores $D11$ e $D52$, isto é, considerando-os como integrantes prévios da equipe. Neste caso, utilizou-se a regra *Alocação complementar* para representar a situação em que o gestor deseja selecionar novos membros que possam complementar as competências pré-existentes. Neste cenário, a saída esperada são os desenvolvedores $D11$, $D49$ e $D52$, os quais como um todo correspondem à equipe mais adequada. Com este resultado, objetiva-se, também, confirmar $D49$ como melhor opção para o cenário.

5.2.1.12 Cenário 12V - Alocação Complementar

No cenário 09V, a saída esperada é $D11$, $D14$ e $D52$ para o projeto $P3$, porém tal formação apenas só deve ser obtida ao zerar o peso das tarefas, caso contrário a saída deveria ser $D11$, $D49$ e $D52$ (cenário 08V). Nos cenários 08V e 09V, $D11$ e $D52$ são consideradas as duas primeiras melhores opções para compor as equipes, sendo $D14$ e $D48$ as opções seguintes, dependendo da configuração utilizada. Neste cenário, o objetivo é fixar $D11$, porém sem zerar as tarefas, com o intuito de confirmar as análises feitas nos cenários mencionados.

5.2.1.13 Cenário 13 - Alocação Complementar

Neste cenário, $P5$ é projeto alvo, com equipe de tamanho três. Sem definir nenhum desenvolvedor fixo, a saída deveria ser igual a do cenário 10V, pois as regras *Alocação Colaborativa* e *Complementar* utilizam a mesma função de *fitness*. Entretanto, optou-se por fixar $D52$, para verificar se $D35$ e $D47$ são escolhidos para complementar a equipe, uma vez que estas são consideradas as duas melhores opções para o projeto alvo. Sendo assim, a saída esperada é uma equipe formada pelos desenvolvedores $D35$, $D47$ e $D52$.

Tabela 5.2: Saídas esperadas para cada cenário.

Cenário	Saída Esperada
Cenário 01V	P3 = {D11,D14,D42,D49,D51,D52} e P5 = {D17,D35,D47}
Cenário 02V	P3 = [D11,D49 e D52} e P5 = {D17,D35,D47}
Cenário 03V	P3 = {D11,D14,D42,D49,D51,D52}
Cenário 04V	P5 = {D35,D47,D52}
Cenário 05V	P3 = {D11,D14,D52} e P5 = {D42,D47,D49}
Cenário 06V	P5 = {D42,D49,D52}
Cenário 07V	P3 = {D11,D14,D52}
Cenário 08V	P3 = {D11,D49,D52]
Cenário 09V	P3 = {D11,D14,D52}
Cenário 10V	P5 = {D17,D35,D47}
Cenário 11V	P3 = {D11,D49,D52]
Cenário 12V	P3 = {D11,D14,D52}
Cenário 13V	P5 = {D17,D35,D47}

5.2.2 Execução da Abordagem nos Cenários de Validação

As saídas esperadas para os cenário estão listadas na Tabela 5.2. Para cada cenário, executou-se a abordagem com 1.728 configurações diferentes, decorrentes dos fatores e níveis estabelecidos (Tabela 5.1). Optou-se por repetir a execução de cada configuração por 30 vezes, resultando um total de 51.840 execuções por cenário.

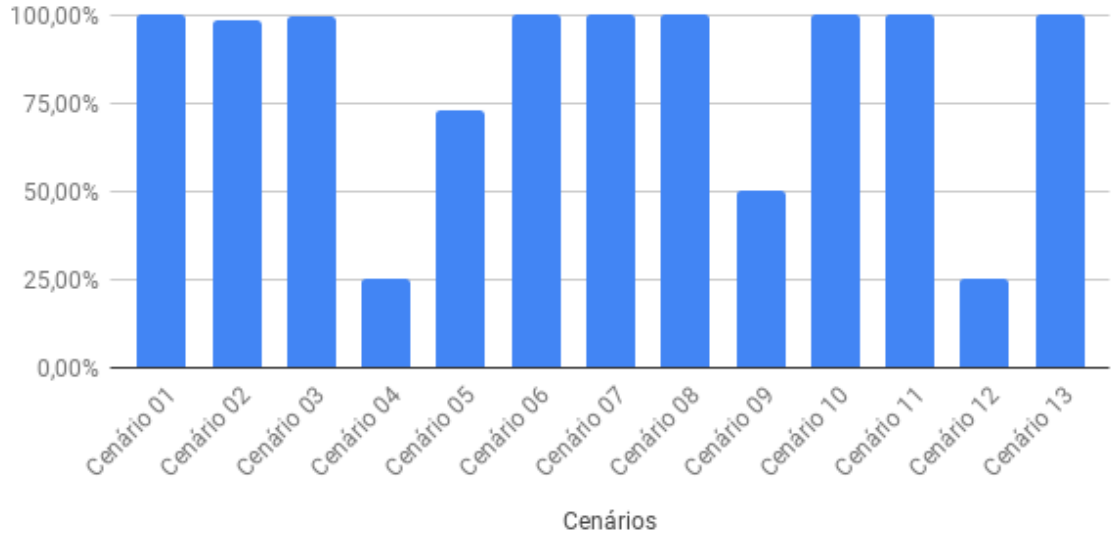
Umas das métricas mais comumente utilizadas para avaliação quantitativa das abordagens para formação de equipes é Precisão, a qual é indicada para mensurar a acurácia das recomendações fornecidas. Neste contexto, tal métrica é representada pela Equação 5.1 e seu valor é dado pela divisão do número de membros sugeridos pela abordagem e considerados relevantes sobre o número de membros sugeridos pela abordagem. Tradicionalmente, ao se utilizar Precisão também se calcula a Revocação dos resultados, porém o número máximo de membros sugeridos é sempre igual ao número máximo de membros relevantes, portanto, não há necessidade de utilização desta métrica.

$$Precisão = \frac{|\{Membros\ relevantes\} \cap \{Membros\ sugeridos\}|}{|\{Membros\ sugeridos\}|} \tag{5.1}$$

Na Figura 5.8, é apresentado um gráfico com a porcentagem de configurações que obti-

veram média de 100% de Precisão nas 30 repetições realizadas. Nota-se que para a maioria dos cenários as configurações consegue alcançar a saída esperada em quase totalidade, porém nos cenários 04V, 05V, 09V e 12V os valores são mais reduzidos. No cenário 04V, as configurações que obtiveram sucesso utilizaram a distância de *Manhattan* como base para o cálculo da similaridade entre vetores. As demais utilizaram as distâncias Euclidiana, de *Canberra* e de *Chebyshev*, divergindo principalmente ao sugerirem $D17$ no lugar de $D52$ para o projeto alvo $P5$. Entretanto, conforme análise descrita na Seção 5.2.1.4, $D52$ representa melhor opção. No cenário 05V, as principais divergências foram registradas por configurações que utilizavam distância de *Chebyshev*, pois nestes casos $D52$ foi recomendado para o projeto $P5$, $D49$ para $P3$ e $D51$ no lugar de $D47$. No cenário 09V, houve discrepância nas configurações que utilizam distância Euclidiana, pois ao invés de $D14$ sugeriu-se D49. No cenário 12V, de forma análoga ao que ocorreu nos resultados do cenário 04V, as configurações divergiram principalmente em relação à métrica de distância, sendo novamente *Manhattan* a mais bem sucedida. A principal diferença registrada foi a sugestão de $D49$ no lugar de $D11$.

Figura 5.8: Porcentagem das configuração que obtiveram média de Precisão igual a 100%.



Fonte: próprio autor.

Uma vez que a porcentagem de configurações que obtiveram 100% de Precisão foi alta, chegando à totalidade em alguns cenário, fez se necessário utilizar uma segunda métrica para ranquear as configurações a fim de escolher a melhor. Desta forma, escolheu-se a média de

tempo de execução (segundos) das configurações como métrica de desempate para ordená-las. A escolha da melhor configuração ocorreu da seguinte forma:

1. As configurações foram filtradas de acordo com o valor de Precisão, ou seja, todas aquelas que não alcançaram 100% foram descartadas;
2. As configurações foram ordenadas de acordo com a média de tempo de execução;
3. Foram escolhidas as k melhores configurações de cada cenário, com base no *ranking* estabelecido previamente;
4. Dentre as k configurações escolhidas previamente, selecionou-se a configuração final com base na ocorrência obtida nos cenários, isto é, escolheu-se aquela que esteve presente em mais cenários.

Inicialmente, pretendia-se selecionar a melhor configuração de cada cenário e em seguida escolher a configuração final com base na maior ocorrência, porém, observou-se que os tempos de execução eram valores muito próximos, não sendo possível escolher as melhores configurações somente com base na comparação simples das médias. Desta forma, foi necessário realizar uma análise estatística pra determinar qual o valor ideal de k, ou seja, as melhores configurações de cada cenário, uma vez que em um mesmo cenário seria possível encontrar configurações estatisticamente iguais em relação ao tempo de execução. A subseção a seguir detalha o processo de análise estatística que resultou na escolha de $k = 20$.

5.2.3 Análise Estatística para Determinação de k

Para determinar o valor de k é necessário definir o tipo de método (paramétrico ou não paramétrico) a ser utilizados para descobrir se existem diferenças entre as médias dos tempos de execução das configurações em cada cenário. Testes paramétricos exigem, dentre outros pré-requisitos, que as amostras sejam provenientes de populações com distribuição Normal. Por outro lado, os testes não paramétricos não demandam normalidade e são indicados para amostras pequenas, porém apresentam menor poder estatístico.

A verificação da normalidade dos dados foi realizada com auxílio do teste de *Shapiro-Wilk* [78], considerado um dos testes de maior poder estatístico para tal finalidade, o qual é capaz de verificar se a distribuição de probabilidade associada aos dados pode ser aproximada pela distribuição normal. Este teste possui as seguintes hipóteses:

- H_0: a amostra provém de uma população que possui distribuição Normal;
- H_1: a amostra não provém de uma população que possui distribuição Normal.

Na Figura 5.9 estão os resultados dos testes de *Shapiro-Wilk*. Para melhor formatação e visualização os dados foram apresentados em notação científica. Em verde, são destacados os p-valores menores que $0,05$ e em vermelho os maiores ou iguais a tal valor. Quando o p-valor é menor que $0,05$ significa que a hipótese nula (H_0) deve ser rejeitada e, neste caso, não é possível afirmar, com 95% de confiança, que a amostra provém de uma população que possui distribuição Normal.

Figura 5.9: Resultados dos teste de normalidade.

	Cenário 01	Cenário 02	Cenário 03	Cenário 04	Cenário 05	Cenário 06	Cenário 07	Cenário 08	Cenário 09	Cenário 10	Cenário 11	Cenário 12	Cenário 13
Rank 01	1,24E-03	1,51E-01	1,49E-03	2,45E-06	3,21E-01	3,83E-05	3,84E-04	1,00E-02	2,06E-03	8,25E-05	1,38E-02	1,39E-01	3,69E-01
Rank 02	2,36E-05	6,67E-01	5,50E-05	1,70E-05	1,99E-01	1,05E-08	2,61E-05	2,40E-03	1,96E-04	2,68E-04	7,95E-01	2,35E-03	6,49E-04
Rank 03	2,98E-07	2,50E-01	8,98E-05	1,34E-05	3,84E-02	2,61E-09	9,08E-05	7,36E-06	2,15E-05	1,53E-03	7,92E-01	8,03E-01	2,92E-03
Rank 04	2,28E-10	5,33E-01	2,09E-06	6,04E-06	3,78E-01	1,19E-08	5,78E-05	3,56E-07	1,16E-05	6,36E-04	5,06E-01	8,06E-03	3,31E-02
Rank 05	4,74E-05	5,49E-01	1,81E-06	2,42E-05	2,41E-01	2,48E-05	1,72E-06	1,10E-06	1,48E-06	4,75E-06	5,56E-01	9,87E-03	1,57E-03
Rank 06	6,56E-05	8,74E-02	6,69E-09	2,04E-04	1,65E-01	2,86E-07	1,58E-04	5,26E-08	3,61E-09	2,24E-04	5,31E-01	4,71E-06	1,18E-01
Rank 07	3,49E-06	4,48E-01	4,12E-07	3,74E-04	7,06E-01	7,40E-06	2,84E-07	1,96E-08	3,99E-05	6,37E-04	5,29E-01	5,43E-01	3,41E-04
Rank 08	9,93E-07	1,90E-02	1,64E-05	6,01E-09	2,71E-01	3,32E-05	5,61E-05	3,80E-04	4,91E-07	3,88E-06	8,61E-02	4,55E-03	7,57E-06
Rank 09	4,33E-08	6,47E-01	3,89E-05	6,99E-06	9,49E-01	1,39E-06	4,38E-06	1,36E-06	5,59E-05	1,70E-05	4,82E-01	1,74E-01	7,56E-01
Rank 10	3,31E-06	1,51E-02	5,75E-05	4,65E-06	7,11E-01	1,12E-06	1,44E-08	7,80E-05	9,64E-09	3,83E-05	3,17E-02	3,04E-02	4,24E-03
Rank 11	4,01E-07	1,13E-01	3,83E-04	1,98E-05	1,40E-02	1,33E-08	2,74E-02	4,91E-07	1,34E-03	6,11E-05	6,13E-01	5,11E-01	6,13E-01
Rank 12	6,76E-05	8,01E-02	6,44E-06	1,93E-04	6,07E-01	1,17E-04	1,73E-06	9,54E-07	1,33E-05	2,33E-06	3,11E-01	9,06E-01	4,90E-04
Rank 13	2,07E-06	7,41E-01	3,34E-07	5,94E-08	3,32E-01	7,71E-05	2,98E-05	1,09E-06	1,34E-04	5,33E-05	8,47E-01	1,45E-04	1,48E-03
Rank 14	3,82E-06	4,03E-01	1,39E-07	1,35E-10	8,68E-02	1,43E-07	2,28E-02	2,67E-03	3,32E-05	1,98E-04	3,03E-03	5,76E-01	2,65E-02
Rank 15	8,94E-07	3,89E-01	4,03E-04	3,16E-05	6,32E-02	1,44E-06	4,58E-07	5,20E-05	9,04E-05	5,52E-06	4,71E-01	1,52E-01	1,92E-03
Rank 16	5,01E-05	5,52E-02	1,64E-06	9,45E-08	9,63E-02	5,90E-08	7,82E-05	1,25E-05	7,30E-05	4,93E-08	9,88E-01	1,16E-01	3,31E-01
Rank 17	1,22E-05	1,63E-01	7,03E-05	9,51E-07	5,48E-02	5,58E-05	1,55E-07	6,94E-05	8,54E-06	2,10E-04	2,29E-01	5,01E-01	1,85E-05
Rank 18	1,39E-08	1,62E-01	1,68E-09	8,35E-09	1,42E-01	1,29E-04	2,29E-04	1,78E-04	2,31E-06	5,59E-06	7,34E-01	4,92E-05	2,36E-04
Rank 19	3,40E-05	3,36E-01	6,19E-07	1,10E-07	5,24E-01	2,71E-06	3,35E-06	2,18E-05	6,64E-07	7,16E-07	1,00E-09	3,47E-03	1,74E-02
Rank 20	1,11E-06	7,15E-01	1,14E-07	6,67E-08	7,31E-01	4,75E-07	2,28E-06	5,99E-06	6,76E-05	5,25E-07	1,73E-03	6,02E-02	6,17E-04
Rank 21	1,14E-05	4,57E-03	1,10E-02	6,85E-09	3,68E-02	6,18E-10	2,76E-05	2,77E-07	7,75E-06	1,68E-06	3,75E-01	2,42E-06	9,32E-01

Fonte: próprio autor.

A escolha do teste paramétrico para determinar se existem diferenças entre as médias dos tempos de execução em cada cenário está condicionada aos p-valores dos testes de *Shapiro* serem maiores ou iguais que $0,05$, isto é, todas as amostras devem ser originadas de populações com distribuições Normais. Entretanto, em nenhum cenário todas as amostram atendem tal condição, por isso, escolheu-se o teste não paramétrico de *Kruskal Wallis* [65]. Este teste permite a comparação de duas ou mais amostras (grupos) independentes, considerando apenas a suposição de que as variáveis sejam dos tipos contínuas ou ordinais. As hipóteses do teste são descritas a seguir:

- H_0: não há diferença entre as média dos grupos;
- H_1: pelo menos um dos grupos apresenta uma média diferente dos demais.

Na Figura 5.10 são apresentados os resultados dos testes de *Kruskall*. A representação das cores é análoga a da Figura 5.9. Em alguns cenários nota-se que desde a comparação das duas primeiras amostras os resultados já apontam que elas são estatisticamente diferentes, ou seja, apresentam p-valores menores que $0,05$, possibilitando a rejeição da hipótese nula, com 95%. Entretanto, no cenário 13V só é possível determinar tal diferença ao se comparar as 21 primeiras amostras, pois até a 20º não havia diferença estatística entre as médias comparadas. Desta forma, para manter as mesmas condições de escolha da configuração final em todos os cenários, optou-se por $k = 20$, ou seja, foram selecionadas as 20 primeiras configurações em cada um dos cenários, apesar de nos demais a diferença estatística já ter sido alcançada com um k menor.

