MATHEUS SILVA FERREIRA

UMA ABORDAGEM VISUAL PARA APOIAR GERENTES DE PROJETOS DE SOFTWARE A COMPREENDER O TRABALHO DOS DESENVOLVEDORES

LAVRAS-MG
2021

MATHEUS SILVA FERREIRA

UMA ABORDAGEM VISUAL PARA APOIAR GERENTES DE PROJETOS DE SOFTWARE A COMPREENDER O TRABALHO DOS DESENVOLVEDORES

Dissertação apresentada à Universidade Federal de Lavras, como parte das exigências do Programa de Pós-Graduação em Ciência da Computação, área de concentração em Engenharia de Software e Banco de dados, para a obtenção do título de Mestre.

Prof. Dr. Heitor Augustus Xavier Costa
Orientador

LAVRAS-MG
2021

Ficha catalográfica elaborada pelo Sistema de Geração de Ficha Catalográfica da Biblioteca Universitária da UFLA, com dados informados pelo(a) próprio(a) autor(a).

Ferreira, Matheus Silva.
Uma Abordagem Visual para Apoiar Gerentes de Projetos de Software a Compreender o Trabalho dos Desenvolvedores / Matheus Silva Ferreira. - 2021.
131 p.

Orientador(a): Heitor Augustus Xavier Costa.

Dissertação (mestrado acadêmico) - Universidade Federal de Lavras, 2021.
Bibliografia.

1. Engenharia de Software. 2. Gerenciamento de Projetos. 3. Compreensão do Trabalho de Desenvolvedores.

Matheus Silva Ferreira

UMA ABORDAGEM VISUAL PARA APOIAR GERENTES DE PROJETOS DE SOFTWARE A COMPREENDER O TRABALHO DOS DESENVOLVEDORES

A VISUAL APPROACH TO SUPPORT SOFTWARE PROJECT MANAGERS TO UNDERSTAND DEVELOPERS' WORK

Dissertação apresentada à Universidade Federal de Lavras, como parte das exigências do Programa de Pós-Graduação em Ciência da Computação, área de concentração em Engenharia de Software e Banco de dados, para a obtenção do título de Mestre.

APROVADA em 26 de agosto de 2021.
Dr. Paulo Afonso Parreira Júnior, UFLA
Dr. Ivan do Carmo Machado, UFBA

Prof. Dr. Heitor Augustus Xavier Costa
Orientador

LAVRAS-MG
2021

AGRADECIMENTOS

Agradeço a Deus pelos dons concedidos que permitiram a concretização deste trabalho. Agradeço também aos meus pais, José Geraldo Ferreira e Jandira Conceição Silva Ferreira, e ao meu irmão, Gustavo Silva Ferreira, por todo apoio e compreensão. Agradeço a Késsia Magalhães Espíndola por seu companheirismo e sua paciência. Agradeço aos amigos da República Tsunami e da Universidade Federal de Lavras. Ressalto que os resultados alcançados foram possíveis graças ao auxílio de todos os profissionais do Programa de Pós-Graduação em Ciência da Computação, em especial ao meu orientador, Prof. Dr. Heitor Augustus Xavier Costa. O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Código de Financiamento 001.

RESUMO

A complexidade inerente ao desenvolvimento de software motiva a utilização das práticas de gerenciamento de projetos. O Gerente de Projetos (GP) é o profissional responsável por executar as práticas de gerenciamento e uma de suas principais atividades é gerenciar as pessoas envolvidas no projeto. Essa atividade implica, entre outros aspectos, na montagem de equipes, reconhecimento pelo trabalho realizado por desenvolvedores e distribuição do conhecimento no projeto. Nesse sentido, para executar um bom gerenciamento de projetos, informações sobre o trabalho dos desenvolvedores podem ser valiosas. Dessa forma, diferentes estratégias foram propostas para fornecer tais informações ao GP, permitindo quantificar o trabalho realizado pelos desenvolvedores. Algumas dessas estratégias incluem a mineração de Sistemas de Controle de Versão (SCV) para obter informações e a aplicação de técnicas de visualização para apresentar os resultados ao GP. Apesar da existência dessas estratégias, ainda existem lacunas a serem exploradas, como considerar a evolução do software ao visualizar o quanto os desenvolvedores trabalharam, bem como fornecer a visualização de diferentes perspectivas, incluindo informações do projeto e do trabalho individual dos desenvolvedores em múltiplos níveis de granularidade. Neste trabalho, o objetivo é apoiar Gerentes de Projetos a compreender o trabalho dos desenvolvedores utilizando uma abordagem para visualização de medidas quantitativas aplicadas sobre informações mineradas de SCV. Para tanto, foi realizada uma pesquisa organizada em três etapas (Fundamentação, Construção e Avaliação). Na etapa Fundamentação, há o levantamento do estado da arte, proporcionando a identificação de diretrizes e lacunas deixadas pelas abordagens existentes. Na etapa Construção, há a elaboração da abordagem e sua implementação como uma ferramenta computacional. Na etapa Avaliação, há a verificação dos efeitos da abordagem proposta sobre a tarefa de quantificar o trabalho dos desenvolvedores. Os resultados do trabalho incluem: i) a organização das informações utilizadas para quantificar o trabalho dos desenvolvedores; ii) a definição da abordagem *Developer Tracker*; iii) a construção de *Developer Tracker App*, um apoio computacional para automatizar o uso da abordagem; iv) o relato de desafios encontrados para conduzir estudos experimentais no contexto da indústria de software; e iv) efeitos do uso da abordagem na compreensão de 16 gerentes de projetos a respeito do trabalho dos desenvolvedores. Em trabalhos futuros podem ser realizadas evoluções na abordagem e no apoio computacional para aumentar sua aderência na indústria de software.

Palavras-chave: Engenharia de Software. Gerenciamento de Projetos. Compreensão do Trabalho de Desenvolvedores.

ABSTRACT

The complexity inherent in software development motivates the use of project management practices. The Project Manager (PM) is the professional responsible for performing the management practices, and one of his main activities is to manage the people involved in the project. That activity implies, among other aspects, building the team, recognizing the developers' work, and distributing the project knowledge. Thus, information about the developers' work can be valuable to perform good project management. Therefore, different strategies have been proposed to provide such information to the PM, allowing quantifying the work done by developers. Some of these strategies include mining Version Control Systems (VCS) to obtain information and apply visualization techniques to present the results to the PM. Despite the existence of these strategies, there are still gaps to be explored, such as considering the software evolution when visualizing how much work developers have done and providing visualization from different perspectives, including information from the project and individual developers' work at multiple levels of granularity. In this paper, the goal is to support Project Managers in understanding developers' work using an approach for visualizing quantitative measures applied on VCS mined information. We conducted one research organized in three stages (Rationale, Construction, and Evaluation). In the Rationale stage, we surveyed state of the art for identifying guidelines and gaps left by existing approaches. In the Construction stage, we elaborated and implemented the approach as a computational tool. In the Evaluation stage, we verified the effects of the approach on the developers' work. The results of the work include: i) the organization of the information is used to quantify the developers' work; ii) the definition of the Developer Tracker approach; iii) the construction of the Developer Tracker App, one computational support to automate the use of the approach; iv) the report of challenges encountered to conduct experimental studies in the context of the software industry; and iv) effects of using the approach on the understanding of 16 project managers about the developers' work. Future work can evolve the approach and computational support to increase its adherence in the software industry.

Keywords: Software Engineering. Project Management. Developer’s Work Understand.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

O desenvolvimento de software é considerado uma tarefa complexa que envolve pessoas e exige diversos recursos, por exemplo, tempo e conhecimento técnico. Tal complexidade motivou as empresas a reconhecer a importância do gerenciamento de projetos (ou gestão de projetos) para apoiar o desenvolvimento de sistemas de software (KERZNER, 2016). Quando gerenciados, os projetos possuem menor chance de falha e, por outro lado, apresentam maior possibilidade de atingirem seus objetivos (PRESSMAN, R.; MAXIM, 2016). Assim, a gestão de projetos é apontada como uma importante área da Engenharia de Software (SOMMERVILLE, 2019). Além disso, é um tema que, progressivamente, vem conquistando espaço na literatura (JUNIOR *et al.*, 2019).

O Gerente de Projetos é o profissional responsável por executar as práticas de gestão, liderando a equipe encarregada de executar um projeto em prol dos objetivos estabelecidos (PMI, 2017). Assim, uma das atividades consideradas mais importantes para os Gerentes de Projetos é gerenciar as pessoas envolvidas com o desenvolvimento de sistemas de software (SOMMERVILLE, 2019). Essa atividade envolve tomar decisões referentes à montagem de equipes, à definição de papéis e responsabilidades, ao suporte para o trabalho colaborativo, à motivação das pessoas e ao reconhecimento pelo esforço dos membros da equipe (FERREIRA, M. S. *et al.*, 2020).

Compreender o trabalho realizado pelos desenvolvedores e os seus impactos no projeto de software pode fornecer informações valiosas para o Gerente de Projetos gerenciar as pessoas. Por exemplo, quantificando o conhecimento que cada desenvolvedor possui sobre o código fonte, é possível identificar a distribuição de conhecimento no projeto. Essa é uma importante medida para controlar riscos relacionados à gestão de pessoas, como a rotatividade de desenvolvedores (FERREIRA, M. *et al.*, 2016; FERREIRA, M.; VALENTE; FERREIRA, 2017; RICCA; MARCHETTO, 2010). Apoiado na distribuição de conhecimento, o Gerente de Projetos pode tomar ações para o domínio sobre o código fonte não ficar concentrado em uma ou poucas pessoas, bem como pode trabalhar para evitar que pessoas importantes deixem a equipe de desenvolvimento antes do previsto.

Outro exemplo é a quantificação da contribuição dos desenvolvedores nos artefatos do projeto. Compreender a contribuição dos desenvolvedores pode fornecer informações sobre o domínio técnico de cada membro da equipe e o quanto um desenvolvedor está participando da evolução do software, apoiando tomadas de decisão referentes à montagem de equipes, à

distribuição de papéis, aos reajustes de remuneração, entre outros (LIMA, J. *et al.*, 2015; LIU; YOKOYAMA, 2015; ZUSER; GRECHENIG, 2003).

1.1 Justificativa

O conhecimento empírico do Gerente de Projetos pode ser insuficiente para boa compreensão do trabalho realizado pelos desenvolvedores do projeto. Nesse sentido, a mineração de informações em Sistemas de Controle de Versão (SCV) tem sido utilizada como forma de obter informações sobre os desenvolvedores e/ou sobre os sistemas de software (DE BASSI *et al.*, 2018; FEINER; ANDREWS, 2018; GAO; LIU, 2015; GONZÁLEZ-TORRES *et al.*, 2016; MOURA; NASCIMENTO; ROSA, 2014; SHARMA; KAULGUD, 2012). Os SCV contêm registros de atividades e de modificações realizadas por desenvolvedores nos artefatos ao longo das versões do projeto. Ao minerar essas informações, medidas quantitativas para compreender o trabalho individual de cada desenvolvedor (JARUCHOTRATTANASAKUL *et al.*, 2016; SCHWIND; WEGMANN, 2008) e medidas para identificar a quantidade de trabalho empregada no projeto podem ser aplicadas (FEINER; ANDREWS, 2018; XIE; POSHYVANYK; MARCUS, 2006), sendo possível combiná-las (GONZÁLEZ-TORRES *et al.*, 2016; SHARMA; KAULGUD, 2012).

Aliada à mineração de SCV, a apresentação visual das informações tem sido utilizada como parte das estratégias de compreensão do trabalho dos desenvolvedores. Particularmente, técnicas de visualização[1] são elaboradas para permitir a interpretação de informações complexas e diversificadas extraídas de SCV (FEINER; ANDREWS, 2018; GAO; LIU, 2015; GONZÁLEZ-TORRES *et al.*, 2016; JARUCHOTRATTANASAKUL *et al.*, 2016; SHARMA; KAULGUD, 2012). Essas técnicas utilizam metáforas visuais que facilitam a identificação de padrões nas informações apresentadas.

Apesar da existência dessas estratégias, ainda existem lacunas a serem exploradas, como considerar a evolução de sistemas de software ao visualizar o quanto os desenvolvedores trabalharam. Dessa forma, é possível, por exemplo, comparar a atuação dos desenvolvedores em versões diferentes do projeto. Outro ponto que pode ser mais explorado é a visualização de diferentes perspectivas, incluindo informações do projeto e do trabalho individual dos desenvolvedores em múltiplos níveis de granularidade.

[1] As técnicas de visualização permitem que as pessoas visualizem grandes quantidades de dados de maneira rápida e eficiente, bem como facilita a percepção de *insights* e de padrões sobre um objeto de análise (COUTO, 2018; SVIOKLA, 2009).

1.2 Objetivo

Neste trabalho, o objetivo é apoiar Gerentes de Projetos a compreender do trabalho dos desenvolvedores utilizando uma abordagem para visualização de medidas quantitativas aplicadas sobre informações mineradas de um SCV. Para atingir esse objetivo, foram definidos os seguintes objetivos específicos:

a) Investigar o estado da arte em relação às estratégias para os Gerentes de Projetos compreenderem o trabalho dos desenvolvedores;
b) Elaborar uma abordagem visual com medidas quantitativas sobre o trabalho dos desenvolvedores;
c) Desenvolver um apoio computacional (sistema *web* que implementa a abordagem proposta, viabilizando sua avaliação);
d) Avaliar a abordagem proposta no contexto da indústria de software.

1.3 Estrutura do Trabalho

O restante do trabalho está organizado da seguinte forma. O método de pesquisa utilizado na condução deste trabalho é descrito no Capítulo 2. Referencial teórico sobre Gestão de Projetos, Sistemas de Controle de Versão (SCV) e Compreensão e Visualização de Sistemas de Software é apresentado no Capítulo 3. Um mapeamento sistemático da literatura sobre as estratégias empregadas para compreender o trabalho dos desenvolvedores e as decisões apoiadas por elas são exibidas do Capítulo 4. Os detalhes da abordagem proposta são descritos no Capítulo 5. O apoio computacional desenvolvido para apoiar a avaliação da abordagem proposta é mostrado no Capítulo 6. A avaliação da abordagem proposta e a discussão dos resultados são relatados no Capítulo 7. Trabalhos relacionados estão resumidos no Capítulo 8. Conclusões, contribuições, limitações e trabalhos futuros são apresentadas no Capítulo 9.

2 MÉTODO DE PESQUISA

Neste capítulo, são apresentadas a classificação da pesquisa e as etapas de desenvolvimento deste trabalho. A classificação da pesquisa sob diferentes pontos de vista é relevante para definir os métodos utilizados (JUNG, 2009), podendo ser classificada em:

a) **Do ponto de vista da natureza**. Uma pesquisa pode ser classificada em (a) básica, que objetiva gerar conhecimentos novos e úteis para o avanço da ciência, sem aplicação prática prevista, ou (b) aplicada, que objetiva gerar conhecimentos para aplicação prática, dirigidos à solução de problemas específicos. Este trabalho pode ser considerado uma pesquisa **aplicada**, visto que o objetivo é gerar conhecimentos que possam ser aplicados na prática para contribuir com a compreensão do trabalho de desenvolvedores, apoiando a gestão de projetos;

b) **Do ponto de vista da abordagem**. Uma pesquisa pode ser classificada em (a) quantitativa, as opiniões e as informações são traduzidas em números de modo a classificá-las e analisá-las, ou (b) qualitativa, há relação dinâmica entre o mundo real e o sujeito, não sendo traduzível em números. Considerando como foi abordado o problema, este trabalho pode ser considerado uma pesquisa **qualitativa**, focando na identificação e na interpretação de fenômenos que ocorrem ao longo do ciclo de vida de projetos de software e são relevantes para o Gerente de Projetos compreender o trabalho dos desenvolvedores;

c) **Do ponto de vista da finalidade**. Uma pesquisa pode ser classificada em (a) exploratória, quando existe pouco conhecimento acumulado e sistematizado, (b) descritiva, quando a pesquisa é direcionada para mostrar características de fenômenos ou populações, (c) explicativa, tenta tornar algo compreensível por meio de justificativa dos motivos, (d) metodológica, refere-se à elaboração de instrumentos de captação ou de manipulação da realidade, ou (e) intervencionista, interfere na realidade estudada visando modificá-la. Este trabalho pode ser considerado uma pesquisa **exploratória**, obtendo o conhecimento necessário para alcançar o objetivo utilizando a técnica Mapeamento Sistemático da Literatura (MSL), e uma pesquisa **metodológica**, visto que a abordagem proposta para apoiar a compreensão do trabalho dos desenvolvedores dispõe de uma ferramenta para automatizá-la;

d) **Do ponto de vista dos meios de investigação**. Uma pesquisa pode ser classificada em (a) de campo, consiste na investigação empírica no local onde ocorre/ocorreu um fenômeno ou que disponibiliza elementos que o explica, (b) de laboratório, a experiência é realizada

em local restrito, (c) telematizada, utiliza computador e telecomunicações, (d) documental, faz investigação em documentos, (e) bibliográfica, o estudo sistematizado desenvolvido baseia-se em material disponível ao público em geral, (f) experimental, a investigação empírica utiliza variáveis independentes de forma a manipulá-las e controlá-las para observar variações que elas surtem em variáveis dependentes, (g) *ex post facto*, refere-se a um fato ocorrido, (h) participante, introduz a fronteira pesquisador/pesquisado ao contexto do problema investigado, (i) pesquisa-ação, supõe intervencionar de maneira participativa na realidade social, (j) estudo de caso, delimitado a uma ou poucas unidades e tem caráter de detalhamento, ou (k) quase-experimento, constitui uma classe de estudos de natureza empírica a que falta duas das características usuais na experimentação (controle completo e a aleatoriedade na seleção dos grupos). Este trabalho pode ser considerado uma pesquisa **bibliográfica** e um **quase-experimento**. Os problemas que a abordagem proposta busca amenizar foram definidos baseando-se em pesquisas bibliográficas. Além disso, o procedimento adotado para avaliação da abordagem pode ser caracterizado como um quase-experimento, sendo executado com um conjunto de voluntários não identificados.

Com base nessa classificação, foi elaborado o método de pesquisa para a realização deste trabalho (Figura 2.1):

a) **Etapa 1 - Fundamentação**. O objetivo é encontrar estudos que abordam temas relacionados à compreensão do trabalho dos desenvolvedores e à gestão de projetos de software. Os resultados obtidos são apresentados no Capítulo 3 e no Capítulo 4. Essa etapa é composta por duas atividades:

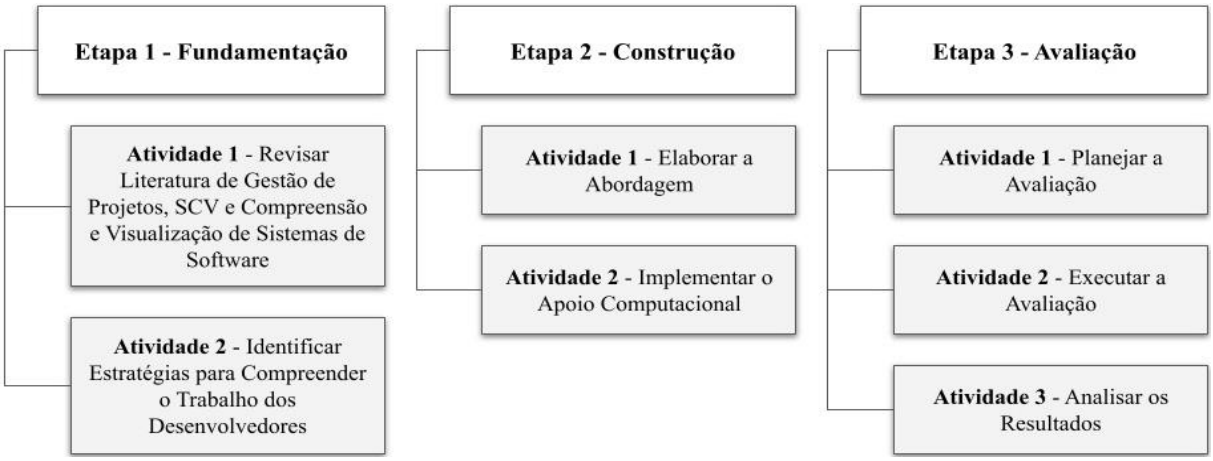
   - **Atividade 1 - Revisar Literatura de Gestão de Projetos, SCV e Compreensão e Visualização de Sistemas de Software**. Consiste na realização de pesquisas *ad-hoc* em livros e em artigos científicos para construir a fundamentação teórica sobre o tema e contextualizar o assunto;
   - **Atividade 2 - Identificar Estratégias para Compreender o Trabalho dos Desenvolvedores**. Consiste em um estudo exploratório, no qual é aplicada a técnica Mapeamento Sistemático da Literatura (MSL) (KITCHENHAM; CHARTERS, 2007). O objetivo desse estudo consiste em identificar quais informações sobre o trabalho dos desenvolvedores são utilizadas pelos Gerentes de Projetos, os meios utilizados para obtê-las e de que modo elas podem apoiar a

prática de gestão de projetos. Os resultados do estudo contribuíram com diretrizes para a elaboração da abordagem proposta neste trabalho;

b) **Etapa 2 - Construção**. O objetivo é elaborar a abordagem e automatizar sua execução por meio de um sistema de software para *web*. Os resultados obtidos são apresentados no Capítulo 5 e no Capítulo 6. Essa etapa é composta por duas atividades:

- **Atividade 1 - Elaborar a Abordagem**. Consiste em utilizar os resultados da Etapa 1 como insumo para a elaboração da abordagem, cujo propósito é fornecer visualmente informações que apoiam a compreensão do trabalho dos desenvolvedores;
- **Atividade 2 - Implementar o Apoio Computacional**. Consiste no desenvolvimento de um apoio computacional (sistema de software para *web*) para (semi) automatizar a abordagem proposta. De modo geral, são coletadas informações sobre o trabalho dos desenvolvedores e apresentadas de forma visual aos Gerentes de Projetos. Esse apoio computacional é um insumo para avaliar a abordagem;

Figura 2.1 - Método de Pesquisa.



Fonte: Do autor (2021).

c) **Etapa 3 - Avaliação**. O objetivo é investigar possíveis benefícios e demais impactos do uso da abordagem proposta para apoiar Gerentes de Projetos a compreender o trabalho dos desenvolvedores. Os resultados são apresentados no Capítulo 7. Essa etapa é composta por três atividades:

- **Atividade 1 - Planejar a Avaliação**. Consiste na elaboração de um plano para coletar de dados que permitam investigar os efeitos da abordagem para seus

usuários. O plano inclui a definição da lista de voluntários (Gerentes de Projetos) e a elaboração do roteiro e do processo de entrevistas;

- **Atividade 2 - Executar a Avaliação.** Consiste em conduzir as entrevistas individualmente com cada voluntário (Gerente de Projetos). Em cada entrevista, é coletado o estado atual da compreensão do trabalho dos desenvolvedores pelo voluntário (sem uso da abordagem). Em seguida, o voluntário deve utilizar a abordagem e são coletados os efeitos (positivos e negativos) do uso da abordagem sobre a compreensão inicial. A utilização da abordagem deve ser realizada com o apoio do sistema de software desenvolvido na Etapa 2. Além disso, os dados devem ser coletados seguindo o roteiro e o processo de entrevistas definidos na atividade anterior;
- **Atividade 3 - Analisar os Resultados.** Os resultados obtidos são analisados de forma qualitativa. A análise qualitativa deve apresentar as percepções e opiniões dos voluntários sobre o uso da abordagem para apoiar a compreensão do trabalho dos desenvolvedores.

3 REFERENCIAL TEÓRICO

3.1 Considerações Iniciais

Neste capítulo, é apresentada a base teórica relacionada aos conceitos abordados nesta pesquisa: Gestão de Projetos, Sistemas de Controle de Versão (SCV) e Visualização de Software.

O restante deste capítulo está organizado da seguinte forma. Na Seção 3.2, são apresentados conceitos relacionados à gestão de projetos e particularidades inerentes aos projetos de software. Na Seção 3.3, são apresentados os recursos existentes em SCV e sua importância para a gestão das atividades relacionadas ao desenvolvimento de sistemas de software. Na Seção 3.4, são exibidos os conceitos básicos de visualização de software, bem como são descritas diretrizes para a utilização de técnicas de visualização.

3.2 Gestão de Projetos

Nesta seção, é apresentada uma base teórica sobre os principais conceitos abordados nesta pesquisa relacionados à gestão de projetos. Para tanto, na Subseção 3.2.1, são descritas diretrizes para realizar a gestão de projetos. Na Subseção 3.2.2, são exibidas as particularidades inerentes à gestão de projetos no contexto de desenvolvimento de software.

3.2.1 Conceitos

O PMBoK (*Project Management Body of Knowledge*) é um documento que possui as melhores práticas para a gestão de projetos (COUTO, 2018), atingindo utilização em mais de 75% dos projetos no mundo (SINGH; LANO, 2014). No PMBoK, projeto é definido como um empreendimento temporário, cuja meta é criar um produto, resultado ou serviço singular (PMI, 2017). A gestão de projetos consiste na aplicação de conhecimentos, de habilidades, de ferramentas e de técnicas que permitem às organizações executarem projetos de forma eficaz e eficiente, caracterizando-se como uma prática de grande importância para as metas do projeto serem alcançadas (PMI, 2017).

Nesse sentido, essa gestão deve ser realizada durante o ciclo de vida do projeto, cuja estrutura tem quatro fases (PMI, 2017): i) início do projeto; ii) organização e preparação; iii) execução do trabalho; e iv) término do projeto. Cada fase pode ser entendida como um estágio que caracteriza uma fatia do projeto, englobando um conjunto de atividades logicamente relacionadas para fornecer uma ou mais entregas. As fases devem passar por um processo de

revisão, sendo uma oportunidade de avaliar o desempenho e tomar medidas necessárias em fases posteriores (PMI, 2017).

Na gestão de projetos, há processos que recebem uma ou mais entradas e, utilizando técnicas e ferramentas apropriadas, produzem uma ou mais saídas (entregas ou resultados). Ao todo, existem quarenta e nove processos definidos no PMBoK. Esses processos devem ser utilizados de forma integrada e, por isso, são agrupados de forma lógica (PMI, 2017):

a) **Processos de Iniciação**. Esses processos englobam os procedimentos necessários para iniciar o projeto ou uma fase em um projeto em andamento;
b) **Processos de Planejamento**. Esses processos permitem traçar como o projeto será conduzido, definindo escopo, objetivos e plano de ações**;**
c) **Processos de Execução**. Esses processos possibilitam a concretização das metas do projeto, executando as tarefas definidas no plano da gestão do projeto;
d) **Processos de Monitoramento e Controle**. Esses processos monitoraram e controlam o progresso do trabalho;
e) **Processos de Encerramento**. Esses processos são destinados à finalização formal do projeto.

Além disso, os processos podem ser organizados por áreas de conhecimento. Essas áreas são interrelacionadas e constituem uma área de gestão de projetos definida por uma série de requisitos e pelos termos dos processos que as compõem (práticas, entradas, saídas, ferramentas e técnicas). O PMBoK considera dez áreas de conhecimento, mas admite que pode haver a necessidade de um projeto específico exigir áreas adicionais (PMI, 2017):

a) **Gerenciamento da Integração do Projeto**. Essa área possui processos para identificar, definir, combinar, unificar e coordenar os demais processos;
b) **Gerenciamento do Escopo do Projeto**. Essa área possui processos para definir o que será e o que não será feito no projeto;
c) **Gerenciamento do Cronograma do Projeto**. Essa área possui processos para apoiar o cumprimento dos prazos;
d) **Gerenciamento dos Custos do Projeto**. Essa área possui processos para realizar planejamento, estimativas, orçamentos, financiamentos, gestão e controle dos custos, de forma que o projeto não exceda o orçamento aprovado;
e) **Gerenciamento da Qualidade do Projeto**. Essa área possui processos para assegurar que o projeto satisfaça as expectativas das partes interessadas;

f) **Gerenciamento dos Recursos do Projeto**. Essa área possui processos para identificar, adquirir e gerenciar recursos, como pessoas, materiais e equipamentos;

g) **Gerenciamento das Comunicações do Projeto**. Essa área possui processos para gerar, coletar, distribuir, armazenar, recuperar e organizar as informações de forma oportuna e adequada;

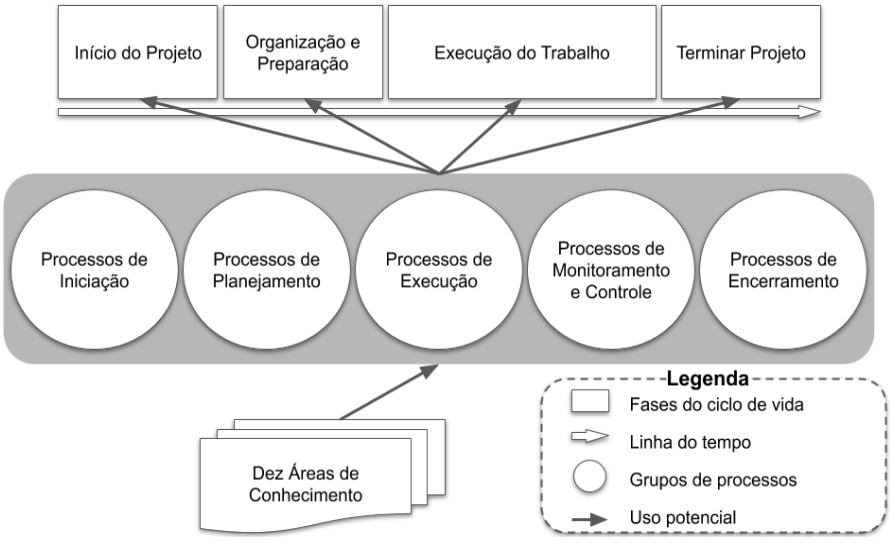
h) **Gerenciamento dos Riscos do Projeto**. Essa área possui processos para planejar, identificar, analisar e responder a riscos do projeto. Os riscos podem ser positivos (oportunidades) ou negativos (ameaças);

i) **Gerenciamento das Aquisições do Projeto**. Essa área possui processos relacionados à compra ou à aquisição de produtos, serviços ou resultados externos à equipe do projeto;

j) **Gerenciamento das Partes Interessadas do Projeto**. Essa área possui processos para gerenciar as partes interessadas nas decisões do projeto. As partes interessadas são pessoas, grupos ou organizações que podem impactar ou serem impactados de alguma forma pelo projeto.

Na Figura 3.1, é apresentada a relação entre as fases do ciclo de vida do projeto, os grupos de processos e as áreas de conhecimento. As áreas de conhecimento fornecem apoio aos processos e cada grupo de processo pode possuir processos para as diversas fases do ciclo de vida do projeto. A relação dos processos com cada área de conhecimento e com cada grupo de processo é exibida na Tabela 3.1.

Figura 3.1 - Inter-relação entre ciclo de vida, grupos de processo e áreas de conhecimento.



Fonte: Adaptado de (PMI, 2017).

A abordagem proposta neste trabalho tem o objetivo de apoiar a compreensão do trabalho dos desenvolvedores. Ter essa compreensão pode auxiliar o Gerente de Projetos a

tomar decisões relacionadas à alocação de pessoas no projeto e à distribuição do conhecimento sobre o código fonte, por exemplo. Essas decisões estão relacionadas às áreas do conhecimento Gerenciamento dos Recursos (considerando recursos humanos) do Projeto e Gerenciamento dos Riscos do Projeto.

Tabela 3.1 - Relação dos processos com os grupos de processos e as áreas de conhecimento.

Áreas de Conhecimento	Grupos de processos de gerenciamento de projetos				
	Iniciação	**Planejamento**	**Execução**	**Monitoramento e Controle**	**Encerramento**
1. Gerenciamento da integração do projeto	1.1 Desenvolver o termo de abertura do projeto	1.2 Desenvolver o plano de gerenciamento do projeto	1.3 Orientar e gerenciar o trabalho do projeto	1.5 Monitorar e controlar o trabalho do projeto	1.7 Encerrar o projeto ou fase
			1.4 Gerenciar o conhecimento do projeto	1.6 Realizar o controle integrado de mudanças	
2. Gerenciamento do escopo do projeto		2.1 Planejar o gerenciamento do escopo		2.5 Validar o escopo	
		2.2 Coletar os requisitos		2.6 Controlar o escopo	
		2.3 Definir o escopo			
		2.4 Criar a estrutura analítica do projeto (EAP)			
3. Gerenciamento do cronograma do projeto		3.1 Planejar o gerenciamento do cronograma		3.6 Controlar o cronograma	
		3.2 Definir as atividades			
		3.3 Sequenciar as atividades			
		3.4 Estimar as durações das atividades			
		3.5 Desenvolver o cronograma			
4. Gerenciamento dos custos do projeto		4.1 Planejar o gerenciamento dos custos		4.4 Controlar os custos	
		4.2 Estimar os custos			
		4.3 Determinar o orçamento			

(continua)

Tabela 3.1 - Relação dos processos com os grupos de processos e as áreas de conhecimento (continuação).

Áreas de Conhecimento	Grupos de processos de gerenciamento de projetos				
	Iniciação	**Planejamento**	**Execução**	**Monitoramento e Controle**	**Encerramento**
5. Gerenciamento da qualidade do projeto		5.1 Planejar o gerenciamento da qualidade	5.2 Gerenciar a qualidade	5.3 Controlar a qualidade	
6. Gerenciamento dos recursos do projeto		6.1 Planejar o gerenciamento dos recursos 6.2 Estimar os recursos das atividades	6.3 Adquirir recursos 6.4 Desenvolver a equipe 6.5 Gerenciar a equipe	6.6 Controlar os recursos	
7. Gerenciamento das comunicações do projeto		7.1 Planejar o gerenciamento das comunicações	7.2 Gerenciar as comunicações	7.3 Monitorar as comunicações	
8. Gerenciamento dos riscos do projeto		8.1 Planejar o gerenciamento dos riscos 8.2 Identificar os riscos 8.3 Realizar a análise qualitativa dos Riscos 8.4 Realizar a análise quantitativa dos riscos 8.5 Planejar as respostas aos riscos	8.6 Implementar respostas aos riscos	8.7 Monitorar os riscos	
9. Gerenciamento das aquisições do projeto		9.1 Planejar o gerenciamento das aquisições	9.2 Conduzir as aquisições	9.3 Controlar as aquisições	
10. Gerenciamento das partes interessadas do projeto	10.1 Identificar as partes interessadas	10.2 Planejar o engajamento das partes interessadas	10.3 Gerenciar o engajamento das partes interessadas	10.4 Monitorar o engajamento das partes interessadas	

Fonte: Adaptado de (PMI, 2017).

3.2.2 Gerente de Projetos de Software

Os projetos de software são desafiadores por três fatores (SOMMERVILLE, 2019): i) o sistema de software é intangível; ii) muitos sistemas de software são únicos e inovadores; e iii) existem variadas metodologias de desenvolvimento e os processos de gestão sofrem adaptações por parte das organizações desenvolvedoras. Os métodos de gestão baseados em fases bem definidas para iniciar, planejar, executar, monitorar e finalizar o projeto são conhecidos como tradicionais (OLIVEIRA, 2020). Quando se utiliza um método de gestão

tradicional, o papel do Gerente de Projetos possui uma definição mais clara, seguindo as diretrizes do PMBoK.

Além desses métodos, existem os métodos de gestão ágil, sendo menos prescritivos que os tradicionais. Enquanto nos métodos de gestão tradicional há a definição de fases para a entrega do projeto, nos métodos de gestão ágil são consideradas entregas parciais em ciclos curtos de tempo, permitindo que o sistema de software seja entregue incrementalmente e de forma contínua durante o desenvolvimento do projeto. Nos métodos de gestão ágil, não é definido explicitamente o papel do Gerente de Projetos, indicando que não é esperado encontrar um profissional exercendo esse papel nos projetos (GANDOMANI *et al.*, 2020). Ao utilizá-los, os integrantes do time que executam o projeto assumem responsabilidades atribuídas ao Gerente de Projetos em métodos de gestão tradicional, como a definição e a organização das atividades de trabalho e o planejamento das entregas.

Contudo, uma abordagem frequentemente utilizada é combinar conceitos de métodos de gestão tradicional e ágil, incluindo o papel do Gerente de Projetos como líder de um time de desenvolvimento ágil (GANDOMANI *et al.*, 2020; OLIVEIRA, 2020; SHASTRI; HODA; AMOR, 2020). Nessa abordagem, o Gerente de Projetos divide a responsabilidade das tomadas de decisão com o time, concedendo autonomia para os desenvolvedores definirem como vão implementar as atividades do projeto ao longo de ciclos curtos de entrega.

Na tentativa de direcionar a atuação do Gerente Projetos no âmbito do desenvolvimento de sistemas de software, pode-se observar os chamados “4 Ps” (Pessoas, Produto, Processo e Projeto) (PRESSMAN, R.; MAXIM, 2016). Nesse desenvolvimento, são considerados o esforço humano e a preparação, motivação e comprometimento das pessoas envolvidas com o projeto. Além disso, as necessidades do cliente precisam ser compreendidas e transmitidas para a equipe de desenvolvimento, fazendo com que esse esforço resulte em um produto que atenda as expectativas de forma adequada. Esse desenvolvimento deve seguir um processo organizado e definido, promovendo mais aproveitamento dos recursos empregados. O plano do projeto deve ser bem elaborado, aumentando o risco de sucesso do projeto. Assim, pode-se considerar que determinado profissional exerce a função de gerente em projetos software, independentemente do método utilizado, quando assume pelo menos uma das seguintes atividades (SOMMERVILLE, 2019): i) planejamento do projeto; ii) geração de relatórios; iii) gestão de riscos; iv) gestão de pessoas; e v) elaboração de propostas.

3.3 Sistemas de Controle de Versão

Nesta seção, são apresentados os conceitos relacionados a Sistemas de Controle de Versão (SCV). Para tanto, na Subseção 3.3.1, são apresentados recursos e definições sobre SCV e como o uso desse tipo de sistema pode apoiar a gestão de projetos. Na Subseção 3.3.2, são destacadas particularidades de repositórios Git[2].

3.3.1 Conceitos

Os SCV têm a finalidade de gerenciar a evolução de um conjunto de dados. Controlar as modificações e os estados dos artefatos do software ao longo do processo de desenvolvimento permite gerenciar, organizar e coordenar as atividades do projeto (ZOLKIFLI; NGAH; DERAMAN, 2018). Nesse sentido, SCV são ferramentas importantes para projetos de software (PMI, 2017). A análise de dados mantidos por SCV pode auxiliar em diversas atividades envolvidas com o desenvolvimento de sistemas de software, por exemplo (XIE; POSHYVANYK; MARCUS, 2006):

a) os Gerentes de Projetos podem utilizar informações de histórico de alterações em artefatos para atribuir tarefas aos desenvolvedores apropriados;
b) testadores podem descobrir quem é responsável pela ocorrência de um *bug*;
c) desenvolvedores podem saber quais partes do sistema de software foram alteradas durante a implementação/correção de uma (um conjunto de) função(ões).

Para compreender o funcionamento de SCV, é importante conhecer os recursos disponíveis nesses sistemas. O principal recurso de um SCV é o repositório, uma biblioteca que centraliza o armazenamento de dados (WU *et al.*, 2004). No contexto de projetos de software, o repositório é comumente chamado de repositório de código. Além disso, o termo "repositório de código" é frequentemente utilizado para referir-se ao próprio SCV. Apesar de o termo incluir a palavra "código", qualquer artefato de software pode ser armazenado no repositório, incluindo registro de *bugs*, itens de documentação, registros de modificações nos dados e histórico de versões. Existem três configurações possíveis para os repositórios (MAJUMDAR *et al.*, 2017):

a) **Repositório Local**. Os dados são armazenados em um banco de dados local no computador do usuário. As versões e o histórico de modificações funcionam como *backup*. Além disso, mesclar as modificações nos dados realizadas de forma paralela por

[2] https://git-scm.com/

pessoas diferentes torna-se uma tarefa árdua e passível de erros, visto que tal tarefa deve ser feita manualmente pelos usuários;

b) **Repositório Centralizado**. Os dados são armazenados em um servidor e computadores clientes podem acessá-los e modificá-los. Após finalizar as modificações, os clientes devem submeter a atualização ao servidor, que verifica a necessidade de mesclar as alterações com outras atualizações realizadas de forma paralela. Essa configuração viabiliza o trabalho paralelo e colaborativo. Os sistemas Subversion[3] e Perforce[4] são exemplos de ferramentas que utilizam essa configuração. Um problema característico dos repositórios centralizados é possível falha no servidor, impossibilitando o acesso aos dados e a submissão de atualizações;

c) **Repositório Distribuído**. Um servidor é utilizado para armazenar o conjunto principal de dados. Computadores clientes podem clonar o repositório por completo e trabalhar em atualizações. Apesar de possuir um servidor como ponto central de dados, essa configuração não é totalmente dependente de tal servidor, visto que as atualizações podem ser confirmadas utilizando o servidor e os clones em clientes. Essa configuração é utilizada em sistemas como o Git e Mercurial[5].

Os SCV possuem outros recursos como relatórios de registros de atividade, ferramentas para comparar versões de um arquivo em cada atualização e suporte à criação de ramificações do repositório (MAJUMDAR *et al.*, 2017; WU *et al.*, 2004). As ramificações possuem os dados da versão principal do repositório, porém evoluem em uma direção diferente da versão principal. Uma situação que exemplifica o uso de ramificações é quando uma função está em desenvolvimento em um arquivo de código. A versão principal do repositório é em uma versão estável do sistema de software e, eventualmente, pode receber manutenções. Ao mesmo tempo, a nova função é desenvolvida na ramificação. Além disso, a ramificação pode ser atualizada com as manutenções realizadas na versão principal. Ao finalizar o desenvolvimento de uma função, a ramificação retorna ao ramo principal (versão principal), incrementando essa função na versão estável do sistema de software.

Os recursos disponíveis em SCV fornecem diversas informações e em diferentes níveis de detalhes sobre a evolução de sistemas de software. Por esse motivo, a mineração de dados em SCV vem sendo utilizada para compreender aspectos como modificações em

[3] https://subversion.apache.org/
[4] https://www.perforce.com/
[5] https://www.mercurial-scm.org/

artefatos, trabalho realizado pelos desenvolvedores e qualidade do código (FEINER; ANDREWS, 2018; GONZÁLEZ-TORRES *et al.*, 2016; JARUCHOTRATTANASAKUL *et al.*, 2016).

3.3.2 Extração de Informações de Repositórios Git

O Git é um SCV *open-source* lançado em 2005 com ampla aceitação da indústria (MAJUMDAR *et al.*, 2017). Além disso, esse SCV é a base dos repositórios vinculados ao GitHub[6], um popular serviço de hospedagem que possui milhões de repositórios Git, incluindo projetos *open-source* (GITHUB, 2020). Existem outros serviços de hospedagem baseados em Git, por exemplo, GitLab[7] e Bitbucket[8].

O Git possui uma aplicação que pode ser instalada em computadores clientes (Git Bash[9]). Essa aplicação permite atualizar os dados e rastrear o histórico de versões do repositório. Além disso, hospedeiros como o GiHub e o GiLab possuem serviços de extração de informações via chamadas HTTP (*Hypertext Transfer Protocol*). Cada hospedeiro possui serviços próprios de extração de informações. Apesar disso, a extração de informações pelo Git Bash funciona da mesma forma para projetos armazenados em qualquer hospedeiro baseado em Git.

3.4 Compreensão e Visualização de Software

Os dados armazenados em SCV contemplam diversos aspectos relacionados a um sistema de software. A quantidade de informações disponíveis torna sua análise uma tarefa árdua. Nesta seção, são apresentadas estratégias para a compreensão de informações complexas relacionadas a esses sistemas. Na Subseção 3.4.1, é apresentado o conceito de compreensão de software. Na Subseção 3.4.2, são descritas estratégias visuais para analisar informações complexas.

3.4.1 Compreensão de Software

A compreensão de software consiste em conhecer a funcionalidade, a estrutura interna e o comportamento de sistemas de software (VON MAYRHAUSER; VANS, 1995). Entre outros aspectos, a compreensão desses sistemas permite aos Gerentes de Projetos e aos desenvolvedores adquirirem entendimento consistente sobre o código fonte (incluindo

[6] https://github.com/
[7] https://about.gitlab.com/
[8] https://bitbucket.org/
[9] https://git-scm.com/downloads

estrutura e qualidade), identificarem os principais contribuidores da equipe e acompanharem a evolução de sistemas de software (FEINER; ANDREWS, 2018). A compreensão desses sistemas não é trivial por causa das seguintes caraterísticas (CARNEIRO, 2011):

a) **Complexidade**. O tamanho e a complexidade da implementação de sistemas de software dificultam a sua compreensão;
b) **Intangibilidade**. Sistemas de software não possuem forma ou tamanho, o que dificulta a sua compreensão utilizando apenas a visão humana;
c) **Evolução**. Sistemas de software evoluem, provocando impactos no tamanho do código, na qualidade da implementação e no aumento e perda de informações (*e.g.*, documentação desatualizada).

A principal fonte de informações sobre sistemas de software é o código fonte (CARNEIRO, 2011), visto que as pessoas envolvida2.1 s no desenvolvimento nem sempre são acessíveis e a documentação, usualmente, apresenta ausência de informações e/ou informações desatualizadas. Esse fator torna o processo de compreensão ainda mais trabalhoso. Desse modo, técnicas de visualização têm sido utilizadas para representar informações de sistemas de software para facilitar a sua compreensão (FEINER; ANDREWS, 2018; GONZÁLEZ-TORRES *et al.*, 2016; JARUCHOTRATTANASAKUL *et al.*, 2016).

3.4.2 Visualização de Software

A visualização de software é uma especialização da visualização de informação, sendo uma área que busca favorecer a compreensão de informações utilizando representações visuais (SPENCE, 2007). Seu objetivo é representar informações de forma visual, considerando os seguintes aspectos de sistemas de software (DIEHL, 2007):

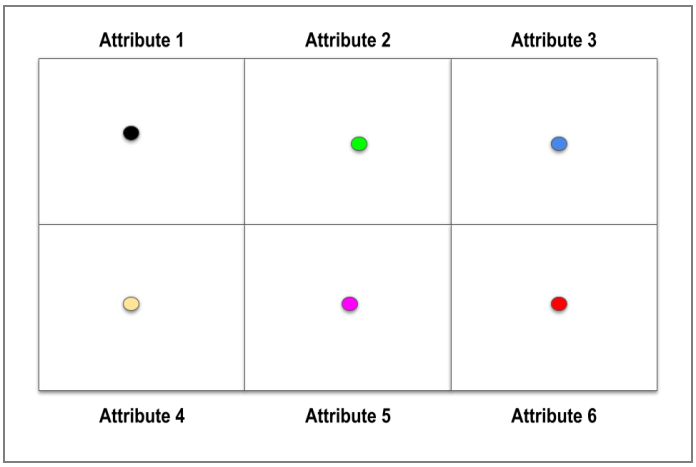
a) **Estáticos**. São constituídos por informações obtidas enquanto os sistemas de software não estão em execução, por exemplo, código fonte e registros de modificações em artefatos. A visualização desses aspectos auxilia na compreensão estrutural desses sistemas;
b) **Dinâmicos**. São definidos por informações obtidas durante a execução de sistemas de software, por exemplo, chamadas de funções e troca de mensagens entre objetos. A visualização desses aspectos auxilia na compreensão comportamental desses sistemas;
c) **Evolutivos**. São compostos por aspectos estáticos organizados em versões, permitindo acompanhar a evolução do desenvolvimento de sistemas de software ao longo do tempo.

A visualização de informações pode ser aplicada para atingir os seguintes propósitos (GARCIA, 2006): i) **análise exploratória**, o objetivo é alcançar conhecimento; ii) **análise**

confirmatória, o objetivo é confirmar ou rejeitar conhecimento prévio; iii) **apresentação**, o objetivo é apresentar informações previamente conhecidas; e iv) **apoio à decisão**, o objetivo é navegar por modelos de dados resultantes de um processo de busca. Na análise exploratória, é comum utilizar técnicas de visualização de informação multidimensional, permitindo a representação de diferentes tipos de dados. Tais técnicas são relevantes para a compreensão dos dados, pois instigam a habilidade humana de encontrar padrões (CARNEIRO, 2011). As técnicas de visualização podem ser classificadas em quatro paradigmas (metáforas de visualização) (KEIM; KRIEGEL, 1996):

a) **Orientado a *pixel***. Um conjunto de dados com `N` atributos é mapeado em `N` *pixels* que recebem cores de acordo com o valor de cada atributo, o que permite identificar padrões ao observar a distribuição e a coloração dos *pixels* em uma janela. Na Figura 3.2, é apresentado um exemplo de seu uso, onde cada *pixel* é posicionado em uma janela;

Figura 3.2 - Técnica baseada no paradigma orientado a *pixel*.



Fonte: Adaptado de (KEIM, 2000).

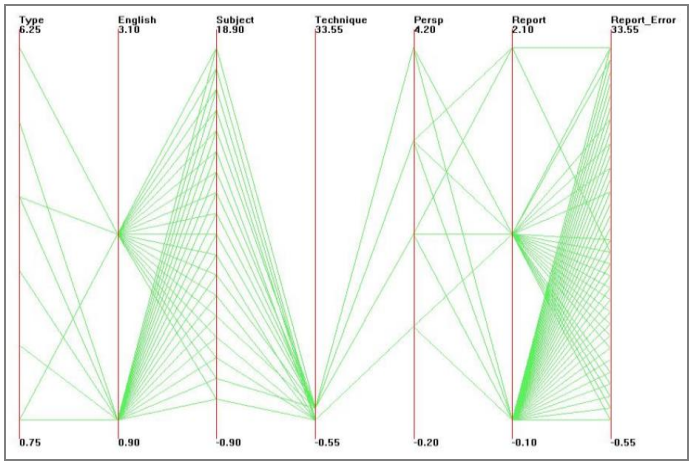
b) **Geométrico**. As projeções têm duas ou três dimensões, permitindo analisar a distribuição dos dados e a relação entre os atributos. Na Figura 3.3, é apresentado um exemplo de seu uso, onde há a visualização de sete atributos em coordenadas paralelas;

c) **Hierárquico**. Os componentes de sistemas de software são apresentados de forma hierárquica (*e.g.*, estrutura de herança entre classes), permitindo visualização estrutural desses sistemas. Na Figura 3.4, é apresentado um exemplo de seu uso, onde há uma árvore organizada hierarquicamente por arquivos. Cada folha é um defeito e o tamanho da folha representa a quantidade de vezes que o defeito foi reportado;

d) **Iconográfico**. As informações são representadas utilizando ícones que possuem conexões entre si. A definição de ícones é livre, mas escolher/criar ícones representativos é essencial para técnicas desse paradigma, visto que ícones pouco significativos podem

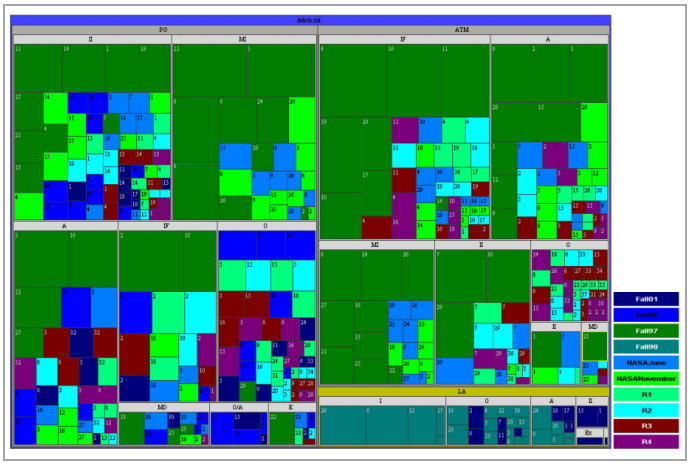
prejudicar a compreensão. Na Figura 3.5, é apresentado um exemplo de seu uso com a técnica iconográfica *SolarSystem*. Os pacotes são ilustrados como “sol” e as classes são apresentadas como planetas do “sistema solar” de cada pacote. O tamanho da órbita de uma classe representa a sua profundidade na hierarquia de herança (YANG; GRAHAM, 2003).

Figura 3.3 - Técnica baseada no paradigma geométrico.



Fonte: Adaptado de (GARCIA, 2006).

Figura 3.4 - Técnica baseada no paradigma hierárquico.



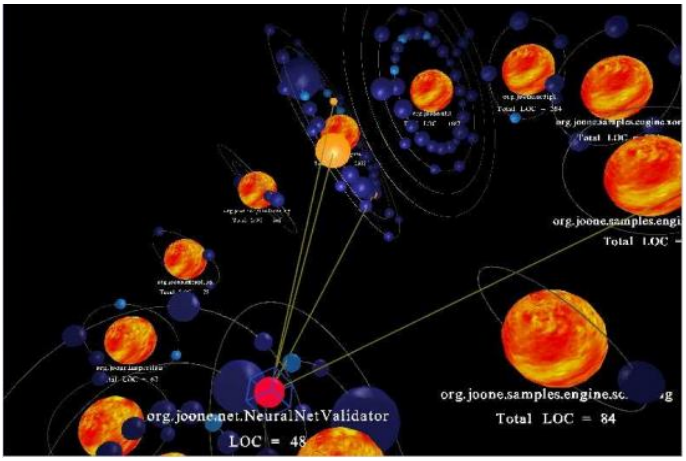
Fonte: Adaptado de (GARCIA, 2006).

Além da utilização de paradigmas visuais, as técnicas de visualização podem incluir recursos interativos, aprimorando a visualização exploratória (KEIM, 2002). Recursos interativos permitem alterar dinamicamente a visualização de três maneiras (CARNEIRO, 2011):

a) ***Zoom***. O campo de visão pode ser ampliado ou focado, tornando possível calibrar o nível de detalhes e de informações disponíveis;

b) **Filtro**. Um subconjunto do conjunto total de dados é exibido, permitindo controlar o fluxo de informações apresentadas ou ocultadas no campo de visão com o intuito de facilitar o processo de análise;
c) **Distorção**. A representação gráfica é exibida com diferentes níveis de detalhes em uma única visão, possibilitando visualizar regiões de interesse detalhadamente sem perder a visão global.

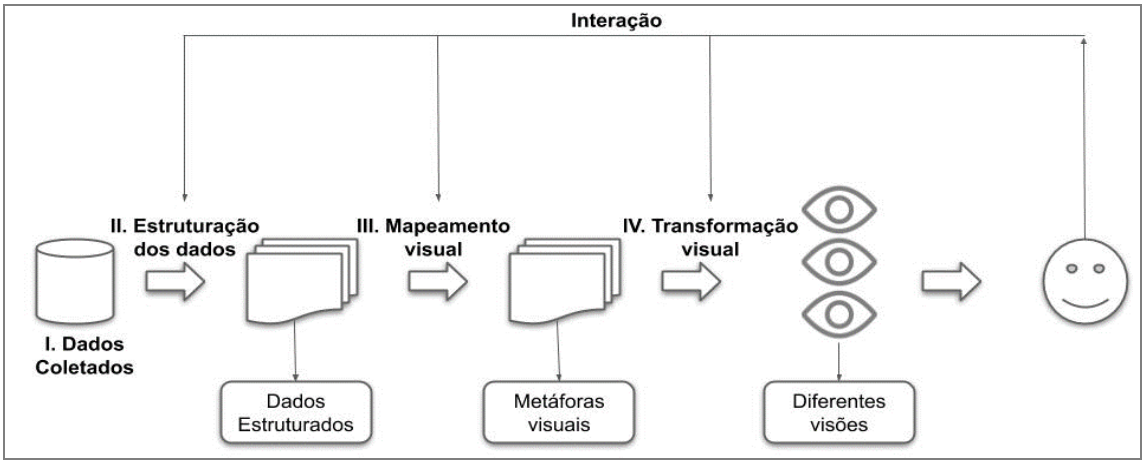
Figura 3.5 - Técnica *SolarSystem* baseada no paradigma iconográfico.



Fonte: (YANG; GRAHAM, 2003)

Em dois trabalhos (CARD, 1999; CARNEIRO, 2011), os autores apresentaram modelos de referência para a visualização de sistemas de software. Em um trabalho (CARD, 1999), são especificadas as seguintes etapas: i) obtenção de dados a partir de repositórios disponíveis; ii) transformação e estruturação dos dados com o objetivo de organizá-los; iii) mapeamento dos dados em estruturas visuais para construir metáforas de visualização; iv) instanciação de metáforas em visualizações concretas. No outro trabalho (CARNEIRO, 2011), é observada a complexidade das informações inerentes aos sistemas de software para adicionar múltiplas perspectivas de visualização para permitir “enxergar” variedade maior de aspectos desses sistemas, o que proporciona melhor compreensão. O modelo de visualização com múltiplas perspectivas é apresentado na Figura 3.6, na qual há a instanciação de diversas visões. Isso permite que diferentes metáforas sejam geradas a partir dos dados, resultando na apresentação de perspectivas distintas, e *zoom* e filtro podem ser utilizados para modificar o conjunto de informações apresentadas. Dependendo da interação, pode ser necessário refazer algumas etapas do modelo para instanciar novamente as visões, considerando os dados transformados.

Figura 3.6 - Modelo de visualização com múltiplas perspectivas.



Fonte: Adaptado de (CARNEIRO, 2011).

3.5 Considerações Finais

Neste capítulo, foram abordados conceitos relacionados à Gestão de Projetos, incluindo as particularidades inerentes aos projetos de software e ao papel do Gerente de Projetos. Além disso, foi apresentada uma visão geral sobre SCV, destacando os dados armazenados por esses sistemas e sua extração para análise. Esses dados podem ser utilizados como insumos para identificar o trabalho realizado em artefatos de software durante a sua evolução. Apesar disso, ao considerar elevado volume de dados e em diferentes níveis de detalhes, aumenta-se a complexidade da tarefa de obter informações úteis de tais dados. Nesse sentido, foi destacada a utilidade da visualização de software para compreender aspectos complexos de sistemas de software. Também, foram abordadas diretrizes para a utilização de técnicas de visualização.

O foco deste trabalho é apoiar Gerentes de Projetos a compreender o trabalho dos desenvolvedores utilizando uma abordagem visual que considera um conjunto de medidas aplicadas sobre dados extraídos de SCV. A abordagem possui uma técnica de visualização baseada no paradigma iconográfico, permitindo realizar uma análise exploratória ao relacionar a quantidade de trabalho executada por desenvolvedores do projeto em cada artefato ao longo das versões do sistema de software. Além disso, há a utilização de *zoom* e de filtros como recursos interativos, bem como duas perspectivas são consideradas. Uma das perspectivas foi definida com o objetivo de fornecer uma visão geral sobre aspectos estruturais de versões de um sistema de software. Outra perspectiva tem a função de apresentar o trabalho realizado pelos membros da equipe do projeto.

4 COMPREENSÃO DO TRABALHO DOS DESENVOVLEDORES - ESTADO DA ARTE

4.1 Considerações Iniciais

Neste capítulo, é descrito o estado da arte referente a tarefa dos Gerentes de Projetos de compreender o trabalho dos desenvolvedores, abordando estratégias existentes e destacando a relevância de SCV e técnicas de visualização de software para a execução dessa tarefa. Como parte desta pesquisa, foi realizado um estudo exploratório com o objetivo de apresentar quais informações são utilizadas para mensurar o trabalho realizado no projeto, os meios utilizados para obtê-las e de que modo elas podem apoiar Gerentes de Projetos a compreender do trabalho dos desenvolvedores (FERREIRA, M. S. *et al.*, 2019). Posteriormente, esse estudo foi atualizado e estendido, incluindo opiniões de quatro Gerentes de Projetos sobre os resultados obtidos (FERREIRA, M. S. *et al.*, 2020). O estudo foi atualizado novamente considerando artigos publicados até março de 2021.

O restante deste capítulo está organizado da seguinte forma. Na Seção 4.2, é apresentada uma visão geral referente ao estudo exploratório sobre a mensuração do trabalho dos desenvolvedores. Na Seção 4.3, há uma discussão sobre a utilidade de SCV e técnicas de visualização de software para apoiar os Gerentes de Projetos a compreender o trabalho dos desenvolvedores.

4.2 Visão Exploratória da Mensuração do Trabalho dos Desenvolvedores

Nesta seção, é apresentada uma visão geral do estudo exploratório sobre a mensuração do trabalho dos desenvolvedores (FERREIRA, M. S. *et al.*, 2020), fornecendo uma visão acadêmica e industrial. Para tanto, na Subseção 4.2.1, é descrito o processo de seleção de estudos na literatura. Na Subseção 4.2.2, são apresentados os resultados do estudo.

4.2.1 Seleção de Estudos

Para realização do estudo exploratório foi utilizada a técnica Mapeamento Sistemático da Literatura (MSL), uma técnica que permite a identificação, a interpretação e a avaliação de evidências disponíveis em estudos referentes a determinado tópico, fenômeno ou conjunto de questões de pesquisa de interesse (KITCHENHAM, 2004). O MSL é realizado em três fases principais (KITCHENHAM; CHARTERS, 2007): i) **Planejamento**, definem-se a motivação, os objetivos e o protocolo de pesquisa; ii) **Execução**, aplica-se a estratégia de busca definida

no protocolo de pesquisa para identificação e seleção de estudos; e iii) **Apresentação dos Resultados**, apresenta-se a análise das informações obtidas com os estudos selecionados. Além da realização do MSL, foram coletadas opiniões de quatro Gerentes de Projetos que trabalham na indústria de software. Essas opiniões são referentes aos resultados obtidos no MSL, fazendo com que o estudo considere a visão da literatura e a visão de algumas pessoas ligadas à indústria.

Na fase Planejamento, foi elaborado o protocolo de pesquisa. De forma simplificada, o protocolo de pesquisa pode ser definido pelos seguintes itens:

a) **Questões de pesquisa**. Foram elaboradas seis questões de pesquisa (QP) que fornecem insumos para atender aos objetivos do MSL e uma visão sobre as características dos estudos científicos encontrados no MSL;
   - **QP1**. "Quais são as medidas utilizadas pelos Gerentes de Projetos para mensurar o trabalho dos desenvolvedores?"

     Justificativa: o objetivo é identificar informações utilizadas para os Gerentes de Projetos mensurarem o trabalho dos desenvolvedores;
   - **QP2**. "Como as medidas são aplicadas pelos Gerentes de Projetos para acompanhar o trabalho dos desenvolvedores?"

     Justificativa: o objetivo é identificar como os Gerentes de Projetos extraem e analisam as informações para mensurar o trabalho dos desenvolvedores;
   - **QP3**. "De que modo as informações sobre o trabalho dos desenvolvedores apoiam o gerenciamento de projetos?"

     Justificativa: o objetivo é identificar a importância das informações sobre o trabalho dos desenvolvedores para apoiar a gestão de projetos;
   - **QP4**. "Qual o tipo de solução frequentemente proposta em estudos nesta área?".

     Justificativa: o objetivo é identificar o tipo de solução proposta em estudos para mensurar o trabalho dos desenvolvedores e apoiar a tomada de decisão do Gerente de Projetos;
   - **QP5**. "Qual é o tipo de metodologia de pesquisa frequentemente utilizada em estudos nesta área?"

     Justificativa: o objetivo é identificar a metodologia de pesquisa utilizada em estudos para verificar sua maturidade;
   - **QP6**. "Como a solução proposta se relaciona com a metodologia de pesquisa nos estudos incluídos?"

Justificativa: o objetivo é verificar a maturidade das soluções apresentadas nos estudos. Por exemplo, saber se a mensuração do trabalho dos desenvolvedores foi verificada empiricamente;

b) **Repositórios de estudos científicos**. Os repositórios utilizados para buscar estudos referentes ao tema foram ACM *Digital Library*[10], *Ei Compendex*[11], IEEE *Xplore*[12], Scopus[13] e *SpringerLink*[14];

c) ***String* de busca.** A *string* de busca utilizada nos repositórios de estudos científicos é apresentada na Tabela 4.1.

Tabela 4.1 - *String* de Busca.

(measure OR measurement OR mensuration OR dimension OR evaluation OR analyze OR analysis OR view OR visualization OR knowledge)
AND
(contribution OR participation OR productivity OR skills OR collaboration OR effort OR knowledge)
AND
(developers OR "software development team" OR "team members")
AND
("software project manager" OR "project manager" OR "project managers" OR "software project" OR "project management")
AND
(tool OR framework OR plugin OR method OR metric OR factors)

Fonte: Do Autor (2021).

d) **Critérios de inclusão e de exclusão**. Foram elaborados critérios para incluir/excluir estudos retornados com a aplicação da *string* de busca nos repositórios de estudos científicos. O critério para incluir estudos consiste em selecionar estudos primários que abordem a mensuração do trabalho dos desenvolvedores para apoiar as atividades do Gerente de Projetos. Os critérios de exclusão consistem em remover estudos que (i) não têm contribuições científicas completas (*e.g.*, resumos), (ii) não são estudos científicos (*e.g.*, normas e índices), ou (iii) têm acesso restrito.

A fase Execução demandou esforços de quatro pesquisadores e foi dividida em cinco etapas (FERREIRA, M. S. *et al.*, 2020): i) **execução da *string* de busca** (1.381 artigos resultantes); ii) **remoção de duplicatas** (1.305 artigos resultantes); iii) **aplicação dos critérios de exclusão** (1.205 artigos resultantes); iv) **aplicação dos critérios de inclusão**

[10] https://dl.acm.org
[11] https://www.engineeringvillage.com/home.url
[12] https://ieeexplore.ieee.org
[13] https://www.scopus.com
[14] https://link.springer.com/

considerando a leitura de título, resumo e palavras-chave (61 artigos resultantes); e v) **aplicação dos critérios de inclusão a partir da leitura completa** (40 artigos resultantes e inclusão de 1 artigo não retornado pelo protocolo de pesquisa[15], totalizando 41 artigos resultantes). É importante ressaltar que essa fase considerou artigos indexados nas bases de dados até novembro de 2019. Nova pesquisa foi realizada, considerando artigos indexados até março de 2021, o que resultou no acréscimo de 2 artigos, totalizando 43 artigos resultantes. Os artigos resultantes são apresentados no ANEXO A, contendo identificador, autores, ano de publicação e repositório em que foi encontrado.

4.2.2 Análise dos Estudos

A análise dos estudos é realizada durante a fase Apresentação dos Resultados do MSL. Assim, são apresentadas respostas para as questões de pesquisa considerando os 43 artigos resultantes do MSL e as opiniões de 4 Gerentes de Projetos.