Figura 5.10: Resultados dos testes de Kruskal Wallis.

	Cenário 01	Cenário 02	Cenário 03	Cenário 04	Cenário 05	Cenário 06	Cenário 07	Cenário 08	Cenário 09	Cenário 10	Cenário 11	Cenário 12	Cenário 13
02 Grupos	2,25E-01	6,66E-02	3,45E-01	3,91E-01	1,48E-09	5,35E-01	1,56E-01	3,37E-01	1,03E-03	5,26E-04	7,06E-01	1,21E-04	8,71E-01
03 Grupos	6,10E-03	2,09E-01	7,90E-02	1,93E-01	6,35E-11	7,03E-02	3,59E-01	4,85E-01	2,30E-03	3,75E-05	8,99E-01	4,87E-09	5,72E-01
04 Grupos	4,10E-03	5,04E-02	2,85E-02	2,85E-01	4,55E-11	1,96E-05	3,37E-01	6,14E-01	1,77E-04	4,82E-05	9,78E-01	4,21E-11	6,91E-01
05 Grupos	1,79E-04	8,85E-05	2,23E-02	2,99E-03	9,47E-12	2,17E-08	2,74E-01	5,15E-01	2,52E-04	8,01E-05	9,82E-01	6,21E-13	7,77E-01
06 Grupos	2,97E-06	5,30E-10	3,42E-04	1,32E-04	2,04E-12	2,30E-08	2,10E-01	5,16E-01	3,75E-04	4,61E-05	9,53E-01	3,18E-14	7,70E-01
07 Grupos	7,43E-06	1,73E-13	5,15E-04	7,25E-05	1,32E-12	9,62E-08	2,05E-01	4,46E-01	1,69E-07	1,24E-04	8,93E-01	8,82E-16	8,23E-01
08 Grupos	4,60E-07	3,54E-16	8,34E-05	1,00E-07	2,25E-12	9,59E-07	7,06E-03	4,63E-01	5,85E-07	2,33E-04	4,54E-01	2,25E-16	8,92E-01
09 Grupos	1,05E-07	2,60E-18	2,44E-04	5,08E-07	6,34E-13	2,54E-06	1,35E-03	4,75E-01	4,20E-07	2,28E-05	3,40E-01	1,90E-17	7,90E-01
10 Grupos	5,04E-08	1,59E-19	2,15E-05	5,65E-09	1,57E-13	2,64E-06	7,20E-06	4,74E-01	2,33E-07	1,14E-06	2,21E-01	1,81E-18	6,54E-01
11 Grupos	1,48E-08	3,65E-20	5,33E-05	1,19E-10	5,86E-13	1,09E-06	3,41E-06	7,18E-02	1,56E-08	5,49E-07	1,59E-01	8,27E-20	4,80E-01
12 Grupos	2,81E-09	1,83E-21	7,28E-05	1,98E-11	4,65E-13	2,58E-09	9,10E-08	8,93E-02	1,98E-10	8,04E-07	1,04E-01	1,38E-20	4,71E-01
13 Grupos	2,81E-09	1,16E-24	1,09E-05	7,69E-14	4,27E-14	2,01E-08	2,33E-08	6,61E-02	4,40E-11	1,14E-06	7,42E-02	2,89E-21	3,17E-01
14 Grupos	1,36E-10	3,08E-26	2,36E-08	4,66E-16	4,06E-14	2,38E-10	1,40E-09	7,24E-03	6,54E-11	2,22E-06	4,27E-02	1,76E-22	1,75E-01
15 Grupos	3,49E-11	5,70E-28	7,51E-12	1,60E-17	5,38E-14	2,79E-10	2,97E-09	1,97E-04	1,49E-10	2,21E-07	1,76E-02	2,07E-23	7,69E-02
16 Grupos	5,15E-11	2,04E-30	1,09E-14	1,65E-20	1,11E-13	3,54E-11	3,63E-09	1,46E-04	9,97E-12	7,75E-08	1,81E-03	4,63E-24	7,29E-02
17 Grupos	3,49E-10	2,02E-31	3,36E-16	1,52E-20	3,17E-14	9,33E-11	4,88E-09	1,31E-04	8,18E-13	5,76E-08	4,01E-04	1,46E-29	1,00E-01
18 Grupos	2,79E-12	7,72E-33	2,51E-18	1,25E-21	5,92E-16	2,83E-10	2,69E-10	6,08E-05	1,03E-13	1,16E-08	2,17E-05	2,46E-33	1,21E-01
19 Grupos	7,69E-16	7,85E-34	4,91E-22	7,40E-23	1,08E-16	6,44E-11	3,80E-10	1,45E-05	1,14E-14	1,64E-08	4,26E-05	2,24E-37	1,41E-01
20 Grupos	1,21E-19	1,02E-35	1,80E-22	2,23E-23	1,86E-17	1,91E-11	3,28E-11	1,41E-05	2,82E-14	1,16E-08	2,45E-06	6,80E-41	6,32E-02
21 Grupos	7,52E-22	1,62E-36	2,10E-25	6,92E-24	1,61E-17	8,70E-13	1,13E-13	1,55E-05	7,04E-16	1,27E-09	4,91E-07	2,93E-44	1,95E-02

Fonte: próprio autor.

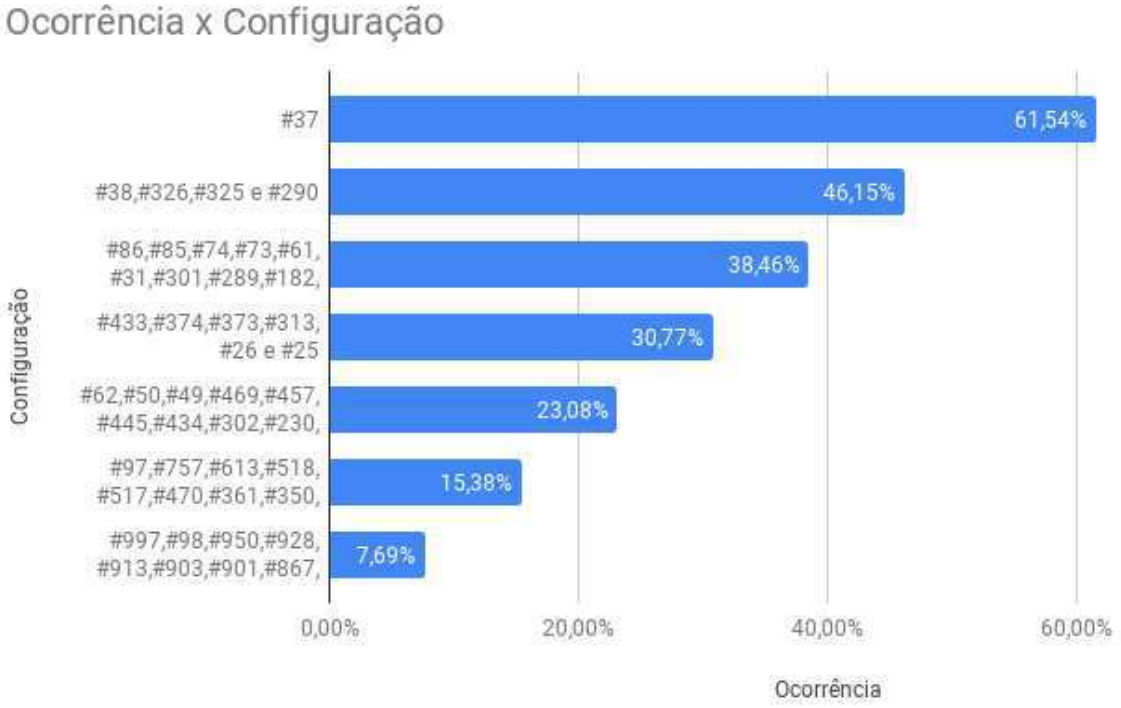
5.2.4 Escolha da Configuração Final

Após a determinação do valor de k, as 20 melhores configurações de cada cenário foram selecionadas e as ocorrências foram contabilizadas. Na Figura 5.11, observa-se o resultado desta operação. A configuração $\#37$ foi a mais recorrente em todos os cenários, apresentando valor de 61,54% sendo, portanto, escolhida como configuração final e detalhada a seguir:

- **Métrica de Distância:** Manhattan
- **Tamanho da População:** 100
- **Método de Cruzamento:** *Partially Mapped Crossover*

- **Taxa de Cruzamento:** 50%
- **Método de Mutação:** *Swap Mutator*
- **Taxa de Mutação:** 5%
- **Método de Seleção:** Torneio (com $n = 3$)
- **Seleção dos sobreviventes:** Elitismo (com $e = 1$)

Figura 5.11: Ocorrências das k melhores configurações.



Fonte: próprio autor.

5.3 Avaliação da Abordagem

Após a realização da validação da abordagem proposta, a qual resultou na definição dos parâmetros a serem utilizados na abordagem final, ocorreu o processo de avaliação. Este processo foi realizado com auxílio de quatro gestores com experiência na gestão de projetos ágeis baseados em Scrum e que trabalharam diretamente em projetos que compõem a base de dados. Nas Tabelas 5.3, 5.4, 5.5 e 5.6 é possível observar os perfis detalhados dos gestores,

com base em tempo de experiência e quantidade de projetos geridos, distribuídos por tipo de projeto.

Tabela 5.3: Perfil detalhado do gestor 01.

Tipo de projeto	Tempo de Experiência	Quantidade de Projetos
Projetos da Plataforma Web	De 2 a menos de 4 anos	De 5 a 7 projetos
Projetos da plataforma Móvel	De 2 a menos de 4 anos	De 2 a 4 projetos
Projetos baseados em Scrum	De 4 a menos 6 anos	8 ou mais projetos
Projetos em geral	De 4 a menos 6 anos	8 ou mais projetos

Tabela 5.4: Perfil detalhado do gestor 02.

Tipo de projeto	Tempo de Experiência	Quantidade de Projetos
Projetos da Plataforma Web	De 4 a menos 6 anos	8 ou mais projetos
Projetos da plataforma Móvel	Menos de 2 anos	1 projeto
Projetos baseados em Scrum	De 2 a menos de 4 anos	De 2 a 4 projetos
Projetos em geral	De 4 a menos 6 anos	8 ou mais projetos

Tabela 5.5: Perfil detalhado do gestor 03.

Tipo de projeto	Tempo de Experiência	Quantidade de Projetos
Projetos da Plataforma Web	Menos de 2 anos	8 ou mais projetos
Projetos da plataforma Móvel	Não tenho experiência	1 projeto
Projetos baseados em Scrum	Menos de 2 anos	De 2 a 4 projetos
Projetos em geral	Menos de 2 anos	8 ou mais projetos

Tabela 5.6: Perfil detalhado do gestor 04.

Tipo de projeto	Tempo de Experiência	Quantidade de Projetos
Projetos da Plataforma Web	6 anos ou mais	De 5 a 7 projetos
Projetos da plataforma Móvel	De 2 a menos de 4 anos	De 5 a 7 projetos
Projetos baseados em Scrum	6 anos ou mais	8 ou mais projetos
Projetos em geral	6 anos ou mais	8 ou mais projetos

O processo foi realizado em duas etapas, a fim de avaliar situações em que havia ou não a disputa por recursos entre os gestores. Vale salientar que para evitar viés na formação das equipes, os gestores não tinham conhecimento da identificação dos desenvolvedores. Ao saber quem são os envolvidos, os gestores poderiam deixar de considerar apenas as competências técnicas apresentadas e passar a considerar aspectos subjetivos (*soft skills*), os quais não fazem parte do escopo do trabalho.

5.3.1 Primeira Etapa da Avaliação

Na primeira etapa, os gestores deveriam (separadamente) formar equipes sem se preocupar com o compartilhamento de recursos da empresa. Para cada gestor, foram apresentados o perfil do projeto em que ele trabalhou mais recentemente e perfis de 15 desenvolvedores escolhidos aleatoriamente da base, para formar equipes com cinco integrantes cada. Os tamanhos das equipes foram escolhidos de acordo com os tamanhos originais das equipes de cada projeto. Optou-se por restringir o número de desenvolvedores disponíveis para alocação em três vezes o tamanho da equipe, para evitar criar cenários complexos, nos quais a escolha da equipe pudesse se tornar uma tarefa exaustiva para uma única pessoa.

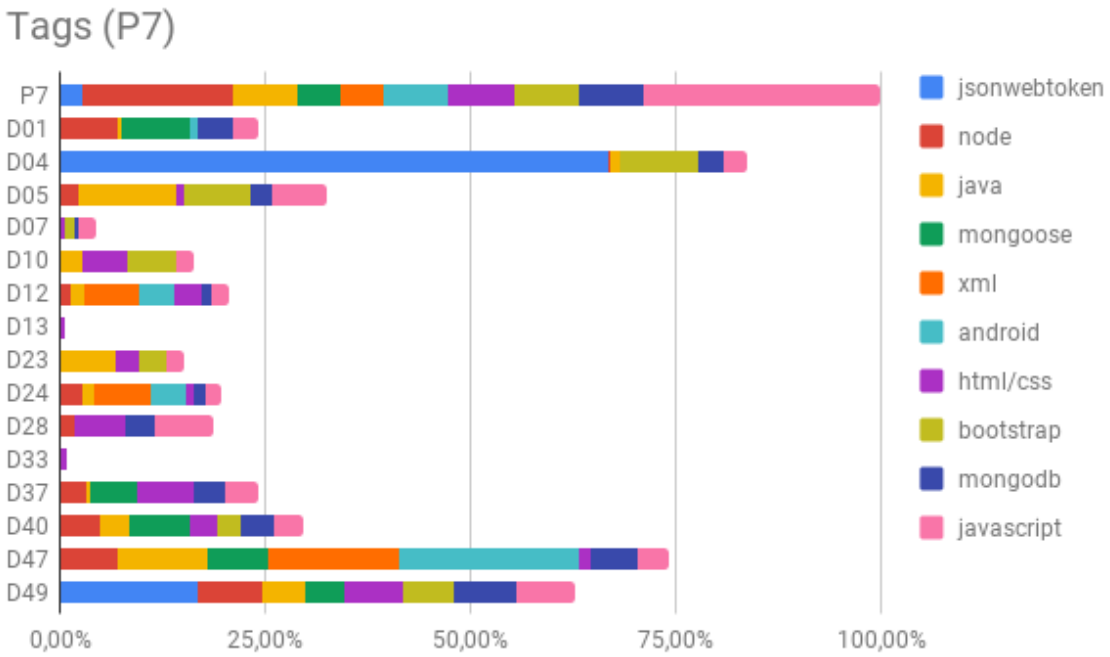
De posse dos perfis dos projetos e dos desenvolvedores, os gestores tinham que formar suas equipes, considerando cinco cenários de avaliação diferentes. No primeiro, foi solicitado que eles escolhessem suas equipes livremente, sem obedecer quaisquer regras, com o intuito de obter uma formação espontânea. No segundo, eles tinham que formar equipes seguindo a regra de *Alocação Individual*, ou seja, a mesma utilizada pela abordagem proposta. No terceiro, precisavam obedecer a regra *Alocação Colaborativa*. No quarto, a regra utilizada foi a *Alocação de Especialistas*. Neste caso, os gestores foram solicitados (previamente) a apontar quais tecnologias consideravam fundamentais aos respectivos projetos. No quinto, tiveram que seguir a regra *Alocação Complementar* e também foram previamente questionados sobre desenvolvedores que julgavam fundamentais às suas equipes, considerando os respectivos projetos alvos. Apesar de individual, a avaliação foi conduzida em paralelo com todos os gestores em uma mesma sala, na qual cada um possuía seu próprio computador. As equipes foram formadas gradativamente e a cada início de cenário a regra utilizada era explicada, sendo todas as dúvidas sanadas.

Ao final da execução dos cinco cenários, utilizou-se a métrica Precisão para avaliar a

acurácia da abordagem em relação às respostas dos gestores. Em seguida, um questionário *online* foi enviado para cada gestor. Nele solicitava-se que o gestor indicasse, para os cenários em que as respostas divergiram, qual o seu nível de concordância (aceitação) em relação a sugestão da abordagem. Foi utilizada uma escala *likert* de cinco níveis (Concordo totalmente, Concordo, Indiferente, Discordo, Discordo totalmente).

Nas Figuras 5.12 e 5.13, estão os perfis do projeto $P7$ e dos desenvolvedores entregues ao gestor 01. Na Tabela 5.7, são apresentados os resultados da avaliação realizada com este gestor. Em negrito estão destacadas as divergências entre os desenvolvedores sugeridos pela abordagem e aqueles escolhidos pelo gestor. No cenário 05A, não constam no cálculo da Precisão os desenvolvedores fixos, ou seja, aqueles que o gestor havia previamente sinalizado como integrantes da equipe (destacados em itálico).