Para responder à questão de pesquisa

QP1. Quais são as medidas utilizadas pelos Gerentes de Projetos para mensurar o trabalho dos desenvolvedores?

são apresentadas as medidas utilizadas para mensurar o trabalho dos desenvolvedores (ANEXO B). Ao todo, foram identificadas 65 medidas organizadas em seis grupos (Figura 4.1), considerando a similaridade entre os significados e a finalidade dessas medidas. Cabe ressaltar que essa organização é um resultado parcial deste trabalho de mestrado. Os grupos são:

a) **Comportamento (CoP)**. As medidas nesse grupo referem-se ao comportamento pessoal dos desenvolvedores, como foco, proatividade e interação com a equipe. A medida Comprometimento é um exemplo de medida desse grupo. Possui 5 medidas em 7 artigos;
b) **Contribuição (Con)**. As medidas nesse grupo referem-se à quantidade e à frequência de trabalho realizado pelo desenvolvedor em artefatos de software. As medidas Quantidade de *Commits* e a Quantidade de Linhas de Código (NLOC) são exemplos de medidas desse grupo. Possui 39 medidas em 34 artigos;
c) **Grau de Importância (ImP)**. As medidas nesse grupo referem-se ao domínio de tecnologias, à reputação do desenvolvedor, ao tempo de participação no projeto e à forma

[15] Dos artigos de controle utilizados para calibrar a *string* de busca, apenas um artigo não foi retornado, sendo incluído como resultado do MSL.

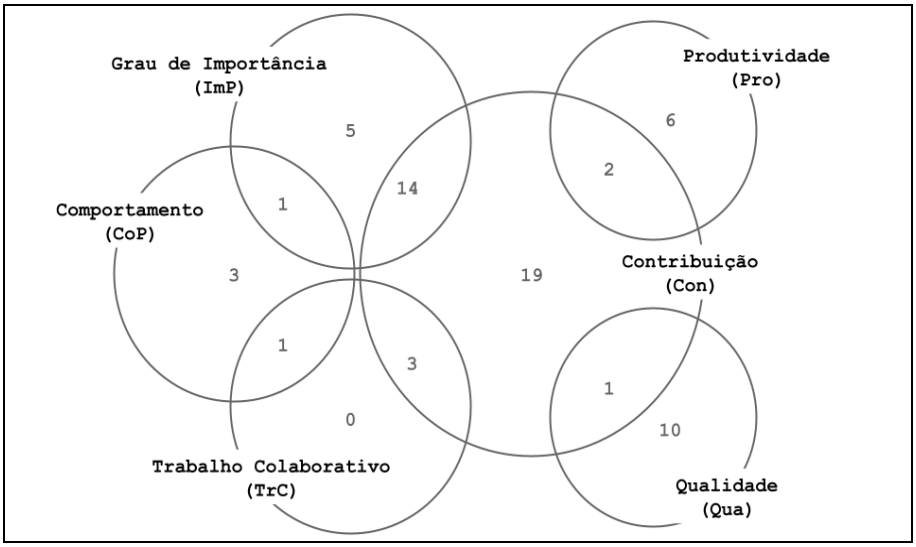
de atuação (*e.g.*, correção de *bugs*). A medida Quantidade de Dias Ativo no Projeto é um exemplo de medida desse grupo. Possui 20 medidas em 23 artigos;

d) **Produtividade (Pro)**. As medidas nesse grupo referem-se à quantidade de tarefas realizadas em um período de tempo. A medida Quantidade de Tarefas Entregues é um exemplo de medida desse grupo. Possui 8 medidas em 11 artigos;

e) **Qualidade (Qua)**. As medidas nesse grupo referem-se à qualidade do trabalho entregue (tarefa ou código-fonte). A medida Complexidade é um exemplo de medida desse grupo. Possui 11 medidas em 9 artigos;

f) **Trabalho Colaborativo (TrC)**. As medidas nesse grupo referem-se ao compartilhamento de informações e ao trabalho em equipe para desenvolver um mesmo artefato de software. A medida Colaboração em Arquivos é um exemplo de medida desse grupo. Possui 4 medidas em 15 artigos.

Figura 4.1 - Organização das medidas em grupos.



Fonte: Do Autor (2021).

Pode-se perceber que a maioria das medidas encontradas tem o objetivo de quantificar a contribuição concreta (em artefatos) do desenvolvedor em um projeto, sendo 39 medidas (60%) do grupo **Contribuição**, que compartilha medidas com os demais grupos criados, com exceção do grupo **Comportamento**. O grupo **Comportamento** considera aspectos como foco e comprometimento e não sobre o trabalho do desenvolvedor em algum artefato.

Além disso, os artigos selecionados no MSL utilizam mais de uma medida para mensurar o trabalho dos desenvolvedores. Isso ocorre por causa da complexidade da tarefa, necessitando de várias informações diferentes para analisar, da forma mais justa, a atuação de cada membro da equipe. Em concordância, os quatro Gerentes de Projetos convidados para

fornecer opiniões sobre as medidas encontradas elegeram diversas medidas como úteis para mensurar o trabalho dos desenvolvedores. Cada convidado escolheu um conjunto diferente de medidas e foi destacado que o contexto de trabalho pode interferir nessa escolha. Por exemplo, um Gerente de Projetos afirmou "escolhi as métricas que já tive contato e foram úteis em projetos que trabalhei.".

Foram encontradas 65 medidas utilizadas para mensurar o trabalho dos desenvolvedores. Essas medidas foram agrupadas de acordo com a informação que elas podem fornecer (Qualidade, Contribuição, Trabalho Colaborativo, Grau de Importância, Produtividade e/ou Comportamento). A maioria das medidas (39 medidas) fornece informações referentes à contribuição dos desenvolvedores em artefatos do projeto (grupo Contribuição).

Para responder à questão de pesquisa

QP2. Como as medidas são aplicadas pelos Gerentes de Projetos para acompanhar o trabalho dos desenvolvedores?

foram consideradas a fonte de dados utilizada para obter informações sobre o trabalho dos desenvolvedores, a forma de extração dessas informações para a aplicação da medida e a forma de apresentação dos resultados para os Gerentes de Projetos. Foram identificadas 5 fontes de dados (Tabela 4.2). Pode-se notar que repositórios de código-fonte (gerenciados por SCV) foram a fonte de dados mais utilizada (27 artigos - 63%). Isso pode ser justificado pelo fato desses repositórios armazenarem informações relacionadas à contribuição dos desenvolvedores em artefatos. Outro ponto interessante é existir fontes de dados impessoais e fontes de dados subjetivos, evidenciando a utilização de tipos variados de informação para a tarefa de mensuração.

Tabela 4.2 - Fontes de dados sobre o trabalho do desenvolvedor.

Fonte de dados	Tipo de fonte de dados	Referências
Repositório de código (SCV)	Impessoal	A8, A10, A11, A13, A16, A18, A19, A20, A22, A23, A24, A26, A27, A28, A30, A29, A33, A36, A38, A39, A3, A4, A7, A14, A31, A40, A34, A37 **(Total: 27 artigos - 62%)**
Repositórios de *bugs*	Impessoal	A8, A10, A15, A19, A26, A27, A38 **(Total: 7 artigos - 16%)**
Pesquisa de opinião	Subjetivo	A1, A9, A25, A32, A17, A41, A6 **(Total: 7 artigos - 16%)**
Repositório de gerenciamento de tarefas	Impessoal	A10, A19, A22, A35, A42, A43 **(Total: 6 artigos - 13%)**

(continua)

Tabela 4.2 - Fontes de dados sobre o trabalho do desenvolvedor (continuação).

Fonte de dados	Tipo de fonte de dados	Referências
Arquivo de código fonte	Impessoal	A5, A15, A21, A12, A2, A42 **(Total: 6 artigos - 13%)**

Fonte: Do Autor (2021).

Todos os Gerentes de Projetos convidados consideraram as medidas obtidas de fontes de dados impessoais como relevantes para mensurar o trabalho do time. Contudo, um dos convidados ressaltou a importância de fontes de dados subjetivas em "equipes menores" (até 5 pessoas), onde é possível acompanhar individualmente o trabalho de cada pessoa. Para as fontes de dados impessoais, foram encontradas duas estratégias para extração de dados e aplicação das medidas (Tabela 4.3): i) utilização de ferramentas automatizadas que coletam informações e aplicam as medidas; e ii) processo manual de extração de dados e aplicação de medidas. Considerando a fonte de dados subjetivos, foi identificada uma estratégia que envolve o uso de pelo menos um dos seguintes componentes: questionários, intuição dos Gerentes de Projetos, grupos focais e entrevistas.

Tabela 4.3 - Estratégias para extração de dados e aplicação das medidas.

Estratégia	Tipo de fonte de dados	Referências
Ferramenta automatizada	Impessoal	A5, A6, A15, A12, A42, A3, A4, A7, A8, A10, A11, A13, A14, A16, A18, A19, A21, A23, A28, A29, A31, A33, A35, A36, A38, A40, A2 **(Total: 27 artigos - 63%)**
Processo manual	Impessoal	A20, A22, A24, A26, A27, A30, A39, A34, A37, A43 **(Total: 10 artigos - 23%)**
Intuições do Gerentes de Projetos, questionários, entrevistas e grupos focais	Subjetivo	A1, A9, A25, A32, A17, A41 **(Total: 6 artigos - 14%)**

Fonte: Do Autor (2021).

A complexidade dos dados disponíveis em SCV e em outros repositórios de dados impessoais pode ser motivo para a utilização de ferramentas automatizadas para extração de dados e aplicação das medidas. Essa foi a estratégia mais utilizada para obter as medidas sobre o trabalho dos desenvolvedores (27 artigos - 63%). As ferramentas automatizadas variaram entre sistemas de software para *web*, sistemas de software executados via terminal de comandos e *plug-ins* para ambientes de desenvolvimento integrados. Dentre as ferramentas automatizadas, os sistemas de software para *web* têm a vantagem de oferecer interface gráfica e não serem limitados a ambientes de desenvolvimento integrado. Além disso, esses sistemas podem ser acessados em qualquer dispositivo e sistema operacional utilizando um navegador *web*.

Além disso, foram identificadas três estratégias para apresentação das informações aos Gerentes de Projetos (Tabela 4.4): i) formato textual; ii) formato de gráficos; e iii) formato de técnicas de visualização. Cabe ressaltar que, em 14 artigos (33%), foram combinadas mais de uma estratégia. Informações mais simples foram apresentadas na forma de texto, o agrupamento de informações foi representado em gráficos e, quando a quantidade de informações exibidas ao mesmo tempo é maior, a estratégia utilizada foi técnicas de visualização.

Tabela 4.4 - Estratégias de apresentação das informações.

Forma de apresentação	Referências
Gráficos	A7, A8, A13, A14, A15, A16, A18, A19, A21, A22, A23, A27, A30, A31, A39, A40, A12, A2, A43 **(Total: 19 artigos - 44%)**
Textual	A1, A5, A13, A14, A16, A19, A21, A22, A23, A25, A27, A29, A30, A31, A33, A35, A36, A38 **(Total: 18 artigos - 42%)**
Técnicas de visualização	A3, A4, A7, A8, A9, A11, A16, A28, A29, A38, A42 **(Total: 11 artigos - 26%)**

Fonte: Do Autor (2021).

As medidas são aplicadas a partir de dados encontrados em repositórios de código, repositórios de *bugs*, pesquisas de opinião, repositórios de gerenciamento de tarefas e arquivos de código. A coleta dos dados e a aplicação das medidas pode ser realizada usando ferramentas automatizadas, processamento manual e/ou por meio de intuições do Gerentes de Projetos, questionários, entrevistas e grupos focais. Por fim, os resultados são apresentados utilizando gráficos, textos e/ou técnicas de visualização.

Para responder à questão de pesquisa

QP3. De que modo as informações sobre o trabalho dos desenvolvedores apoiam o gerenciamento de projetos?

foram identificadas as práticas de gestão de projetos que podem ser apoiadas pelas informações obtidas com a mensuração do trabalho dos desenvolvedores. Foram encontradas 7 atividades para apoiar a gestão de projetos (Tabela 4.5), as quais estão relacionadas às Áreas de Conhecimento descritas no PMBoK (PMI, 2017).

Cabe ressaltar que essas atividades é um resultado parcial deste trabalho de mestrado. Pode-se notar que a maioria dessas atividades é relacionada às áreas Gerenciamento de Recursos do Projeto (pessoas) e/ou Gerenciamento de Riscos do Projeto (entre 5 e 7 atividades).

Tabela 4.5 - Atividades dos desenvolvedores.

Atividades	Área de conhecimento	Referências
Identificação das habilidades e perfil do desenvolvedor	Gerenciamento de Recursos do Projeto (pessoas) e Gerenciamento de Riscos do Projeto	A9, A10, A30, A35, A31, A27, A32, A18, A22, A26, A25, A24, A1, A3, A4, A7, A8, A11, A14, A20, A29, A17, A40, A41, A12, A42 **(Total: 26 artigos - 60%)**
Estimar custos e prazos de projetos e identificar anomalias no desempenho do desenvolvedor	Gerenciamento de Recursos do Projeto (pessoas)	A10, A11, A22, A30, A32, A14, A1, A15, A8, A19, A21, A28, A12, A2, A6, A43 **(Total: 16 artigos - 37%)**
Melhoria no desempenho das equipes	Gerenciamento de Recursos do Projeto (pessoas)	A9, A11, A26, A32, A22, A1, A19, A16, A21, A17, A34, A37 **(Total: 12 artigos - 28%)**
Compreensão e controle da distribuição do conhecimento	Gerenciamento de Riscos do Projeto	A10, A13, A23, A26, A32, A36, A33, A34, A37, A42 **(Total: 10 artigos - 23%)**
Identificação de necessidade de investimento (treinamento ou equipamentos)	Gerenciamento de Recursos do Projeto (materiais e equipamentos)	A10, A15, A24, A28, A17 **(Total: 5 artigos - 12%)**
Planejamento de melhoria da qualidade de código	Gerenciamento da Qualidade do Projeto	A22, A24, A14, A16 **(Total: 4 artigos - 9%)**
Reajustes de remuneração	Gerenciamento de Recursos do Projeto (pessoas)	A19, A26 **(Total: 2 artigos - 5%)**

Fonte: Do Autor (2021).

Foram analisadas as contribuições citadas nos artigos do MSL para a gestão de projetos. A partir dessa análise, foram definidas 7 atividades apoiadas pelas informações fornecidas pelas medidas.

Para responder à questão de pesquisa

QP4. Qual o tipo de solução frequentemente proposta em estudos nesta área?

foram mapeadas soluções propostas nos estudos para condução da pesquisa (Tabela 4.6). O resultado ajuda a avaliar se a comunidade teve mais foco na proposição de novas métricas ou no desenvolvimento de novas ferramentas para coletar as métricas existentes. A solução **Métrica** descreve novas métricas ou um plano de medição. A solução **Ferramenta** descreve e disponibiliza uma ferramenta computacional. A solução **Método** define fluxos de trabalho, regras ou procedimentos sobre como realizar uma atividade. A solução **Modelo** descreve representações conceituais com abstração formal de detalhes e notações. A solução **Visão Geral** descreve e compara informações para fornecer uma visão geral do assunto. Cabe destacar que a solução **Métrica** foi utilizada na maioria dos artigos (11 artigos - 26%) e solução menos utilizada foi **Visão Geral**, em 4 artigos (9%).

Tabela 4.6 - Soluções propostas nos artigos selecionados no MSL.

Soluções Propostas	Referências
Métrica	A8, A15, A20, A23, A10, A22, A26, A27, A39, A42, A43 **(Total: 11 artigos - 26%)**
Ferramenta	A3, A4, A11, A13, A14, A28, A29, A31, A38, A40 **(Total: 10 artigos - 23%)**
Método	A1, A25, A34, A37, A12, A7, A16, A35, A6 **(Total: 9 artigos - 21%)**
Modelo	A9, A24, A30, A41, A5, A18, A19, A21, A2 **(Total: 9 artigos - 21%)**
Visão Geral	A32, A17, A33, A36 **(Total: 4 artigos - 9%)**

Fonte: Do Autor (2021).

Os artigos foram analisados e os tipos de soluções encontrados foram Métrica (25,6%), Método (20,9%), Modelo (20,9%), Visão geral (9,3%) e Ferramenta (23,3%).

Para responder à questão de pesquisa

QP5. Qual é o tipo de metodologia de pesquisa frequentemente utilizada em estudos nesta área?

os artigos foram mapeados de acordo com a metodologia de pesquisa utilizada (WIERINGA, N. *et al.*, 2006) (Tabela 4.7). A metodologia **Pesquisa de Validação** investiga uma solução proposta em determinado contexto por meio de experimentos, pesquisas ou entrevistas para responder a uma determinada questão de pesquisa e foi encontrada na maioria dos artigos (16 artigos - 37%). Esta categoria não requer métodos experimentais mais formais (*e.g.*, teste de hipótese, experimento de controle). Na metodologia **Proposta de Solução**, uma técnica de solução é proposta e sua relevância é defendida com um pequeno exemplo ou uma boa linha de argumentação. A metodologia **Pesquisa de Avaliação** que investiga a relação entre fenômenos por meio de métodos experimentais formais onde as propriedades casuais são estudadas empiricamente, como por meio de estudos de caso, estudos de campo e experimentos de campo. A metodologia **Estudos de Experiência** explica por meio da experiência pessoal do autor como algo foi feito na prática. A metodologia **Estudos de Opinião** reporta a opinião do autor de como as coisas devem ser feitas. A metodologia **Estudos Filosóficos** estrutura as informações existentes sobre determinado campo na forma de uma taxonomia ou arcabouço conceitual, resultando em uma nova forma de ver as coisas existentes. As metodologias **Estudos de Experiência**, **Estudos de Opinião** e **Estudos Filosóficos** não foram encontradas nos artigos.

Tabela 4.7 - Metodologias de pesquisa utilizadas nos artigos selecionados no MSL.

Metodologia	Referências
Pesquisa de Validação	A4, A7, A8, A9, A10, A11, A13, A21, A26, A29, A31, A40, A41, A34, A12, A6 **(Total: 16 artigos - 37%)**
Proposta de Solução	A1, A3, A5, A14, A16, A20, A24, A25, A28, A35, A38, A39, A2, A42, A43 **(Total: 15 artigos - 35%)**
Pesquisa de Avaliação	A15, A18, A19, A22, A23, A27, A30, A32, A33, A36, A17, A37 **(Total: 12 artigos - 28%)**
Estudos de Experiência	**(Total: 0 artigos - 0%)**
Estudos de Opinião	**(Total: 0 artigos - 0%)**
Estudos Filosófico	**(Total: 0 artigos - 0%)**

Fonte: Do Autor (2021).

Os artigos foram analisados e os tipos de metodologia de pesquisa utilizados foram Proposta de Solução (34,9%), Pesquisa de Validação (37,2%) e Pesquisa de Avaliação (27,9%).

Para responder à questão de pesquisa

QP6. Como a solução proposta se relaciona com a metodologia de pesquisa nos estudos incluídos?

foram relacionadas as informações sobre a quantidade de estudos por metodologia de pesquisa e tipo de solução primária (Figura 4.2). Pode-se observar que muitos estudos apresentam Métrica, Ferramenta, Método e Modelo como solução nas metodologias Proposta de Solução, Pesquisa de Validação e Pesquisa de Avaliação. No entanto, muitas interseções têm valores zero. Por exemplo, em estudos que apresentam a solução Ferramenta, apenas avaliações de baixo e médio rigor foram realizadas utilizando as metodologias Proposta de Solução e Pesquisa de Validação. Portanto, estudos empíricos robustos são necessários para essa solução. Além disso, não há soluções relacionadas às metodologias Estudos de Experiência e Estudos de Opinião e Estudos Filosóficos, o que evidencia a necessidade de estudos nesta área.

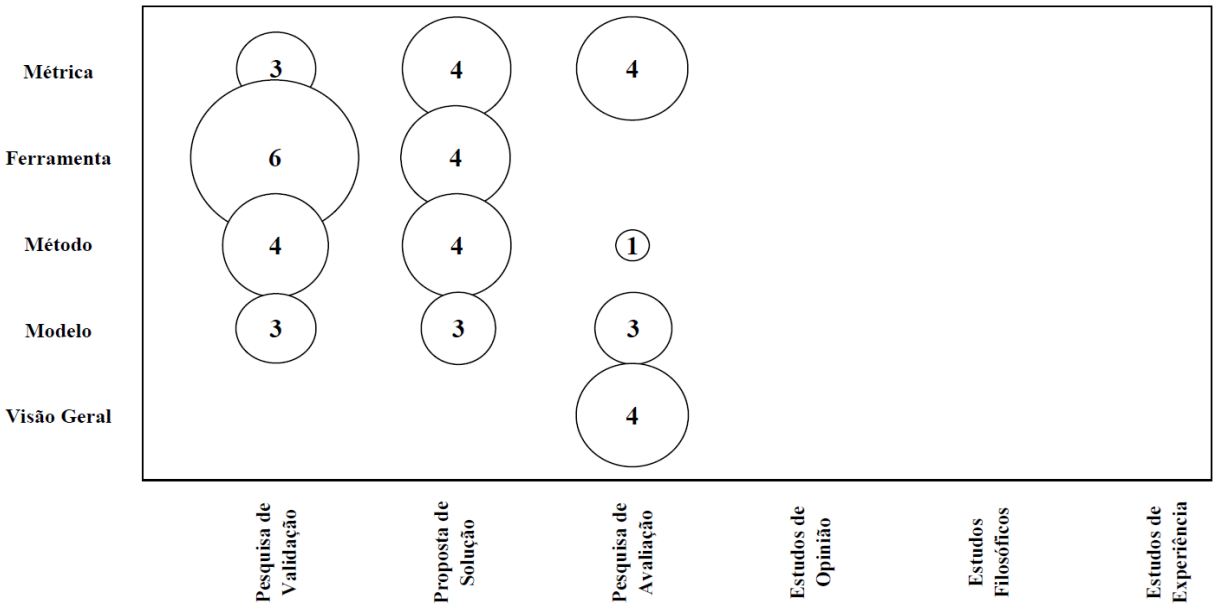
Ao relacionar os tipos de solução e as metodologias de pesquisa utilizadas é possível notar que que muitos estudos apresentam Métrica, Ferramenta, Método e Modelo como solução nas metodologias de Proposta de Solução, Pesquisa de Validação e Pesquisa de Avaliação.

4.3 Compreender o Trabalho dos Desenvolvedores

As medidas quantitativas fornecem informações impessoais que permitem aos Gerentes de Projetos compreender o trabalho dos desenvolvedores de forma imparcial. Essas

informações podem ser utilizadas de maneira conjunta com as impressões empíricas que os Gerentes de Projetos possuem sobre os membros da equipe, tornando a compreensão mais assertiva (LIMA, J. *et al.*, 2015). Observando os resultados do estudo exploratório descrito na Seção 4.2, diversas informações impessoais podem ser obtidas utilizando mineração em SCV. É o caso de duas medidas mais utilizadas nos artigos selecionados no MSL: NLOC (12 artigos - 28%) e Quantidade de *Commits* (10 artigos - 23%). A relevância dessas medidas torna-se mais evidente ao observar a quantidade de novas medidas criadas a partir delas: 45 medidas (Tabela 4.8). Outras medidas não derivadas de NLOC e/ou de Quantidade de *Commits* também podem ser utilizadas extraindo dados de SCV. É o caso das medidas *change code documentation*, *start a new thread* - MST, *start a new wiki page* - WSP, *update a wiki page* - WUP, *link a wiki page from documentation file* - WLP, QCTE, CQI, medidas CK, medidas de Martin, medidas de tamanho e CDI.

Figura 4.2 - Relação entre metodologias e propostas de solução.



Fonte: Do Autor (2021).

Apesar da principal finalidade dos SCV ser o controle de versão dos dados, esses sistemas caracterizam-se como valiosas fontes de informações de diversos tipos, como as contribuições em artefatos de código e de documentação, o tempo de atuação no projeto ou em determinada versão do projeto e a quantidade de correções realizadas no código. Além de serem armazenadas, essas informações podem ser observadas em diferentes perspectivas. Por exemplo, observar (i) o quanto o sistema de software evoluiu ao identificar a sua quantidade de *commits* e (ii) a quantidade de linhas de código. Utilizando essas duas medidas e

relacionando-as a um desenvolvedor, pode-se observar o quanto esse desenvolvedor contribuiu com a evolução do projeto.

Tabela 4.8 - Medidas derivadas das medidas NLOC e Quantidade de *Commits*.

Autoria por culpa	*Commits Versatility*	Fator de Contribuição
Close a bug (BCL)	Custo	Fragmentação do programador
Close a bug that is then reopened (BCR)	*Contribuition start*	Heroi
Close a lingering thread (MCT)	*Contribution Duration*	*Knowledge at Risk* (KaR)
Colaboração (arquivos)	DOA	*Knowledge Loss*
Colaboração (*CodeChurn*)	Domínio de tecnologias	MTBC
Comment on a bug report (BCR)	Duplicação de código	Número de dias ativo
Commit binary files (CBF)	Entrega de tarefas	Quantidade de *code churn*
Commit code that closes a bug (CCB)	Esforço mensal	*Participate in a flamewar* (MFW)
Commit comment that includes a bug report num (CBN)	Esforço no *commit*	% de linhas de código
Commit documentation files (CDF)	Esforço por modificação	% de *commits*
Commit fixes code style (CSF)	Experiência	*Source Abandoned*
Commit multiples files in a single commit (CMF)	*Expertise Breadth of a Developer* (EBD)	*Status*
Commit new source file or directory (CNS)	*Expertise of a Developer* (ED)	*Truck Factor*
Commit with empty commit comment (CEC)	*Expected Shortfall* (ES)	Último *commiter* “leva tudo”

Fonte: Do Autor (2021).

Os artigos encontrados no MSL mostraram a aplicação da mineração em SCV considerando perspectivas diferentes. Alguns autores procuraram minerar informações considerando o projeto como um todo, permitindo avaliar o quanto a equipe se esforçou para produzir a versão do sistema de software (FEINER; ANDREWS, 2018; GONZÁLEZ-TORRES *et al.*, 2016). Por outro lado, há autores que procuraram identificar as contribuições individuais de cada desenvolvedor para compreender o quanto ele entende do projeto ou de determinada tecnologia (FERREIRA, M. *et al.*, 2016; FERREIRA, M.; VALENTE; FERREIRA, 2017; JARUCHOTRATTANASAKUL *et al.*, 2016). Observando as estratégias existentes e as diferentes informações e perspectivas disponíveis, nota-se a inexistência de consenso ao compreender o trabalho dos desenvolvedores. Apesar disso, a combinação de informações variadas é considerada uma boa estratégia (LIMA, J. *et al.*, 2015).

A tarefa de compreender um conjunto de informações variado e volumoso é complexa. As técnicas de visualização facilitam a identificação de padrões em grandes conjuntos de dados, favorecendo a realização da tarefa de compreensão. Nos artigos resultantes do MSL, a aplicação de técnicas de visualização ocorreu quando foi necessário exibir muitas informações, por exemplo, para verificar o quanto cada desenvolvedor trabalhou em determinados arquivos de código em conjunto com outros desenvolvedores (GONZÁLEZ-TORRES *et al.*, 2016), o quanto as linhas de código de diferentes arquivos evoluíram ao longo do tempo (FEINER; ANDREWS, 2018) e quantas modificações aconteceram em

classes e pacotes em determinada versão do sistema de software (GAO; LIU, 2015). Alguns trabalhos incluíram interações de *zoom* e filtros, permitindo expandir ou focar o campo de visão, bem como ocultar ou exibir informações em dado momento.

Apesar da eficiência das técnicas de visualização, é importante ressaltar que algumas informações podem ser apresentadas de forma textual ou em gráficos. Trata-se de uma estratégia relevante por apresentar as informações sem que seja preciso interpretar ou aprender a interagir com uma metáfora (vinculada a uma técnica de visualização). Por exemplo, textos e gráficos foram utilizados para apresentar as tecnologias que um desenvolvedor domina e a quantidade de modificações realizadas no projeto (JARUCHOTRATTANASAKUL *et al.*, 2016; SHARMA; KAULGUD, 2012).

4.4 Considerações Finais

Os Gerentes de Projetos são profissionais encarregados da tarefa de conduzir os projetos de software ao sucesso. Assim, as práticas de gestão de projetos devem ser aplicadas. Para apoiá-lo, diversos estudos na literatura propuseram estratégias de mensurar o trabalho dos desenvolvedores. Com o objetivo de apresentar o estado da arte e organizar os conceitos referentes a essa temática, foi realizado um estudo exploratório aplicando a técnica Mapeamento Sistemático da Literatura (MSL) (FERREIRA, M. S. *et al.*, 2020). O estudo foi atualizado e apresentado de forma resumida, respondendo as seis questões de pesquisa.

Além disso, neste capítulo, foi apresentada uma discussão acerca dos resultados do estudo exploratório, destacando a relevância da aplicação de medidas quantitativas para a obtenção de informações impessoais sobre trabalho dos desenvolvedores. Foi mostrado como SCV são fontes preciosas dessas informações. Além disso, foi destacada a importância da aplicação de técnicas de visualização para facilitar a compreensão de dados extraídos de SCV, visto que esses dados são complexos e podem ser extraídos considerando perspectivas diferentes.

5 *DEVELOPER TRACKER*: UMA ABORDAGEM PARA APOIAR A COMPREENSÃO DO TRABALHO DOS DESENVOLVEDORES

5.1 Considerações Iniciais

Neste capítulo, é apresentada *Developer Tracker*, uma abordagem visual proposta neste trabalho para fornecer a visualização de dados quantitativos, apoiando a compreensão do trabalho dos desenvolvedores. A abordagem segue um modelo de referência para visualização de sistemas de software e é dividida em cinco etapas.

O restante deste capítulo está organizado da seguinte forma. Na Seção 5.2, é exibida uma visão geral da abordagem. Na Seção 5.3, são apresentadas as etapas da abordagem proposta.

5.2 Visão Geral

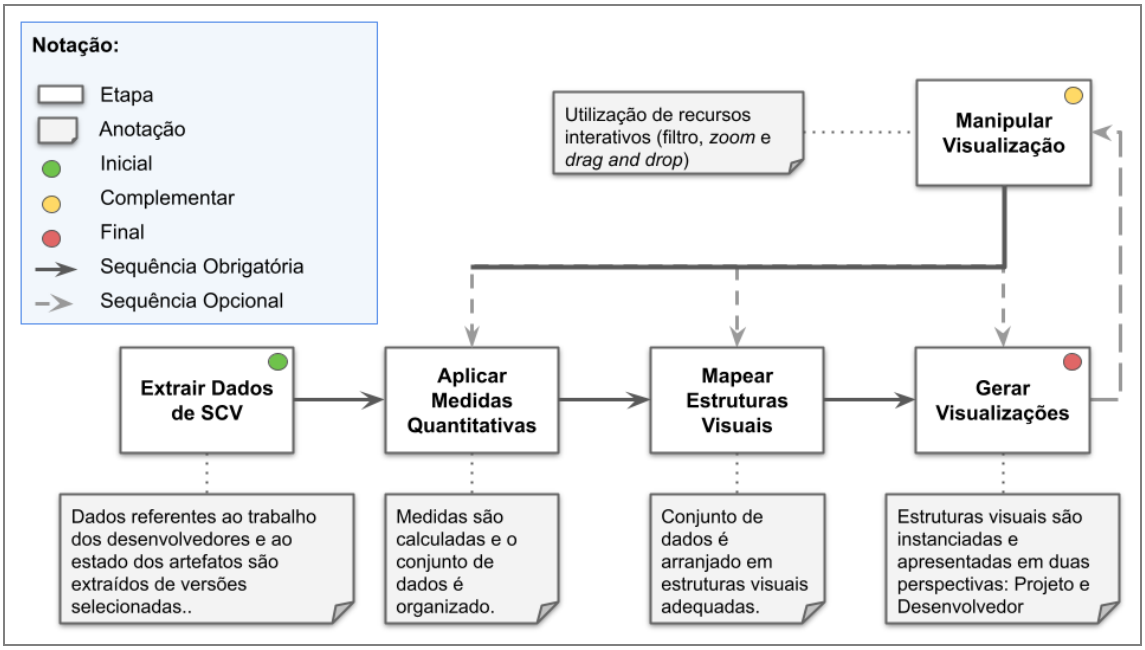
Developer Tracker é uma abordagem que consiste em um conjunto de etapas para coletar informações registradas em repositórios de código, aplicar medidas quantitativas e apresentar visualmente os resultados para os Gerentes de Projetos. O objetivo de apresentar medidas quantitativas de maneira visual é auxiliar a interpretação de informações relacionadas ao trabalho dos desenvolvedores e à evolução de sistemas de software. O conjunto de etapas da *Developer Tracker* é baseado em um modelo de referência para a visualização de sistemas de software (CARNEIRO, 2011). Na etapa inicial desse modelo, deve ser realizada a coleta dos dados de interesse. Em seguida, os dados devem ser submetidos a um processo de transformação, que consiste em padronizá-los e organizá-los. Ao término desse processo, os dados são mapeados em estruturas de visualização, como tabelas, gráficos e/ou metáforas visuais. A etapa seguinte consiste em realizar a instanciação das visualizações mapeadas e, em seguida, utilizar recursos interativos para manipulá-las. Dependendo da manipulação, pode ser necessário refazer alguma etapa anterior. Ao refazer uma etapa, as etapas subsequentes são impactadas e, portanto, devem ser refeitas. A manipulação da visualização pode impactar a transformação dos dados, o mapeamento das estruturas visuais e/ou a instanciação das visualizações.

5.3 Etapas

A *Developer Tracker* possui cinco etapas (Figura 5.1): i) Extrair Dados de SCV; ii) Aplicar Medidas Quantitativas; iii) Mapear Estruturas Visuais; iv) Gerar Visualizações; e v)

Manipular Visualização. É importante ressaltar que a última etapa é complementar, visto que sua execução não é obrigatória e, caso ocorra, só poderá ser executada após a etapa Gerar Visualizações. Assim, a etapa Gerar Visualizações é considerada etapa final da abordagem.