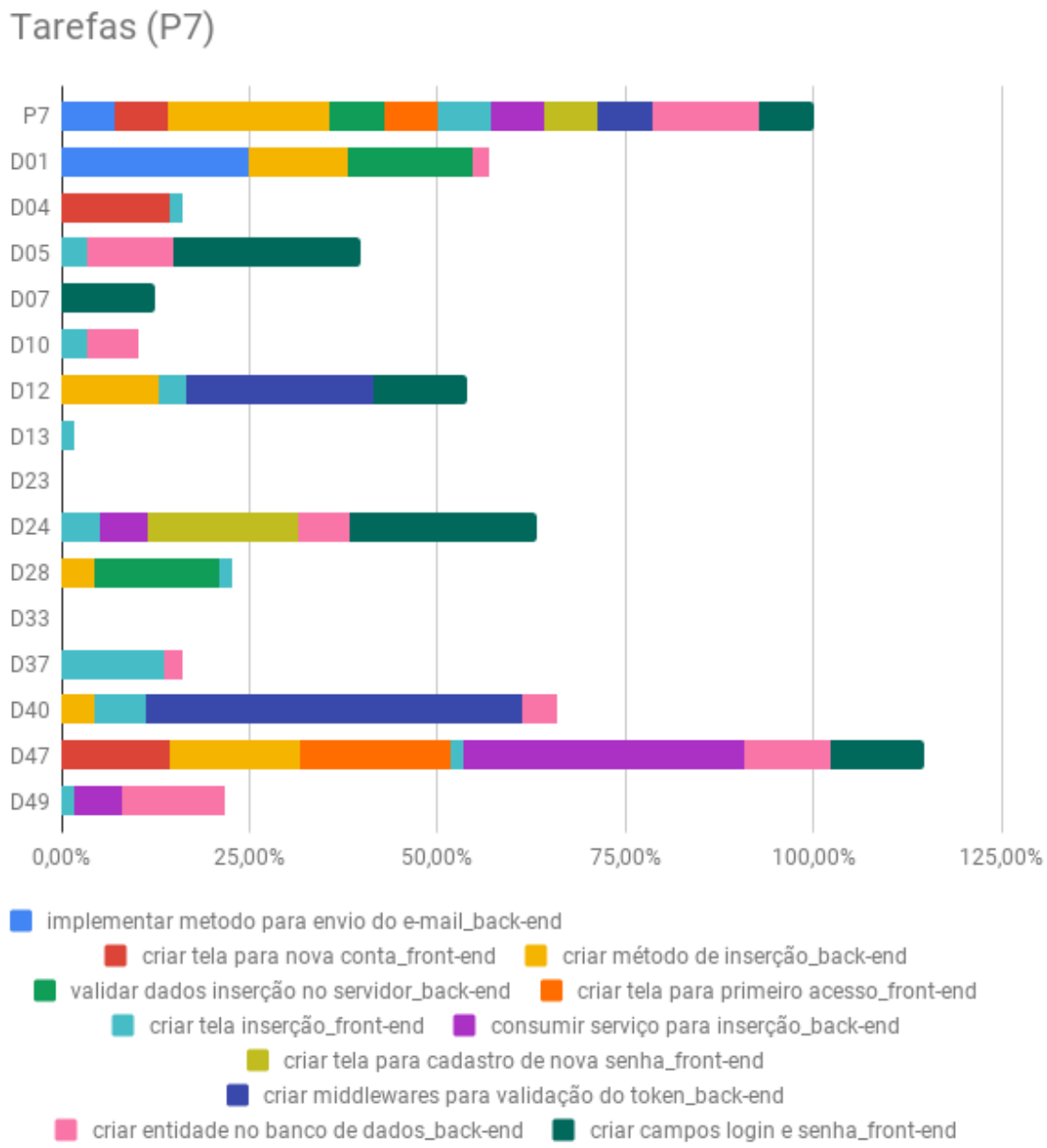
Figura 5.12: Perfis de projeto e desenvolvedores para avaliação com gestor 01 - Tags.



Fonte: próprio autor.

Em relação ao gestor 01, a média de Precisão foi de 90%, havendo divergências apenas nos cenários 03A e 04A. No cenário 03A, o gestor alegou que escolheu $D24$, porém também considerou $D05$ como opção válida e que inclusive esta teria sido uma dúvida que teve durante a escolha dos membros. Em sua resposta ao questionário, sinalizou *Concordo* em

Figura 5.13: Perfis de projeto e desenvolvedores para avaliação com gestor 01 - Tarefas.



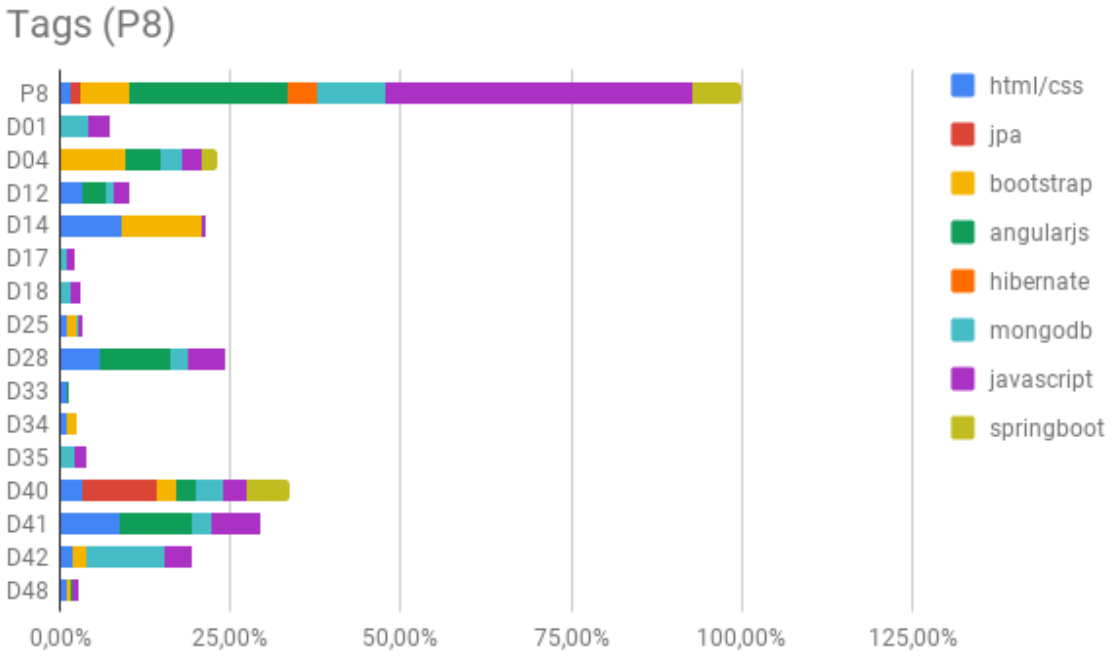
Fonte: próprio autor.

Tabela 5.7: Comparação dos resultados com o gestor 01.

Cenário	Formação do Gestor	Sugestão da Abordagem	Precisão
Cenário 01A	{D47,D49,D40,D05,D01}	-	-
Cenário 02A	{D47,D49,D40,D05,D01}	{D47,D49,D40,D05,D01}	100%
Cenário 03A	{D47,D49,D40,**D24**,D01}	{D47,D49,D40,**D05**,D01}	80%
Cenário 04A	{D47,D49,D40,D05,**D01**}	{D47,D49,D40,D05,**D23**}	80%
Cenário 05A	{*D05*,D47,D49,D40,D01}	{*D05*,D47,D49,D40,D01}	100%
	Média		90%

relação à sugestão da abordagem. No cenário 04A, o gestor também disse que concordou (*Concordo*) com a recomendação, pois a escolha de $D01$ não teve importância significativa, sendo portanto $D23$ uma opção válida.

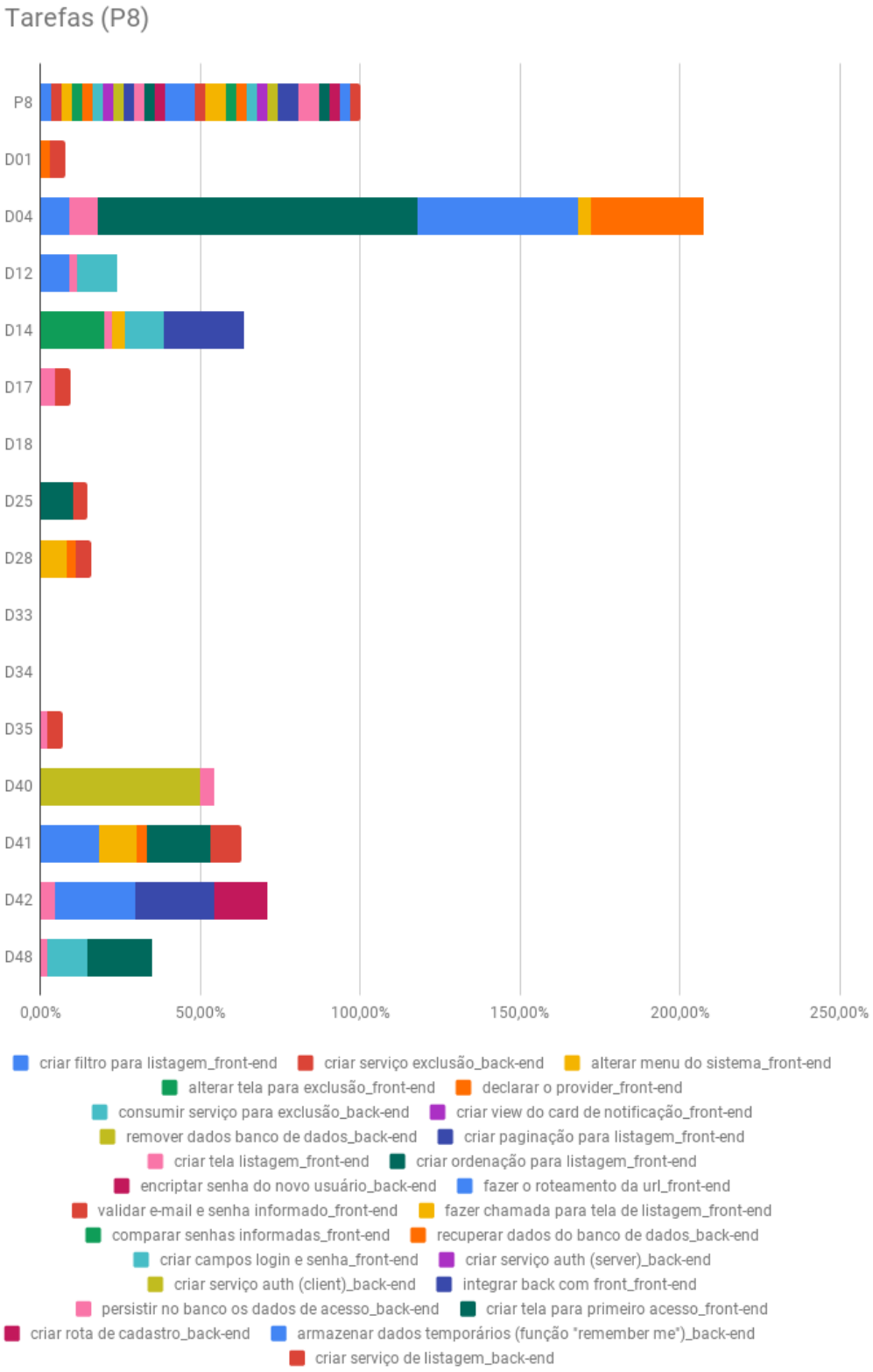
Figura 5.14: Perfis de projeto e desenvolvedores para avaliação com gestor 02 - Tags.



Fonte: próprio autor.

Nas Figuras 5.14 e 5.15, estão os perfis do projeto $P8$ e dos desenvolvedores entregues ao gestor 02. Na Tabela 5.8, estão os resultados da avaliação com este gestor. A média de Precisão foi de 95%, sendo a mais alta entre todos os gestores. A única divergência registrada foi no cenário 04A, no qual o gestou afirmou que o desenvolvedor sugerido ($D12$) tem perfil semelhante ao que ele escolheu, e por isso concordou (*Concordo*) com a recomendação

Figura 5.15: Perfis de projeto e desenvolvedores para avaliação com gestor 02 - Tarefas.



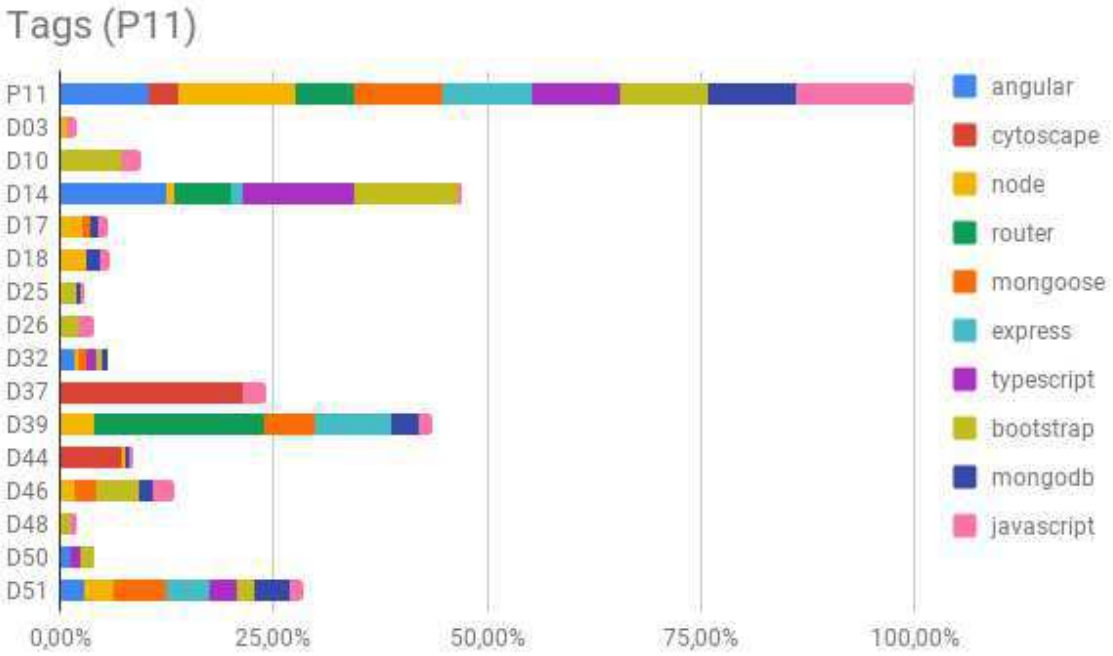
Fonte: próprio autor.

apresentada.

Tabela 5.8: Comparação dos resultados com o gestor 02.

Cenário	Formação do Gestor	Sugestão da Abordagem	Precisão
Cenário 01A	{D41,D28,D42,D04,D40}	-	-
Cenário 02A	{D41,D28,D42,D04,D40}	{D41,D28,D42,D04,D40}	100%
Cenário 03A	{D41,D28,D42,D04,D40}	{D41,D28,D42,D04,D40}	100%
Cenário 04A	{D41,D28,**D42**,D04,D40}	{D41,D28,**D12**,D04,D40}	80%
Cenário 05A	{*D28*,*D41*,D42,D04,D40}	{*D28*,*D41*,D42,D04,D40}	100%
	Média		95%

Figura 5.16: Perfis de projeto e desenvolvedores para avaliação com gestor 03 - Tags.

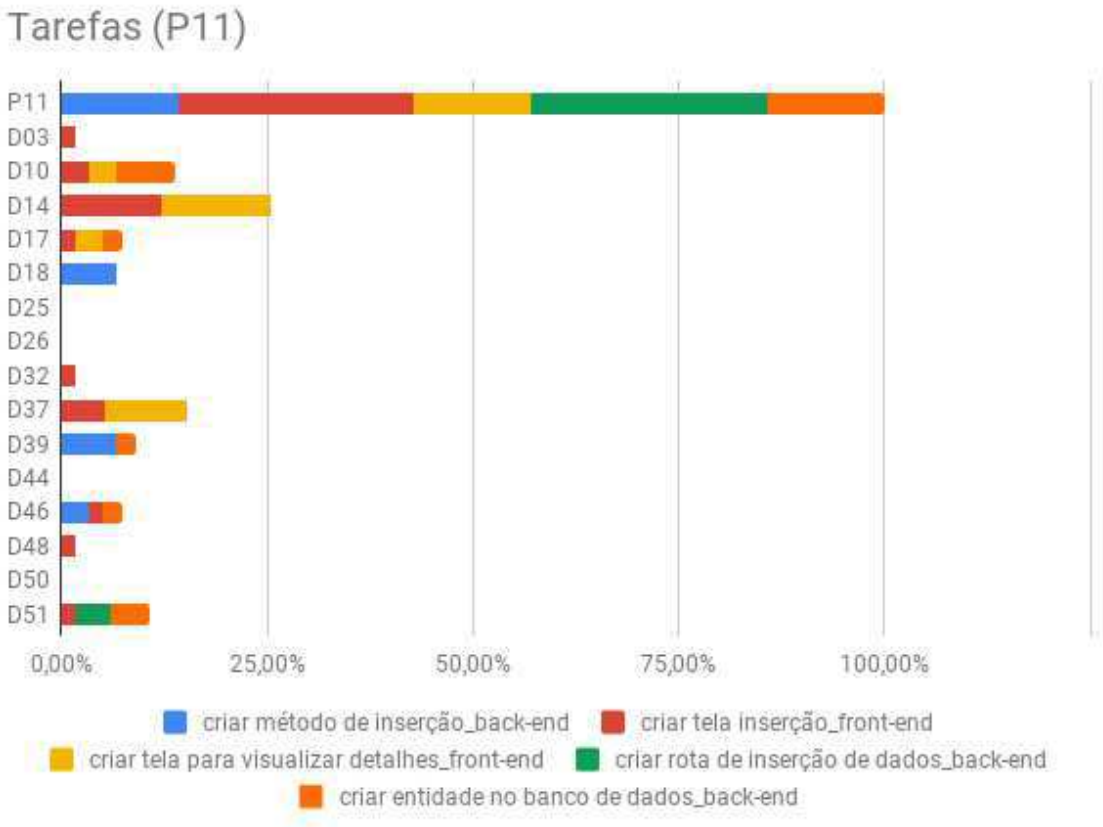


Fonte: próprio autor.

Nas Figuras 5.16 e 5.17, estão os perfis do projeto $P11$ e dos desenvolvedores entregues aos gestor 03. Na Tabela 5.9, podem ser observados os resultados da avaliação com este gestor, sendo a média de Precisão igual a 84%. No cenário 02A, o gestor afirmou que discordou (*Discordo*) da sugestão apresentada. Ele disse que considerando apenas os conhecimentos necessários para o projeto alvo, aceitaria a recomendação, porém ao considerar os conhecimentos e habilidades, não trocaria, pois $D37$ é o único da equipe com conhecimento

em *Cytoscape*, uma API que foi utilizada na maior parte das tarefas do projeto. Ele complementou dizendo que $D10$ não possui conhecimentos relevantes para as principais tarefas do seu projeto. No cenário 04A, o gestor afirmou que aceitou a recomendação (*Concordo*), pois apesar da sua escolha inicial ($D46$) ter mais conhecimentos sobre as tecnologias, o desenvolvedor sugerido ($D18$) apresenta maior conhecimento nas tecnologias que são mais demandadas pelo projeto. No cenário 05A, o gestor sinalizou que também concordou (*Concordo*) com a recomendação, uma vez que $D39$ é superior em relação aos conhecimentos e $D10$ superior em relação às habilidades. Entretanto, a sugestão da abordagem o fez refletir sobre sua escolha, pois considerando a regra estabelecida para o cenário, $D39$ se enquadra melhor no perfil colaborativo, podendo proporcionar mais benefícios à equipe.

Figura 5.17: Perfis de projeto e desenvolvedores para avaliação com gestor 03 - Tarefas.



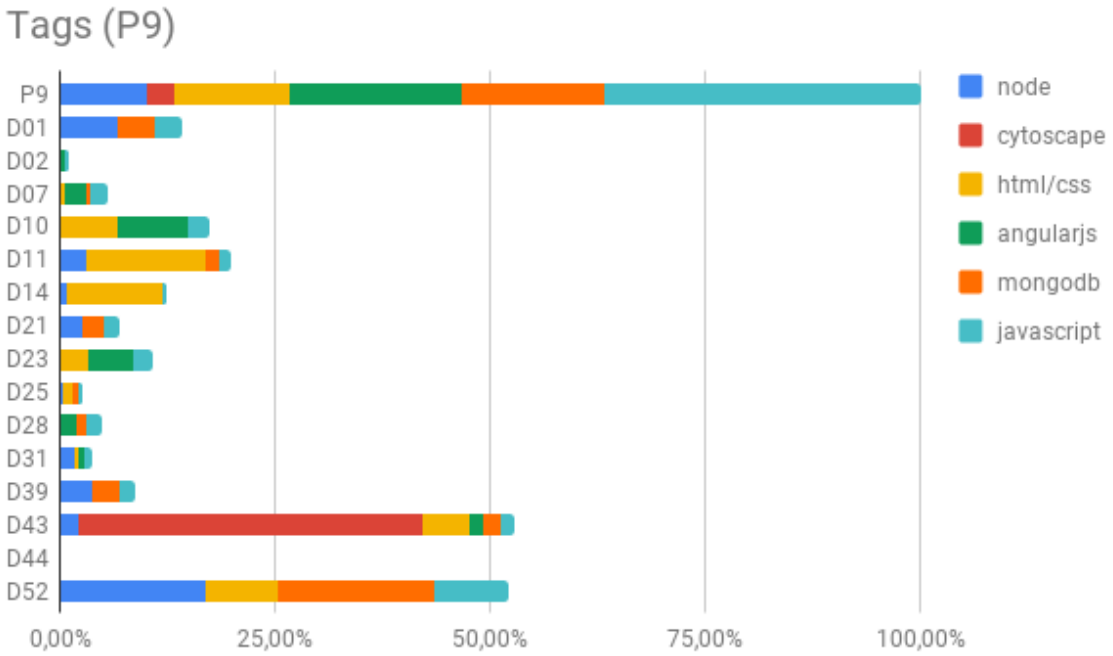
Fonte: próprio autor.

Nas Figuras 5.18 e 5.19 estão os perfis do projeto $P9$ e dos desenvolvedores entregues aos gestor 04. Na Tabela 5.10, são apresentados os resultados da avaliação com este gestor.

Tabela 5.9: Comparação dos resultados com o gestor 03.

Cenário	Formação do Gestor	Sugestão da Abordagem	Precisão
Cenário 01A	{D37,D39,D14,D03,D44}	-	-
Cenário 02A	{D14,D51,D39,**D37**,D46}	{D14,D51,D39,**D10**,D46}	80%
Cenário 03A	{D14,D51,D39,D46,D10}	{D14,D51,D39,D46,D10}	100%
Cenário 04A	{D39,D51,D14,D37,**D46**}	{D39,D51,D14,D37,**D18**}	80%
Cenário 05A	{*D37*,D46,D14,D51,**D10**}	{*D37*,D46,D14,D51,**D39**}	75%
	Média		84%

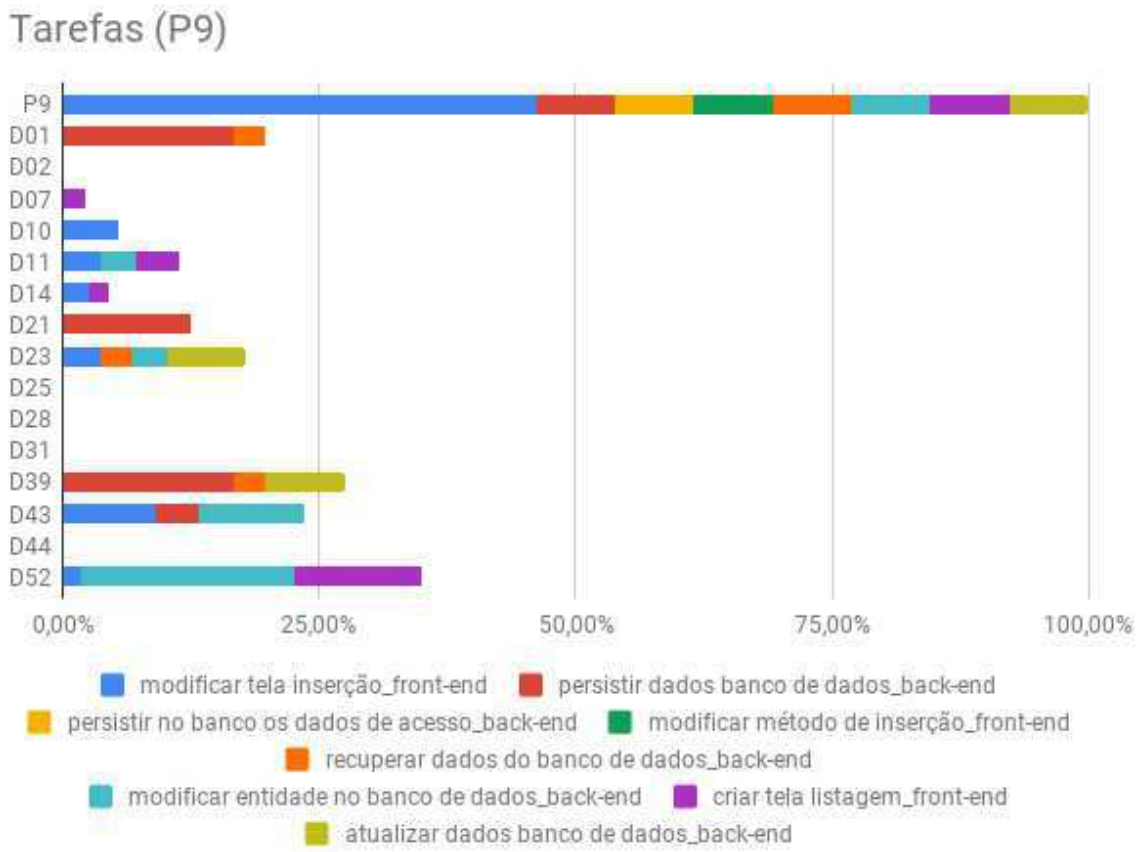
Figura 5.18: Perfis de projeto e desenvolvedores para avaliação com gestor 04 - Tags.



Fonte: próprio autor.

A média de Precisão foi de 77%, sendo a menor taxa registrada. Nos cenários 02A e 03A, o gestou utilizou a mesma justificativa e afirmou que concordou (*Concordo*) com as sugestões oferecidas, pois uma vez que $D52$ possui todas as competências necessárias ao projeto, ele selecionou $D01$ por este ter mais conhecimento em *Node* do que $D11$. $D52$ seria capaz de fazer as tarefas de *front-end*, juntamente com $D43$ e $D23$, enquanto que $D01$ faria as de *back-end*. Apesar disto, o gestor também considerou $D11$ como opção válida, pois ele também seria capaz de atender satisfatoriamente às tarefas de *back-end* com *Node* e *MongoDB*. No cenário 04A, o gestor discordou da sugestão (*Discordo*), afirmando que não

Figura 5.19: Perfis de projeto e desenvolvedores para avaliação com gestor 04 - Tarefas.



Fonte: próprio autor.

trocaria $D01$ por $D07$, pois este último apresenta menos competências e somente em tarefas de *front-end*. No cenário 05A, o gestor alegou que selecionou $D01$ devido às competências referentes ao *back-end*, mas que isso poderia ser atendido pelos desenvolvedores $D52$ e $D43$ com foco no *front-end*, mais especificamente em *Cytoscape* que foi considerada uma importante API para atender as demandas do projeto. Ao analisar todos os cenários, observa-se que o gestor 04 alocou a mesma equipe para 02A, 03A e 04A. De forma análoga, a abordagem proposta alocou equipes iguais para 02A e 03A, e com apenas uma divergência para 04A. Esse comportamento similar indica uma baixa diversidade nos perfis dos desenvolvedores disponibilizados para alocação, embora eles tenham sido escolhidos aleatoriamente.

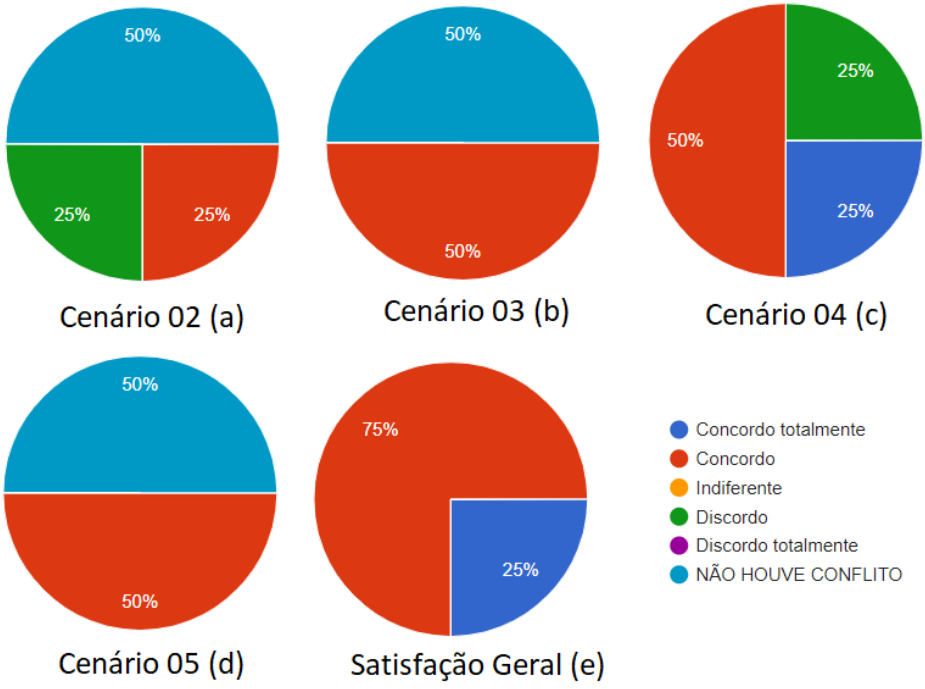
Na Figura 5.20 estão os gráficos resultantes da aplicação do questionário aos gestores. A opção *"NÃO HOUVE CONFLITO"* representa os casos em que a abordagem proposta sugeriu a mesma equipe que o gestor, ou seja, obteve Precisão de 100%. Nestes casos,

Tabela 5.10: Comparação dos resultados com o gestor 04.

Cenário	Formação do Gestor	Sugestão da Abordagem	Precisão
Cenário 01A	{D52,D10,D01,D43,D23}	-	-
Cenário 02A	{D52,D10,**D01**,D43,D23}	{D52,D10,**D11**,D43,D23}	80%
Cenário 03A	{D52,D10,**D01**,D43,D23}	{D52,D10,**D11**,D43,D23}	80%
Cenário 04A	{D52,D10,**D01**,D43,D23}	{D52,D10,**D07**,D43,D23}	80%
Cenário 05A	{*D28*,*D44*,D52,D10,**D01**}	{*D28*,*D44*,D52,D10,**D43**}	67%
	Média		77%

considerou-se que houve concordância total. Nos demais, é possível perceber que na maioria dos cenários houve alto nível de concordância, significando que mesmo quando houve divergências, os gestores tendiam a aceitar as sugestões oferecidas. No questionário foi solicitado também que os gestores indicassem seu nível de satisfação geral em relação às recomendações apresentadas pela abordagem. Na Figura 5.20(e) observa-se que todos os gestores ficaram Totalmente satisfeitos ou Satisfeitos com as recomendações, o que aponta para um forte indício de receptividade na utilização de uma ferramenta semelhante em um contexto apropriado.

Figura 5.20: Resultados da aplicação do questionário aos gestores.



Fonte: próprio autor.

O cenário 01A, no qual as equipes foram formadas livremente pelos gestores, teve como objetivo verificar com qual tipo de regra os gestores apresentavam maior semelhança. Ao analisar as Tabelas 5.7, 5.8, 5.9 e 5.10, percebeu-se que o gestor 01, ao utilizar regras *Alocação Individual* e *Complementar*, formou as mesmas equipe do cenário 01A. No caso dos gestores 02, todas as equipes formadas foram iguais, sem distinção de regras. Em relação ao gestor 03, em todos os cenários foram formadas equipes diferentes. Por fim, o gestor 04 formou as mesmas equipes, com exceção do cenário 05. Desta forma, não foi possível obter conclusão significativa em relação ao cenário 01A.

5.3.2 Segunda Etapa da Avaliação

Na segunda etapa da avaliação, os gestores foram reunidos novamente em uma sala e foram apresentados os mesmos perfis de projetos, porém com os 52 desenvolvedores disponíveis para alocação. Neste caso, os gestores precisaram formar suas respectivas equipes, com cinco membros cada, tendo que compartilhar os recursos, ou seja, um desenvolvedor não poderia fazer parte de duas equipes simultaneamente. Nas Figuras 5.21 e 5.22, estão os perfis dos projetos e dos desenvolvedores.