Figura 5.1 - Visão Geral da Abordagem *Developer Tracker*



Fonte: Do Autor (2021).

Na Seção 5.3.1, é descrita a etapa de seleção de versões do software em um SCV. Na Seção 5.3.2, é relatada a etapa de aplicação de medidas quantitativas. Na Seção 5.3.3, é apresentada a etapa de mapeamento das estruturas visuais. Na Seção 5.3.4, é exibida a etapa de geração de visualizações. Na Seção 5.3.5, é descrita uma etapa complementar que permite a alteração das visualizações geradas.

5.3.1 Etapa 1: Extrair Dados de SCV

A coleta de dados consiste em selecionar dados de interesse em um repositório disponível. Nesse sentido, a etapa **Extrair Dados de SCV** consiste na extração de dados de versões de um sistema de software armazenado em um SCV. Os SCV são ricas fontes de informação sobre o sistema de software desenvolvido e sobre a atuação dos desenvolvedores (MOURA; NASCIMENTO; ROSA, 2014; XIE; POSHYVANYK; MARCUS, 2006). Os SCV registram informações de mudanças realizadas nos artefatos de software (incluindo arquivos de código fonte e documentação) e oferecem suporte para determinar onde, quando e quanto os membros da equipe de desenvolvimento atuaram em determinado artefato (WU *et al.*, 2004).

Nessa etapa, as informações de registros de atividades e de arquivos armazenados no SCV devem ser capturadas de uma ou duas versões de um sistema de software. Essas informações incluem aspectos de cada versão, permitindo identificar os estados dos artefatos e o trabalho dos desenvolvedores. Quando uma única versão for selecionada pode-se visualizar informações de uma versão específica do sistema de software. Quando duas versões forem selecionadas pode-se visualizar comparativamente as informações dessas versões.

5.3.2 Etapa 2: Aplicar Medidas Quantitativas

Após a coleta, é iniciado um processo de transformação dos dados. Nesse sentido, a etapa **Aplicar Medidas Quantitativas** consiste em organizar o montante de dados coletados, calculando medidas que quantificam o trabalhado realizado na(s) versão(ões) selecionadas. Alguns autores direcionaram a aplicação de medidas aos dados referentes a uma versão do sistema de software (FEINER; ANDREWS, 2018; GAO; LIU, 2015). Outros autores direcionaram a aplicação de medidas aos registros individuais de atividade dos desenvolvedores (JARUCHOTRATTANASAKUL *et al.*, 2016). Além disso, existem autores que direcionaram a aplicação de medidas às duas aplicações anteriores (GONZÁLEZ-TORRES *et al.*, 2016; SHARMA; KAULGUD, 2012).

É importante ressaltar que não existe consenso de qual(is) informação(ões) ou qual(is) medida(s) pode(m) quantificar o trabalho realizado pelos desenvolvedores (LIMA, J. *et al.*, 2015). Contudo, utilizar um conjunto diversificado de informações e de medidas é uma estratégia apropriada para quantificar esse trabalho (FEINER; ANDREWS, 2018; LIMA, J. *et al.*, 2015). Desse modo, na *Developer Tracker*, a aplicação das medidas é direcionada em duas perspectivas complementares:

a) **`Projeto`**. O objetivo é quantificar o trabalho considerando um montante de informações referentes à versão do sistema de software;
b) **`Desenvolvedor`**. O objetivo é quantificar o trabalho do membro da equipe considerando os registros individuais de atividade.

Por exemplo, ao observar que um desenvolvedor é autor de 1.000 linhas de código de uma versão do sistema de software (aplicação da medida NLOC na perspectiva `Desenvolvedor`), os Gerentes de Projetos podem interessar-se em conhecer a dimensão total da versão desse sistema, em linhas de código, para melhor interpretar essa informação (aplicação da medida NLOC na perspectiva `Projeto`). É importante ressaltar que uma mesma medida pode ser utilizada em perspectivas diferentes, bem como medidas específicas

podem ser utilizadas em cada perspectiva. Um diferencial da *Developer Tracker* é a possibilidade de utilizar as duas perspectivas simultaneamente para duas versões do sistema de software, permitindo realizar a comparação do trabalho realizado em versões diferentes. Na Tabela 5.1, são apresentadas medidas utilizadas na *Developer Tracker* com o objetivo das medidas em cada perspectiva.

Tabela 5.1 - Medidas para quantificar individualmente o trabalho dos desenvolvedores

Medida	Objetivo Perspectiva `Desenvolvedor`	Objetivo Perspectiva `Projeto`
Colaboração (em arquivos)	Quantificar o trabalho em conjunto entre os desenvolvedores.	-
Domínio de Tecnologias	-	Quantificar a proporção de linhas de código para cada linguagem de programação utilizada na versão do projeto.
NLOC	Quantificar o quanto cada desenvolvedor possui de autoria sobre o código fonte na versão do projeto.	Quantificar o tamanho da versão do projeto quanto a quantidade de linhas de código.
Quantidade de *Commits*	Quantificar quantos incrementos (inclusões/ exclusões/ modificações) foram realizados pelo desenvolvedor na versão do projeto.	Quantificar o tamanho da versão do projeto quanto a quantidade de *commits*.
Truck Factor	Quantificar o grau de importância do desenvolvedor para a versão do projeto.	Quantificar a concentração de conhecimento sobre o código fonte da versão do projeto.

Fonte: Do Autor (2021).

Ao utilizar a medida NLOC na perspectiva `Desenvolvedor`, é contabilizada a quantidade de linhas de código de um membro da equipe (desenvolvedor), indicando sua contribuição e sua autoria cada membro da equipe. Considerando a perspectiva `Projeto`, a mesma medida contabiliza a dimensão total do projeto quanto às linhas de código. Essa medida é comumente utilizada para quantificar o trabalho dos desenvolvedores (FERREIRA, M. S. *et al.*, 2020). Da mesma forma, a medida Quantidade de *Commits* é uma das mais utilizadas para representar o trabalho realizado pelos desenvolvedores (FERREIRA, M. S. *et al.*, 2020). Com a sua utilização, pode-se identificar a frequência de trabalho realizado sobre os artefatos do projeto. Essa medida é aplicada para a perspectiva `Desenvolvedor` e para a perspectiva `Projeto`. A aplicação dessas medidas nas duas perspectivas auxilia na compreensão da proporção do trabalho realizado pelos desenvolvedores, contrastando o percentual de trabalho do desenvolvedor com o valor total. Essas medidas são utilizadas como base para diversas outras medidas, evidenciando a importância de suas informações.

As demais medidas incluídas na *Developer Tracker* são baseadas em NLOC e Quantidade de *Commits*. Uma delas é a medida *Truck Factor*, resultante de um cálculo utilizando a medida DOA (*degree-of-authorship*), que define o grau de autoria em arquivos ao contabilizar valores de NLOC e Quantidade de *Commits* (FERREIRA, M.; VALENTE; FERREIRA, 2017). O valor da medida DOA é obtido após refinamento dessas duas medidas,

caracterizando as ações (*commits*) de criação e de modificação de cada arquivo e observando a quantidade de linhas de código (NLOC) adicionadas pelos contribuidores. Assim, o valor dessa medida é calculado para cada desenvolvedor em cada arquivo, permitindo estimar o quanto cada desenvolvedor tem de conhecimento sobre o artefato. O valor da medida DOA varia entre 0 e 1, quando esse valor para um desenvolvedor é 0,75 ou superior em determinado arquivo, ele é considerado como um autor desse artefato. Note que vários desenvolvedores podem ser autores de um artefato.

Após a definição dos autores dos arquivos, é simulada a saída de desenvolvedores do projeto, iniciando pela saída do desenvolvedor que possui autoria em mais arquivos até o desenvolvedor que é autor de menos arquivos (ordem decrescente de quantidade de arquivos). A cada desenvolvedor retirado do projeto, arquivos podem se tornar "órfãos" (sem autor). A quantidade de desenvolvedores retirados é definida como o valor da medida *Truck Factor* para o projeto quando, pelo menos, 50% dos arquivos tornam-se "órfão". Logo, quanto menor o valor da medida *Truck Factor*, maior é a concentração de conhecimento. Além disso, é possível identificar o grau de importância de um desenvolvedor e a distribuição de conhecimento existente na versão do projeto (FERREIRA, M. S. *et al.*, 2020). Essa informação pode guiar a análise de outras informações sobre os desenvolvedores, visto que os Gerentes de Projetos podem ter o interesse de distribuir o conhecimento sobre o código fonte do sistema de software (BOEHM, 1991; FERREIRA, M.; VALENTE; FERREIRA, 2017). A distribuição de conhecimento é importante para mitigar riscos, como a rotatividade de desenvolvedores. Quando um ou poucos desenvolvedores concentram o conhecimento sobre o código fonte de um software, há o risco de fracasso do projeto com a saída dessas pessoas.

O valor da medida Domínio de Tecnologia é obtida pela quantidade de linhas de código (NLOC) e/ou quantidade de *commits* referentes a determinada linguagem de programação. Essa medida foi aplicada na perspectiva `Projeto`, permitindo quantificar a proporção de linhas de código escritas em cada linguagem de programação utilizada na versão do projeto. Ao identificar o domínio de tecnologias demandadas no projeto, pode-se direcionar contratações de novos profissionais, realocar colaboradores e identificar o perfil dos desenvolvedores do projeto (JARUCHOTRATTANASAKUL *et al.*, 2016).

Outra medida utilizada é Colaboração (em arquivos). O desenvolvimento de sistemas de software envolve a colaboração de muitas pessoas, inclusive entre os desenvolvedores de uma equipe. Essa medida está presente na perspectiva `Desenvolvedor` e sua utilização pode fornecer, para cada artefato do projeto, o quanto os desenvolvedores trabalharam em conjunto. O seu cálculo consiste em utilizar NLOC para cada artefato do software e relacioná-

lo aos desenvolvedores que contribuíram com o desenvolvimento. Ao observar os desenvolvedores com participação na construção de um mesmo artefato, pode-se inferir que houve colaboração, pois diversos indivíduos empregaram esforços para produzir a versão do artefato em questão. O domínio técnico da equipe e a sua capacidade de trabalhar em conjunto são informações valorizadas na indústria de software, pois podem representar a capacidade de gerar soluções de qualidade e de proporcionar o fechamento de contratos (SOMMERVILLE, 2019).

Algumas das medidas utilizadas na perspectiva `Desenvolvedor` podem ser aplicadas em diferentes níveis de granularidade, fornecendo maior quantidade de informações sobre o trabalho dos indivíduos. É o caso das medidas NLOC, Quantidade de *Commits* e Colaboração (em arquivos). Essas medidas estão intimamente relacionadas às atividades desempenhadas pelos desenvolvedores em artefatos do software. Esses artefatos estão distribuídos em uma hierarquia de diretórios e arquivos do repositório do SCV. Assim, na *Developer Tracker*, o valor dessas medidas é considerado em três níveis de granularidade: i) **Projeto**, considera todos os artefatos da versão do projeto; ii) **Diretório**, considera todos os arquivos e subdiretórios incluídos no diretório ou pasta; e iii) **Arquivo**, considera um artefato do tipo arquivo. Os artefatos são os arquivos de diretórios armazenados no repositório do SCV.

Com essa granularidade, pode-se identificar o quanto um desenvolvedor trabalhou em determinados arquivos ou setores do projeto. Por exemplo, considere um sistema de software para *web* armazenado em um SCV. Os artefatos do código fonte estão divididos nos diretórios *frontend* (contém artefatos de código da interface do sistema) e *backend* (contém artefatos de código da configuração e do atendimento de requisições *web*). Na granularidade **Projeto**, as medidas possuem valor considerando todo o sistema de software para *web*. Na granularidade **Diretório**, as medidas possuem valor considerando os dois diretórios (*frontend* e *backend*). Nesse momento, pode-se identificar a atuação dos desenvolvedores nos artefatos de interface e nos artefatos de configuração e atendimento de requisições *web*. Algumas situações a serem analisadas pelos Gerentes de Projetos são:

a) **um dos diretórios pode possuir valor superior de NLOC**. A diferença em linhas de código pode ocorrer por causa do uso de linguagens de programação diferentes entre os diretórios ou pode indicar que esforço maior está sendo demandando em determinado setor do projeto;
b) **um dos diretórios pode possuir valor superior de Quantidade de *Commits***. A diferença em Quantidade de *Commits* pode ocorrer por causa da necessidade de muitas

correções ou evoluções em determinado setor do projeto ou que mais esforço é demandando;

c) **um desenvolvedor pode apresentar valor de NLOC e Quantidade de *Commits* superior aos demais desenvolvedores em determinado diretório**. Pode indicar que esse desenvolvedor concentra o conhecimento de um diretório e tem o domínio técnico da tecnologia utilizada naquele setor do projeto.

Observando a granularidade **Arquivo**, pode-se identificar a atuação do desenvolvedor em arquivos específicos (*e.g.*, algum arquivo de configuração do servidor *web*), o que pode mostrar a capacidade do desenvolvedor em definir o comportamento da aplicação ao responder requisições. Ao utilizar granularidade, torna-se possível observar a atuação dos desenvolvedores com "rico" nível de detalhamento. Além disso, o uso de diferentes níveis de granularidade possibilita melhor integração das diferentes informações fornecidas pelas medidas. Por exemplo, dependendo da organização dos artefatos, há a possibilidade de identificar o nível de conhecimento do desenvolvedor em determinada tecnologia do projeto e, ao mesmo tempo, verificar a proporção de linhas de código do projeto representada por essa tecnologia (medida presente na perspectiva `Projeto`). Outro exemplo é identificar que determinado desenvolvedor tem grau de importância alto no projeto (medida *Truck Factor* presente na perspectiva `Projeto`) e, na perspectiva `Desenvolvedor`, identificar outro desenvolvedor (que não aparece em destaque no *Truck Factor*) que detém o conhecimento de um aspecto chave do sistema de software, por exemplo, a configuração do servidor *web*.

5.3.3 Etapa 3: Mapear Estruturas Visuais

Após a padronização e a organização, os dados devem ser preparados para serem apresentados visualmente. Nesse sentido, com a *Developer Tracker*, é sugerido o mapeamento dos dados em estruturas visuais agrupando-os conforme a estratégia de visualização a ser utilizada. A escolha da estratégia de visualização depende da complexidade dos dados a serem apresentados (FERREIRA, M. S. *et al.*, 2020). A complexidade cresce de acordo com a quantidade de combinações necessárias para mostrar a informação. As informações compostas por um único dado são preparadas para serem apresentadas de forma direta (*e.g.*, texto), pois não demandam uma estrutura visual elaborada. As informações compostas pela combinação de dois ou mais dados são preparadas para serem apresentadas com alguma estrutura de agrupamento, por exemplo, estruturas bidimensionais (*e.g.*, tabelas e gráficos)

e/ou técnicas de visualização baseadas em paradigma visual. Os paradigmas visuais fazem uso de metáforas, que facilitam a compreensão de um conjunto de dados (SPENCE, 2007).

Na Tabela 5.2, são exibidas as estruturas visuais mapeadas para as medidas da abordagem proposta (*Developer Tracker*). Na perspectiva `Projeto`, as medidas NLOC e Quantidade de *Commits* são aplicadas para fornecerem informações precisas e diretas sobre o tamanho da versão do sistema de software (projeto), portanto devem ser exibidas de forma direta com uma estrutura unidimensional (*e.g.*, texto). A medida Domínio de Tecnologias é resultante da combinação das informações linguagens de programação utilizadas no projeto e proporção de linhas de código que cada linguagem representa. Da mesma forma, a medida *Truck Factor* do projeto é composta pelas informações de desenvolvedores que trabalharam na versão e de quais desenvolvedores concentram o conhecimento. Assim, essas últimas duas medidas devem ser exibidas utilizando uma estrutura de agrupamento bidimensional (*e.g.*, tabela ou gráfico).

Tabela 5.2 - Mapeamento das medidas em estruturas visuais.

Perspectiva	Medida	Estrutura Visual
Desenvolvedor	Colaboração (em arquivos)	Técnica de Visualização Baseada em Paradigma Visual
Desenvolvedor	NLOC	Técnica de Visualização Baseada em Paradigma Visual
Desenvolvedor	Quantidade de *Commits*	Técnica de Visualização Baseada em Paradigma Visual
Desenvolvedor	*Truck Factor*	Técnica de Visualização Baseada em Paradigma Visual
Projeto	Domínio de Tecnologias	Bidimensional
Projeto	NLOC	Unidimensional
Projeto	Quantidade de *Commits*	Unidimensional
Projeto	*Truck Factor*	Bidimensional

Fonte: Do Autor (2021).

Na perspectiva `Desenvolvedor`, as medidas Colaboração (em arquivos), NLOC e Quantidade de *Commits* possuem informações que podem ser exibidas em conjunto e que representam grande volume de combinações. Em um projeto, podem existir diversos desenvolvedores que podem escrever linhas de código, realizar *commits* em diversos artefatos diferentes e ter trabalhado em variados artefatos em conjunto com desenvolvedores distintos. Além disso, essas medidas são calculadas considerando diferentes níveis de granularidade. Apesar de não considerar diferentes níveis de granularidade, a medida *Truck Factor* também está presente na perspectiva `Desenvolvedor` e, portanto, deve ser exibida em conjunto com as demais medidas, permitindo relacionar o grau de importância de um desenvolvedor com as demais informações que representam seu trabalho. As combinações entre essas medidas e a possível necessidade de utilizar recursos interativos (*e.g.*, *zoom* e filtro) para aprimorar a visualização implicam na utilização de uma técnica de visualização baseada em algum paradigma visual para representá-las.

5.3.4 Etapa 4: Gerar Visualizações

Após o mapeamento das estruturas visuais, é necessário realizar a instanciação das estruturas e gerar as visualizações. Nesse sentido, com a *Developer Tracker*, são geradas visualizações nas perspectivas `Desenvolvedor` e `Projeto`. Além disso, as visualizações são instanciadas a partir de três tipos de estrutura:

a) **Unidimensional**. A sugestão é instanciar com o valor da medida e um rótulo descritivo. Por exemplo, "Quantidade de *Commits*: 100";
b) **Bidimensional**. A sugestão é instanciar relacionando um rótulo descritivo ao valor da medida. São exemplos:
    - **Tabela**: cada linha da primeira coluna contém os nomes dos desenvolvedores (rótulo descritivo) e cada linha da segunda coluna evidencia se o desenvolvedor da respectiva linha está incluído ou não no *Truck Factor* (valor da medida);
    - **Gráfico de setores**: gráfico circular fatiado, onde cada fatia do círculo representa uma linguagem de programação (pode ser utilizada uma legenda, relacionando cor e linguagem de programação - rótulo descritivo) e o tamanho da fatia representa a proporção de linhas de código da linguagem de programação (pode ser inserido em cada fatia o valor percentual que representa);
c) **Técnica de Visualização baseada em paradigma visual**. Na *Developer Tracker* é sugerida a utilização de uma técnica de visualização exclusiva. Essa técnica de visualização é um resultado parcial deste trabalho de mestrado e é apresentada no ANEXO C. O objetivo dessa técnica é permitir a visualização conjunta das medidas NLOC, Quantidade de *Commits*, Colaboração (em arquivos) e *Truck Factor* e suas relações com desenvolvedores e artefatos do projeto. Essa técnica é baseada no paradigma Iconográfico. Para representar as informações presentes na *Developer Tracker*, foram elaborados cinco ícones:
    - **Ícone Maleta**, representa a versão do sistema de software com todos os artefatos do projeto;
    - **Ícone Envelope**, representa um artefato do tipo diretório - armazenado no repositório;
    - **Ícone Documento,** representa um artefato do tipo arquivo - armazenado no repositório;
    - **Ícone Pessoa**, representa um desenvolvedor do projeto;

- **Ícone Relação de Trabalho**, representa uma relação de trabalho entre um desenvolvedor e um artefato.

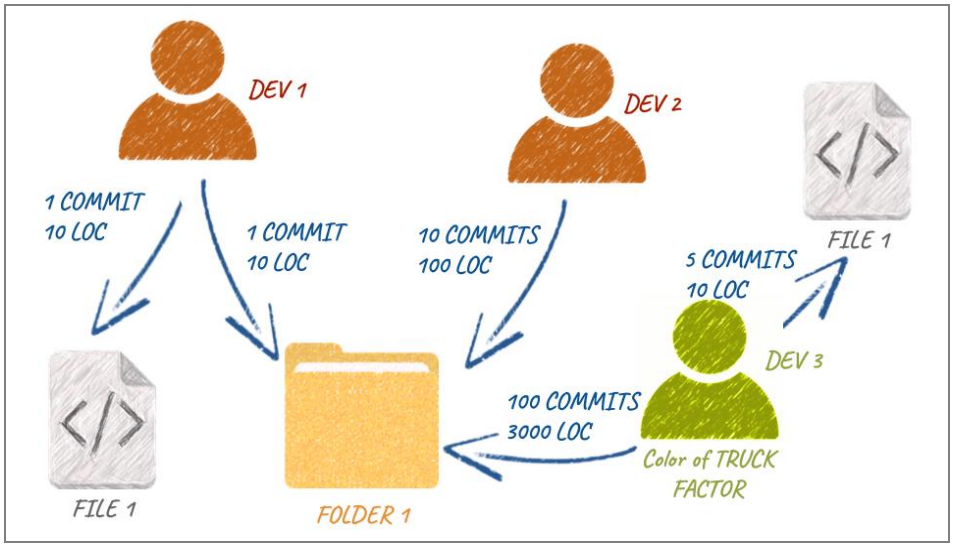
O ícone **Pessoa** conecta-se aos três primeiros ícones por meio do ícone **Relação de Trabalho**, cujo objetivo é mostrar que um desenvolvedor trabalhou em determinado projeto/diretório/arquivo, escrevendo pelo menos uma linha de código (medida NLOC) e/ou realizando pelo menos um *commit* (medida Quantidade de *Commits*). Se dois desenvolvedores estão conectados a um artefato, significa que tal artefato possui pelo menos uma linha de código ou um *commit* com autoria de cada um dos dois desenvolvedores e que ambos colaboraram com o desenvolvimento (medida Colaboração (em arquivos)). Além disso, cada conexão deve possuir um rótulo, que exibe a quantidade de linhas de código e/ou a quantidade de *commits* que a caracteriza. Em outras palavras, a conexão de um desenvolvedor com um sistema de software (projeto) deve possuir um rótulo que mostra a **"quantidade de linhas código"/"quantidade de *commits*"** de sua autoria no projeto. Da mesma forma, a conexão de um desenvolvedor com um artefato deve possuir um rótulo que mostra a **"quantidade de linhas de código"/"quantidade de *commits*"** sua autoria no artefato em questão. Outra definição é o ícone **Pessoa** deve receber uma cor de destaque quando o desenvolvedor representado estiver "incluído" na medida *Truck Factor*.

Além disso, essa técnica permite visualizar medidas e suas relações com os artefatos considerando duas versões do projeto de software. Ao considerar duas versões, há a caracterização das conexões como **novas**, **removidas**, **mantidas** ou **modificadas**, explicitando as diferenças das conexões entre as duas versões. Na Figura 5.2, é apresentado um exemplo hipotético da técnica de visualização elaborada neste trabalho, no qual os desenvolvedores *Dev 1*, *Dev 2* e *Dev 3* e suas respectivas relações de trabalho com um diretório (*Folder 1*) e dois arquivos (*File 1* e *File 2*) de uma versão de um projeto de software. Além disso, o *Dev 3* possui uma coloração diferente e um rótulo explicativo, indicando que esse desenvolvedor está "incluído" na medida *Truck Factor*.

5.3.5 Etapa 5: Manipular Visualização

Após a geração das visualizações, há as manipulações visuais utilizando recursos interativos. Nesse sentido, com a *Developer Tracker*, pode-se realizar alterações nas visualizações da perspectiva `Desenvolvedor`. Na perspectiva `Projeto`, informações exibidas são mais objetivas e com pouco grau de combinação, sendo desnecessária a manipulação da visualização.

Figura 5.2 - Técnica Iconográfica da *Developer Tracker*



Fonte: Do Autor (2021).

O uso de recursos interativos é comum em técnicas de visualização como a técnica Iconográfica utilizada na *Developer Tracker*, incluindo o uso de filtros (enfatizar visualmente um conjunto específico de informações) e/ou de *zoom* (ampliar ou reduzir um elemento visual) (BASTOS, 2016; KEIM, 2002). Por exemplo, se Gerentes de Projetos quiserem quantificar a medida Colaboração (em arquivos) em um projeto contendo dez desenvolvedores, eles podem perceber que dois desenvolvedores estão trabalhando juntos em muitos arquivos. Assim, seria interessante ocultar os outros oito desenvolvedores da visualização para concentrar-se somente nos dois desenvolvedores que apresentaram colaboração destacada.

Por outro lado, a ação de ocultar os dois desenvolvedores com mais colaboração para "limpar" a quantidade de informações exibidas e visualizar a colaboração entre os oito desenvolvedores restantes exige menos carga cognitiva. Essa ação tem o objetivo de filtrar a informação exibida. Assim, o filtro de desenvolvedor foi incluído na *Developer Tracker*. Outra forma de "limpar" a quantidade de informações exibidas é filtrar artefatos e conexões entre artefatos e desenvolvedores. Por exemplo, pode-se filtrar arquivos de interesse ou de alguma linguagem de programação (filtro de artefato) e ver alguma conexão de forma isolada para concentrar-se nos detalhes dos rótulos (filtro de relação de trabalho). Portanto, os filtros de artefato e de relação de trabalho foram incluídos na *Developer Tracker*.

Além de recursos de filtro, foram incluídos recursos para manipular a visualização sem ocultar ou "limpar" informações. Esses recursos são o *zoom* e o *drag and drop*, cuja utilidade é permitir a modificação da posição dos elementos apresentados nas perspectivas. O principal objetivo é facilitar a visualização de certos detalhes. Por exemplo, ao utilizar *zoom*,

pode-se “aproximar” a visão em algumas regiões da visualização e explorar com mais precisão as conexões entre desenvolvedores e artefatos específicos, e, ao utilizar *drag and drop*, pode-se posicionar um conjunto específico de elementos no centro do espaço visual.

5.4 Considerações Finais

Neste capítulo, foi apresentada a *Developer Tracker*, cujo objetivo é fornecer a visualização de dados quantitativos, apoiando a compreensão do trabalho dos desenvolvedores. De forma geral, a abordagem pode ser descrita em cinco etapas: i) **Extrair Dados de SCV** (coletar dados de versões de um software armazena em SCV); ii) **Aplicar Medidas Quantitativas** (são aplicadas medidas, organizando as informações referentes às atividades dos desenvolvedores e referentes a cada versão selecionada); iii) **Mapear Estruturas Visuais** (as informações medidas são organizadas em estruturas unidimensionais, bidimensionais e técnica de visualização baseada em paradigma visual); iv) **Gerar Visualizações** (cada estrutura visual é instanciada, incluindo uma técnica baseada no paradigma iconográfico); e v) **Manipular Visualizações** (etapa complementar que possibilita a alteração das visualizações geradas utilizando recursos de filtros, *zoom* e *drag and drop*).

6 APOIO COMPUTACIONAL

6.1 Considerações Iniciais

Neste capítulo, é apresentado o apoio computacional *Developer Tracker App*, uma implementação da *Developer Tracker* como um sistema de software para *web*.

O restante deste capítulo está organizado da seguinte forma. Na Seção 6.2, são descritas as tecnologias utilizadas para o desenvolvimento. Na Seção 6.3, é apresentada a arquitetura. Na Seção 6.4, são exibidas as telas e é descrito o funcionamento.

6.2 Tecnologias Utilizadas

Para um usuário acessar um sistema de software para *web*, não é necessária a instalação de sistemas de software e não é exigido um sistema operacional específico. Esses sistemas podem ser acessados utilizando dispositivos que possuem *browser* (navegador), por exemplo, *smartphones*, *desktops*, *laptops* e *tablets*. Contudo, uma premissa para um software *web* ser acessível em um *browser* é ser executado em um servidor *web*. Um servidor *web* pode ser implantado no computador local do usuário ou em um computador externo. Essa implantação habilita a execução do software *web*, gerando um endereço eletrônico para acesso via *browser*. Assim, usuários com posse desse endereço eletrônico podem acessar o software *web* referenciado por esse endereço.

Para facilitar o acesso dos Gerentes de Projetos às informações sobre o trabalho dos desenvolvedores fornecidas pela utilização da *Developer Tracker*, o *Developer Tracker App* (um software para *web*) foi desenvolvido, utilizando as seguintes tecnologias:

a) **Java**. Linguagem de programação que permite o desenvolvimento de sistemas de software para *web* e possui grande comunidade de utilizadores e mantenedores (LUCKOW; DE MELO, 2010);
b) ***Spring Boot***. *Framework* de código aberto baseado no *Spring Framework*[16]. Esse *framework* acelera o desenvolvimento de sistemas Java para ambiente *web*, diminuindo a complexidade para a criação de novas aplicações (MALIPENSE; ZUCHI, 2018);
c) **HTML (*Hyper TextMarkup Language*)**. Linguagem de marcação para hipertexto, cujo objetivo é permitir a escrita de documentos ou páginas *web*, que podem ser lidas por navegadores de *internet* (*browser*) (SILVA, 2007);

[16] *Framework* para facilitar o desenvolvimento de sistemas baseados na linguagem *Java*, sendo considerado um dos mais maduros e mais utilizados para tal finalidade (MALIPENSE; ZUCHI, 2018).

d) **CSS (*Cascading Style Sheets*)**. Linguagem de programação para estilizar páginas HTML (SILVA, 2007);

e) ***JavaScript***. Linguagem de programação para *web* utilizada para construção de páginas *web*, sendo interpretada pelo *browser* do usuário, sendo possível sua utilização no servidor *web* (FLANAGAN, 2004). Para desenvolver o *Developer Tracker App*, essa linguagem foi utilizada para construção de páginas web;

f) **Vue.js**. *Framework* popular baseado na linguagem de programação *JavaScript* que permite o desenvolvimento rápido de interfaces *web* e interações com usuários (LIMA, L. G.; PETRUCELLI; DO ESPÍRITO SANTO, 2019);

g) **Git**. Sistema de controle de versão que armazena arquivos de um projeto em um repositório distribuído, sendo utilizado para gerenciar sistemas de software (DE QUEIROZ; FERREIRA; DA SILVA, 2015);

h) ***ECharts***. Biblioteca *JavaScript* para a geração de diversos tipos de gráficos e com recursos de customização (APACHE, 2020). Dois gráficos dessa biblioteca foram utilizados: i) "*pie*" (permitiu instanciar um gráfico de setores que exibe o domínio de tecnologias na perspectiva `Projeto`); ii) "*graph*" (um tipo genérico de gráfico que permite construir gráficos customizados, sendo utilizado como base para a implementação da técnica visual exclusiva da *Developer Tracker*, apresentada na perspectiva `Desenvolvedor`);

i) **CLOC - *Count Lines of Code***. Sistema de software de código aberto utilizado para contabilizar as linhas (de código, de comentários e/ou em branco) existentes em arquivos de diversas linguagens de programação (DANIAL, 2017). Esse sistema possui 15 versões e cerca de 9 mil estrelas no GitHub (onde está hospedado). Para o desenvolvimento do *Developer Tracker App*, a versão 1.84-1 foi utilizada para aplicação da medida Domínio de Tecnologias, calculando a proporção de código fonte em cada linguagem de programação do projeto. Esse sistema é instalado durante a implantação do servidor *web* para o *Developer Tracker App* acionar seus recursos utilizando o comando "*cloc*" no terminal;

j) ***Truckfactor-tool***. Sistema de software de código aberto que implementa o cálculo do valor da medida *Truck Factor* (AVELINO, 2016). A versão *V1.1 ICPC* foi acoplada ao *Developer Tracker App*, permitindo realizar o cálculo da medida e identificar os desenvolvedores que concentram o conhecimento sobre o código fonte de um projeto de software;

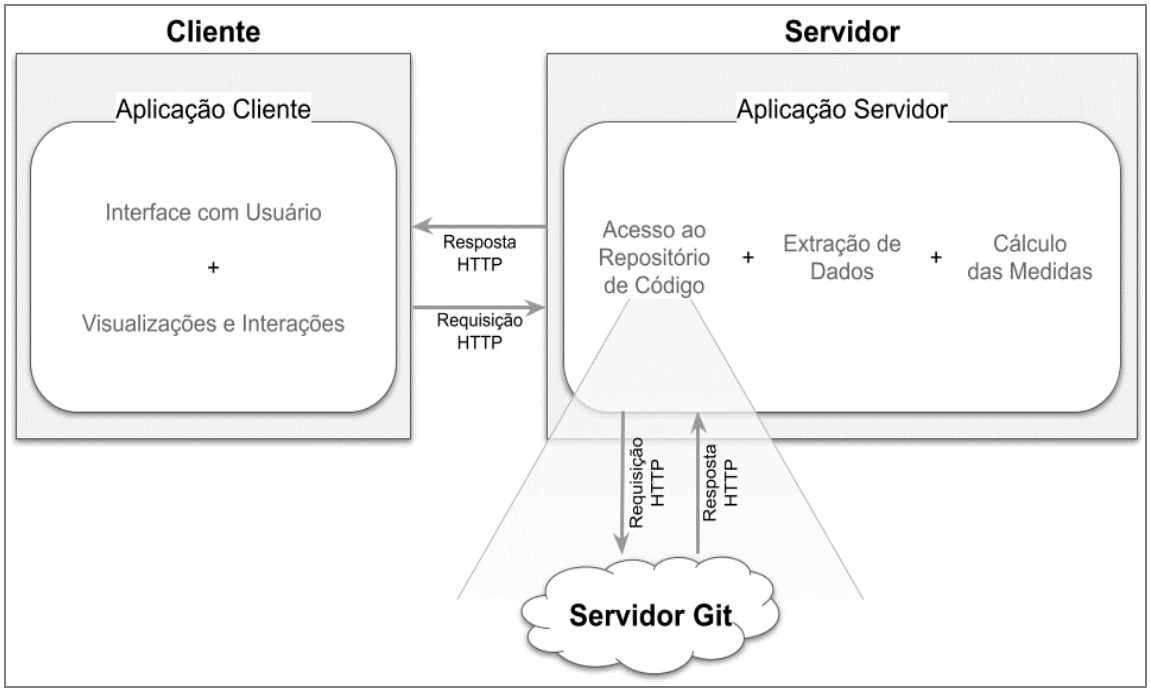
k) ***Docker***. É uma ferramenta de virtualização, cuja utilização é similar à de máquinas virtuais, porém mais leve do que uma máquina virtual convencional (ANDERSON, 2015). *Docker* foi utilizado para facilitar a implantação do *Developer Tracker App* em um servidor *web*, virtualizando a imagem de um sistema operacional que contém suas bibliotecas e dependências tecnológicas.

6.3 Arquitetura

A arquitetura empregada no sistema é um modelo cliente/servidor que consiste em uma estrutura de aplicação distribuída, dividindo tarefas entre os servidores (fornecedores de um recurso) e os clientes (requerentes dos recursos). Um servidor (servidor *web*) é um computador ou *host* que compartilha recursos com diversos clientes. Um cliente é um computador ou *host* que solicita recursos, serviços e/ou conteúdo de um servidor (MALIPENSE; ZUCHI, 2018; PRESSMAN, R. S., 2002).

A arquitetura do *Developer Tracker App* é apresentada na Figura 6.1. A aplicação cliente é responsável por apresentar a interface aos usuários. A interface é composta pelas telas que geram as visualizações e permitem a interação dos usuários. Para gerar as visualizações, a aplicação cliente precisa receber dados organizados. Por isso, são realizadas requisições via protocolo HTTP para a aplicação servidor. A aplicação servidor é responsável por acessar e extrair dados de um repositório de código e, em seguida, calcular o valor das medidas e enviá-lo como resposta a uma requisição HTTP. Para extrair os dados, é necessário clonar o projeto a ser utilizado como fonte de dados. A aplicação servidor tem suporte para clonar repositórios baseados em Git (*e.g.*, *GitLab*). A clonagem é realizada via requisição HTTP para o servidor GIT onde o projeto está armazenado. Em seguida, os dados são extraídos e as medidas são calculadas. Por fim, as medidas são organizadas e enviadas em resposta HTTP para a aplicação cliente. Como o usuário pode interagir e manipular as visualizações na aplicação cliente, novas requisições podem ser realizadas para a aplicação servidor. É importante ressaltar que o clone de um projeto é realizado somente na primeira requisição da aplicação cliente para a aplicação servidor. Assim, as requisições seguintes utilizam como fonte de dados o projeto clonado até o usuário encerrar o uso do *Developer Tracker App*.

Figura 6.1 - Visão Arquitetural do Apoio Computacional *Developer Tracker App*.



Fonte: Do Autor (2021).

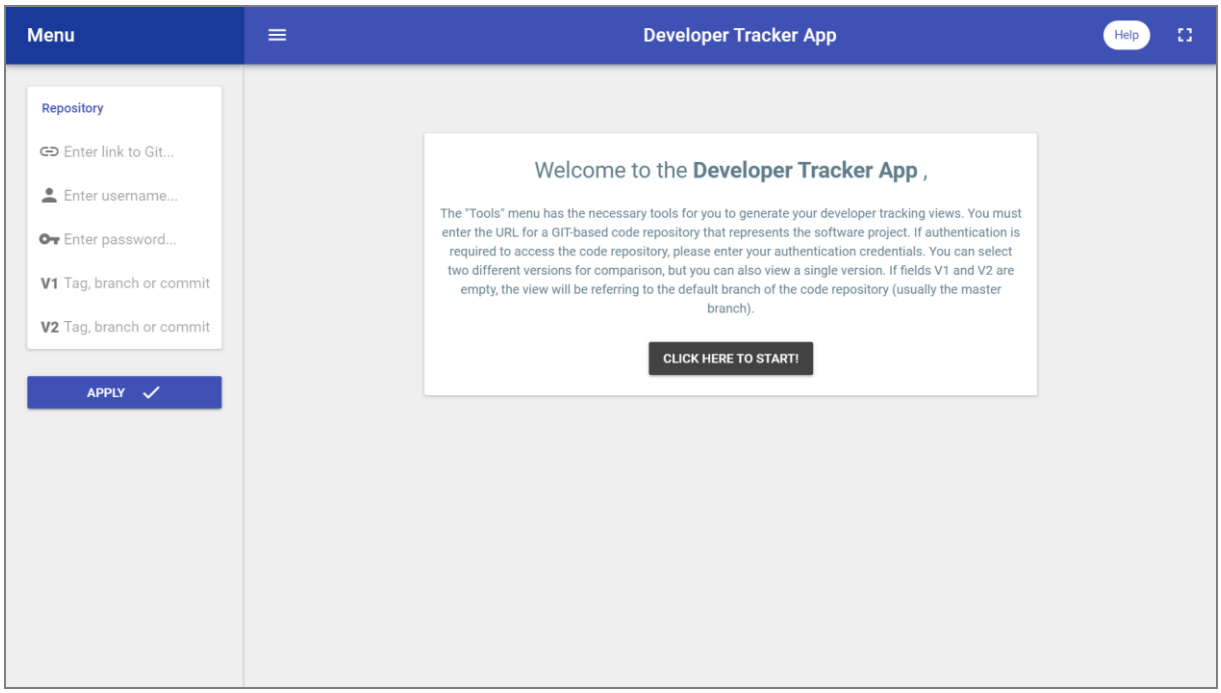
6.4 Funcionamento

O *Developer Tracker App* possui duas telas. A primeira tela (Tela Inicial) é a tela de apresentação, na qual o usuário recebe algumas instruções de como utilizar a ferramenta (Figura 6.2). Além disso, o usuário pode clicar no botão "*CLICK HERE TO START*". Ao realizar essa ação, o componente *MENU* é aberto, surgindo no canto esquerdo da tela. O *Menu* possui algumas opções de campos para preenchimento do usuário. O preenchimento do primeiro campo deve ser o *link* do Git para clone do repositório de código ("*Enter link to Git...*"). O preenchimento dos dois campos seguintes consiste nas credenciais de acesso ao repositório (usuário - "*Enter username*" e senha - "*Enter password*"). Por fim, existem outros dois campos ("*V1"* e "*V2"*) cujo preenchimento deve ser um identificador de versão em cada campo. Um identificador de versão pode ser uma ramificação (*branch*), um rótulo (*tag*) ou um identificador *hash* de *commit*.

É importante ressaltar que o preenchimento do primeiro campo é obrigatório ("*Enter link to Git*"). Além disso, as credenciais precisam ser informadas somente quando há necessidade de autorização para acessar o repositório Git (*e.g.*, projetos privados). Se o campo "*V1*" for preenchido e o campo "*V2*" não for preenchido, então será gerada a visualização referente à versão "*V1*". Ao não preencher o campo "*V1*" e preencher o campo "*V2*", é apresentada uma mensagem de erro ao usuário. Caso os campos "*V1*" e "*V2*" sejam preenchidos, as visualizações são geradas de forma comparativa entre as duas versões.

Quando esses campos não forem preenchidos, automaticamente, será utilizada a *branch* padrão do repositório como versão selecionada para o campo "*V1*". Geralmente, essa versão consiste na *branch* chamada *master*. Após preencher corretamente os campos do *Menu*, o usuário pode clicar no botão "*APPLY*" para disparar uma requisição HTTP para a aplicação servidor. No momento em que a resposta da requisição chegar à aplicação cliente, a segunda tela do *Developer Tracker App* é apresentada.

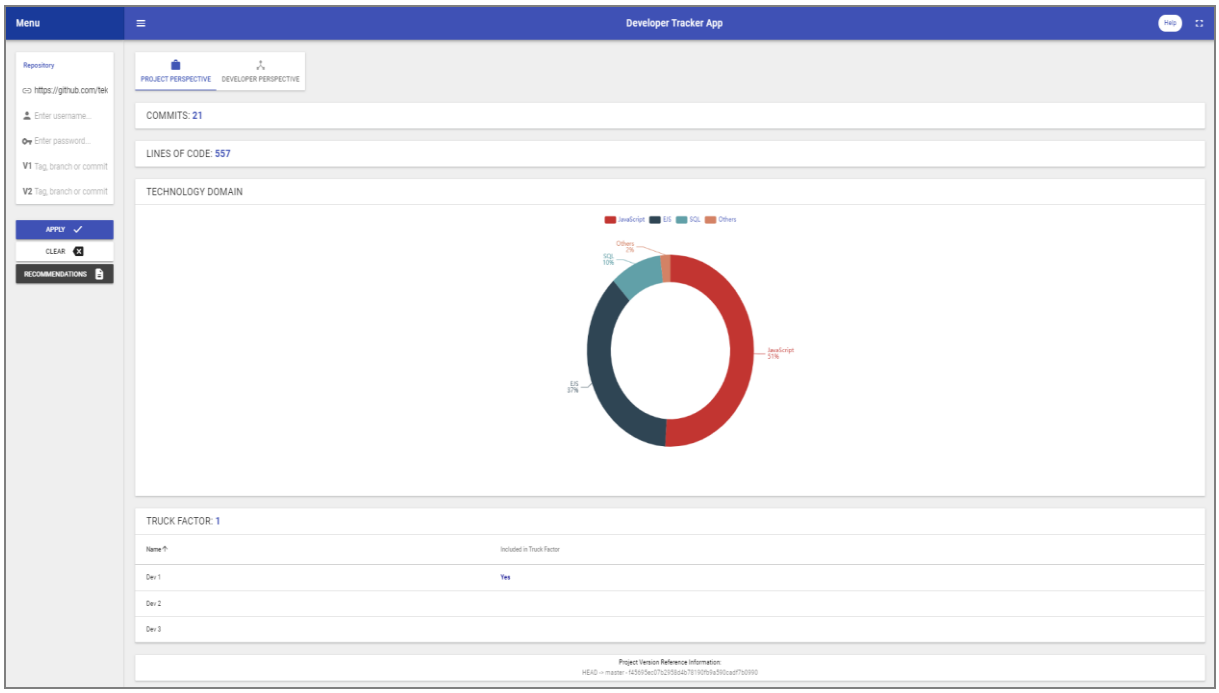
Figura 6.2 - Tela Inicial - Apoio Computacional *Developer Tracker App*.



Fonte: Do Autor (2021).

A segunda tela (Figura 6.3) é utilizada para apresentar as informações geradas, exibindo as medidas calculadas conforme as opções preenchidas no *Menu*. Nessa tela, é apresentada a opção de visualizar as perspectivas `Projeto` e `Desenvolvedor`, que podem ser alternadas acionando os botões suspensos ("*Project Perspective*" e "*Developer Perspective*", respectivamente). Na perspectiva `Projeto` (perspectiva apresentada na Figura 6.3), o valor das medidas é exibido utilizando duas estruturas visuais (unidimensional e bidimensional). O valor das medidas Quantidade de *Commits* e NLOC são instanciadas como cartões, contendo o valor das respectivas medidas e um rótulo descritivo. O valor da medida Domínio de Tecnologias é instanciado como um gráfico de setores. O valor da medida *Truck Factor* é instanciado como um rótulo posicionado no cabeçalho de uma tabela, listando os desenvolvedores e indicando com o *label* "*Yes*" os "incluídos" (considerados) no *Truck Factor*.

Figura 6.3 - Tela para visualização dos resultados na perspectiva `Projeto`.



Fonte: Do Autor (2021).

Ainda na Figura 6.3, pode-se notar dois novos botões no *Menu*. Um dos botões é "*CLEAR*" que, quando clicado, atualiza a página, limpando dados preenchidos e visualizados e permitindo ao usuário iniciar a utilização do *Developer Tracker App*. Outro botão é "*RECOMMENDATIONS*" que, quando clicado, gera um arquivo de texto contendo algumas recomendações ao Gerente de Projetos. São geradas quatro recomendações. As três primeiras recomendações são geradas independentemente do valor das medidas e a quarta recomendação é gerada de acordo com a medida NLOC. De maneira geral, as recomendações são:

a) **Recomendação 1.** É apresentado o diagnóstico de quantos desenvolvedores trabalharam no projeto e quantos estão concentrando o conhecimento de acordo com a medida *Truck Factor*. A seguinte recomendação é apresentada: "*When a small number of people on the team concentrate the knowledge in parts of the implementation, the project can be dependent on these people. Thus, you should investigate that knowledge in the source code and rearrange the people for the knowledge to be more homogeneous among team members.*";

b) **Recomendação 2.** É apresentado o diagnóstico de porcentagem de linhas de código para cada linguagem de programação utilizada no projeto. A seguinte recomendação é apresentada: "*Project Managers can need human resources in specific programming languages. Thus, you should know who developers who work on the software modules use*

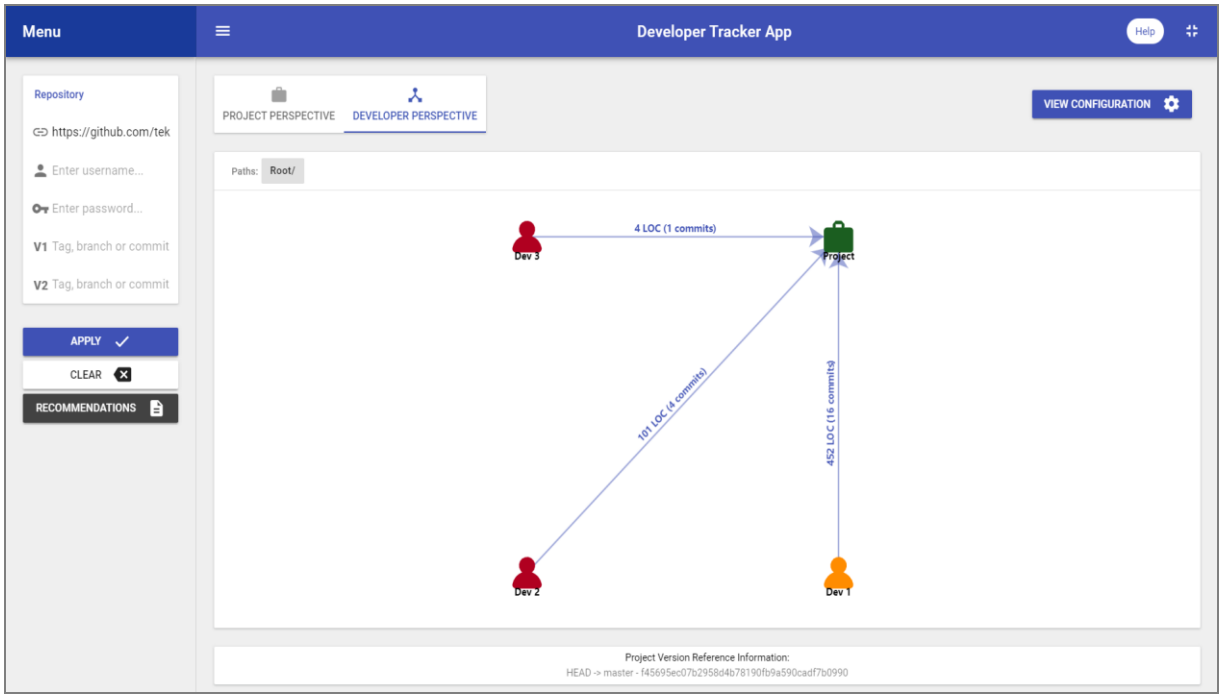
that programming language because they can be allocated to other projects. This knowledge can help you optimize people's allocation, avoiding underutilization or work overload of developers.";

c) **Recomendação 3.** É apresentado o diagnóstico de quantas linhas de código e de quantos *commits* a(s) versão(ões) selecionada(s) possui(em). Independentemente dos resultados, a seguinte recomendação é apresentada: "*When looking at LOC (project and developer perspective), you should consider that the team must follow the appropriate code standards of the programming language. Also, it would help if you considered defining a code review process so that other developers can evaluate the solutions implemented by a team member, avoiding LOC inappropriate or excessive solutions. When observing commits, you should consider that the team must follow a commits pattern. Thus, you can compare the difference in dimensions of the two versions. Also, note if the technological demand and the performance of the developers were different in the two versions. These results can indicate the results of decisions taken throughout the project and the release of versions.*";

d) **Recomendação 4.** É apresentada, para cada linguagem de programação, quais são os três desenvolvedores que mais escreveram linhas de código (começando pelo que mais escreveu), indicando que esses desenvolvedores podem ser avaliados como possíveis especialistas.

Na Figura 6.4, a segunda tela é apresentada com a perspectiva `Desenvolvedor` ativada, na qual são mostrados os desenvolvedores que trabalharam no projeto conectados a ele. Cada conexão possui rótulo indicando o valor das medidas NLOC e Quantidade de *Commits* do desenvolvedor na versão desse projeto. O ícone de projeto (uma "maleta") pode ser clicado para entrar no próximo nível da hierarquia de arquivos e pastas. No nível atual da hierarquia, a "maleta" encontra-se fechada. Ao clicar, a "maleta" é aberta e o conteúdo interno é apresentado. O ícone de desenvolvedores, que representam o *Truck Factor* do projeto, possui coloração de destaque (laranja), enquanto o ícone dos demais desenvolvedores possuem a cor padrão (vermelho). No *Developer Tracker App*, há dois recrusos: i) filtragem por desenvolvedores, artefatos e conexões (relacionamento entre desenvolvedor e artefato), utilizando o botão "*VIEW CONFIGURATION*" ou posicionando *mouse* sobre o ícone que se deseja obter mais informações; e ii) *zoom* para aproximar ou distanciar a visualização, sendo possível arrastar a visualização (toda a visualização, não um único ícone) para posicioná-la em um local diferente dentro do espaço disponível.

Figura 6.4 - Tela para visualização dos resultados na perspectiva `Desenvolvedor`.

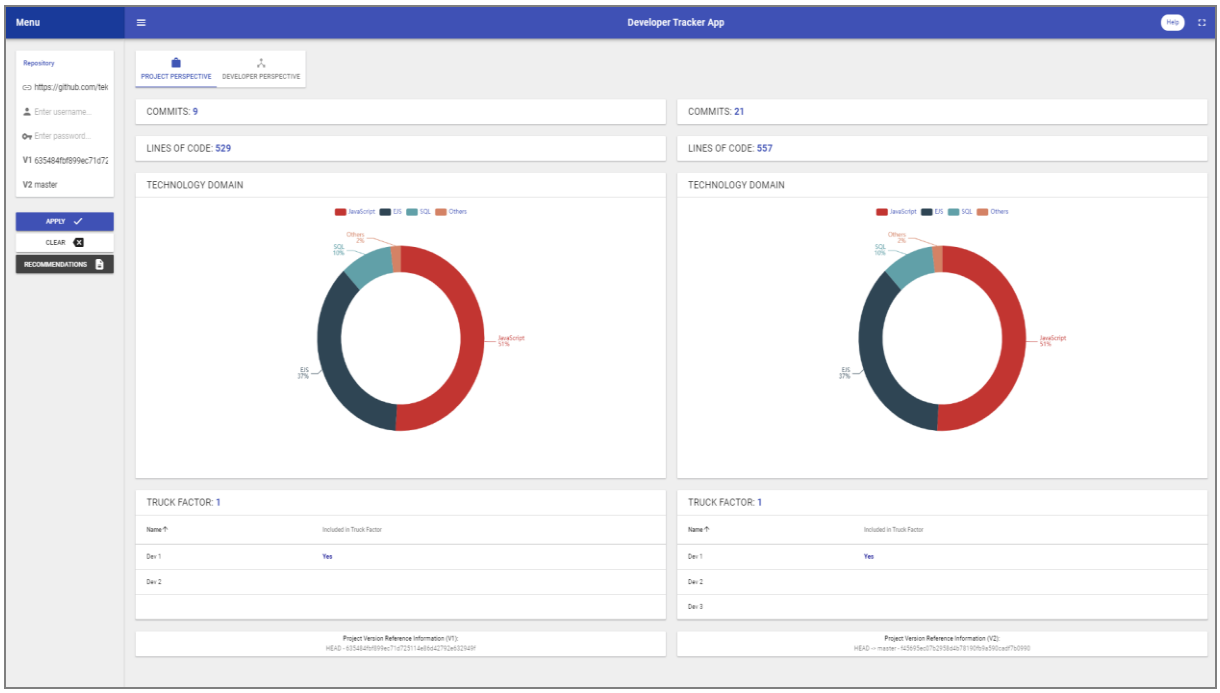


Fonte: Do Autor (2021).

Na Figura 6.5, é apresentada uma visualização na perspectiva `Projeto` quando o usuário compara duas versões de um sistema de software. Nessa visualização, as medidas de "*V1*" ficam à esquerda e as medidas de "*V2*" à direita na tela ("lado a lado"). Essa visualização também é utilizada na perspectiva `Desenvolvedor` (Figura 6.6). Nessa perspectiva, é apresentada uma visualização "diferencial", conforme definido na técnica visual da *Developer Tracker*. O usuário pode intercalar entre as visualizações "lado a lado" e "diferencial" acessando as opções do botão "*VIEW CONFIGURATION*". O mesmo exemplo da Figura 6.6 é apresentado na Figura 6.7, porém com a visualização "diferencial" habilitada. As versões em comparação são um identificador *hash* de *commit* ("*V1*") e a *branch master* ("*V2*"). Pode-se observar que o *Dev 3* não integrava a equipe em "*V1*", mas ele está presente em "*V2*" e o desenvolvedor o *Dev 1* é incluído na medida *Truck Factor* em ambas versões.

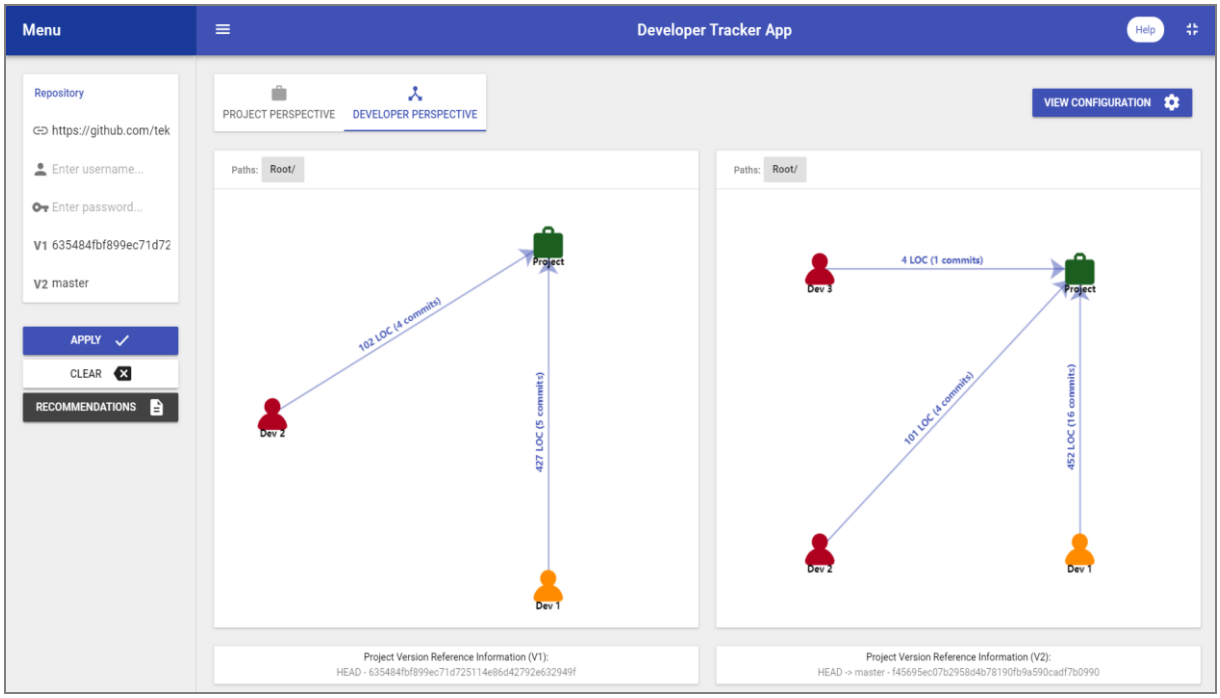
Por fim, há os botões "*Help*" e "*In/Out Full Screen*" localizados no canto superior direito das duas telas do no *Developer Tracker App*. O botão "*Help*" é acionado ao posicionar o *mouse* sobre ele, abrindo um cartão com informações de ajuda e legendas para ícones. O botão "*In/Out Full Screen*" é acionado com um clique, fazendo com que a janela do navegador ocupe toda a tela do usuário (*In Full Screen*) ou fique normal (*Out Full Screen*).

Figura 6.5 - Tela para visualização dos resultados comparativos na perspectiva `Projeto`.



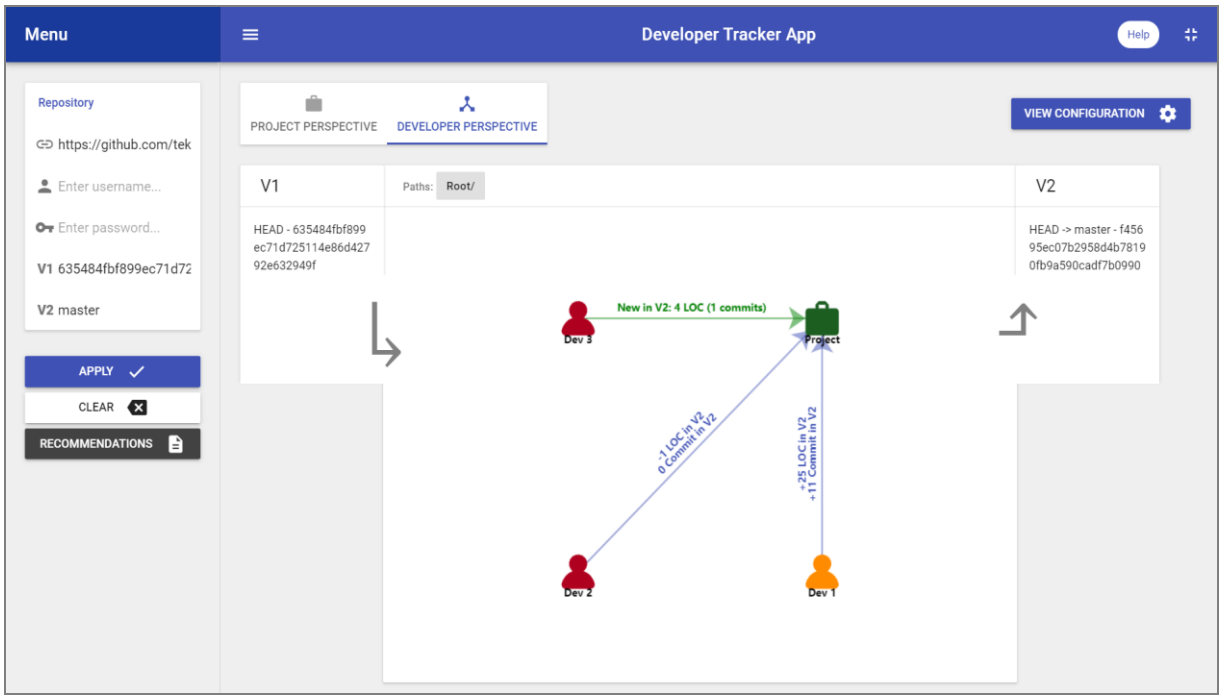
Fonte: Do Autor (2021).

Figura 6.6 - Tela para visualização dos resultados comparativos na perspectiva `Desenvolvedor` (visualização “lado a lado”).



Fonte: Do Autor (2021).

Figura 6.7 - Tela para visualização dos resultados comparativos na perspectiva `Desenvolvedor` (visualização “diferencial”).



Fonte: Do Autor (2021).

6.5 Considerações Finais

Neste capítulo, foi apresentado o sistema de software para *web Developer Tracker App*, um apoio computacional à abordagem proposta neste trabalho. Nesse sistema, podem ser selecionadas versões em SCV baseados em Git e as versões podem ser representadas por ramificações (*branches*), rótulos (*tags*) ou identificador *hash* de *commits*. Além disso, pode-se alternar entre as perspectivas `Desenvolvedor` e `Projeto`, bem como utilizar os três níveis de granularidade (projeto, diretório e artefato). Cabe ressaltar que, no *Developer Tracker App*, há a visualização comparativa entre duas versões de um sistema de software e a geração de um relatório de recomendações ao Gerente de Projetos.

7 AVALIAÇÃO DA ABORDAGEM *DEVELOPER TRACKER*

7.1 Considerações Iniciais

A *Developer Tracker* foi avaliada com a realização de um estudo empírico. O estudo consistiu na investigação dos efeitos da sua utilização na compreensão de Gerentes de Projetos sobre o trabalho dos desenvolvedores. Neste capítulo, são apresentados os procedimentos adotados e os resultados obtidos com a realização desse estudo.

O restante do capítulo está organizado da seguinte forma. O planejamento do estudo é descrito na Seção 7.2. A execução do estudo é relatada na Seção 7.3. Os resultados obtidos são apresentados e discutidos na Seção 7.4. As ameaças à validade são detalhadas na Seção 7.5.

7.2 Planejamento

A *Developer Tracker* é baseada na utilização de técnicas visuais para apresentação de informações. De modo geral, a finalidade das técnicas visuais é facilitar a percepção humana sobre determinados aspectos de sistemas de software. Para avaliar abordagens baseadas nessas técnicas, é indicada a condução de estudos empíricos (DI LUCCA; DI PENTA, 2006). Nesse sentido, a *Developer Tracker* foi avaliada por meio de um estudo empírico, cujo objetivo foi investigar os efeitos que as visualizações provocam na compreensão do Gerente de Projetos sobre o trabalho dos desenvolvedores.

De modo geral, o estudo empírico consistiu em realizar avaliações individuais com voluntários. Em cada avaliação, o voluntário executa um conjunto de tarefas e responde questões **não utilizando** e **utilizando** *Developer Tracker App*. As respostas das questões foram analisadas qualitativamente, permitindo identificar efeitos provocados pela utilização da abordagem, bem como as opiniões dos voluntários sobre ela. Os voluntários são Gerentes de Projetos que atuam na indústria de software e, portanto, são o público-alvo de *Developer Tracker* e/ou o *Developer Tracker App*.

Esta seção está organizada da seguinte forma. Na Subseção 7.2.1, é apresentado o protocolo de avaliação utilizado no estudo. Na Subseção 7.2.2, é detalhada a condução de um "estudo preliminar" no qual uma versão inicial do protocolo de avaliação foi aplicada.

7.2.1 Protocolo de Avaliação

O protocolo de avaliação utilizado foi composto pelos seguintes itens: i) termo de consentimento (ANEXO D); ii) conjunto de tarefas a serem executadas pelos participantes que consistem em compreender determinados aspectos do trabalho realizado pelos desenvolvedores; iii) roteiro de entrevistas; e iv) sequência de atividades para executar a avaliação. As tarefas a serem realizadas pelos participantes são:

a) **Tarefa 1. Compreender qual(is) é(são) a(s) tecnologia(s) mais demandada(s) no projeto de software.** O participante deve buscar informações sobre quais linguagens de programação (tecnologias) são utilizadas no projeto, bem como qual o grau de importância (quanto à quantidade de trabalho demandado) de cada uma. O participante conclui a tarefa quando afirmar "qual(is) é(são) a(s) tecnologia(s) mais demanda(s) no projeto" ou que "todas tecnologias são demandas de forma semelhante";
b) **Tarefa 2. Compreender quem é(são) o(s) desenvolvedor(es) que concentra(m) o conhecimento sobre o código do projeto de software.** O participante deve buscar informações sobre quem são os desenvolvedores e seus respectivos graus de importância para o projeto (quanto ao domínio/conhecimento do código). Se os desenvolvedores mais importantes deixarem o time antes do previsto, então aumentará o risco de o projeto fracassar, visto que esses desenvolvedores possuem alto grau de conhecimento sobre o código. O participante conclui a tarefa quando afirmar "quem é(são) o(s) desenvolvedor(es) que concentra(m) o conhecimento sobre o código" ou que "não há desenvolvedores concentrando o conhecimento sobre o código, ou seja, todos possuem conhecimento semelhante";
c) **Tarefa 3. Compreender qual(is) setor(es) do código demanda(m) mais quantidade de esforço dos desenvolvedores do time.** O participante deve buscar informações sobre os setores do código e o quanto os desenvolvedores têm trabalhado em cada setor. Um setor do código pode ser um diretório ou componente (*e.g.*, *backend*), um módulo (*e.g.*, módulo de gestão de usuários) ou um recurso específico (*e.g.*, exportação de relatórios em planilhas). A quantidade de esforço pode ser relacionada à frequência de demanda de trabalho ou à complexidade de implementar soluções em determinado setor do código. O participante conclui a tarefa quando afirmar "qual(is) setor(es) do código demanda(m) mais quantidade de esforço dos desenvolvedores" ou que "todos os setores do código demandam quantidade semelhante de esforço dos desenvolvedores";

d) **Tarefa 4. Compreender qual(is) setor(es) do código cada desenvolvedor tem mais ou maior atuação.** O participante deve buscar informações sobre quais setores do código cada desenvolvedor tem atuado com mais frequência. O participante conclui a tarefa quando afirmar "qual é o desenvolvedor com mais ou maior atuação em cada setor do código" ou que "todos os desenvolvedores possuem atuação semelhante em todos os setores do código";

e) **Tarefa 5. Compreender qual(is) mudança(s) ocorreu(ram) na atuação dos desenvolvedores no período entre o desenvolvimento de duas versões diferentes do projeto de software**. O participante deve buscar informações históricas de duas versões quaisquer de um sistema de software e como os desenvolvedores atuaram em cada uma. Uma versão pode ser referente a uma data, a uma ramificação do repositório (*branch*) ou a uma *tag* (*e.g.* v1.0.0). A mudança de atuação pode ser qualquer situação referente ao trabalho dos desenvolvedores. Por exemplo: *Dev1* é o principal desenvolvedor do módulo *M* na versão *V-A*, mas, durante o desenvolvimento da versão *V-B,* houve uma iniciativa para diversificar a atuação de desenvolvedores no módulo *M* e, como consequência, *Dev1* passou a trabalhar em outros módulos, deixando de ser o único desenvolvedor principal do módulo *M*. O participante conclui a tarefa quando afirmar que "consegue identificar uma mudança na atuação de, pelo menos, um desenvolvedor que ocorreu entre quaisquer duas versões do projeto" ou que "não houve mudanças na atuação dos desenvolvedores entre todas as versões do projeto".