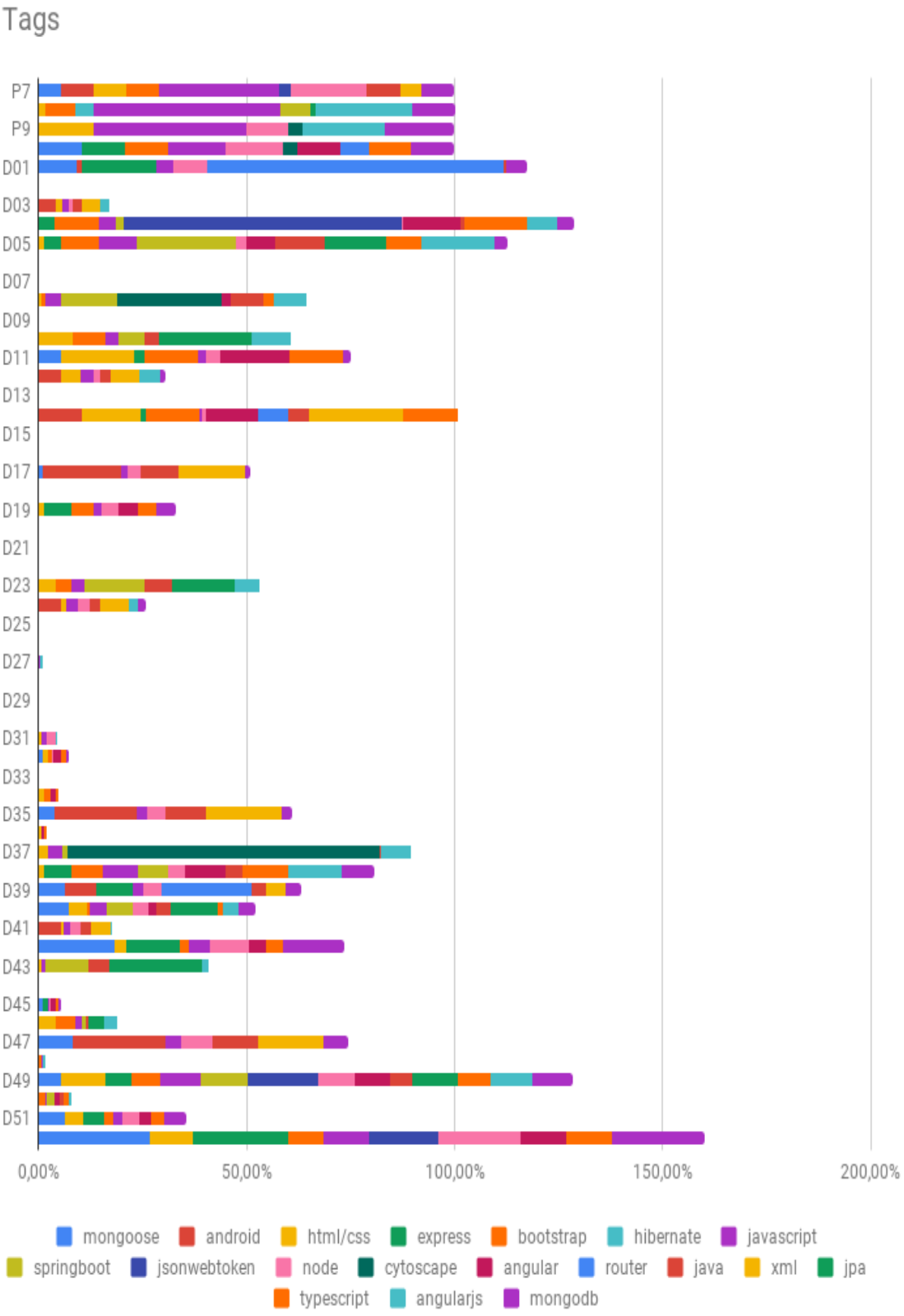
Tabela 5.11: Taxas de conflito registradas durante a formação simultânea.

			Taxas de Conflito			
Gestor	**Projeto Alvo**	**Formação**	**P7**	**P8**	**P9**	**P11**
Gestor 01	P7	{D52,D49,D11,D42,D05}	-	100%	60%	40%
Gestor 02	P8	{D52,D49,D11,D42,D05}	100%	-	60%	40%
Gestor 04	P9	{D52,D49,D14,D42,D47}	60%	60%	-	40%
Gestor 03	P11	{D52,D11,D14,D07,D37}	40%	40%	40%	-
Média			67%	67%	53%	40%

Na Tabela 5.11, observam-se as equipes formadas por cada gestor. Apesar de estarem reunidos e terem sido orientados a negociar entre si a escolha dos membros para suas respectivas equipes, nota-se que em todas as formações houve conflito, ou seja, os mesmos desenvolvedores foram alocados para diversas equipes. Uma vez que os gestores não conseguiram chegar a um consenso por conta própria, foi oferecida a solução gerada pela abordagem, a qual formou as equipes utilizando a regra *Alocação Individual*, regra padrão da abordagem.

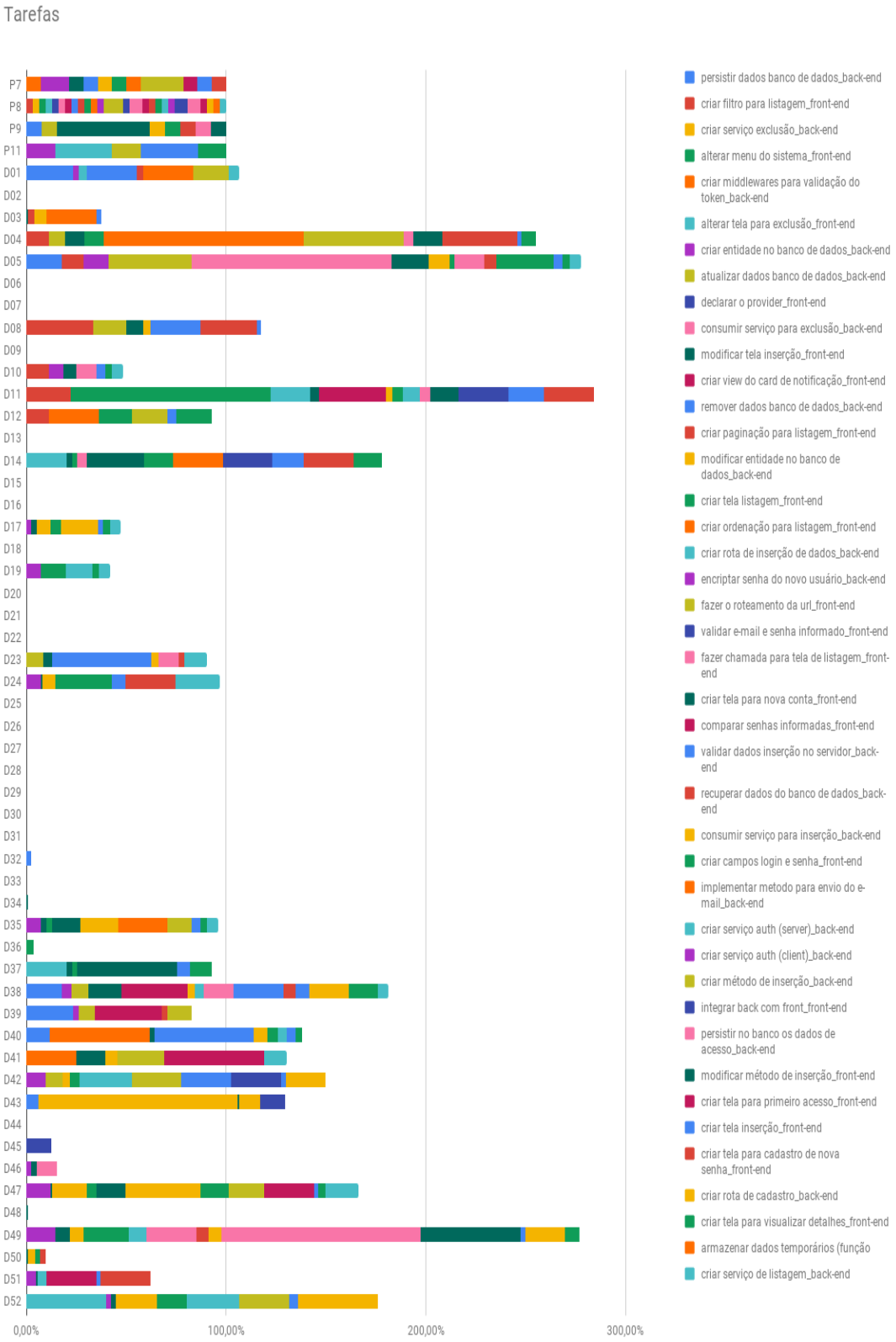
Na Tabela 5.12, é possível observar a solução oferecida pela abordagem. Cada gestor

Figura 5.21: Perfis de projeto e desenvolvedores para avaliação de todos os gestores - Tags.



Fonte: próprio autor.

Figura 5.22: Perfis de projeto e desenvolvedores para avaliação todos os gestores - Tags.



Fonte: próprio autor.

analisou a sugestão que lhe foi oferecida e após negociarem novamente entre si, chegaram a uma formação global final. A taxa de aceitação representa a porcentagem de desenvolvedores de cada equipe que foram sugeridos pela abordagem e aceitos como membros da equipe pelo gestor. No geral, houve uma média de 75% de aceitação a nível global, sendo unânime o relato dos gestores no sentido de que a solução oferecida simplificou o processo de formação das equipes e ajudou a minimizar os conflitos entre eles.

5.3.3 Ameaças à Validade

A análise de ameaças à validade da pesquisa tem como objetivo examinar a relação entre as conclusões alcançadas e a realidade, a fim de mitigar prováveis ameaças que possam afetar os resultados da pesquisa. A seguir são detalhados os principais tipos de ameaças à validade detectados de acordo com a classificação de Wohlin et al. [107].

- **Ameaça à validade interna:** a aplicação do modelo de especificação de tarefas à base de dados históricos não foi realizada em tempo real para todos os projetos, pois alguns já haviam sido finalizados. Ao aplicar o modelo pós-término de um projeto, é possível que algumas tarefas não sejam rotuladas e/ou padronizadas fielmente, pois com o passar do tempo os integrantes do projeto podem não se recordar completamente de todos os conhecimentos e habilidades utilizados durante a implementação. Para mitigar tal ameaça, solicitou-se que após os processos de Rotulação e Padronização, cada um dos membros das equipes revisasse o resultado dos colegas e em caso de divergência, toda a equipe deveria se reunir para discutir as possíveis alterações. Além disto, após aplicação do modelo, foram selecionados alguns desenvolvedores com mais tempo de experiência em projetos, para revisar o resultado final. As alterações propostas eram repassadas à equipe dos respectivos projetos e estas eram responsáveis por decidir se acatariam ou não as sugestões.

- **Ameaça à validade externa:** a base de dados históricos é composta apenas por projetos das plataformas móvel e web. Entretanto, sabe-se da existência de outras plataformas como embarcada, desktop etc, as quais podem possuir características diferentes das utilizadas, o que dificultaria a generalização dos resultados. Para mitigar tal ameaça, pretende-se continuar a construção da base, buscando a adição de projetos de

Tabela 5.12: Taxa de aceitação da solução oferecida pela abordagem proposta.

Gestor	Projeto Alvo	Solução Oferecida	Formação Final	Taxa de Aceitação
Gestor 01	P7	{D24,D47,D01,D41,D35}	{D24,D47,D01,D41,D35}	100%
Gestor 02	P8	{D04,D23,D12,D40,D37}	{D04,D23,D12,D11,D05}	60%
Gestor 04	P9	{D38,D08,D05,D49,D10}	{D38,D08,D39,D49,D10}	80%
Gestor 03	P11	{D14,D52,D42,D11,D19}	{D14,D52,D42,D07,D37}	60%
		Média		75%

diferentes plataformas, domínios e tecnologias, além da inclusão de desenvolvedores com perfis mais diversificados em relação aos conhecimentos e habilidades.

- **Ameaça à validade de construção:** a escolha da configuração final da abordagem foi resultado da avaliação das métricas de similaridade e dos parâmetros do AG. Sabe-se que existem outras métricas e parâmetros além dos escolhidos, porém a adição de mais níveis aos fatores aumentaria exponencialmente o número de configurações possíveis, tornando o experimento mais custoso. A fim de mitigar esta ameaça, optou-se por escolher as métricas e parâmetro mais recorrentes na literatura, pois, em geral, são os que fornecem os melhores resultados.

- **Ameaça à validade de conclusão:** embora a abordagem tenha sido avaliada mediante execução em diversos cenários criados a partir de dados reais e com auxílio de especialistas, é possível que os cenários definidos não reflitam as características exatas de um ambiente real. Para mitigar esta ameaça, pretende-se aprimorar a abordagem para que possa ser utilizada em ambiente real.

5.4 Considerações Finais do Capítulo

Neste capítulo, foram descritos os processos de validação e avaliação da abordagem proposta, bem como a montagem de uma base de dados históricos provenientes de projetos reais de desenvolvimento de software.

A validação foi realizada com a execução da abordagem em 13 cenários distintos, criados com dados reais provenientes da base histórica. Os resultados da validação estão de

acordo com as saídas planejadas para cada cenário, indicando que a abordagem apresentou o comportamento esperado em todas as situações.

O processo de avaliação foi realizado com o auxílio de gestores com experiência na gestão de projetos ágeis de software, sendo dividido em duas etapas: no primeiro, o objetivo consistiu em verificar a acurácia das sugestões apresentadas pela abordagem em relação às equipes escolhidas pelos gestores. Os resultados indicam que a abordagem conseguiu uma média de 86,4% de Precisão, considerando todos os gestores. Na segunda etapa, o intuito foi verificar a aceitação das sugestões apresentadas, diante de um cenário de maior complexidade, no qual os recursos tinham que ser compartilhados. Neste caso, registrou-se uma taxa de aceitação de 75% sobre as recomendações oferecidas, representando um resultado promissor. É importante ressaltar que a quantidade reduzida gestores participantes da avaliação ocorreu em função de dificuldade de se reunir, presencialmente, todos aqueles que atuaram diretamente em projetos que compõe a base de dados, uma que estes profissionais possuem compromissos de trabalho com as instituições em que atuam.

Capítulo 6

Considerações Finais

6.1 Conclusões

Neste trabalho, foi apresentada uma abordagem de apoio à decisão para formação de múltiplas equipes para projetos ágeis baseados em Scrum, a partir da realocação dos recursos humanos disponíveis na empresa, visando obter a melhor distribuição global, considerando as demandas de cada projeto e as competências dos desenvolvedores.

Foi apresentado um modelo de especificação de tarefas que possibilita a criação de perfis técnicos a partir de atributos de baixa granularidade, durante a execução dos projetos conduzidos na empresa, baseando-se nos processos de Rotulação e Padronização de Tarefas que ocorrem durante a reunião de *Sprint Planning* que é um evento típico do *Scrum*. Este modelo representa uma das contribuições do trabalho, uma vez que representa uma forma de gerar perfis capazes de refletir, mais fielmente, as competências técnicas adquiridas durante o desenvolvimento de software.

Para validar e avaliar a solução, foi montada uma base de dados históricos com informações reais provenientes de 12 projetos de desenvolvimento de software, executados entre os anos de 2015 e 2018. Ao todo, a base é composta por 1.063 tarefas rotuladas e padronizadas, implementadas por 52 desenvolvedores diferentes. A base de dados também é uma das contribuições do trabalho, uma vez que apresenta potencial de utilização para ouras pesquisas. Por exemplo, seria possível recomendar de tarefas com base nas competências técnicas dos desenvolvedores, de modo a reduzir o tempo necessário de implementação das mesmas; melhorar a estimativa de esforço de acordo com o perfil técnico da equipe; e a partir da aná-

lise da base de dados, seria possível recomendar cursos de capacitação em novas tecnologias para os desenvolvedores da empresa.

A validação do trabalho foi realizada mediante execução da abordagem em 13 cenários distintos, criados a partir de dados de quatro projetos das plataformas web e móvel. Para cada cenário, utilizou-se uma determinada regra de negócio, almejando-se formar equipes multidisciplinares com características específicas. Os objetivos da validação foram determinar se a abordagem é capaz de formar equipes de acordo com as saídas esperadas para cada cenário e definir a melhor configuração dos parâmetros para tal.

A avaliação do trabalho foi dividida em duas etapas, conduzidos com o auxílio de quatro gestores com experiência em gestão ágil de projetos de software, os quais trabalharam em projetos da base que foram utilizados para criar cenários de avaliação. No primeiro, os gestores tinham que formar equipes de acordo com determinadas regras. Nesta etapa, o objetivo foi determinar a acurácia da abordagem, ou seja, verificar se as recomendações eram semelhantes às equipes formadas pelos gestores. Na segunda etapa, foi apresentado aos gestores um cenário de maior complexidade, no qual eles tinham que negociar os recursos disponíveis para alocação, a fim de formar suas respectivas equipes.

No tocante às questões de pesquisa, os resultados obtidos corroboram para a aceitação das hipóteses estabelecidas. O modelo de especificação de tarefas foi aplicado em 1.063 tarefas, implementadas por dezenas de desenvolvedores em diversos projetos reais de desenvolvimento de software. Desta forma, possibilitou-se modelar os conhecimentos e habilidades técnicas dos integrantes, dando suporte às hipóteses H_{1-1} e H_{1-2}.

Os resultados da validação da abordagem dão suporte às hipóteses H_{2-1}, H_{2-2} e H_{3-2}, uma vez que o Algoritmo Genético, utilizando os dados dos perfis técnicos gerados a partir da base de dados, foi capaz de formar equipes de acordo com as saídas esperadas em todos os cenários, com base em diferentes regras de negócios e funções de *fitness*.

Por fim, os resultados da avaliação dão suporte à hipótese H_{3-1}, uma vez que a abordagem conseguiu expressivos valores de Precisão, obtendo média de 86,4% entre os gestores e 75% de aceitação.

6.2 Limitações e Sugestões para Trabalhos Futuros

Os projetos que compõem a base de dados históricos foram executados seguindo a metodologia de desenvolvimento ágil Scrum. No Scrum, a Equipe de Desenvolvimento é orientada a quebrar as US em tarefas técnicas de tamanho similar, não devendo demandar mais do que um dia de trabalho. Entretanto, a abordagem proposta não engloba métricas de esforço ou qualidade para diferenciar tarefas implementadas. A aplicação do modelo de especificação de tarefas originou uma base de dados que contribuiu diretamente para que os resultados obtidos estivessem de acordo com o esperado, porém não foi possível avaliar o impacto deste processo no desenvolvimento de software em tempo real.

Apesar de não fazer parte do escopo do trabalho, outra limitação identificada diz respeito à criação dos perfis técnicos dos desenvolvedores, os quais são originados gradativamente, ao longo das participações nos projetos. Desta forma, a abordagem utiliza apenas dados considerados de maior confiabilidade, ou seja, dados que podem ser verificados dentro da empresa, como por exemplo as tarefas implementadas que são revisadas por outros desenvolvedores da equipe. Esta limitação impacta, principalmente, em um cenário em que a abordagem tenha sido recém implantada e não haja nenhuma informação disponível sobre os perfis dos desenvolvedores. Neste caso, a execução imediata da abordagem resultaria na alocação aleatória de desenvolvedores, formando equipes inadequadas. Uma alternativa para reduzir este problema seria converter as informações do currículo em *tags*, ponderando-as de acordo com o tempo de experiência do indivíduo nas tecnologias presentes no documento. Entretanto, seria necessária uma investigação adicional para descobrir quais pesos teriam os dois tipos de perfis criados e em quais momentos poderiam ser utilizadas apenas as informações internas.

Como trabalho futuro, pretende-se dar continuidade à pesquisa no sentindo de minimizar algumas limitações identificadas. Por exemplo, em relação à diferenciação das tarefas, seria possível utilizar estimativa de esforço para auxiliar a mitigar tal limitação. A estimativa de esforço visa determinar o esforço necessário durante o desenvolvimento de software. Neste caso, uma métrica comumente utilizada é quantidade de horas que o desenvolvedor precisou para finalizar a tarefa, o que ajudaria a determinar a complexidade de tal artefato. Outra possibilidade é a utilização de métricas de qualidade. É comum que em tarefas que tenham

sido consideradas prontas, sejam encontrados defeitos no código fonte gerado, o que pode indicar que a qualidade da implementação está aquém do esperado. Dessa forma, o número de *bugs* reportados para uma tarefa pode representar uma métrica em potencial para auxiliar na diferenciação destes artefatos. Infelizmente, não foi possível integrar as sugestões ao escopo, pois a base de dados montada não possuía as métricas mencionadas em volume suficiente. Por exemplo, somente em uma minoria de projetos havia registro da quantidade de horas para cada tarefa.

Outra sugestão de trabalho futuro, consiste na investigação dos custos e benefícios para adição de aspectos sociais *(soft skills*) à abordagem. Na versão atual, são utilizados somente fatores técnicos (*hard skills*), porém é possível que as interações entre os indivíduos possam aumentar o nível de aceitação das recomendações oferecidas aos gestores, uma vez que desenvolvedores que trabalharam juntos em diversos projetos têm maior potencial de apresentar melhor convivência dentro da equipe. Por outro lado, também é possível que a utilização de *soft skills* proporcione uma subjetividade prejudicial ao processo de formação de equipes, uma vez que esse tipo de atributo é de difícil aprendizado e mensuração.

Por fim, pretende-se integrar a abordagem proposta a uma ferramenta de gerenciamento e planejamento de projetos, chamada Turmalina. Atualmente, essa ferramenta é utilizada por dezenas de projetos e centenas de desenvolvedores no VIRTUS. Dessa forma, seria possível realizar um estudo de caso para avaliar a abordagem em larga escala, o que auxiliaria na mitigação das ameaças à validade reportadas no Capítulo 5.

Bibliografia

[1] Muhammad Ovais Ahmad, Jouni Markkula, and Markku Oivo. Kanban in software development: A systematic literature review. In *Software Engineering and Advanced Applications (SEAA), 2013 39th EUROMICRO Conference on*, pages 9–16. IEEE, 2013.

[2] Nauman Bin Ali, Kai Petersen, and Claes Wohlin. A systematic literature review on the industrial use of software process simulation. *Journal of Systems and Software*, 97:65 – 85, 2014.

[3] Mohammad Y Allaho, Wang-Chien Lee, and De-Nian Yang. Staffing open collaborative projects based on the degree of acquaintance. In *International Conference on Database Systems for Advanced Applications*, pages 385–400. Springer, 2013.

[4] Scrum Alliance. The state os scrum report. `https://www.scrumalliance.org/why-scrum/state-of-scrum-report/2017-state-of-scrum`, 2017. [Online; acessado em 09 de outubro de 2017].

[5] Christine M Anderson-Cook. Practical genetic algorithms, 2005.

[6] Margarita André, María G Baldoquín, and Silvia T Acuña. Formal model for assigning human resources to teams in software projects. *Information and Software Technology*, 53(3):259–275, 2011.

[7] Michael Arias, Jorge Munoz-Gama, and Marcos Sepúlveda. A multi-criteria approach for team recommendation. In *International Conference on Business Process Management*, pages 384–396. Springer, 2016.

[8] Abhinaya Arunachalam, Nandhini Priya Nagarajan, Vanathi Mohan, Monica Reddy, and Chamundeswari Arumugam. Resolving team selection in agile development using nsga-ii algorithm. *CSI Transactions on ICT*, 4(2):83–86, 2016.

[9] Celso Ishida Aurora Pozo, Andrea Cavalheiro. Computação evolutiva. `http://www.inf.ufpr.br/aurora/tutoriais/Ceapostila.pdf`, 2017. [Online; acessado em 15 de setembro de 2017].

[10] Derya Avci and Akif Dogantekin. An expert diagnosis system for parkinson disease based on genetic algorithm-wavelet kernel-extreme learning machine. *Parkinson's disease*, 2016, 2016.

[11] L.K.J. Baartman and E Bruijn. Integrating knowledge, skills and attitudes: Conceptualising learning processes towards vocational competence. *Educational Research Review*, 6:125–134, 12 2011.

[12] Ahilton Barreto, Márcio Barros, and Cláudia Werner. Staffing a software project: a constraint satisfaction approach. In *ACM SIGSOFT Software Engineering Notes*, volume 30, pages 1–5. ACM, 2005.

[13] Ahilton Barreto, Márcio de O Barros, and Cláudia ML Werner. Staffing a software project: A constraint satisfaction and optimization-based approach. *Computers & Operations Research*, 35(10):3073 – 3089, 2008. Part Special Issue: Search-based Software Engineering.

[14] K. Beck, M. Beedle, A. van Bennekum, A. Cockburn, W. Cunningham, M. Fowler, J. Grenning, J. Highsmith, A. Hunt, R. Jeffries, Jon Kern, Brian Marick, Robert C. Martin, Steve Mallor, Ken Shwaber, and Jeff Sutherland. The Agile Manifesto. Technical report, The Agile Alliance, 2001.

[15] Pearl Brereton, Barbara A. Kitchenham, David Budgen, Mark Turner, and Mohamed Khalil. Lessons from applying the systematic literature review process within the software engineering domain. *J. Syst. Softw.*, 80(4):571–583, April 2007.

[16] Ricardo Britto, Pedro Santos Neto, Ricardo Rabelo, Werney Ayala, and Thiago Soares. A hybrid approach to solve the agile team allocation problem. In *Evolutionary Computation (CEC), 2012 IEEE Congress on*, pages 1–8. IEEE, 2012.

[17] David Cohen, Mikael Lindvall, and Patricia Costa. Agile software development. *DACS SOAR Report*, 11, 2003.

[18] Mike Cohn. *User stories applied: For agile software development*. Addison-Wesley Professional, 2004.

[19] Thelma Elita Colanzi, Silvia Regina Vergilio, Wesley Klewerton Guez AssunçãO, and Aurora Pozo. Search based software engineering: Review and analysis of the field in brazil. *Journal of Systems and Software*, 86(4):970–984, 2013.

[20] Ricardo Colomo-Palacios, Israel González-Carrasco, José Luis López-Cuadrado, and Ángel García-Crespo. Resyster: A hybrid recommender system for scrum team roles based on fuzzy and rough sets. *International Journal of Applied Mathematics and Computer Science*, 22(4):801–816, 2012.

[21] Alexandre Costa, Felipe Ramos, Mirko Perkusich, Arthur Freire, Hyggo Almeida, and Angelo Perkusich. A search-based software engineering approach to support multiple team formation for scrum projects. In *The 30th International Conference on Software Engineering & Knowledge Engineering*, San Francisco Bay, USA, 2018.

[22] Fabio QB da Silva, A Cesar C Franca, Tatiana B Gouveia, Cleviton VF Monteiro, Elisa SF Cardozo, and Marcos Suassuna. An empirical study on the use of team building criteria in software projects. In *Empirical Software Engineering and Measurement (ESEM), 2011 International Symposium on*, pages 58–67. IEEE, 2011.

[23] Charles Darwin and William F Bynum. *The origin of species by means of natural selection: or, the preservation of favored races in the struggle for life*. AL Burt, 2009.

[24] Jesse Davis and Mark Goadrich. The relationship between precision-recall and roc curves. In *Proceedings of the 23rd international conference on Machine learning*, pages 233–240. ACM, 2006.

[25] Massimiliano Di Penta, Mark Harman, and Giuliano Antoniol. The use of search-based optimization techniques to schedule and staff software projects: an approach and an empirical study. *Software: Practice and Experience*, 41(5):495–519, 2011.

[26] R Dillibabu and K Krishnaiah. Cost estimation of a software product using cocomo ii. 2000 model–a case study. *International Journal of Project Management*, 23(4):297–307, 2005.

[27] Torgeir Dingsøyr, Sridhar Nerur, VenuGopal Balijepally, and Nils Brede Moe. A decade of agile methodologies: Towards explaining agile software development. *Journal of Systems and Software*, 85(6):1213 – 1221, 2012. Special Issue: Agile Development.

[28] Marguerite Doman, Andrew Besmer, and Anne Olsen. Managing software engineering student teams using pellerin's 4-d system. *Journal of Information Systems Education*, 26(4):257, 2015.

[29] Tore Dybå and Torgeir Dingsøyr. Empirical studies of agile software development: A systematic review. *Inf. Softw. Technol.*, 50(9-10):833–859, August 2008.

[30] Lúcio Camara e Silva and Ana Paula Cabral Seixas Costa. Decision model for allocating human resources in information system projects. *International Journal of Project Management*, 31(1):100–108, 2013.

[31] Mehdi Farhangian, Martin Purvis, Maryam A. Purvis, and Bastin Tony Roy Savarimuthu. Personalities and software development team performance: a psycholinguistic study. *24th European Conference on Information Systems*, 2016.

[32] A César C França, Évisson Fernandes Lucena, and Fabio QB da Silva. A quantitative assessment on team building criteria for software project teams. In *Proceedings of 6th Experimental Software Engineering Latin American Workshop (ESELAW 2009)*, page 12. Citeseer, 2009.

[33] Nir Friedman, Dan Geiger, and Moises Goldszmidt. Bayesian network classifiers. *Machine learning*, 29(2-3):131–163, 1997.

[34] Mangesh Gharote, Rahul Patil, and Sachin Lodha. Scatter search for trainees to software project requirements stable allocation. *Journal of Heuristics*, 23(4):257–283, 2017.

[35] MS Gharote, RJ Patil, and SP Lodha. Minimal cost stable workforce allocation in presence of ties. In *Industrial Engineering and Engineering Management (IEEM), 2016 IEEE International Conference on*, pages 1146–1150. IEEE, 2016.

[36] Abdul Rehman Gilal, Jafreezal Jaafar, Shuib Basri, Mazni Omar, and Muhammad Zahid Tunio. Making programmer suitable for team-leader: Software team composition based on personality types. In *Mathematical Sciences and Computing Research (iSMSC), International Symposium on*, pages 78–82. IEEE, 2015.

[37] Abdul Rehman Gilal, Jafreezal Jaafar, Luiz Fernando Capretz, Mazni Omar, Shuib Basri, and Izzatdin Abdul Aziz. Finding an effective classification technique to develop a software team composition model. *Journal of Software: Evolution and Process*, 30(1):e1920, 2018.

[38] Abdul Rehman Gilal, Jafreezal Jaafar, Mazni Omar, Shuib Basri, and Izzatdin Abdul Aziz. A set of rules for constructing gender-based personality types' composition for software programmer. *The Second International Conference on Advanced Data and Information Engineering*, 2015.

[39] Abdul Rehman Gilal, Jafreezal Jaafar, Mazni Omar, Shuib Basri, and Izzatdin Abdul Aziz. Balancing the personality of programmer: software development team composition. *Malaysian Journal of Computer Science*, 29(2):145–155, 2016.

[40] Abdul Rehman Gilal, Jafreezal Jaafar, Mazni Omar, Shuib Basri, and Ahmad Waqas. A rule-based model for software development team composition: Team leader role with personality types and gender classification. *Information and Software Technology*, 74:105–113, 2016.

[41] Abdul Rehman Gilal, Mazni Omar, and Kamal Imran Sharif. Discovering personality types and diversity based on software team roles. In *International Conference on Computing and Informatics, ICOCI*, volume 2013, pages 259–264, 2013.

[42] Narasimhaiah Gorla and Yan Wah Lam. Who should work with whom?: Building effective software project teams. *Commun. ACM*, 47(6):79–82, June 2004.

[43] Alan Gray, Andrew Jackson, Ioanna Stamouli, and Shiu Lun Tsang. Forming successful extreme programming teams. In *Agile Conference, 2006*, pages 10–pp. IEEE, 2006.

[44] Jimmy H Gutiérrez, César A Astudillo, Pablo Ballesteros-Pérez, Daniel Mora-Melià, and Alfredo Candia-Véjar. The multiple team formation problem using sociometry. *Computers & Operations Research*, 75:150–162, 2016.

[45] Mark Harman, S Afshin Mansouri, and Yuanyuan Zhang. Search-based software engineering: Trends, techniques and applications. *ACM Computing Surveys (CSUR)*, 45(1):11, 2012.

[46] Rashina Hoda, James Noble, and Stuart Marshall. Self-organizing roles on agile software development teams. *IEEE Transactions on Software Engineering*, 39(3):422–444, 2013.

[47] Rashina Hoda, Norsaremah Salleh, and John Grundy. The rise and evolution of agile software development. *IEEE Software*, 35(5):58–63, 2018.

[48] Rashina Hoda, Norsaremah Salleh, John Grundy, and Hui Mien Tee. Systematic literature reviews in agile software development: A tertiary study. *Information and Software Technology*, 85:60–70, 2017.

[49] Jianbin Huang, Ze Lv, Yu Zhou, He Li, Heli Sun, and Xiaolin Jia. Forming grouped teams with efficient collaboration in social networks. *The Computer Journal*, pages 1–16, 2016.

[50] RK Jana, MK Sanyal, and Saikat Chakrabarti. Binary fuzzy goal programming for effective utilization of it professionals. In *Proceedings of the First International Conference on Intelligent Computing and Communication*, pages 395–405. Springer, 2017.

[51] Avnish Singh Jat, Purtee Kohli, and Devpriya Soni. Team member selection in agile. *4th Internacional Conference on Science, Technology and Management*, 2016.

[52] Enrique Jiménez-Domingo, Ricardo Colomo-Palacios, and Juan Miguel Gómez-Berbís. A multi-objective genetic algorithm for software personnel staffing for hcim solutions. *International Journal of Web Portals (IJWP)*, 6(2):26–41, 2014.

[53] Piers Johnson, Luke Vandewater, William Wilson, Paul Maruff, Greg Savage, Petra Graham, Lance S Macaulay, Kathryn A Ellis, Cassandra Szoeke, Ralph N Martins, et al. Genetic algorithm with logistic regression for prediction of progression to alzheimer's disease. *BMC bioinformatics*, 15(16):S11, 2014.

[54] Leila Kallel, Bart Naudts, and Alex Rogers. *Theoretical aspects of evolutionary computing*. Springer Science & Business Media, 2013.

[55] George J Klir and Baozung Yuan. *Fuzzy sets and fuzzy logic: theory and applications*, volume 574. Prentice Hall PTR New Jersey, 1995.

[56] Roberto Latorre and Javier Suárez. Measuring social networks when forming information system project teams. *Journal of Systems and Software*, 134:304–323, 2017.

[57] Sherlock Licorish, Anne Philpott, and Stephen G MacDonell. Supporting agile team composition: A prototype tool for identifying personality (in) compatibilities. In *Proceedings of the 2009 ICSE Workshop on Cooperative and Human Aspects on Software Engineering*, pages 66–73. IEEE Computer Society, 2009.