A conclusão de uma tarefa deve refletir a percepção que o participante possui a respeito dos aspectos abordados. Assim, quando o participante conclui uma tarefa, ele possui convicção de que compreende os aspectos abordados na tarefa. Por outro lado, quando o participante não consegue concluir uma tarefa, ele não conseguiu atingir o grau de compreensão necessário para completá-la. É importante ressaltar que a única informação registrada sobre cada tarefa é se o participante conseguiu ou não concluir, visto que informações adicionais são confidenciais, como nomes de desenvolvedores e tecnologias do projeto. O roteiro de entrevistas é composto por cinco grupos de questões (ANEXO E):

a) **Grupo 1**. Contém questões para caracterizar a experiência e as atividades executadas pelo participante;

b) **Grupo 2**. Contém questões para caracterizar o projeto de software quanto à duração do projeto e à quantidade de desenvolvedores do time;

c) **Grupo 3**. Contém questões para verificar a compreensão que o participante tem a respeito do trabalho dos desenvolvedores, considerando as tarefas concluídas sem utilizar o *Developer Tracker App*;
d) **Grupo 4**. Contém questões para verificar os efeitos da utilização do *Developer Tracker App* na compreensão que o participante possui a respeito do trabalho dos desenvolvedores. Nessas questões, foram considerados quatro efeitos sobre a compreensão dos participantes
    - **Aprimora a compreensão**. Esse efeito ocorre quando o participante julga ter obtido novas informações e novas percepções que aprimoram sua capacidade de compreensão;
    - **Confirma a compreensão**. Esse efeito ocorre quando o participante julga ter aumentado a confiança em sua compreensão, apesar de não a ter aprimorado com novas informações e/ou percepções;
    - **Não interfere na compreensão**. Esse efeito ocorre quando o participante não consegue concluir a tarefa e/ou julga ter recebido informações que não apoiam nem atrapalham a sua compreensão;
    - **Piora a compreensão**. Esse efeito ocorre quando o participante julga ter obtido informações consideradas equivocadas ou confusas, piorando a sua compreensão.
e) **Grupo 5**. Contém questões referentes à utilidade geral da abordagem e como ela pode ajudar e/ou atrapalhar o participante.

A execução da avaliação envolve a aplicação do roteiro de entrevistas, a execução das tarefas e outras atividades. Contudo, anteriormente à execução da avalição, é preciso identificar potenciais participantes para enviar um convite contendo explicação da pesquisa e os procedimentos da avaliação. Ao aceitar o convite, é agendada uma entrevista com esse participante para realizar a avaliação que seguiu a seguinte sequência:

a) Leitura e concordância do termo de consentimento pelo participante;
b) Aplicação das questões do Grupo 1 e do Grupo 2 do roteiro de entrevista;
c) Execução do conjunto de tarefas sem utilizar o *Developer Tracker App* (o participante pode consultar suas fontes de informação para compreender o trabalho dos desenvolvedores, por exemplo, documentos, ferramentas, pessoas, métricas, conhecimento empírico do gerente);
d) Aplicação das questões do Grupo 3 do roteiro de entrevista;

e) Execução das tarefas utilizando o *Developer Tracker App* (o participante pode consultar suas fontes informação e visualizações geradas pelo *Developer Tracker App*. O participante receberá instruções sobre a funcionalidade do *Developer Tracker App*);
f) Aplicação das questões do Grupo 4 do roteiro de entrevista;
g) Aplicação das questões do Grupo 5 do roteiro de entrevista.

7.2.2 Estudo Preliminar

Foi estabelecida a realização de um "estudo preliminar" com o objetivo de identificar e efetuar melhorias na *Developer* Tracker, no *Developer Tracker App* e no protocolo de avaliação. O estudo preliminar foi conduzido com três Gerentes de Projetos, de maneira de remota, atuantes em duas empresas diferentes. Esses participantes foram Gerentes de Projetos diferentes dos entrevistados para o MSL (Capítulo 4). Os principais resultados foram:

a) **Identificação de oportunidades de melhoria na abordagem *Developer Tracker***. A técnica visual baseada no paradigma iconográfico foi criticada por um participante ao visualizar, comparativamente, duas versões de um sistema de software. A visualização comparativa consistia em mostrar dois gráficos lado a lado, sendo um gráfico para cada versão (Figura 6.6). Nesse sentido, o participante sugeriu inserir a opção de visualizar a diferença das conexões entre os desenvolvedores e os artefatos considerando duas versões. Assim, a visualização apresentada na Figura C.4 foi definida e adicionada à técnica visual;
b) **Identificação de oportunidades de melhoria no apoio computacional *Developer Tracker App***. A nova visualização comparativa inserida na técnica visual da *Developer Tracker* foi implementada no *Developer Tracker App* (Figura 6.7). Além disso, os três participantes criticaram o tempo de espera para a ferramenta gerar as visualizações. Até então, a ferramenta havia sido testada em projetos com menos artefatos e menos desenvolvedores do que a realidade dos projetos disponibilizados pelos participantes. Com o intuito de melhorar a performance em projetos maiores, alguns processamentos do *Developer Tracker App* foram modificados, paralelizando os cálculos das medidas e diminuindo o tempo de espera. Outra sugestão de um dos participantes foi a possibilidade de filtrar, além de artefatos e/ou desenvolvedores, os arquivos de determinada linguagem de programação. O objetivo desse filtro é focar a visualização somente em tecnologias de interesse. Assim, o filtro de artefatos foi melhorado para permitir a seleção de extensões de arquivos de linguagens de programação de interesse;
c) **Identificação de possíveis desafios para realização do estudo empírico**. Uma preocupação comum entre os três participantes foi a garantia de segurança ao compartilhar

o código do projeto, que deve ser importado para o *Developer Tracker App*. Três opções foram oferecidas aos Gerentes de Projetos importarem o código do projeto: i) utilizar o computador próprio (necessita que o participante instale o servidor do *Developer Tracker App* em seu computador); ii) utilizar um *link on-line* (necessita que o pesquisador hospede o *Developer Tracker App* em um serviço de nuvem); e iii) utilizar o computador do pesquisador (necessita que o pesquisador instale o *Developer Tracker App* em seu computador e conceda o controle do computador ao participante). Todos os participantes escolheram a terceira opção. O problema da primeira opção é a necessidade de tempo de instalação e a segunda opção foi considerada insegura por ser uma operação realizada em um serviço de nuvem (sem controle total do pesquisador e do participante). Para viabilizar a terceira opção o setor de segurança da empresa de um dos participantes foi consultado. Assim, o aplicativo *AnyDesk*[17] foi sugerido como o mais seguro e com licença gratuita para ceder o controle do computador do pesquisador ao participante. Outra dificuldade foi agendar a avaliação com os participantes. As avaliações precisaram ser remarcadas até três vezes por causa de imprevistos e urgências no trabalho dos participantes;

d) **Identificação de oportunidades de melhoria no termo de consentimento**. Uma dúvida que havia antes da realização do "estudo preliminar" era a quantidade de tempo necessária para a avaliação com cada participante. O tempo foi medido durante as avaliações e a duração aproximada foi 90 minutos. Assim, essa informação foi adicionada ao termo de consentimento para o participante reservar 90 minutos para participar da avaliação;
e) **Confirmação de que o conjunto de tarefas está bem definido**. Não foram identificadas necessidades de adicionar, remover ou modificar o conjunto de tarefas.
f) **Identificação de oportunidades de melhoria no roteiro de entrevistas**. Foi identificada a necessidade de adicionar uma nova questão ao Grupo 1 (questão 1.3) com o objetivo de registrar quantos projetos ativos está sob a gerência do participante em sua organização. A quantidade de projetos gerenciados simultaneamente pode influenciar no tempo de dedicação do Gerente de Projetos para cada projeto, podendo provocar impactos na execução das tarefas. Além disso, foi observado que todos os projetos disponibilizados pelos participantes possuem duração superior a 4 anos e possuem time com mais de 5 desenvolvedores. Assim, as alternativas de resposta das questões dos grupos 1 e 2 foram modificadas para permitir melhor caracterização dos projetos que possuem duração superior a 4 anos e times com mais de 5 desenvolvedores;

[17] https://anydesk.com/

g) **Confirmação de que a sequência de atividades para execução da avaliação está bem definida.** Não foram identificadas necessidades de adicionar, remover ou modificar as atividades e suas respectivas sequências.

Quanto aos resultados referentes aos efeitos da *Developer Tracker*, cabe destacar que: i) utilizando a *Developer Tracker*, os três participantes aprimoram ou confirmaram seu grau de compreensão; ii) para os três participantes, a principal utilidade do uso da *Developer Tracker* é a identificação dos desenvolvedores que concentram o conhecimento no projeto; e iii) dois participantes consideraram que o *Developer Tracker App* é útil e um participante o considerou extremamente útil.

7.3 Execução

O estudo empírico foi realizado com profissionais da indústria de software que atuam como Gerentes de Projetos e que não participaram do "estudo preliminar" (Seção 7.2.2) nem das entrevistas referentes ao MSL (Capítulo 4). É importante ressaltar que um participante não necessariamente tem o cargo "Gerente de Projetos", mas executa pelo menos uma das cinco atividades sugeridas para Gerentes de Projeto de Software (SOMMERVILLE, 2019): i) planejamento do projeto; ii) geração de relatórios; iii) gestão de riscos; iv) gestão de pessoas; e v) elaboração de propostas. Assim, profissionais da indústria de software foram convidados a participar da avaliação. Convites individuais contendo o termo de consentimento foram enviados para 20 profissionais. Esses profissionais foram selecionados por conveniência. Desses convites, 16 convites foram aceitos (80%), 3 convites foram recusados (15%) e 1 convite não foi respondido (5%). A justificativa dos três profissionais que recusaram o convite é relacionada à restrição de importar o código do projeto para o *Developer Tracker App*. Contudo, a quantidade de participantes que aceitaram o convite está de acordo com as recomendações sugeridas para estudos experimentais que envolvem entrevistas e análises qualitativas (MARSHALL *et al.*, 2013).

Foram agendadas avaliações individuais com cada participante. A dificuldade de agendamento percebida ao executar o "estudo preliminar" ocorreu da mesma forma. Além disso, a duração de aproximadamente 90 minutos foi respeitada em todas as avaliações. As avaliações ocorreram de forma remota e todos os participantes escolheram a opção de controlar o computador do pesquisador utilizando o software *AnyDesk*. A sequência de atividades estabelecida no protocolo de avaliação foi seguida rigorosamente. Por fim, as

respostas das questões do roteiro de entrevistas foram armazenadas em um formulário do *Google Forms*[18] para facilitar a análise dos resultados.

7.4 Resultados e Discussão

Nesta seção, os resultados são apresentados e discutidos em três etapas. A primeira etapa é referente à caracterização dos participantes e dos projetos que eles disponibilizaram para a avaliação. Na segunda etapa, são apresentados e discutidos os resultados referentes à compreensão dos participantes a respeito do trabalho dos desenvolvedores no projeto. Na terceira etapa, são abordados os efeitos do uso da *Developer Tracker*, por meio do *Developer Tracker App*.

Esta seção está organizada da seguinte forma. Na Subseção 7.4.1, são tratados os resultados da primeira etapa. Na Subseção 7.4.2, são tratados os resultados da segunda etapa. Na Subseção 7.4.3, são tratados os resultados da terceira etapa.

7.4.1 Caracterização de Participantes e Projetos

7.4.1.1 Resultados

Os participantes foram caracterizados considerando as respostas do Grupo 1 do roteiro de entrevistas. Na Tabela 7.1, é apresentado um resumo dessas respostas dos 16 participantes. Pode-se observar que os participantes estão distribuídos em quatro empresas[19] diferentes, sendo 7 participantes (43,75%) na empresa A, 7 participantes (43,75%) na empresa B, 1 participante (6,25%) na empresa C e 1 participante (6,25%) na empresa D. Com relação aos três profissionais que recusaram o convite para participar da avaliação e ao profissional que não respondeu ao convite, dois deles pertenciam a uma dessas empesas e outros dois pertenciam a empresas diferentes.

A função dos participantes na empresa em que trabalham foi caracterizada de acordo com as atividades de gerenciamento de projetos que eles exercem. Os 16 participantes[20] realizam o “planejamento do projeto” ou a “gestão de pessoas”. Além disso, 14 participantes (87,50%) executam ambas atividades. De forma mais específica, apenas P7 não realiza a atividade “planejamento do projeto” e P11 não realiza a atividade “gestão de pessoas”. As atividades “geração de relatórios” e “elaboração de propostas” são executadas por 11

[18] https://www.google.com/intl/pt-BR/forms/about/
[19] Para manter o anonimato das empresas, letras foram utilizadas as quais não possuem relação com o nome ou qualquer outra referência à empresa.
[20] Os participantes são referenciados pelo identificador Pn, sendo `n` um número sequencial de 1 a 16.

participantes (68,75%) cada uma. A atividade “gestão de riscos” é executada por 8 participantes (50%). Com relação ao tempo de experiência em executar as atividades informadas, destaca-se que 8 participantes (50%) possuem entre 3 e 8 anos de experiência, 1 participante (6,2%) possui mais de 8 anos de experiência, 4 participantes (25%) possuem entre 1 e 3 anos e 3 participantes (18,8%) possuem menos de 1 ano.

Tabela 7.1 - Resumo das respostas do Grupo 1 de questões do roteiro de entrevistas.

Participante	Empresa	Atividades executadas	Experiência (anos)	Experiência (projetos)	Quantidade de projetos simultâneos
P1	**A**	Planejamento; Relatórios; Pessoas; Propostas	Entre 3 e 8	+ de 8	+ de 4
P2	**A**	Planejamento; Relatórios; Riscos; Pessoas; Propostas	+ de 8	Entre 6 e 8	2
P3	**A**	Planejamento; Relatórios; Riscos; Pessoas; Propostas	Entre 3 e 8	Entre 4 e 5	3
P4	**A**	Planejamento; Pessoas; Propostas	Entre 3 e 8	Entre 4 e 5	1
P5	**A**	Planejamento; Relatórios; Riscos; Pessoas; Propostas	- de 1	Entre 2 e 3	2
P6	**A**	Planejamento; Relatórios; Pessoas; Propostas	Entre 3 e 8	Entre 2 e 3	1
P7	**A**	Relatórios; Pessoas	Entre 3 e 8	+ de 8	2
P8	**B**	Planejamento; Pessoas; Propostas	Entre 1 e 3	Entre 4 e 5	1
P9	**B**	Planejamento; Relatórios; Riscos; Pessoas; Propostas	Entre 3 e 8	Entre 4 e 5	3
P10	**B**	Planejamento; Relatórios; Pessoas	Entre 3 e 8	Entre 6 e 8	1
P11	**B**	Planejamento; Riscos; Propostas	- de 1	Entre 2 e 3	1
P12	**B**	Planejamento; Relatórios; Pessoas	Entre 1 e 3	1	1
P13	**B**	Planejamento; Relatórios; Riscos; Pessoas	- de 1	1	1
P14	**B**	Planejamento; Relatórios; Pessoas; Propostas	Entre 3 e 8	+ de 8	1
P15	**C**	Planejamento; Riscos; Pessoas	Entre 1 e 3	Entre 2 e 3	1
P16	**D**	Planejamento; Riscos; Pessoas; Propostas	Entre 1 e 3	+ de 8	+ de 4

Legenda da coluna “Atividades executadas”:
Planejamento: *Planejamento do projeto;* ***Relatórios:*** *Geração de relatórios;* ***Riscos:*** *Gerenciamento de riscos;* ***Pessoas:*** *Gerenciamento de pessoas;* ***Propostas:*** *Elaboração de propostas*

Legenda da Escala de cores

	Maior valor da coluna
	Segundo maior valor da coluna
	Terceiro maior valor da coluna
	Quarto maior valor da coluna
	Menor valor da coluna

Fonte: Do Autor (2021).

A quantidade de projetos que cada participante gerenciou durante o tempo de experiência também foi respondido. Dessa forma, 4 participantes (25%) gerenciaram mais de 8 projetos, 2 participantes (12,50%) gerenciaram entre 6 e 8 projetos, 4 participantes (25%) gerenciaram 4 ou 5 projetos, 4 participantes (25%) gerenciaram 2 ou 3 projetos e 2

participantes (12,50%) gerenciaram 1 projeto. Por fim, os participantes foram questionados sobre quantos projetos gerenciam simultaneamente ao projeto compartilhado na avaliação. Assim, 2 participantes (12,50%) gerenciaram outros quatro projetos, 2 participantes (12,50%) gerenciaram outros dois projetos simultaneamente, 3 participantes (18,80%) gerenciaram um outro projeto e 9 participantes (56,20%) gerenciam apenas o projeto disponibilizado.

As respostas do Grupo 2 do roteiro de entrevistas foram utilizadas para caracterizar os projetos disponibilizados pelos participantes para avaliação. Na Tabela 7.2, para os participantes e sua respectiva empresa, são listadas a duração de contrato e a quantidade de desenvolvedores do projeto. Os projetos da empresa A tem pelo menos 6 desenvolvedores. Além disso, apenas 2 projetos da empresa B não possuem mais de 8 desenvolvedores. O projeto com menor duração (menos de 1 ano) e com menos desenvolvedores (entre 2 e 3 desenvolvedores) é o projeto disponibilizado pelo participante da empresa C.

Tabela 7.2 - Resumo das respostas do Grupo 2 de questões do roteiro de entrevistas.

Participante	Empresa	Duração do projeto (anos)	Quantidade de desenvolvedores
P1	**A**	+ de 5	+ de 8
P2	**A**	+ de 5	Entre 6 e 8
P3	**A**	Entre 4 e 5	+ de 8
P4	**A**	Entre 2 e 4	+ de 8
P5	**A**	Entre 2 e 4	Entre 6 e 8
P6	**A**	+ de 5	+ de 8
P7	**A**	Entre 4 e 5	+ de 8
P8	**B**	Entre 2 e 4	+ de 8
P9	**B**	Entre 2 e 4	+ de 8
P10	**B**	Entre 4 e 5	Entre 6 e 8
P11	**B**	Entre 1 e 2	Entre 2 e 3
P12	**B**	+ de 5	+ de 8
P13	**B**	Entre 2 e 4	+ de 8
P14	**B**	Entre 4 e 5	+ de 8
P15	**C**	- 1 ano	Entre 2 e 3
P16	**D**	Entre 2 e 4	Entre 6 e 8

Legenda da Escala de cores
Maior valor da coluna
Segundo maior valor da coluna
Terceiro maior valor da coluna
Quarto maior valor da coluna
Menor valor da coluna

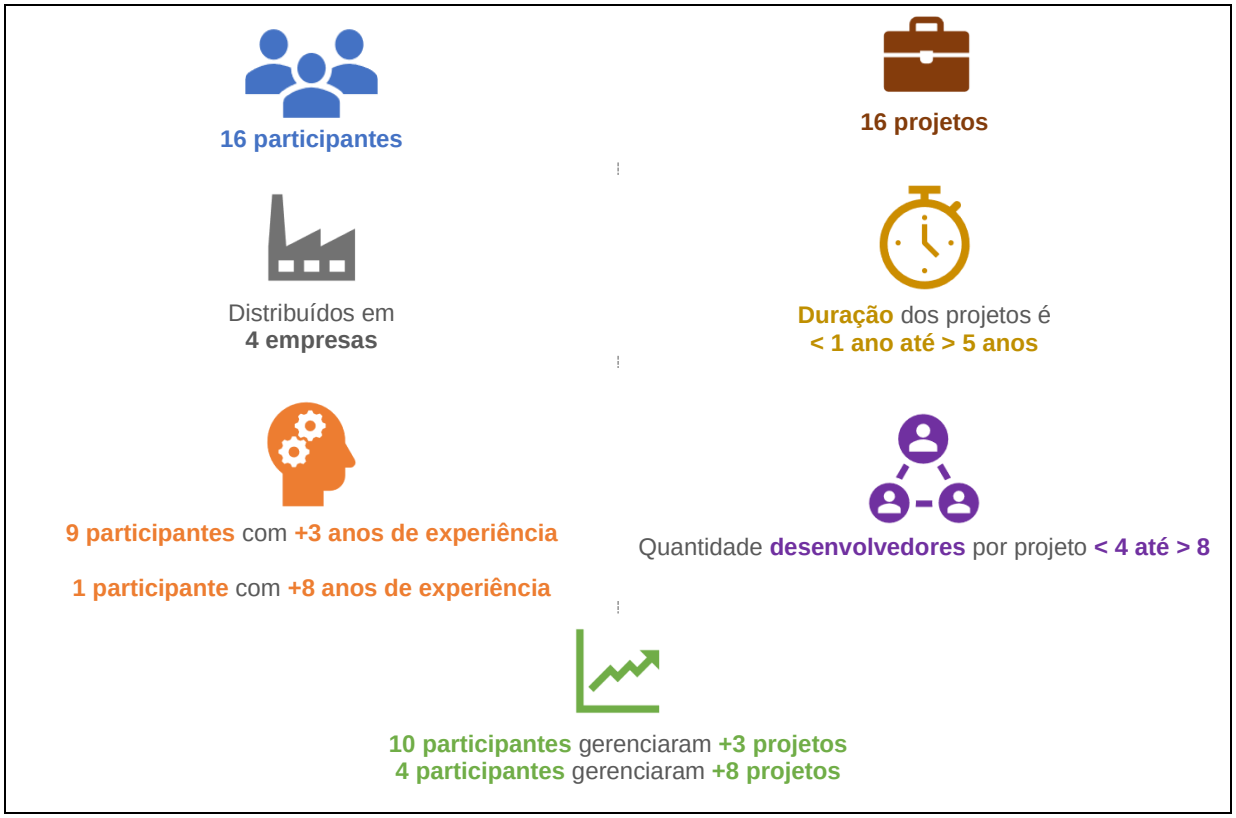
Fonte: Do Autor (2021).

7.4.1.2 Discussão

A caracterização dos participantes foi obtida com a análise das respostas às questões do Grupo 1 do roteiro de entrevistas. Enquanto para caracterizar os projetos disponibilizados

pelos participantes foram analisadas as repostas referentes às questões do Grupo 2. Na Figura 7.1, é apresentado um resumo das características dos participantes e dos projetos. Cada um dos 16 participantes disponibilizou um projeto para realização da avaliação individual. Os projetos possuem diferentes tempos de duração e diferentes quantidades de desenvolvedores. De forma geral, os projetos com menos tempo de duração (2 anos ou menos) possuem 2 ou 3 desenvolvedores, enquanto os projetos com maior duração (os projetos chegam a ter mais de 5 anos de duração) possuem mais de 8 desenvolvedores. Isso pode indicar que, dentre os projetos disponibilizados, contratos mais extensos demandaram um time maior. Além disso, pode indicar que a amostra possui projetos de diferentes características.

Figura 7.1 - Resumo da caracterização de participantes e projetos



Fonte: Do Autor (2021).

Um ponto interessante é não ser possível fazer relação entre a quantidade projetos que um participante tem experiência em gerenciar com as dimensões do projeto disponibilizado para avaliação. Por exemplo, P1 gerenciou mais de 8 projetos e disponibilizou um projeto que possui mais de 5 anos de duração e mais de 8 desenvolvedores. Em contrapartida, P12 gerenciou apenas 1 projeto, que também possui mais de 5 anos de duração e mais de 8 desenvolvedores. Ainda sobre a experiência dos participantes, pode-se notar variedade da amostra. Há participantes menos experientes (com menos de 3 anos de experiência), mas a maioria possui experiência igual ou superior a 3 anos (9 participantes).

Os participantes estão distribuídos em 4 empresas diferentes, sendo que 14 participantes estão distribuídos em duas dessas empresas (7 participantes na empresa A e 7 participantes na empresa B). Apesar dessa concentração de participantes, essas empresas possuem diferentes projetos em contextos diversos. As empresas C e D possuem menos projetos e não possuíam outros gerentes disponíveis para participar da avaliação. Outro questionamento realizado aos participantes foi referente a quantidade de projetos que eles gerenciam simultaneamente ao projeto disponibilizado para avaliação. O total de 9 participantes (56%) gerenciam apenas 1 projeto (projeto disponibilizado), tendo dedicação exclusiva ao time do projeto. Além disso, há participantes que gerenciam mais de quatro projetos simultaneamente (incluindo o projeto disponibilizado). Pode ser que a dedicação do Gerente de Projetos ao projeto pode influenciar sua capacidade de acompanhar o trabalho dos desenvolvedores e compreender os aspectos abordados no conjunto de tarefas da avaliação.

Concluindo, a caracterização dos participantes e dos projetos contém variados fatores que podem influenciar a compreensão do participante sobre o trabalho dos desenvolvedores. Esses fatores são: i) diferentes níveis de experiência dos participantes; ii) diferentes contextos relacionados às empresas e aos projetos dos participantes; iii) diferentes características de projetos (duração de contrato e quantidade de desenvolvedores); iv) diferentes níveis de dedicação dos participantes ao projeto disponibilizado para avaliação.

7.4.2 Compreensão Sobre o Trabalho dos Desenvolvedores

7.4.2.1 Resultados

Os participantes realizaram cinco tarefas sem utilizar o *Developer Tracker App*. Em seguida, responderam questões para confirmar se concluíram ou não as tarefas. Uma tarefa é concluída quando o participante afirma que compreende o trabalho realizado pelos desenvolvedores, referente ao aspecto abordado na tarefa. A **Tarefa 1** é referente às tecnologias demandadas no projeto. A **Tarefa 2** é referente à concentração de conhecimento entre os desenvolvedores do projeto. A **Tarefa 3** é referente ao trabalho demandado aos desenvolvedores em cada setor do código. A **Tarefa 4** é referente à relação de trabalho entre desenvolvedores e os setores do código. A **Tarefa 5** é referente à comparação da atuação dos desenvolvedores entre duas versões. Na Tabela 7.3, é apresentada a conclusão de cada participante em cada tarefa.

Pode-se notar que, a Tarefa 2 e a Tarefa 5 foram concluídas por todos os participantes e 4 participantes (P2, P7, P8 e P10) não concluíram alguma tarefa (25%). P10 foi o

participante com menos tarefas concluídas (apenas, Tarefa 2 e a Tarefa 5), sendo o único participante que não concluiu a Tarefa 1. A Tarefa 3 e a Tarefa 4 foram as que tiveram menor incidência de conclusão, sendo 3 participantes (Tarefa 3 - P2, P8 e P10; Tarefa 4 - P2, P7 e P10) não conseguiram concluí-las (18% dos participantes para cada tarefa).

Tabela 7.3 - Resumo das respostas do Grupo 3 de questões do roteiro de entrevistas.

Participante	Tarefa 1	Tarefa 2	Tarefa 3	Tarefa 4	Tarefa 5
P1	✓	✓	✓	✓	✓
P2	✓	✓	x	x	✓
P3	✓	✓	✓	✓	✓
P4	✓	✓	✓	✓	✓
P5	✓	✓	✓	✓	✓
P6	✓	✓	✓	✓	✓
P7	✓	✓	✓	x	✓
P8	✓	✓	x	✓	✓
P9	✓	✓	✓	✓	✓
P10	x	✓	x	x	✓
P11	✓	✓	✓	✓	✓
P12	✓	✓	✓	✓	✓
P13	✓	✓	✓	✓	✓
P14	✓	✓	✓	✓	✓
P15	✓	✓	✓	✓	✓
P16	✓	✓	✓	✓	✓

Tarefa 1. Compreender qual(is) é(são) a(s) tecnologia(s) mais demandada(s) no projeto de software.
Tarefa 2. Compreender quem é(são) o(s) desenvolvedor(es) que concentra(m) o conhecimento sobre o código do projeto de software.
Tarefa 3. Compreender qual(is) setor(es) do código demanda(m) maior quantidade de esforço dos desenvolvedores do time.
Tarefa 4. Compreender qual(is) setor(es) do código cada desenvolvedor tem maior atuação.
Tarefa 5. Compreender qual(is) mudança(s) ocorreu(ram) na atuação dos desenvolvedores no período entre o desenvolvimento de duas versões diferentes do projeto de software.

✓**-** Tarefa concluída pelo participante
x **-** Tarefa não concluída pelo participante

Fonte: Do Autor (2021).

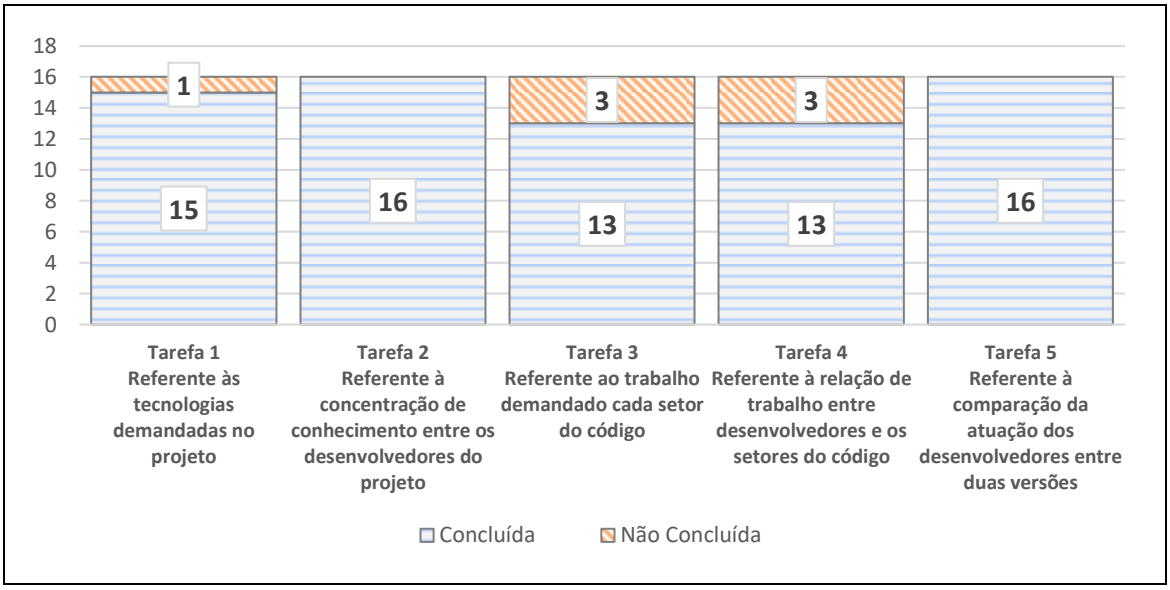
A execução das tarefas envolve a busca de informações sobre o trabalho dos desenvolvedores. Todos os participantes, quando questionados sobre quais fontes de informações utilizaram, responderam que fizeram uso de conhecimento empírico, baseando-se em percepções pessoais e acompanhamento da rotina diária do time. Adicionalmente ao conhecimento empírico, P1 afirmou ter consultado o software de gerenciamento de atividades do time, P5 disse que consultou um desenvolvedor do time e o P11 falou ter visto os registros de *log* do repositório de código do projeto.

7.4.2.2 Discussão

De modo geral, a quantidade de tarefas concluídas foi superior à quantidade de tarefas não concluídas (Figura 7.2). Destaque é 12 participantes (75%) terem concluído todas as tarefas e todos os participantes terem utilizado conhecimento empírico para realizar as tarefas. A utilização do conhecimento empírico pode ser explicada considerando a relevância de

compreender os aspectos abordados nas tarefas na rotina dos participantes. Geralmente, um Gerente de Projetos acompanha a rotina de trabalho do time e lida com a alocação de pessoas e a gestão de atividades. Assim, as tarefas podem ser consideradas essenciais para esses participantes e, portanto, eles possuem as respostas de forma quase imediata. Nesse sentido, P5 justifica "*... acompanho o projeto desde o início e já são pontos de preocupação em minha rotina*". Além de afirmar que as tarefas abordam "*pontos de preocupação*", a justificativa de P5 inclui o tempo de acompanhamento do projeto. De modo semelhante, outros participantes citaram o tempo de acompanhamento do projeto como justificativa. Em resumo, os participantes consideram que uma visão do "*histórico*" (P13, P15, P16), obtida pelo "*prazo*" (P14) de acompanhamento, muitas vezes, "*desde o início*" (P5, P6) do projeto, incrementado à "*vivência*" (P4) e à participação no "*dia a dia da equipe*" (P6, P9, P10, P12) fornecem insumos para compreender os aspectos a respeito do trabalho dos desenvolvedores abordados nas tarefas.