[58] Lowell Lindstrom and Ron Jeffries. Extreme programming and agile software development methodologies. *Information systems management*, 21(3):41–52, 2004.

[59] Anirban Majumder, Samik Datta, and KVM Naidu. Capacitated team formation problem on social networks. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1005–1013. ACM, 2012.

[60] Isa Maleki, Ali Ghaffari, and Mohammad Masdari. A new approach for software cost estimation with hybrid genetic algorithm and ant colony optimization. *International Journal of Innovation and Applied Studies*, 5(1):72, 2014.

[61] Robert Cecil Martin. *Agile Software Development: Principles, Patterns, and Practices*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2003.

[62] Luis G Martínez, Guillermo Licea, Antonio Rodríguez, Juan R Castro, and Oscar Castillo. Using matlab's fuzzy logic toolbox to create an application for ramset in software engineering courses. *Computer Applications in Engineering Education*, 21(4):596–605, 2013.

[63] Luis G Martínez, Guillermo Licea, Antonio Rodríguez-Díaz, and Juan R Castro. Experiences in software engineering courses using psychometrics with ramset. In *Proceedings of the fifteenth annual conference on Innovation and technology in computer science education*, pages 244–248. ACM, 2010.

[64] Gurpreet Singh Matharu, Anju Mishra, Harmeet Singh, and Priyanka Upadhyay. Empirical study of agile software development methodologies: A comparative analysis. *ACM SIGSOFT Software Engineering Notes*, 40(1):1–6, 2015.

[65] Patrick E McKight and Julius Najab. Kruskal-wallis test. *The corsini encyclopedia of psychology*, pages 1–1, 2010.

[66] Phil McMinn. Search-based software test data generation: a survey. *Software testing, Verification and reliability*, 14(2):105–156, 2004.

[67] Santanu Kr Misra and Amitava Ray. Software developer selection: A holistic approach for an eclectic decision. *Int. J. Comput. Appl*, 47(1):12–18, 2012.

[68] N. B. Moe, T. Dingsøyr, and T. Dybå. Understanding self-organizing teams in agile software development. In *19th Australian Conference on Software Engineering (aswec 2008)*, pages 76–85, March 2008.

[69] Mazin Abed Mohammed, Mohd Khanapi Abd Ghani, Raed Ibraheem Hamed, Salama A. Mostafa, Mohd Sharifuddin Ahmad, and Dheyaa Ahmed Ibrahim. Solving vehicle routing problem by using improved genetic algorithm for optimal solution. *Journal of Computational Science*, 21:255 – 262, 2017.

[70] Tais Dall Oca. Testes de software e scrum. `https://taisdalloca.blogspot.com/2015/09/testes-de-software-e-scrum.html`, 2015. [Online; acessado em 11 de marçp de 2019].

[71] Mazni Omar, Bikhtiyar Hasan, Mazida Ahmad, Azman Yasin, Fauziah Baharom, Haslina Mohd, and Norida Mohd Darus. Applying fuzzy technique in software team formation based on belbin team role. *Journal of Telecommunication, Electronic and Computer Engineering (JTEC)*, 8(8):109–113, 2016.

[72] Mazni Omar, Bikhtiyar Hasan, Mazida Ahmad, Azman Yasin, Fauziah Baharom, Haslina Mohd, and Norida Muhd Darus. Towards a balanced software team formation based on belbin team role using fuzzy technique. In *AIP Conference Proceedings*, volume 1761, page 020082. AIP Publishing, 2016.

[73] Carlos E Otero, Luis D Otero, Ira Weissberger, and Abrar Qureshi. A multi-criteria decision making approach for resource allocation in software engineering. In *Computer Modelling and Simulation (UKSim), 2010 12th International Conference on*, pages 137–141. IEEE, 2010.

[74] Mario Andrés Paredes-Valverde, María del Pilar Salas-Zárate, Ricardo Colomo-Palacios, Juan Miguel Gómez-Berbís, and Rafael Valencia-García. An ontology-based approach with which to assign human resources to software projects. *Science of Computer Programming*, 156:90–103, 2018.

[75] Saswat K Pujari, Gargi Bhattacharjee, and Soumyakanta Bhoi. A hybridized model for image encryption through genetic algorithm and dna sequence. *Procedia Computer Science*, 125:165 – 171, 2018. The 6th International Conference on Smart Computing and Communications.

[76] D. Castellanos-Nieves R. Valencia-Garcia, F. Garcia-Sanchez. Exploitation of social semantic technology for software development team configuration. *Iet Software*, 4(6):373–385, 2010.

[77] Anitha Rao, Sandeep Kumar Hegde, A Rao, K Hegde, IAnitha Rao, K Hegde, A Rao, and SK Hegde. Literature survey on travelling salesman problem using genetic algorithms. *International Journal of Advanced Research in Eduation Technology (IJARET)*, 2(1):42, 2015.

[78] Nornadiah Mohd Razali, Yap Bee Wah, et al. Power comparisons of shapiro-wilk, kolmogorov-smirnov, lilliefors and anderson-darling tests. *Journal of statistical modeling and analytics*, 2(1):21–33, 2011.

[79] Abdul Rehman Gilal, Mazni Omar, and Kamal Imran Sharif. A rule-based approach for discovering effective software team composition. *Journal of Information & Communication Technology*, 13, 2014.

[80] Jian Ren, Mark Harman, and Massimiliano Di Penta. Cooperative co-evolutionary optimization of software project staff assignments and job scheduling. *Search Based Software Engineering*, pages 127–141, 2011.

[81] Pilar Rodríguez, Jouni Markkula, Markku Oivo, and Kimmo Turula. Survey on agile and lean usage in finnish software industry. In *Proceedings of the ACM-IEEE International Symposium on Empirical Software Engineering and Measurement*, ESEM '12, pages 139–148, New York, NY, USA, 2012. ACM.

[82] Daniel Schall. Skill-based team formation in software ecosystems. In *International Workshop on Quality Assurance in Computer Vision and the International Workshop on Digital Eco-Systems*, 2016.

[83] Ken Schwaber and Mike Beedle. *Agile software development with Scrum*, volume 1. Prentice Hall Upper Saddle River, 2002.

[84] Ken Schwaber and Jeff Sutherland. The Scrum guide. `https://www.scrumguides.org/`, 2017. [Online; acessado em 10 de agosto de 2018].

[85] Mary Shaw. Writing good software engineering research papers: Minitutorial. In *Proceedings of the 25th International Conference on Software Engineering*, ICSE '03, pages 726–736, Washington, DC, USA, 2003. IEEE Computer Society.

[86] Brajesh Kumar Singh and AK Misra. Software effort estimation by genetic algorithm tuned parameters of modified constructive cost model for nasa software projects. *International Journal of Computer Applications*, 59(9), 2012.

[87] Vedant Singh, Somesh K Sharma, and S Vaibhav. Transport aircraft conceptual design optimization using real coded genetic algorithm. *International Journal of Aerospace Engineering*, 2016, 2016.

[88] Ian Sommerville. *Software Engineering (5th Ed.)*. Addison Wesley Longman Publishing Co., Inc., Redwood City, CA, USA, 1995.

[89] Laabadi Soukaina, Naimi Mohamed, El Amri Hassan, and Achchab Boujemâa. A hybrid genetic algorithm for solving 0/1 knapsack problem. In *Proceedings of the International Conference on Learning and Optimization Algorithms: Theory and Applications*, LOPAL '18, pages 51:1–51:6, New York, NY, USA, 2018. ACM.

[90] Mark Staples and Mahmood Niazi. Experiences using systematic review guidelines. *J. Syst. Softw.*, 80(9):1425–1437, September 2007.

[91] Stavros Stavru. A critical examination of recent industrial surveys on agile method usage. *Journal of Systems and Software*, 94:87–97, 2014.

[92] Damjan Strnad and Nikola Guid. A fuzzy-genetic decision support system for project team formation. *Applied soft computing*, 10(4):1178–1187, 2010.

[93] Constantinos Stylianou and Andreas S Andreou. A multi-objective genetic algorithm for software development team staffing based on personality types. In *IFIP International Conference on Artificial Intelligence Applications and Innovations*, pages 37–47. Springer, 2012.

[94] Constantinos Stylianou and Andreas S Andreou. A multi-objective genetic algorithm for intelligent software project scheduling and team staffing. *Intelligent Decision Technologies*, 7(1):59–80, 2013.

[95] Constantinos Stylianou, Simos Gerasimou, and Andreas S Andreou. A novel prototype tool for intelligent software project scheduling and staffing enhanced with personality factors. In *Tools with Artificial Intelligence (ICTAI), 2012 IEEE 24th International Conference on*, volume 1, pages 277–284. IEEE, 2012.

[96] Pratibha Sukhija, Sunny Behal, and Pritpal Singh. Face recognition system using genetic algorithm. *Procedia Computer Science*, 85:410 – 417, 2016. International Conference on Computational Modelling and Security (CMS 2016).

[97] Didi Surian, Nian Liu, David Lo, Hanghang Tong, Ee-Peng Lim, and Christos Faloutsos. Recommending people in developers' collaboration network. In *Reverse Engineering (WCRE), 2011 18th Working Conference on*, pages 379–388. IEEE, 2011.

[98] Hsien-Tang Tsai, Herbert Moskowitz, and Lai-Hsi Lee. Human resource selection for software development projects using taguchi's parameter design. *European Journal of operational research*, 151(1):167–180, 2003.

[99] Tzu-Liang Bill Tseng, Chun-Che Huang, How-Wei Chu, and Roger R Gung. Novel approach to multi-functional project team formation. *International Journal of Project Management*, 22(2):147–159, 2004.

[100] Tutorials Point. Genetic algorithms - introduction, 2018. `https://www.tutorialspoint.com/genetic_algorithms/genetic_algorithms_quick_guide.htm`, Last accessed on 2018-09-03.

[101] June Verner, Jennifer Sampson, and Narciso Cerpa. What factors lead to software project failure? In *2008 Second International Conference on Research Challenges in Information Science*, pages 71–80. IEEE, 2008.

[102] Xuewu Wang, Yingpan Shi, Dongyan Ding, and Xingsheng Gu. Double global optimum genetic algorithm–particle swarm optimization-based welding robot path planning. *Engineering Optimization*, 48(2):299–316, 2016.

[103] RA Welikala, Muhammad Moazam Fraz, Jamshid Dehmeshki, Andreas Hoppe, V Tah, S Mann, Thomas H Williamson, and Sarah A Barman. Genetic algorithm based feature selection combined with dual classification for the automated detection of proliferative diabetic retinopathy. *Computerized Medical Imaging and Graphics*, 43:64–77, 2015.

[104] Roel Wieringa, Neil Maiden, Nancy Mead, and Colette Rolland. Requirements engineering paper classification and evaluation criteria: a proposal and a discussion. *Requirements Engineering*, 11(1):102–107, 2006.

[105] Gerhard J Woeginger. Exact algorithms for np-hard problems: A survey. In *Combinatorial optimization—eureka, you shrink!*, pages 185–207. Springer, 2003.

[106] Claes Wohlin. Guidelines for snowballing in systematic literature studies and a replication in software engineering. In *Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering*, EASE '14, pages 38:1–38:10, New York, NY, USA, 2014. ACM.

[107] Claes Wohlin, Per Runeson, Martin Hst, Magnus C. Ohlsson, Bjrn Regnell, and Anders Wessln. *Experimentation in Software Engineering*. Springer Publishing Company, Incorporated, 2012.

[108] Junchao Xiao, Qing Wang, Mingshu Li, Qiusong Yang, Lizi Xie, and Dapeng Liu. Value-based multiple software projects scheduling with genetic algorithm. In *International Conference on Software Process*, pages 50–62. Springer, 2009.

[109] Yuming Xu, Kenli Li, Jingtong Hu, and Keqin Li. A genetic algorithm for task scheduling on heterogeneous computing systems using multiple priority queues. *Information Sciences*, 270:255 – 287, 2014.

[110] Fadi A Zaraket, Majd Olleik, and Ali A Yassine. Skill-based framework for optimal software project selection and resource allocation. *European Journal of Operational Research*, 234(1):308–318, 2014.

Apêndice A

Lista Detalhada dos Estudos do Mapeamento Sistemático

Estudo	Título	Ano	Local de Publicação	Questão de pesquisa
E01	Team Member Selection In Agile	2016	International Journal Of Science Technology & Management	Método ou meio de desenvolvimento
E02	Resolving team selection in agile development using NSGA-II algorithm	2016	CSI Transactions on ICT	Método ou meio de desenvolvimento
E03	Skill-based Team Formation in Software Ecosystems	2016	International Workshop on Quality Assurance in Computer Vision and the International Workshop on Digital Eco-Systems	Método ou meio de desenvolvimento
E04	Applying Fuzzy Technique In Software Team Formation Based On Belbin Team Role	2016	Journal of Telecommunication, Electronic and Computer Engineering	Método ou meio de desenvolvimento
E06	A hybrid approach to solve the agile team allocation problem	2012	Congress on Evolutionary Computation	Método ou meio de desenvolvimento
E07	Capacitated team formation problem on social networks	2012	International conference on Knowledge discovery and data mining	Método ou meio de desenvolvimento
E08	The use of search-based optimization techniques to schedule and staff software projects: An approach and an empirical study	2011	Journal: Software - Practice & Experience	Método para análise ou avaliação
E09	Novel approach to multi-functional project team formation	2004	International Journal of Project Management	Método ou meio de desenvolvimento
E10	Who should work with whom? Building effective software project teams	2004	Communications of the ACM - Wireless sensor networks CACM	Design, avaliação ou análise de uma instância particular
E12	Software developer selection: A holistic approach for an eclectic decision	2012	International Journal of Computer Applications	Método ou meio de desenvolvimento
E13	Exploitation of social semantic technology for software development team configuration	2010	Institution of Engineering and Technology	Método ou meio de desenvolvimento
E14	Staffing a software project: A constraint satisfaction and optimization-based approach	2008	Computers and Operations Research	Método ou meio de desenvolvimento
E15	Human resource selection for software development projects using Taguchi‘s parameter design	2003	European Journal of Operational Research	Método ou meio de desenvolvimento
E16	Forming Grouped Teams with Efficient Collaboration in Social Networks	2016	The Computer Journal	Método ou meio de desenvolvimento
E17	A multi-objective genetic algorithm for intelligent software project scheduling and team staffing	2013	Intelligent Decision Technologies	Método ou meio de desenvolvimento
E19	Cooperative co-evolutionary optimization of software project staff assignments and job scheduling	2011	International conference on Search based software engineering	Método ou meio de desenvolvimento
E20	A fuzzy-genetic decision support system for project team formation	2010	Applied Soft Computing	Método ou meio de desenvolvimento

Estudo	Título	Ano	Local de Publicação	Questão de pesquisa
E21	An empirical study on the use of team building criteria in software projects	2011	International Symposium on Empirical Software Engineering and Measurement	Método para análise ou avaliação
E22	Formal model for assigning human resources to teams in software projects	2011	Information and Software Technology	Método ou meio de desenvolvimento
E23	Forming successful extreme programming teams	2006	Agile Conference	Método para análise ou avaliação
E24	Managing Software Engineering Student Teams Using Pellerin's 4-D System	2015	Journal of Information Systems Education	Método ou meio de desenvolvimento
E25	Personalities And Software Development Team Performance, A Psycholinguistic Study	2016	European Conference on Information Systems	Método ou meio de desenvolvimento
E26	Supporting agile team composition: A prototype tool for identifying personality (In) compatibilities	2009	Workshop on Cooperative and Human Aspects of Software Engineering	Método ou meio de desenvolvimento
E11	Using MatLab's fuzzy logic toolbox to create an application for RAMSET in software engineering courses	2013	Computer Applications in Engineering Education	Método ou meio de desenvolvimento
E27	Resyster: A hybrid recommender system for scrum team roles based on fuzzy and rough sets	2012	International Journal of Applied Mathematics and Computer Science	Método ou meio de desenvolvimento
E18	A Multi-objective Genetic Algorithm for Software Development Team Staffing Based on Personality Types	2012	International Conference on Artificial Intelligence Applications and Innovations	Método ou meio de desenvolvimento
E28	A multi-criteria approach for team recommendation	2017	International Conference on Business Process Management	Método ou meio de desenvolvimento
E29	A multi-objective genetic algorithm for software personnel staffing for HCIM solutions	2014	International Journal of Web Portals	Método ou meio de desenvolvimento
E30	Decision model for allocating human resources in information system projects	2013	International Journal of Project Management	Método para análise ou avaliação
E31	Minimal cost stable workforce allocation in presence of ties	2016	International Conference on Industrial Engineering and Engineering Management	Método ou meio de desenvolvimento
E32	Scatter search for trainees to software project requirements stable allocation	2017	Journal of Heuristics	Método ou meio de desenvolvimento
E33	Skill-based framework for optimal software project selection and resource allocation	2014	European Journal of Operational Research	Método ou meio de desenvolvimento
E34	Staffing open collaborative projects based on the degree of acquaintance	2013	International Conference on Database Systems for Advanced Applications	Método para análise ou avaliação
E35	A multi-criteria decision making approach for resource allocation in software engineering	2010	International Conference on Computer Modelling and Simulation	Método ou meio de desenvolvimento

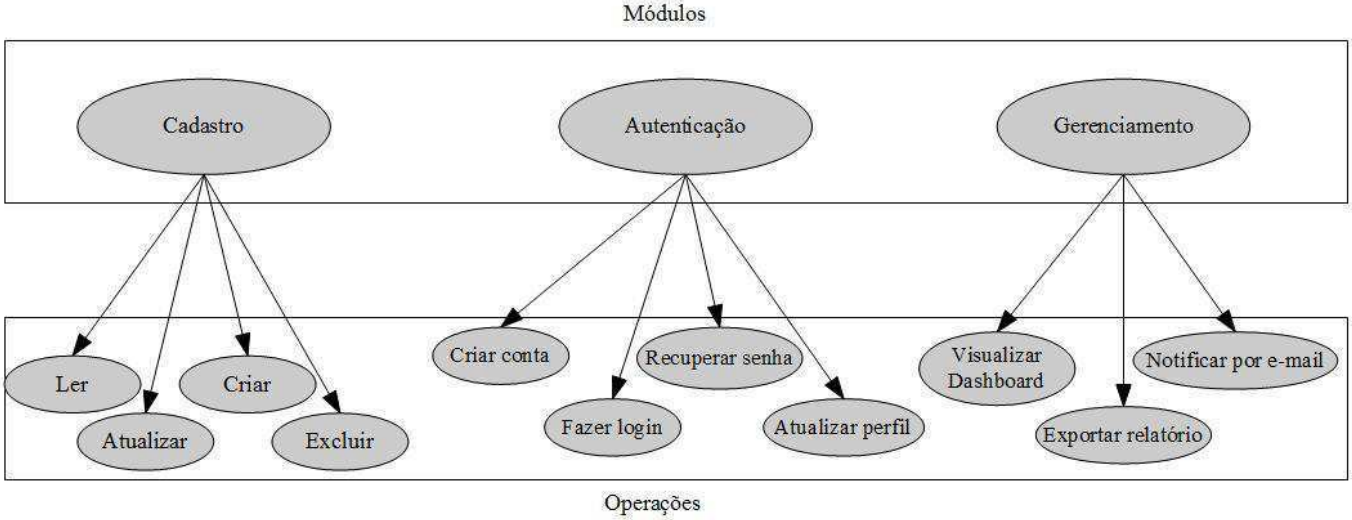
Estudo	Título	Ano	Local de Publicação	Questão de pesquisa
E36	An ontology-based approach with which to assign human resources to software projects	2018	Science of Computer Programming	Método ou meio de desenvolvimento
E37	Binary fuzzy goal programming for effective utilization of IT professionals	2016	International Conference on Intelligent Computing and Communication	Método ou meio de desenvolvimento
E38	Measuring social networks when forming information system project teams	2017	Journal of Systems and Software	Método ou meio de desenvolvimento
E05	Finding an effective classification technique to develop a software team composition model	2017	Journal of Software: Evolution and Process	Método ou meio de desenvolvimento
E39	Value-based multiple software projects scheduling with genetic algorithm	2009	International Conference on Software Process	Método ou meio de desenvolvimento
E40	Recommending people in developers' collaboration network	2011	Working Conference on Reverse Engineering	Método ou meio de desenvolvimento

Estudo	Resultado da Pesquisa	Validação da Pesquisa	Tipo de Atributo	Tipo de Técnica	Pontuação
E01	Procedimento ou técnica	Afirmação	Hard Skills	Algoritmo Próprio	2
E02	Procedimento ou técnica	Exemplo	Hard Skills	Busca e Otimização	3
E03	Procedimento ou técnica	Análise	Ambas	Busca e Otimização	9
E04	Procedimento ou técnica	Afirmação	Soft Skills	Raciocínio Probabilístico	3
E06	Procedimento ou técnica	Avaliação	Ambas	Busca e Otimização / Raciocínio Probabilístico	10
E07	Procedimento ou técnica	Análise	Soft Skills	Algoritmo Próprio	8
E08	Procedimento ou técnica	Análise	Hard Skills	Busca e Otimização / Busca e Otimização	10
E09	Procedimento ou técnica	Exemplo	Hard Skills	Raciocínio Probabilístico / Tomada de Decisão	6
E10	Relatório	Avaliação	Soft Skills	Outra	6
E12	Procedimento ou técnica	Exemplo	Hard Skills	Tomada de Decisão / Raciocínio Probabilístico	4
E13	Procedimento ou técnica	Avaliação	Ambas	Tomada de Decisão	9
E14	Ferramenta ou notação	Avaliação	Ambas	Busca e Otimização	11
E15	Procedimento ou técnica	Análise	Hard Skills	Outra	9
E16	Procedimento ou técnica	Análise	Ambas	Busca e Otimização	9
E17	Procedimento ou técnica	Análise	Hard Skills	Busca e Otimização	9
E19	Procedimento ou técnica	Análise	Hard Skills	Busca e Otimização	9
E20	Procedimento ou técnica	Avaliação	Hard Skills	Raciocínio Probabilístico / Busca e Otimização	7
E21	Modelo empírico	Análise	Ambas	Outra	10
E22	Ferramenta ou notação	Avaliação	Ambas	Ferramenta Psicológica / Ferramenta Psicológica	9
E23	Solução específica, protótipo, resposta ou julgamento	Avaliação	Ambas	Outra	9
E24	Ferramenta ou notação	Avaliação	Soft Skills	Ferramenta Psicológica	7
E25	Modelo empírico	Análise	Soft Skills	Ferramenta Psicológica	8

Estudo	Resultado da Pesquisa	Validação da Pesquisa	Tipo de Atributo	Tipo de Técnica	Pontuação
E26	Ferramenta ou notação	Avaliação	Soft Skills	Ferramenta Psicológica	4
E11	Ferramenta ou notação	Avaliação	Soft Skills	Ferramenta Psicológica	5
E27	Procedimento ou técnica	Avaliação	Ambas	Raciocínio Probabilístico / Raciocínio Probabilístico	9
E18	Procedimento ou técnica	Análise	Ambas	Busca e Otimização	9
E28	Modelo empírico	Análise	Hard Skills	Busca e Otimização	6
E29	Ferramenta ou notação	Avaliação	Ambas	Busca e Otimização	5
E30	Modelo descritivo ou qualitativo	Análise	Ambas	Busca e Otimização	5
E31	Procedimento ou técnica	Exemplo	Hard Skills	Algoritmo Próprio	6
E32	Procedimento ou técnica	Análise	Hard Skills	Busca e Otimização	7
E33	Ferramenta ou notação	Análise	Hard Skills	Busca e Otimização	7
E34	Procedimento ou técnica	Análise	Hard Skills	Algoritmo Próprio	9
E35	Procedimento ou técnica	Exemplo	Hard Skills	Busca e Otimização	2
E36	Procedimento ou técnica	Análise	Hard Skills	Tomada de Decisão	6
E37	Procedimento ou técnica	Exemplo	Hard Skills	Raciocínio Probabilístico	4
E38	Procedimento ou técnica	Avaliação	Ambas	Outra	11
E05	Modelo empírico	Análise	Soft Skills	Busca e Otimização	10
E39	Procedimento ou técnica	Análise	Hard Skills	Busca e Otimização	5
E40	Procedimento ou técnica	Análise	Hard Skills	Busca e Otimização	9

Apêndice B

Modelo Estruturado de User Stories e Tarefas


Módulos
Cadastro
Autenticação
Gerenciamento
Ler
Criar
Atualizar
Excluir
Criar conta
Recuperar senha
Fazer login
Atualizar perfil
Visualizar Dashboard
Notificar por e-mail
Exportar relatório
Operações

Módulo	Operação	Tarefas
Autenticação	Fazer login com usuário e senha	Criar campos Login e Senha
Autenticação	Fazer login com usuário e senha	Armazenar dados temporários (Função "Remember me")
Autenticação	Fazer login com usuário e senha	Criar serviço Auth (Server)
Autenticação	Fazer login com usuário e senha	Criar serviço Auth (Client)
Autenticação	Fazer login com usuário e senha	Declarar o provider
Autenticação	Fazer login com usuário e senha	Integrar back com front
Autenticação	Fazer login com usuário e senha	Criptografar senha no banco
Autenticação	Fazer login com usuário e senha	Configurar API
Autenticação	Fazer login com usuário e senha	Fazer logout
Autenticação	Fazer login com usuário e senha	Verificação de senha expirada
Autenticação	Fazer login com usuário e senha	Implementar timeout de sessão
Autenticação	Fazer login com usuário e senha	Implementar Captcha
Autenticação	Fazer login com usuário e senha	Fazer o roteamento da URL
Autenticação	Fazer login com OAuth	Criar serviço
Autenticação	Fazer login com OAuth	Fazer roteamento de URL
Autenticação	Fazer login com OAuth	Consumir API externa
Autenticação	Recuperar senha	Alterar tela de login
Autenticação	Recuperar senha	Validar e-mail e senha informado
Autenticação	Recuperar senha	Criar rota de recuperação de senha
Autenticação	Recuperar senha	Gerar senha padrão
Autenticação	Recuperar senha	Criar serviço de notificação
Autenticação	Recuperar senha	Criar tela para cadastro de nova senha
Autenticação	Acessar pela primeira vez o sistema	Criar tela para primeiro acesso
Autenticação	Acessar pela primeira vez o sistema	Comparar senhas informadas
Autenticação	Acessar pela primeira vez o sistema	Validar e-mail e senha informado
Autenticação	Acessar pela primeira vez o sistema	Integrar back com front
Autenticação	Acessar pela primeira vez o sistema	Criar rota de cadastro
Autenticação	Acessar pela primeira vez o sistema	Encriptar senha do novo usuário
Autenticação	Acessar pela primeira vez o sistema	Persistir no banco os dados de acesso

Autenticação	Acessar pela primeira vez o sistema	Fazer o roteamento da URL
Autenticação	Validar Permissão do Usuário	Adicionar atributo de proteção no provider
Autenticação	Validar Permissão do Usuário	Recuperar dados do usuário logado
Autenticação	Validar Permissão do Usuário	Validar permissão do usuário na aplicação
Autenticação	Validar Permissão do Usuário	Criar middlewares para validação do token
Autenticação	Validar Permissão do Usuário	Bloquear rotas
Autenticação	Atualizar perfil	Criar tela para atualizar perfil
Autenticação	Atualizar perfil	Integrar back com front
Autenticação	Atualizar perfil	Criar serviço user
Autenticação	Atualizar perfil	Declarar o provider
Autenticação	Atualizar perfil	Validar senha com confirmar senha
Autenticação	Atualizar perfil	Criar rota de cadastro
Autenticação	Atualizar perfil	Persistir no banco os dados de acesso
Autenticação	Criar conta	Criar tela para nova conta
Autenticação	Criar conta	Criar entidade no banco de dados
Autenticação	Criar conta	Integrar back com front
Autenticação	Criar conta	Atualizar serviço para nova conta
Autenticação	Criar conta	Declarar o provider
Autenticação	Criar conta	Criar rota de cadastro
Autenticação	Criar conta	Encriptar senha do usuário
Autenticação	Criar conta	Persistir no banco os dados de acesso
Autenticação	Remover conta	Integrar back com front
Autenticação	Remover conta	Criar serviço
Cadastro	Recuperação de Dados	Criar tela listagem
Cadastro	Recuperação de Dados	Criar paginação para listagem
Cadastro	Recuperação de Dados	Criar ordenação para listagem
Cadastro	Recuperação de Dados	Criar filtro para listagem
Cadastro	Recuperação de Dados	Criar Gráfico
Cadastro	Recuperação de Dados	Criar tela para visualizar detalhes
Cadastro	Recuperação de Dados	Alterar menu do sistema

Cadastro	Recuperação de Dados	Expandir informações de item da listagem
Cadastro	Recuperação de Dados	Fazer chamada para tela de listagem
Cadastro	Recuperação de Dados	Criar serviço de listagem
Cadastro	Recuperação de Dados	Criar rotas para listagem
Cadastro	Recuperação de Dados	Consumir serviço para listagem
Cadastro	Recuperação de Dados	Criar tela para visualizar detalhes
Cadastro	Recuperação de Dados	Recuperar dados do banco de dados
Cadastro	Atualizar dados	Criar tela alteração
Cadastro	Atualizar dados	Fazer validação de dados no cliente.
Cadastro	Atualizar dados	Fazer chamada para tela de alteração
Cadastro	Atualizar dados	Criar serviço alteração
Cadastro	Atualizar dados	Validar Dados no Servidor
Cadastro	Atualizar dados	Consumir serviço para alteração
Cadastro	Atualizar dados	Associar entidades no banco de dados
Cadastro	Atualizar dados	Atualizar dados banco de dados
Cadastro	Inserir dados	Criar tela inserção
Cadastro	Inserir dados	Criar entidade no banco de dados
Cadastro	Inserir dados	Criar rota de inserção de dados
Cadastro	Inserir dados	Fazer validação de dados no cliente
Cadastro	Inserir dados	Fazer chamada para tela de inserção
Cadastro	Inserir dados	Criar método de inserção
Cadastro	Inserir dados	Validar dados inserção no servidor
Cadastro	Inserir dados	Consumir serviço para inserção
Cadastro	Inserir dados	Persistir dados banco de dados
Cadastro	Modificar inserção de dados	Modificar tela inserção
Cadastro	Modificar inserção de dados	Modificar entidade no banco de dados
Cadastro	Modificar inserção de dados	Modificar rota de inserção de dados
Cadastro	Modificar inserção de dados	Modificar validação de dados no cliente
Cadastro	Modificar inserção de dados	Modificar chamada para tela de inserção
Cadastro	Modificar inserção de dados	Modificar método de inserção

Cadastro	Modificar inserção de dados	Modificar validação dados inserção no servidor
Cadastro	Modificar inserção de dados	Consumir serviço para inserção
Cadastro	Modificar inserção de dados	Persistir dados banco de dados
Cadastro	Remover dados	Alterar tela para exclusão
Cadastro	Remover dados	Criar serviço exclusão
Cadastro	Remover dados	Criar rotas para remoção de dados
Cadastro	Remover dados	Consumir serviço para exclusão
Cadastro	Remover dados	Criar modal de confirmação e toast de feedback
Cadastro	Remover dados	Remover dados banco de dados
Gerencial	Visualizar Dashboard	Criar tela do dashboard
Gerencial	Visualizar Dashboard	Declarar no provider
Gerencial	Visualizar Dashboard	Adicionar verificação de autorização
Gerencial	Visualizar Dashboard	Adicionar menus no dashboard
Gerencial	Exportar relatório em PDF	Adicionar botão para exportação do relatório
Gerencial	Exportar relatório em PDF	Criar Relatório
Gerencial	Exportar relatório em PDF	Modificar relatório
Gerencial	Exportar relatório em PDF	Criar dialog para pegar o titulo do PDF
Gerencial	Exportar relatório em PDF	Adicionar campo de titulo no dialog
Gerencial	Exportar relatório em PDF	Integrar back com front
Gerencial	Exportar relatório em PDF	Criar serviço para exportação de relatórios
Gerencial	Exportar relatório em PDF	Exportar Gráfico
Gerencial	Exportar relatório em PDF	Implementar metodo para exportação de relatório em PDF
Gerencial	Exportar relatório em XLS	Adicionar botão para exportação do relatório
Gerencial	Exportar relatório em XLS	Criar dialog para pegar o título do relatório
Gerencial	Exportar relatório em XLS	Adicionar campo de titulo no dialog
Gerencial	Exportar relatório em XLS	Integrar back com front
Gerencial	Exportar relatório em XLS	Criar serviço para exportação de relatórios
Gerencial	Exportar relatório em XLS	Implementar método para exportação de relatório
Gerencial	Notificar via E-mail	Criar serviço de envio de e-mail
Gerencial	Notificar via E-mail	Implementar metodo para envio do e-mail

Gerencial	Notificar via E-mail	Agendar notificação
Gerencial	Notificar via E-mail	Recuperar dados para envio do e-mail
Gerencial	Notificar via E-mail	Consumir API externa
Gerencial	Notificar via E-mail	Criar Layout do E-mail
Gerencial	Notificar via E-mail	Recuperar dados referentes ao e-mail
Gerencial	Notificar via aplicação	Adicionar no dashboard uma aba de notificações
Gerencial	Notificar via aplicação	Criar view do card de notificação
Gerencial	Notificar via aplicação	Criar serviço de notificação via Firebase no backend
Gerencial	Notificar via aplicação	Recuperar dados do usuário que receberá a notificação
Gerencial	Notificar via aplicação	Recuperar dados da notificação
Gerencial	Notificar via aplicação	Criar serviço para conexão com firebase no frontend
Gerencial	Notificar via aplicação	Atualizar aba notificações com os novos cards de notificação