Figura 7.2 - Quantidade de tarefas concluídas e não concluídas pelos participantes.



Fonte: Do Autor (2021).

Outra possibilidade para explicar o fato da utilização do conhecimento empírico pelos participantes é a ideia de as tarefas serem "triviais" e não demandarem esforços para serem concluídas. Em outras palavras, apenas o histórico e o acompanhamento do time são suficientes para concluir as tarefas. Nesse sentido, P14 afirmou "*... algumas respostas são simples e óbvias para mim, mas algumas tenho hipóteses que considero confiáveis.*". Contudo, algumas opiniões de outros participantes contrariam a ideia de as tarefas serem "*simples*". Alguns participantes mostraram que o uso do conhecimento empírico é justificado pela conveniência. P10 considera que se trata da alternativa mais "*fácil*" e ponderou que essa

não é a melhor fonte de informação possível. Uma das preocupações de P10 com o uso do conhecimento empírico é a possibilidade de surgirem "*dificuldades*" quando, por exemplo, há trocas de Gerentes de Projetos, visto que as informações estão armazenadas somente na "*cabeça*" do atual gerente. Nesse sentido, P2 afirmou que pretende implantar processos para melhorar sua compreensão sobre o trabalho dos desenvolvedores e o uso do conhecimento empírico é justificado pela "*atual*" ausência de "*... estratégia para pensar sobre esses temas.”*. P9 considerou que "*... não usar ferramentas muito formais de gestão tem riscos, mas pela observação é possível perceber esses pontos com um bom custo-benefício.*". Em resumo, para alguns participantes, apesar dos riscos de usar conhecimento empírico, pode-se compreender de maneira "*fácil*" e com grau de precisão aceitável o trabalho realizado pelos desenvolvedores. Além disso, P2 e P9 afirmaram que ações devem ser tomadas para ampliar o grau de compreensão, implementando estratégias e processos que permitam consultar informações relevantes sobre o trabalho dos desenvolvedores.

Outro resultado interessante é duas tarefas (Tarefa 2 e Tarefa 5) terem sido concluídas por todos os participantes. Em geral, os participantes concluíram prontamente a Tarefa 2, justificando que há concentração de conhecimento sobre o código em um ou em até três desenvolvedores do time. Os participantes mostraram preocupação com a concentração de conhecimento, bem como citaram alguns inconvenientes provocados por ela, por exemplo, a dificuldade de encontrar substitutos ao "*planejar férias*" (P1, P6, P11, P16) dos desenvolvedores mais importantes. Com relação à Tarefa 5, os participantes justificaram, em muitos casos, que era possível perceber a mudança na atuação de algum desenvolvedor após a saída de outro desenvolvedor do time. Assim, conseguiram concluir a tarefa atendendo o requisito de citar pelo menos uma mudança na atuação de pelo menos um desenvolvedor em quaisquer duas versões do software.

Sobre a não conclusão de alguma tarefa, P10 foi o único a não concluir a Tarefa 1. Os demais participantes concluíram tal tarefa sem necessidade de grandes esforços. Segundo P10, apesar de conhecer as tecnologias do projeto, não possui capacidade de classificar a importância de cada uma considerando a demanda de trabalho, visto que não participa das decisões técnicas do time. Embora não tenha o feito, o participante poderia consultar diversas fontes de informação, tais como desenvolvedores do time e o próprio repositório de código do projeto. Assim, o P10 limitou a não concluir a tarefa.

A Tarefa 3 e a Tarefa 4 são relacionadas ao trabalho demandado em setores do código. Apesar da liberdade de definir os setores do código, alguns participantes não concluíram essas tarefas. Considerando todas as avaliações realizadas, foi possível notar que, nessas tarefas,

houve maior dificuldade de conclusão pelos participantes. P10 apresentou dificuldades em compreender aspectos relacionados à parte técnica do projeto, visto que não conseguiu concluir a Tarefa 1, a Tarefa 3 e a Tarefa 4. Cabe destacar que P10 possui dedicação exclusiva ao projeto, mas suas atividades estão mais focadas na alocação de recursos, planejamento e geração de relatórios. Outro participante com dedicação exclusiva ao projeto que relatou dificuldade na Tarefa 3 e na Tarefa 4 foi P8. Esse participante possui entre 1 e 3 anos de experiência, trabalhando em 4 ou 5 projetos ao longo desse período. P8 não conseguiu identificar o setor do código que demanda mais trabalho do time (Tarefa 3). Contudo, conseguiu citar os principais desenvolvedores de cada setor. Isso pode indicar que P8 compreende as tecnologias e a atuação dos desenvolvedores de acordo com as demandas, mas não possui clareza em determinar quais setores do código são mais críticos para o trabalho do time. P2 não conseguiu concluir as mesmas tarefas que P8 não concluiu e justificou que não acompanha estratégias de desenvolvimento do time.

Concluindo, a maioria dos participantes cumpriu os requisitos para concluir as tarefas. Para tanto, a principal fonte de informação utilizada foi o conhecimento empírico, sendo esse conhecimento fruto do acompanhamento do projeto e do time, muitas vezes, desde o início das atividades. Os principais motivadores para utilizar o conhecimento empírico são a conveniência e a relação custo-benefício. A Tarefa 1 não demandou grande esforço de 11 participantes, com exceção de P10, que não conseguiu compreender quais são as tecnologias mais demandadas no projeto. A Tarefa 2, em geral, foi concluída prontamente pelos participantes, visto que todos consideram que uma pequena parcela do time de desenvolvimento concentra o conhecimento sobre o código. A Tarefa 3 e a Tarefa 4 foram as que os participantes mostraram maior dificuldade para concluir. Tais tarefas estão relacionadas ao trabalho em setores do código. Por fim, a Tarefa 5 foi concluída, por muitos participantes, citando situações onde algum desenvolvedor deixou o time entre o desenvolvimento de duas versões diferentes.

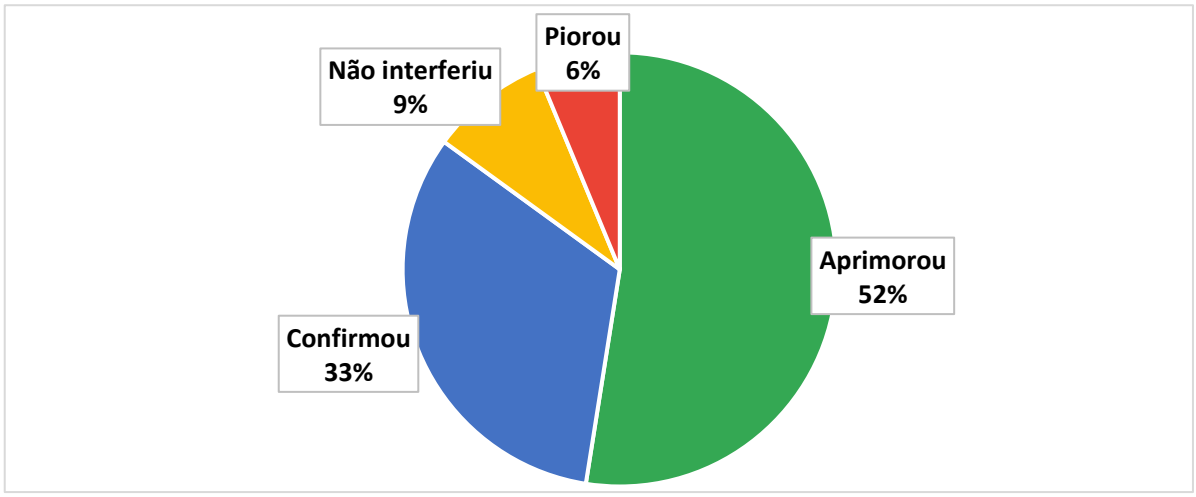
7.4.3 Efeitos do Uso de *Developer Tracker App*

7.4.3.1 Resultados

Os participantes executaram o conjunto de tarefas utilizando *Developer Tracker App*. Em seguida, foram questionados (questões do Grupo 4 do roteiro de entrevistas) sobre os seus efeitos na compreensão sobre os aspectos abordados nas tarefas. Os efeitos sobre a compreensão são: aprimoramento, confirmação, não interferência e piora. Na Figura 7.3, é

apresentada a proporção de ocorrência dos quatro efeitos. Assim, em mais da metade (52%) das oportunidades o uso do *Developer Tracker App* aprimorou a compreensão dos participantes sobre alguma tarefa. A confirmação da compreensão foi relatada em 33% das oportunidades. Em 9% das vezes, os participantes disseram que o uso do *Developer Tracker App* não inferiu em sua compreensão. Em algumas oportunidades (6%), os participantes consideraram que sua compreensão foi piorada ou confundida.

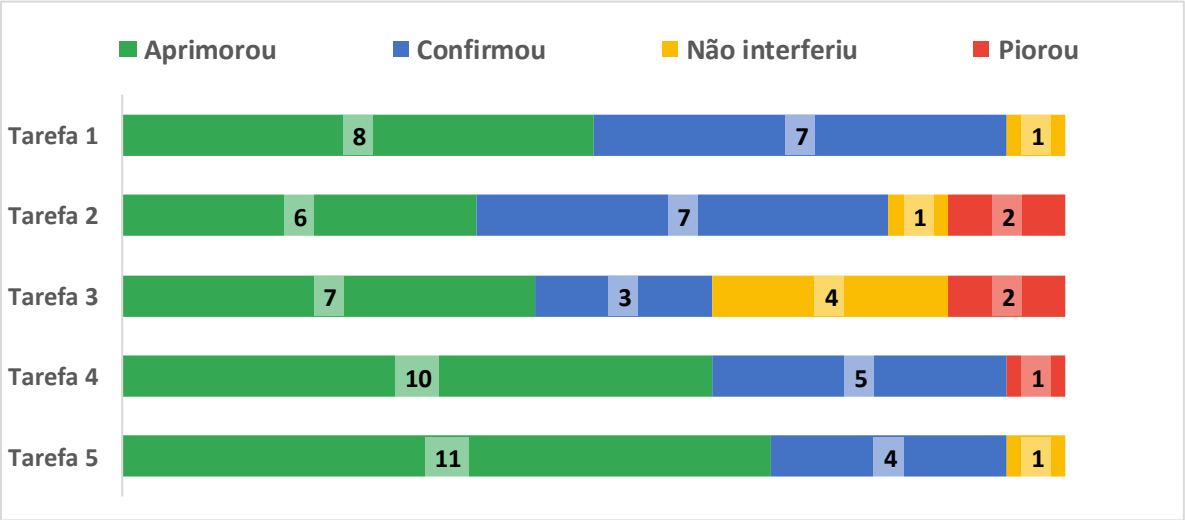
Figura 7.3 - Proporção de ocorrência dos quatro efeitos considerando as respostas dos participantes sobre todas tarefas.



Fonte: Do Autor (2021).

Com relação aos efeitos em cada uma das tarefas, pode-se observar, na Figura 7.4, que a tarefa com maior ocorrência de aprimoramento da compreensão foi a Tarefa 5 (11 vezes). Na Tarefa 1, na Tarefa 3 e na Tarefa 4, o aprimoramento foi o efeito mais relatado pelos participantes (8, 7 e 10 vezes, respectivamente). Contudo, na Tarefa 2, o efeito que mais ocorreu foi a confirmação da compreensão (7 vezes), seguido do aprimoramento (6 vezes). Além disso, na Tarefa 1, na Tarefa 2, na Tarefa 3 e na Tarefa 5, alguns participantes relataram que o uso do *Developer Tracker App* não interferiu na compreensão e, na Tarefa 2, na Tarefa 3 e na Tarefa 4, houve ocorrência de piora na compreensão dos participantes. Na Tabela 7.4, na qual estão listados os efeitos percebidos pelos participantes em cada tarefa, observa-se que cinco participantes (P5, P8, P9, P12, P14) relataram que não houve interferência na compreensão e quatro participantes (P3, P6, P7, P9) relataram que a compreensão piorou. Chama atenção as percepções dos P2 e P13, que afirmaram efeito de aprimoramento em todas as tarefas, do P9, que não percebeu efeito de aprimoramento em nenhuma tarefa, e do P11, que confirmou sua compreensão em todas as tarefas.

Figura 7.4 - Relação de efeitos ocorridos por tarefa.



Fonte: Do Autor (2021).

Tabela 7.4 - Relação de efeitos por participante em cada tarefa.

Participante	Tarefa 1	Tarefa 2	Tarefa 3	Tarefa 4	Tarefa 5
P1	Confirmou	Aprimorou	Aprimorou	Aprimorou	Confirmou
P2	Aprimorou	Aprimorou	Aprimorou	Aprimorou	Aprimorou
P3	Aprimorou	Confirmou	Piorou	Piorou	Confirmou
P4	Confirmou	Aprimorou	Aprimorou	Aprimorou	Aprimorou
P5	Aprimorou	Confirmou	Não interferiu	Confirmou	Não interferiu
P6	Aprimorou	Piorou	Confirmou	Aprimorou	Aprimorou
P7	Confirmou	Confirmou	Piorou	Aprimorou	Aprimorou
P8	Confirmou	Aprimorou	Não interferiu	Aprimorou	Aprimorou
P9	Não interferiu	Piorou	Não interferiu	Confirmou	Confirmou
P10	Aprimorou	Confirmou	Aprimorou	Aprimorou	Aprimorou
P11	Confirmou	Confirmou	Confirmou	Confirmou	Confirmou
P12	Confirmou	Não interferiu	Confirmou	Aprimorou	Aprimorou
P13	Aprimorou	Aprimorou	Aprimorou	Aprimorou	Aprimorou
P14	Aprimorou	Confirmou	Não interferiu	Aprimorou	Aprimorou
P15	Aprimorou	Confirmou	Aprimorou	Confirmou	Aprimorou
P16	Confirmou	Aprimorou	Aprimorou	Confirmou	Aprimorou

Fonte: Do Autor (2021).

Com relação às questões do Grupo 5 do roteiro de entrevistas, os participantes foram questionados sobre quais de suas atividades poderiam ser apoiadas ou prejudicadas pelo uso do *Developer Tracker App*, bem como qual é o grau de utilidade do *Developer Tracker App*. As respostas de todos os participantes para essas questões estão listadas no ANEXO F. Na Tabela 7.5, as respostas foram agrupadas com relação a sua similaridade, considerando o significado das atividades apoiadas pela compreensão do trabalho dos desenvolvedores (Tabela 4.5). Algumas atividades citadas nas respostas não possuem relação de similaridade com as atividades e, por isso, não aparecem na Tabela 7.5. O mapeamento das atividades agrupadas é apresentado no ANEXO G.

Tabela 7.5 - Agrupamento das atividades apoiadas pelo uso do *Developer Tracker App*

Atividade	Significado	Participantes
Identificação das habilidades e perfil do desenvolvedor	Consiste em identificar a capacidade técnica de um desenvolvedor em tecnologias de interesse com o objetivo de apoiar a montagem de equipes e a alocação de tarefas	P1, P2, P6, P7, P9, P10, P11, P13, P14, P15, P16
Compreensão e controle da distribuição do conhecimento	Consiste em identificar os desenvolvedores que concentram o conhecimento sobre a implementação e tomar ações para gerenciar os riscos relacionados	P1, P2, P3, P4, P5, P6, P12, P14, P16
Estimativa de custos e prazos de projetos e identificação de anomalias no desempenho do desenvolvedor	Consiste em criar planos de entrega baseado em dados históricos da atuação do time em tipos de demandas diferentes e perceber mudanças na atuação de determinado desenvolvedor ou do time	P3, P4, P8, P9, P10, P11
Melhoria no desempenho das equipes	Consiste em medir e promover ações para melhorar a performance dos desenvolvedores do projeto	P2, P5
Reajustes de remuneração	Consiste em realizar avaliações e tomar ações para promover ou reconhecer de forma financeira algum desenvolvedor	P9

Fonte: Do Autor (2021).

Além de apresentar as atividades apoiadas pelo uso do *Developer Tracker App*, os participantes P1 e P11 justificaram que a grande vantagem em o utilizar é captura dos dados de forma "*fácil e prática*". Nenhum dos 16 participantes comentou sobre o tempo de espera do processamento, como ocorreu na execução do "estudo preliminar". Entretanto, os participantes tiveram a oportunidade de opinar sobre como o uso do *Developer Tracker App* pode atrapalhar suas atividades (Tabela 7.6). A principal preocupação de P2, P9, P12 e P16 é a ausência de aspectos de qualidade nas medidas NLOC e Quantidade de *Commits*, visto que maiores valores para essas medidas não significam contribuições melhores. Esse foi o mesmo argumento de P5 sobre a ausência de aspectos de complexidade do trabalho realizado.

Tabela 7.6 - Opinião dos participantes sobre como o uso do *Developer Tracker App* pode atrapalhar suas atividades.

Participante	Como o uso do *Developer Tracker App* pode atrapalhar as atividades?
P2, P9, P12, P16	Ausência de aspectos de qualidade nas medidas NLOC e Quantidade de *Commits*.
P3, P15	Concentração de conhecimento em arquivos irrelevantes.
P10, P11	Acúmulo do histórico do valor das medidas NLOC e Quantidade de *Commits* pode confundir a visão recente.
P14	Aplicação prática seria preterida em detrimento do processo da empresa.
P5	Ausência do aspecto complexidade nas medidas NLOC e Quantidade de *Commits*.
P6	Limitação de um único repositório pode trazer dificuldades em alguns projetos.
P1	Usar essas medidas como julgamento final é um erro.

** P4, P7, P8, P13 consideram que não são atrapalhados pelo uso do Developer Tracker App.*

Fonte: Do Autor (2021).

Ainda sobre as medidas NLOC e Quantidade de *Commits*, P10 e P11 afirmaram que aplicá-las em todo o histórico do projeto pode trazer um descompasso em relação à situação atual do código, visto que aparecem contribuições de desenvolvedores antigos que não compõe mais o time. Além disso, destaca-se a argumentação de P3 e P15, referente à inclusão, na medida *Truck Factor*, das contribuições em setores do código relacionados aos testes ou aos arquivos binários (esses setores seriam irrelevantes, segundo os participantes).

Outros pontos de preocupação foram levantados: i) P14 falou sobre a inclusão do *Developer Tracker App* no processo da empresa, afirmando que não teria boa adesão; ii) P6 falou sobre a limitação de visualização de um único repositório de código por projeto, alertando que há projetos que possuem mais de um repositório e não poderiam ser analisados ao mesmo tempo; iii) P1 citou que confiar unicamente nos valores das medidas pode levar a compreensões falhas a respeito do trabalho dos desenvolvedores; e iv) P4, P7, P8 e P13 afirmaram que o uso do *Developer Tracker App* não atrapalharia suas atividades.

Por fim, os participantes expressaram sua opinião sobre o grau de utilidade do *Developer Tracker App* que ser "Extremamente útil", "Útil", "Inútil" ou "Inútil e prejudicial". As opiniões individuais estão listadas no ANEXO F. Das respostas, 9 participantes (56,6%) consideraram o *Developer Tracker App* "Útil" com a principal justificativa de eles conseguirem informações valiosas, mas usá-lo não é essencial para garantir boas tomadas de decisão, 7 participantes (43,4%) consideraram o *Developer Tracker App* "Extremamente útil" e nenhum participante considerou as opções "Inútil" e "Inútil e prejudicial".

7.4.3.2 Discussão

A utilização do *Developer Tracker App* e as respostas para as questões do Grupo 4 do roteiro de entrevistas geraram resultados a respeito dos efeitos da compreensão do trabalho dos desenvolvedores. Em 52% das oportunidades, os participantes relataram o efeito de aprimoramento de sua compreensão. Houve também oportunidades em que foram relatados os efeitos "confirmação", "não interferiu" e "piorou". Cabe discutir, esses efeitos em cada uma das tarefas.

Os efeitos na Tarefa 1 foram aprimoramento da compreensão (P2, P3, P5, P6, P10, P13, P14, P15), confirmação da compreensão (P1, P4, P7, P8, P11, P12, P16) e não interferiu na compreensão (P9). Dos 8 participantes que consideraram o efeito de aprimoramento, apenas P10 não concluiu a tarefa sem o uso do *Developer Tracker App*. Assim, para esse participante, o aprimoramento foi nítido. P2, P3, P5 e P6 assumiram que sua compreensão anterior ao uso do *Developer Tracker App* estava errada e, portanto, fez mais sentido considerar os resultados visuais. P6, P14 e P15 afirmaram que a observação do valor quantitativo (porcentual de linhas código em cada linguagem programação) foi decisivo para o aprimoramento de suas percepções empíricas sobre as tecnologias demandadas no projeto. Os participantes que confirmaram sua compreensão justificaram que principais tecnologias estavam dentro de uma margem de variação esperada, mas o grau de confiança aumentou ao

acessar as visualizações do *Developer Tracker App*, especialmente ao observar as tecnologias menos demandadas no projeto.

Ainda sobre a Tarefa 1, cabe ressaltar que P9 considerou que a visualização das tecnologias demandas no projeto não interferiu em sua compreensão, visto que possui plena convicção desse aspecto e não necessita do *Developer Tracker App* para apoiá-lo no projeto em questão. Alguns participantes ligaram a Tarefa 1 à atividade "Identificação das habilidades e perfil do desenvolvedor" e similares. A ligação existe porque os desenvolvedores do projeto possuem habilidade nas tecnologias demandas no projeto e, além disso, pode-se identificar as especialidades de cada desenvolvedor ao observar o relatório de recomendações e as visualizações da perspectiva *Developer*.

> Sobre a Tarefa 1, na opinião dos participantes, o *Developer Tracker App* apoiou sua compreensão, especialmente evitando erros oriundos de percepções empíricas e detalhando o percentual de linhas de código de cada tecnologia demandada no projeto. As visualizações podem ajudar os participantes a realizar atividades relacionadas ao planejamento de equipes.

Na Tarefa 2, diferentemente da Tarefa 1, houve participantes que consideraram que sua compreensão piorou a respeito da concentração de conhecimento. P6 afirmou que seu projeto é referente a um software utilizado em vários outros projetos. Assim, havia um desenvolvedor de outro time incluído no *Truck Factor*. Por esse motivo, o participante não reconheceu o grau de autoria de tal desenvolvedor sobre o código do projeto. De forma análoga, P9 não concordou com a inclusão no *Truck Factor* de um desenvolvedor com menos tempo de atuação no projeto que outros desenvolvedores. Após investigar o caso, foi detectado que o desenvolvedor contestado por P9 realizou uma tarefa de refatoração e "ganhou" a autoria de vários arquivos. Adicionalmente, o desenvolvedor utilizou recurso de um *framework* do projeto para gerar, de forma automática, diversos arquivos de configuração. Logo, esse desenvolvedor também assumiu a autoria dos arquivos gerados pelo *framework*. Contudo, os arquivos gerados pelo *framework* não requerem manutenção do time e, portanto, na visão de P9, deveriam ser irrelevantes para o cálculo de autoria. Além disso, P9 citou que as medidas NLOC e Quantidade de *Commits*, utilizadas para o cálculo de autoria, não abrangem aspectos de qualidade de código, podendo levar a compreensões equivocadas.

Ainda na Tarefa 2, P12 considerou que não houve interferência do uso do *Developer Tracker App* em sua compreensão, visto que possui convicção clara das pessoas que concentram o conhecimento, independente do resultado confirmativo apresentado. P3, P5, P7, P10, P11, P14 e P15 relataram efeito de confirmação. Entretanto, P15 alertou que alguns setores do código contendo arquivos binários e de testes automatizados não eram tão

importantes para ele, mas influenciaram o cálculo do *Truck Factor* e, por isso, sua opinião poderia ser diferente, dependendo dos resultados. Alguns participantes aprimoraram suas compreensões e o principal motivo foi visualizar desenvolvedores que não fazem mais parte do time, mas possuem alto grau de autoria sobre código, indicando que suas implementações ainda seguem intactas. Esse resultado está diretamente ligado à atividade "Compreensão e controle da distribuição do conhecimento", citada por 9 participantes, como apoiada pelas visualizações de *Developer Tracker App*.

> Sobre a Tarefa 2, na opinião dos participantes, o *Developer Tracker App* apoiou a compreensão quando mostrou a relevância de desenvolvedores que não fazem mais parte do time e confirmando a relevância dos desenvolvedores atuais. Algumas preocupações surgiram em relação ao escopo de arquivos considerados no cálculo do *Truck Factor* e no uso de NLOC e Quantidade de *Commits* para o cálculo de autoria. Em geral, as visualizações poderiam ajudar os participantes a realizar atividades relacionadas à distribuição do conhecimento, mas é preciso ter atenção às características do projeto para evitar decisões equivocadas.

A Tarefa 3 e a Tarefa 4 estão fortemente relacionadas. Ambas abordam aspectos inerentes à demanda de trabalho em setores do código e o quanto os desenvolvedores atuam nesses setores. Os desafios relacionados a essas tarefas inspiraram as definições da técnica visual incluída na *Developer Tracker*. Essas tarefas abordam aspectos como a especialidade técnica dos desenvolvedores ("Identificação das habilidades e perfil do desenvolvedor"), o domínio sobre código, especialmente quando considerados determinados setores e tipos de demandas específicos ("Compreensão e controle da distribuição do conhecimento" e "Estimativa de custos e prazos de projetos e identificação de anomalias no desempenho do desenvolvedor"), bem como abordam aspectos relacionados à performance e às entregas do desenvolvedor ("Reajustes de remuneração"). Essas tarefas tiveram o maior índice de não conclusão sem o uso do *Developer Tracker App*, visto que três participantes não conseguiram concluir cada uma delas. A suspeita é que essas tarefas são mais complexas, por envolverem diversos aspectos sobre o trabalho dos desenvolvedores.

Considerando a Tarefa 3, P5, P8, P9 e P14 não notaram interferência em sua compreensão. Para P5, P9 e P14, há uma visão bem clara dos setores que mais demandam esforço do time. Para P8, há dificuldade de determinar quais setores do código são mais críticos. Tal participante não concluiu a Tarefa 3 sem o uso do *Developer Tracker App* e não se convenceu com as visualizações e as medidas baseadas em NLOC e Quantidade de *Commits* para determinar a demanda de esforço nos setores do código. Outro caso marcante é referente ao participante P7, pois considerou que o uso do *Developer Tracker App* piorou sua

compreensão. Ele citou uma característica da técnica visual como determinante para esse efeito: à primeira vista, as várias conexões que chegam a um artefato (ou a um setor do código) fornecem a impressão de ter sido demandado mais esforço no artefato. Porém, a interpretação é a existência de vários desenvolvedores trabalhando colaborativamente no artefato. Para determinar o esforço, seria preciso analisar os rótulos das conexões, que contém os valores de NLOC e Quantidade de *Commits*. Assim, P7 considerou que esse esforço cognitivo gera confusão e atrapalha sua compreensão. Para solucionar esse problema, P7 e P5 fizeram sugestão semelhantes: substituir o rótulo por uma conexão caracterizada de acordo com a grandeza das medidas, podendo variar na espessura e/ou na coloração. Os participantes que relataram aprimoramento de sua compreensão, justificam que a visualização das conexões entre desenvolvedores e artefatos em diversos níveis ajuda a compreender o quanto tem sido demandado trabalho nesses artefatos.

> Sobre a Tarefa 3, na opinião dos participantes, o *Developer Tracker App* apoiou a compreensão quando permitiu a navegação em diversos níveis de artefatos. Mas, a visualização das conexões em artefatos pode gerar confusão no primeiro momento, visto que muitas conexões em um artefato/setor do código não significam maior esforço, mas maior colaboração. Em geral, as visualizações poderiam ajudar os participantes a realizar diversas atividades relacionadas à compreensão do trabalho dos desenvolvedores, mas é preciso ter atenção aos rótulos das conexões para identificação mais correta do esforço demandado.

Considerando a Tarefa 4, 10 participantes (62%) relataram o efeito de aprimoramento. Isso deve-se à navegação nos artefatos em diversos níveis, permitindo visualizar, por exemplo, desenvolvedores da especialidade técnica *X* auxiliando no desenvolvimento da especialidade *Y*. Outro exemplo, é a visualização de artefatos onde muitas pessoas estão trabalhando e outros artefatos onde somente uma pessoa está trabalhando (pode indicar concentração "local" de conhecimento e o nível de colaboração no artefato). Contudo, P3 considerou que o uso do *Developer Tracker App* piorou sua compreensão, pois ele demorou "*para ver o que já sabia*". Ele tinha uma visão confiável da atuação dos desenvolvedores nos setores do código, mas precisou navegar em diversos níveis da visualização para confirmar. Então, no caso de P3, não foi uma informação falha que o atrapalhou, mas a dificuldade de obter uma informação que possuía.

Sobre a Tarefa 4, na opinião dos participantes, o *Developer Tracker App* apoiou a compreensão quando permitiu visualizar a atuação dos desenvolvedores em diversos setores do código. Mas, é necessário navegar em cada nível da hierarquia de diretórios para visualizar as conexões. O tempo de navegação pode ser considerado excessivo para Gerentes de Projetos que têm compreensão consolidada sobre a atuação dos desenvolvedores. Em geral, as visualizações ajudaram os participantes a identificar os locais do código onde os desenvolvedores têm atuado e, portanto, poderiam auxiliar em decisões como a delegação de tarefas para os membros do time.

Na Tarefa 5, ao comparar duas versões, 10 participantes (62%) mostraram satisfação em aprimorar sua compreensão (P4, P6, P7, P8, P10, P12, P13, P14, P15, P16). Essa foi a tarefa que mais ocorreu o efeito aprimoramento. Além disso, nenhum dos participantes percebeu o efeito de piora. Em geral, os participantes conseguiram visualizar que determinado desenvolvedor "*tornou-se importante*" (P4, P11, P14) e "*quem saiu e quem entrou*" (P4, P14) no time. Alguns participantes identificaram a evolução de determinado setor (P6, P15, P16) e notaram determinado desenvolvedor "*assumir a dianteira*" (P4) em uma tecnologia no intervalo entre duas versões. A não interferência relatada por P5 é relacionada a sua convicção consolidada sobre o aspecto abordado na tarefa. Além disso, P5 vê um problema relacionado ao uso das medidas NLOC e Quantidade de *Commits*, pois nem sempre essas medidas são sinônimo de complexidade e qualidade. Contudo, P5 afirmou que pode ser importante utilizar a visualização comparativa do *Developer Tracker App* quando o objetivo for verificar os impactos de alguma ação de distribuição de conhecimento sobre o código, por exemplo.

Sobre a Tarefa 5, na opinião dos participantes, o *Developer Tracker App* apoiou a compreensão de alterações na atuação dos desenvolvedores entre duas versões. Além disso, as visualizações ajudaram a verificar impactos de ações tomadas pelo Gerente de Projetos em relação a atuação do time. Mas, é necessário atentar às imperfeições das medidas NLOC e Quantidade de *Commits*, por não incluírem aspectos de qualidade e complexidade.

No final da avaliação, os participantes classificaram o grau de utilidade do *Developer Tracker App*. No geral, eles o consideraram útil, mas não essencial. A confiança no conhecimento empírico foi notada nos 16 participantes. Ainda assim, alguns participantes o consideraram essencial, pois pode aliar o conhecimento empírico aos dados quantitativos. A combinação de informações subjetivas e quantitativas é uma prática considerada eficiente para compreender o trabalho dos desenvolvedores (FERREIRA *et al.*, 2020; FEINER; ANDREWS, 2018). P14 falou a respeito da aceitação do uso do *Developer Tracker App* no

contexto de sua empresa, visto que ele deve seguir diversas diretrizes e tomar decisões de forma muito rápida, dificultando a implantação do *Developer Tracker App*.

Concluindo, o *Developer Tracker App* foi considerado útil para apoiar os participantes em diversas atividades relacionadas à compreensão do trabalho desenvolvedores. Contudo, para sua implantação na indústria de software podem ser realizadas melhorias na aplicação das medidas e na geração das visualizações, bem como pode ser necessária uma adaptação do processo de trabalho de Gerentes de Projetos em determinadas empresas.

7.5 Ameaças à Validade

É comum a existência de fatores que impactam ou limitam a validade dos resultados de estudos. Tais fatores são conhecidos como ameaças à validade e podem ser classificadas quanto à (WOHLIN *et al.*, 2012):

a) **Validade interna**. Consiste em ameaças que prejudicam a definição do grau em que os resultados do estudo refletem a realidade observada (relação de causa e efeito). Neste estudo, as possíveis ameaças à validade interna são:

   - Durante a entrevista, há a possibilidade de os participantes sofrerem influência de sentimentos pessoais. Por exemplo, sentimento de competição entre a *Developer Tracker* e os métodos que eles usam para obter informações sobre o trabalho dos desenvolvedores e o projeto de software. O sentimento de competição poderia influenciar nas respostas de duas formas: i) Privilegiar a forma utilizada para obter informações (o participante pode resistir em reconhecer aspectos positivos da *Developer Tracker* em determinadas questões da entrevista); e ii) Privilegiar a *Developer Tracker* (o participante pode ignorar possíveis impactos negativos provocados pelo uso de *Developer Tracker*). Para contornar essa ameaça, para cada opinião emitida durante a entrevista, os participantes foram incentivados a fornecer justificativas. Ao justificar sua opinião, os participantes puderam esclarecer seus pensamentos, proporcionando opiniões mais convictas e com menos influências sentimentais (como a competição);
   - As perguntas definidas no roteiro de entrevista são pontuais, ou seja, referem-se a algumas situações específicas relacionadas ao trabalho dos desenvolvedores e ao projeto. Essas questões foram definidas dessa forma com objetivo de reduzir o esforço dos participantes na execução do estudo. Desse modo, não é possível garantir que seriam obtidos resultados semelhantes ao: i) considerar projetos diferentes, mas gerenciados pelos mesmos participantes; ii) entrevistar

participantes diferentes, mas que gerenciaram os mesmos projetos (concomitantemente ou em momentos diferentes do projeto). Para amenizar essa ameaça, os participantes tiveram espaço para elogiar, criticar e sugerir melhorias à abordagem ao responder e justificar as questões referentes ao Grupo 4 do roteiro de entrevistas;

- Questões mal formuladas podem levar a diferentes interpretações, interferindo nos resultados obtidos no estudo. Neste estudo, apesar de um pesquisador mais experiente ter revisado o roteiro de entrevista e ter sido realizado um estudo preliminar com o objetivo de identificar e efetuar melhorias no roteiro, não é possível garantir que os participantes tiveram a mesma interpretação das questões. Para contornar essa ameaça, o pesquisador que conduziu as entrevistas esteve atento ao entendimento das questões pelos participantes, certificando-se (por meio de questionamentos adicionais) de que o objetivo principal de cada questão ficou claro para o participante;

b) **Validade externa**. Consiste em ameaças que limitam a capacidade de generalização dos resultados em contextos externos ao contexto avaliado no estudo. Neste estudo, as possíveis ameaças à validade externa são:

    - Os participantes da avaliação final podem não representar adequadamente a população de possíveis usuários reais da abordagem, comprometendo a generalização dos resultados. A aplicação do estudo com maior quantidade de profissionais pode amenizar essa ameaça. Na tentativa de amenizar essa ameaça, foram convidados Gerentes de Projetos de empresas diferentes (4 empresas) e com níveis de experiência diferentes. Essas características são referentes aos 16 Gerentes de Projetos participantes para ter amostra diversificada para o estudo;
    - A quantidade limitada de questões definidas no roteiro de entrevista pode não abordar todos os assuntos necessários para avaliação da abordagem. Para contornar essa ameaça, foram definidas questões específicas para avaliar cada uma das cinco tarefas propostas aos participantes. Cabe ressaltar que essas tarefas representarem atividades necessárias para consultar informações sobre o trabalho dos desenvolvedores fornecidas pelas medidas inclusas na abordagem;

c) **Validade de construção**. Consiste em ameaças que afetam a validação do instrumento do estudo, definindo se o tratamento e os resultados refletem, respectivamente, a causa e o

efeito de forma adequada. Neste estudo, uma possível a ameaça à validade de construção é:

- Apesar da tentativa de evitar respostas dadas ao acaso considerando as justificativas das questões no momento da entrevista, não é possível garantir que todos participantes se comprometeram com o estudo e realizaram as atividades da maneira proposta. Para mitigar essa ameaça, durante a entrevista, o pesquisador tentou manter um clima tranquilo e seguro para realização da avaliação, passando as informações para o participante pausadamente e de forma incremental. Além disso, foi realizado um discurso para conscientizar o participante sobre a relevância da pesquisa para Engenharia de Software e para possíveis aplicações à indústria, bem como foi ressaltada a necessidade de opiniões sinceras (positivas ou negativas) para avaliação mais correta possível do objeto de estudo;

d) **Validade de conclusão**. Consiste em ameaças que afetam a habilidade de alcançar conclusões corretas a respeito dos relacionamentos entre o tratamento e o resultado do estudo. Neste estudo, a possível ameaça à validade de conclusão e:

- A ausência de testes estatísticos adequados e o tamanho limitado da amostra podem comprometer a obtenção de conclusões significativas. Para contornar essa ameaça, pode-se repetir o estudo empírico com maior quantidade de pessoas e realizar uma análise quantitativa dos resultados. Na tentativa de amenizar essa ameaça, houve caracterização dos participantes (questões do Grupo 1 do roteiro de entrevistas) para identificação dos perfis dos profissionais. Em caso de identificação de homogeneidade de perfis, poderia ser necessário convidar novos participantes para o estudo. Além disso, em benefício do tamanho da amostra, foram extraídas informações detalhadas sobre a opinião dos participantes, para permitir análise qualitativa mais aprofundada sobre os efeitos do uso da *Developer Tracker*.

7.6 Considerações Finais

Neste Capítulo, foi apresentada a avaliação da abordagem *Developer Tracker*. A avaliação da abordagem foi realizada por meio de um estudo empírico e os resultados foram apresentados e discutidos de forma qualitativa. Foi realizado um “estudo preliminar” para aprimorar o planejamento do estudo e coletar resultados preliminares. No estudo final, 16 participantes realizaram um conjunto de tarefas sem e com o uso do *Developer Tracker App*.

Além disso, os participantes foram entrevistados para coletar efeitos do seu uso na compreensão do trabalho realizado pelos desenvolvedores.

Os resultados mostraram que a abordagem, considerando alguns aspectos relacionados ao trabalho dos desenvolvedores, aprimorou 52% da compreensão dos participantes do estudo. Além disso, ajudou a aumentar as convicções em 33%. No geral, a abordagem foi considerada útil por gerar visualizações a respeito das tecnologias do projeto, da atuação dos desenvolvedores e permitir navegação em diversos níveis da hierarquia de diretórios relacionando com o trabalho de cada desenvolvedor. Entretanto, para implantar o *Developer Tracker App* em empresas, podem ser necessárias melhorias na aplicação das medidas NLOC e Quantidade de *Commits*, na geração das visualizações e na adaptação do processo de trabalho de Gerentes de Projetos em determinadas empresas.

8 TRABALHOS RELACIONADOS

Podem ser encontrados na literatura diversos estudos que abordam formas de apoiar a gestão de projetos utilizando alguma estratégia para compreender o trabalho dos desenvolvedores. Neste capítulo, são apresentados alguns desses estudos, evidenciando o diferencial e a relevância do presente trabalho.

Em um trabalho (GONZÁLEZ-TORRES *et al.*, 2016), os autores realizaram estudos para identificar quais programadores participam no desenvolvimento de partes específicas do código fonte de um sistema de software (projeto). Para tanto, elaboraram uma abordagem para descrever um processo que consiste na extração de informações de SCV, na aplicação de medidas de software e na apresentação visual dos resultados. Essa abordagem é a base para a ferramenta computacional intitulada *Maleku* (*plugin* para o *Eclipse*[21]). A utilização do *Maleku* fornece informações usando técnicas de visualização que incluem quais desenvolvedores realizaram mais modificações no projeto e quais criaram mais arquivos. Além disso, pode-se obter o grau de colaboração entre os desenvolvedores de acordo com os artefatos de código em que trabalharam. *Maleku* foi testada em vários projetos de código aberto em um ambiente de laboratório por alguns usuários experientes. As respostas obtidas foram satisfatórias para apoiar a gestão da equipe do projeto.

Quanto à abordagem, o processo de definição da *Developer Tracker* e da abordagem utilizada para definir a funcionalidade de *Maleku* é semelhante, pois ambas incluem a extração dados de SCV, a aplicação de medidas, a apresentação visual dos resultados e o foco é na visualização da atuação dos desenvolvedores e da evolução dos artefatos do projeto. Contudo, a abordagem de *Maleku* tem como diferencial a visualização de características de implementação referentes ao paradigma de orientação a objetos, como informações de métodos, classes e seus relacionamentos. Por outro lado, *Developer Tracker* tem três diferenciais: i) uso de duas perspectivas (projeto e desenvolvedor); uso de técnica visual exclusiva; e iii) suporte para softwares de qualquer paradigma e/ou linguagem de programação.

Quanto ao apoio computacional/ferramenta, a *Maleku* possui recurso para filtrar, de forma temporal, o estado dos artefatos (filtro de data), e o *Developer Tracker App* possui recurso parecido, permitindo observar versões de um mesmo projeto geradas em épocas diferentes (seleção de versões). Adicionalmente, no *Developer Tracker App*, duas versões são

[21] https://www.eclipse.org/

visualizadas simultaneamente, enquanto na *Maleku* o filtro permite visualizar apenas uma versão. Além disso, a *Maleku* possui um uso limitado por ser um *plugin* para o *Eclipse*. Por outro lado, *Developer Tracker App* é um software *web* e pode ser acessado em um navegador de internet.

Em outro estudo (SHARMA; KAULGUD, 2012), os autores buscaram preencher a lacuna entre o entendimento intuitivo do Gerente de Projetos sobre a equipe e a sua real situação. Para tanto, foi desenvolvida uma ferramenta intitulada *PIVoT*. A sua utilização fornece graficamente relatórios com abrangência holística do projeto. Essa ferramenta foi implementada considerando 30 fatores divididos em seis categorias: i) qualidade do código; ii) qualidade de testes; iii) eficiência de desenvolvimento; iv) rotatividade de código; v) análise da equipe; e vi) gráficos compostos. Dessa forma, tem-se suporte à análise riscos relacionados à rotatividade de desenvolvedores e à gestão da equipe (observando as competências e a qualidade do trabalho de cada desenvolvedor).

As categorias de qualidade de código e qualidade de testes não estão presentes em *Developer Tracker* e consequentemente no apoio computacional *Developer Tracker App*, que se limitam ao uso de medidas quantitativas. No entanto, as medidas de qualidade são específicas para sistemas de software orientados a objetos, provocando limitação na *PIVoT* não presente no *Developer Tracker App*. Outras duas limitações são a *PIVoT* não comparar versões, portanto não são visualizadas informações evolutivas do sistema de software, e não fornece visualização em múltiplos níveis de granularidade dos artefatos. Ambas limitações não estão presentes *Developer Tracker App*.

Em outro trabalho (GAO; LIU, 2015), os autores seguiram a linha de analisar o sistema de software de modo a identificar a contribuição do desenvolvedor analisando os diferentes trechos de código modificados e em qual período foi modificado. Eles desenvolveram a ferramenta *web TeamWatch*, que permite minerar informações de repositórios de código fonte e exibi-las aplicando avançadas técnicas de visualização. Essa ferramenta oferece uma visão geral do sistema de software, o quanto cada artefato é mais modificado e os desenvolvedores que trabalharam em cada artefato. Apesar disso, *TeamWatch* não tem medidas que forneçam informações sobre a versão do sistema de software nem considera aspectos evolutivos, configurando-se como diferenciais de *Developer Tracker App* em relação à *TeamWatch*. Outro diferencial é *TeamWatch* incluir uma técnica visual específica que auxilia o usuário a identificar padrões em grandes conjuntos de dados. No entanto, é importante ressaltar que *Developer Tracker App* possui uma técnica visual definida em *Developer Tracker* que permite a navegação pela árvore de diretórios do projeto.

Em um trabalho encontrado na literatura (FEINER; ANDREWS, 2018), os autores estudaram a dificuldade que Gerentes de Projetos e desenvolvedores têm em manter um *overview* da estrutura, evolução e *status* dos sistemas de software. Eles destacaram que os repositórios de código armazenam informações importantes para o tema em questão, porém a exibição dessas informações consiste em estatísticas resumidas ou representações visuais simples de gráficos. Nesse contexto, a ideia dos autores foi fornecer uma visualização abrangente desse sistema para analisar a sua evolução e a do trabalho realizado no código fonte. Para tanto, foi apresentada a ferramenta *RepoVis*, um sistema de software para *web* para minerar informações do Git e repositórios de *bugs*, apresentando-as utilizando técnicas de visualização avançadas. Foram implementados filtros de granularidade, permitindo navegar entre partes do sistema de software, como pacotes, arquivos e linhas de código. Apesar da exibição de uma visão abrangente da evolução do projeto, há poucos recursos para analisar o trabalho de desenvolvedores individualmente (recurso presente em *Developer Tracker App*). Isso ocorre porque as técnicas de visualização presentes em *RepoVis* são focadas em retratar a evolução das linhas de código e são limitadas para representar a contribuição de cada desenvolvedor da equipe. Um outro diferencial entre *RepoVis* e *Developer Tracker App* é o primeiro possuir fonte de informação adicional, os repositórios de *bugs*.

Alguns autores estudaram meios para apoiar o gerenciamento de equipe no que diz respeito ao recrutamento de desenvolvedores (JARUCHOTRATTANASAKUL *et al.*, 2016). Eles destacaram a importância de considerar a experiência prática e a *expertise* de cada candidato e que o currículo tradicional contém poucas evidências sobre suas atividades de desenvolvimento reais executas por um desenvolvedor. Para auxiliar o Gerente de Projetos a avaliar melhor um candidato para um projeto, é proposta uma abordagem para extrair atividades práticas dos desenvolvedores de seus sistemas de software de fonte aberta (OSS - *Open Source Software*) e gerar as biografias que refletem suas contribuições para o OSS. Foi implementada uma ferramenta computacional para a abordagem. Consiste em um sistema de software para *web* que captura informações de linguagens de programação dominadas pelos desenvolvedores a partir de seus trabalhos em projetos no *Github*. Com a utilização dessa ferramenta, são exibidas informações considerando a quantidade de linhas de código e as épocas em que o desenvolvedor trabalhou com determinada linguagem de programação. A principal diferença entre essa abordagem e *Developer Tracker* é o escopo. Enquanto a primeira tem recursos focados em identificar o domínio de tecnologias de cada desenvolvedor, a segunda concentra-se em identificar mais fatores além do domínio de

tecnologias, como o grau de importância a quantidade de contribuição, bem como aspectos evolutivos de versões do projeto.

Os trabalhos existentes apresentam soluções para obter diversas informações para os Gerentes de Projetos. Apesar disso, possuem algumas diferenças em relação à abordagem *Developer Tracker* e ao apoio computacional *Developer Tracker App*. Os principais diferencias da abordagem proposta neste trabalho são: i) visualização de um conjunto de informações que inclui aspectos do projeto e de cada desenvolvedor individualmente; ii) comparação entre duas versões de um sistema de software, permitindo observar características evolutivas; e iii) utilização de uma técnica visual exclusiva, fornecendo perspectivas diferentes e com múltiplos níveis de granularidade ao navegar em artefatos e relacioná-los a cada desenvolvedor.

9 CONSIDERAÇÕES FINAIS

A utilização de práticas de gestão de projetos é motivada pela complexidade inerente ao desenvolvimento de software. O Gerente de Projetos é o responsável por executar as práticas de gestão e uma de suas principais atividades é gerenciar o time do projeto. Essa atividade envolve aspectos como montagem de equipes, reconhecimento pelo trabalho realizado por desenvolvedores e distribuição do conhecimento no projeto. Por isso, informações sobre o trabalho dos desenvolvedores podem ser valiosas para os Gerentes de Projetos. Apesar de existirem estratégias para obter essas informações e apresentá-las de forma visual, ainda existem lacunas a serem exploradas, como considerar a evolução do software ao visualizar o quanto os desenvolvedores trabalharam, bem como fornecer a visualização de diferentes perspectivas, incluindo informações do projeto e do trabalho individual dos desenvolvedores em múltiplos níveis de granularidade.

Neste trabalho, o objetivo foi apoiar Gerentes de Projetos a compreender do trabalho dos desenvolvedores utilizando uma abordagem para visualização de medidas quantitativas aplicadas sobre informações mineradas de SCV. Para tanto, foi elaborada a abordagem *Developer Tracker*, composta por uma técnica de visualização de informações.

9.1 Conclusões

A abordagem *Developer Tracker* foi avaliada por meio de um estudo empírico e os resultados foram apresentados e discutidos de forma qualitativa. No estudo, 16 participantes realizaram cinco tarefas sem e com o uso do apoio computacional *Developer Tracker App* que implementa a abordagem *Developer Tracker*. Além da execução de cinco tarefas, ocorreram entrevistados com o objetivo de caracterizar os participantes e coletar os efeitos do uso da abordagem na compreensão desses participantes sobre o trabalho realizado pelos desenvolvedores. Os participantes são Gerentes de Projetos que atuam na indústria de software.

Os efeitos de uso do *Developer Tracker App* podem aprimorar, confirmar, não interferir ou piorar a compreensão sobre o trabalho dos desenvolvedores. Os resultados obtidos mostraram que a *Developer Tracker App*, considerando aspectos relacionados ao trabalho dos desenvolvedores, aprimorou 52% da compreensão dos participantes do estudo. Além disso, ajudou a confirmar em 33% suas percepções. Em resumo, o *Developer Tracker App* foi considerado útil por gerar visualizações a respeito das tecnologias demandas no projeto, da atuação dos desenvolvedores e por permitir a navegação na hierarquia de

diretórios e a visualização das relações de trabalho de cada desenvolvedor. Contudo, para implantar o *Developer Tracker App* em empresas, podem ser necessárias mudanças no uso das medidas NLOC e Quantidade de *Commits*, bem como na geração das visualizações. Além disso, pode ser interessante adaptar o processo de trabalho de Gerentes de Projetos em determinadas empresas.

9.2 Contribuições

As contribuições deste trabalho são:

a) Mapeamento Sistemático da Literatura (MSL) para identificar estratégias para compreender o trabalho dos desenvolvedores;
b) As medidas encontradas no MSL foram categorizadas e organizadas em seis grupos: i) Comportamento (CoP); ii) Contribuição (Con); iii) Grau de Importância (ImP); iv) Produtividade (Pro); v) Qualidade (Qua); e vi) Trabalho Colaborativo (TrC);
c) As atividades encontradas no MSL para apoiar a gestão de projetos foram organizadas em sete grupos: i) Identificação das habilidades e perfil do desenvolvedor; ii) Estimar custos e prazos de projetos e identificar anomalias no desempenho do desenvolvedor; iii) Melhoria no desempenho das equipes; iv) Compreensão e controle da distribuição do conhecimento; v) Identificação de necessidade de investimento (treinamento ou equipamentos); vi) Planejamento de melhoria da qualidade de código; e vii) Reajustes de remuneração;
d) Definição da abordagem *Developer Tracker*, que tem o propósito obter informações armazenadas em SCV, aplicar medidas quantitativas e fornecer resultados visualmente para apoiar a compreensão do trabalho dos desenvolvedores;
e) Definição de uma técnica de visualização baseada no paradigma iconográfico. Essa técnica representa visualmente e interativamente as informações coletadas de versões de sistemas de software armazenadas em SCV;
f) Desenvolvimento do *Developer Tracker*, um software *web*, que automatiza a execução da abordagem proposta;
g) Avaliação qualitativa da abordagem proposta no contexto da indústria de software.

9.3 Limitações

As principais limitações da abordagem *Developer Tracker* são:

a) A abordagem permite extrair informações de um repositório de código armazenado em algum SCV. Porém, um projeto pode possuir diversos repositórios de código. Atualmente,

para analisar diversos repositórios de código é necessário instanciar a abordagem uma vez para cada repositório e as informações são visualizadas separadamente em cada instância;

b) A abordagem não permite modificar o escopo de artefatos considerados nas medidas, pois todos os artefatos monitorados pelo SCV são considerados nos cálculos das medidas. Ou seja, não é possível remover dos cálculos determinados setores do código, como testes automatizados e arquivos binários;
c) As medidas NLOC e Quantidade de *Commits* são uma das medidas mais utilizadas para compreender o trabalho dos desenvolvedores. Contudo é interessante adicionar determinados componentes a essas medidas para que elas contemplem informações de qualidade e complexidade da implementação. Atualmente, *Developer Tracker* não possui suporte para visualizar qualidade e complexidade.

9.4 Perspectivas Futuras

Algumas sugestões de trabalhos futuros que podem contribuir na evolução, no refinamento e na avaliação da abordagem *Developer Tracker*:

a) Permitir visualizar mais de um repositório para uma instância da abordagem;
b) Permitir modificar o escopo de artefatos considerados no cálculo das medidas;
c) Inserir suporte à visualização de qualidade e complexidade ao aplicar medidas NLOC e Quantidade de *Commits*;
d) Inserir formas de customizar o uso da abordagem para promover melhor aderência à processos de compreensão do trabalho dos desenvolvedores em empresas da indústria de software;
e) Evoluir `Developer Tracker App` para incluir a seleção de mais de um repositório de código, permitindo visualizar informações em projetos que possuem diversos repositórios de código. Outra possível evolução é permitir a seleção de projetos em tipos diferentes de repositórios (*e.g.* repositórios de tarefas, bugs e outros tipos de SCV), permitindo maior aceitação na indústria. Por fim, há necessidade de realizar avaliação da acurácia das recomendações, bem como de avaliar a geração de alertas para valores de medidas com alto potencial de risco ao projeto.

9.5 Publicações

Trabalhos publicados ao longo desta pesquisa de mestrado:

a) Artigo “Measuring Developer Work to Support the Software Project Manager: An Exploratory Study” referente ao MSL no Simpósio Brasileiro de Qualidade de Software (SBQS 2019). O artigo foi premiado como segundo melhor trabalho do evento;
b) Artigo “Uma Abordagem para Apoiar o Gerente de Projetos de Software a Quantificar o Trabalho dos Desenvolvedores” referente a proposta de pesquisa no Workshop de Teses e Dissertações em Qualidade de Software (WTDQS 2019);
c) Artigo “How is the Work of Developers Measured? An Industrial and Academic Exploratory View” estendido do artigo premiado no Journal of Software Engineering Research and Development (JSERD) em 2020;
d) Artigo “Developer Tracker App: Uma Ferramenta para Visualizar o Trabalho dos Desenvolvedores” na Trilha de Ferramentas do Simpósio Brasileiro de Engenharia de Software (SBES 2021).

REFERÊNCIAS

ANDERSON, C. Docker [software engineering]. **Ieee Software**, v. 32, n. 3, p. 102-c3, 2015.

APACHE. **ECharts**. 2020. Disponível em: https://echarts.apache.org/en/index.html. Acesso em: 22 março de 2021.

AVELINO, G. **V1.1 ICPC release**. 2016. Disponível em: https://github.com/aserg-ufmg/Truck-Factor/releases/tag/v1.1. Acesso em: 22 março de 2021.

BASTOS, C. **Uma Abordagem para Visualização da Evolução de Código Morto em Sistemas de Software Orientados a Objetos**. 86 p. Dissertação (mestrado) - Universidade Federal de Lavras, 2016.

BOEHM, B. W. Software risk management: principles and practices. **IEEE software**, IEEE, v. 8, n. 1, p. 32-41, 1991.

CARD, M. **Readings in information visualization: using vision to think**. [*S. l.*]: Morgan Kaufmann, 1999.

CARNEIRO, Glauco de Figueiredo. **SourceMiner: Um Ambiente Integrado para Visualização Multi-Perspectiva de Software**. 2011. Tese (Doutorado) – Universidade Federal da Bahia, [S. l.], 2011.

COUTO, J. M. C. **Técnicas de visualização de dados em gerenciamento de projetos de desenvolvimento de software: proposta de extensão do PMBOK**. 100 p. Dissertação (mestrado) - Escola Politécnica da Pontifícia Universidade Católica do Rio Grande do Sul, 2018.

DANIAL, A. **CLOC**. 2017. Disponível em: https://github.com/AlDanial/cloc. Acesso em: 22 março de 2021.

DE BASSI, P. R. *et al.* Measuring Developers' Contribution in Source Code using Quality Metrics. **2018 IEEE 22nd International Conference on Computer Supported Cooperative**

Work in Design (CSCWD), IEEE, p. 39-44, 2018. Disponível em: https://doi.org/10.1109/CSCWD.2018.8465320.

DE QUEIROZ, F. B.; FERREIRA, F. S.; DA SILVA, L. S. **Análise de usabilidade dos Sistemas de Controle de Versão Subversion e Git**. [*S. l.: s. n.*], 2015.

DI LUCCA, G. A.; DI PENTA, M. Experimental settings in program comprehension: Challenges and open issues. **International Conference on Program Comprehension,** IEEE, p. 229-234, 2006.

DIEHL, S. **Software visualization: visualizing the structure, behaviour, and evolution of software**. [*S. l.*]: Springer Science & Business Media, 2007. 187 p.

FEINER, J.; ANDREWS, K. RepoVis: Visual Overviews and Full-Text Search in Software Repositories. **2018 IEEE Working Conference on Software Visualization (VISSOFT)**, IEEE, p. 1-11, 2018. Disponível em: https://doi.org/10.1109/VISSOFT.2018.00009.

FERREIRA, M. *et al.* A Comparative Study of Algorithms for Estimating Truck Factor. **2016 X Brazilian Symposium on Software Components, Architectures and Reuse (SBCARS)**, p. 91-100, 2016. Disponível em: https://doi.org/10.1109/SBCARS.2016.20.

FERREIRA, M. S. *et al.* How is the Work of Developers Measured? An Industrial and Academic Exploratory View. **Journal of Software Engineering Research and Development**, v. 8, p. 1-8, 2020.

FERREIRA, M. S. *et al.* Measuring Developer Work to Support the Software Project Manager: An Exploratory Study. **Proceedings of the XVIII Brazilian Symposium on Software Quality,** p. 79-88, 2019.

FERREIRA, M.; VALENTE, M. T.; FERREIRA, K. A Comparison of Three Algorithms for Computing Truck Factors. **2017 IEEE/ACM 25th International Conference on Program Comprehension (ICPC)**, p. 207-217, 2017. Disponível em: https://doi.org/10.1109/ICPC.2017.35.

FLANAGAN, D. **JavaScript: o guia definitivo**. [*S. l.*]: Bookman Editora, 2004.

GANDOMANI, T. J. *et al.* The role of project manager in agile software teams: A systematic literature review. **IEEE Access**, v. 8, p. 117109-117121, 2020.

GAO, M.; LIU, C. TeamWATCH Demonstration: A Web-based 3D Software Source Code Visualization for Education. **Proceedings of the 1st International Workshop on Code Hunt Workshop on Educational Software Engineering**, New York, NY, USA, p. 12-15, 2015. Disponível em: https://doi.org/10.1145/2792404.2792408.

GARCIA, R. E. **VIDAese: processo de visualização exploratória para apoio a estudos empíricos em verificação, validação e teste de software**. 163 p. 2006. Tese (Doutorado) - Universidade de São Paulo, 2006.

GITHUB. **Diretrizes da comunidade do GitHub**. 2020. Disponível em: https://help.github.com/pt/github/site-policy/github-community-guidelines. Acesso em: 22 mar. 2021.

GONZÁLEZ-TORRES, A. *et al.* Knowledge discovery in software teams by means of evolutionary visual software analytics. **Science of Computer Programming**, v. 121, p. 55-74, 2016. Disponível em: https://doi.org/10.1016/j.scico.2015.09.005.

JARUCHOTRATTANASAKUL, T. *et al.* Open Source Resume (OSR): A Visualization Tool for Presenting OSS Biographies of Developers. **2016 7th International Workshop on Empirical Software Engineering in Practice (IWESEP)**, p. 57-62, 2016. Disponível em: https://doi.org/10.1109/IWESEP.2016.17.

JUNG, C. F. Metodologia aplicada a projetos de pesquisa: Sistemas de Informação & Ciência da Computação. **Proposta de TCC e Projeto de Pesquisa**, 2009.

JUNIOR, S. *et al.* Gestão de valor em projetos de TI: um estudo sobre organizações no Brasil. **Gestão & Produção**, v. 26, n. 2, 2019.

KEIM, D. A. Designing pixel-oriented visualization techniques: Theory and applications. **IEEE Transactions on visualization and computer graphics**, v. 6, n. 1, p. 59-78, 2000.

KEIM, D. A. Information visualization and visual data mining. **IEEE transactions on Visualization and Computer Graphics**, v. 8, n. 1, p. 1-8, 2002.

KEIM, D. A.; KRIEGEL, H.-P. Visualization techniques for mining large databases: A comparison. **IEEE Transactions on knowledge and data engineering**, v. 8, n. 6, p. 923-938, 1996.

KERZNER, H. **Gestão de Projetos: As Melhores Práticas**. 3. ed. [*S. l.*]: Bookman Editora, 2016. 726p.

KITCHENHAM, B. Procedures for performing systematic reviews. **Keele, UK, Keele University**, v. 33, n. 2004, p. 1-26, 2004.

KITCHENHAM, B.; CHARTERS, S. Guidelines for performing systematic literature reviews in software engineering. 2007.

LIMA, J. *et al.* Assessing developer contribution with repository mining-based metrics. **2015 IEEE International Conference on Software Maintenance and Evolution (ICSME)**, p. 536-540, 2015. Disponível em: https://doi.org/10.1109/ICSM.2015.7332509.

LIMA, L. G.; PETRUCELLI, E. E.; DO ESPÍRITO SANTO, F. Visão geral sobre o gerenciamento de estado no Vue.js com a biblioteca vuex. **Revista Interface Tecnológica**, v. 16, n. 1, p. 56-66, 2019.

LIU, G.; YOKOYAMA, S. Proposal for a Quantitative Skill Risk Evaluation Method Using Fault Tree Analysis. **IEEE Transactions on Engineering Management**, v. 62, n. 2, p. 266-279, 2015. Disponível em: https://doi.org/10.1109/TEM.2015.2413357.

LUCKOW, D. H.; DE MELO, A. A. **Programação Java para a WEB**. [*S. l.*]: Novatec Editora, 2010.

MAJUMDAR, R. *et al.* Source code management using version control system. **2017 6th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions)(ICRITO)**. [*S. l.: s. n.*], 2017. p. 278-281.

MALIPENSE, L. M.; ZUCHI, J. D. Um estudo sobre o conceito e a aplicação da arquitetura de microserviços. **Revista Interface Tecnológica**, v. 15, n. 1, p. 122-134, 2018.

MARSHALL, B. *et al.* Does Sample Size Matter in Qualitative Research?: A Review of Qualitative Interviews in is Research. **Journal of Computer Information Systems**, v. 54, n. 1, p. 11-22, 2013. Disponível em: https://doi.org/10.1080/08874417.2013.11645667.

MOURA, M. H. D. de; NASCIMENTO, H. A. D. do; ROSA, T. C. Extracting New Metrics from Version Control System for the Comparison of Software Developers. **2014 Brazilian Symposium on Software Engineering**, p. 41-50, 2014. Disponível em: https://doi.org/10.1109/SBES.2014.25.

OLIVEIRA, L. M. de. MODELO DE GERENCIAMENTO ÁGIL DE PROJETOS UTILIZANDO A METODOLOGIA KANBAN: aplicação em uma empresa de software. 2020.

PMI. **Guide to the Project Management Body of Knowledge (PMBOK® Guide)**. 6. ed. [*S. l.*]: Project Management Institute, 2017. 726 p.

PRESSMAN, R.; MAXIM, B. **Engenharia de Software: uma abordagem profissional**. 8. ed. [*S. l.*]: McGraw Hill Brasil, 2016. 968 p.

PRESSMAN, R. S. **Arquitetura de Software: desenvolvimento orientado para arquitetura**. [*S. l.*]: McGraw-Hill, 2002.

RICCA, F.; MARCHETTO, A. Are Heroes Common in FLOSS Projects? **Proceedings of the 2010 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement**, New York, NY, USA, ESEM '10, p. 55:1--55:4, 2010. Disponível em: https://doi.org/10.1145/1852786.1852856.

SCHWIND, M.; WEGMANN, C. SVNNAT: Measuring Collaboration in Software Development Networks. **2008 10th IEEE Conference on E-Commerce Technology and the Fifth IEEE Conference on Enterprise Computing, E-Commerce and E-Services**, p. 97-104, 2008. Disponível em: https://doi.org/10.1109/CECandEEE.2008.100.

SHARMA, V. S.; KAULGUD, V. PIVoT: Project insights and Visualization Toolkit. **2012 3rd International Workshop on Emerging Trends in Software Metrics (WETSoM)**, p. 63-69, 2012. Disponível em: https://doi.org/10.1109/WETSoM.2012.6226995.

SHASTRI, Y.; HODA, R.; AMOR, R. The role of the project manager in agile software development projects. **Journal of Systems and Software**, p. 110871, 2020.

SILVA, M. S. **Construindo sites com CSS e (X) HTML: sites controlados por folhas de estilo em cascata**. [*S. l.*]: Novatec Editora, 2007.

SINGH, R.; LANO, K. Defining and formalising project management models and processes. **Science and Information Conference**, p. 720-731, 2014.

SOMMERVILLE, I. **Engenharia de Software**. 10. ed. [*S. l.*]: Pearson Universidades, 2019. 756 p.

SPENCE, R. **Information Visualization: Design for Interaction**. 2. ed. [*S. l.*]: ACM Press, Inc., Harlow, 2007. 120p.

SVIOKLA, J. Swimming in data? Three benefits of visualization. **HBR Blog Network, Harvard Business Review**, 2009.

WIERINGA N.; MAIDEN N. M. R.; ROLLAND, C. Requirements Engineering Paper Classification and Evaluation Criteria: A Proposal and a Discussion. **Requirements engineering**, v. 11, n. 1, p. 102-107, 2006.

VON MAYRHAUSER, A.; VANS, A. M. Program comprehension during software maintenance and evolution. **Computer**, v. 28, n. 8, p. 44-55, 1995.

WOHLIN, C. *et al*. **Experimentation in software engineering**. 1. ed. [*S. l.*]: Springer Science & Business Media, 2012. 236 p.

WU, X. *et al.* A reverse engineering approach to support software maintenance: version control knowledge extraction. **11th Working Conference on Reverse Engineering**, p. 90-99, 2004. Disponível em: https://doi.org/10.1109/WCRE.2004.8.

XIE, X.; POSHYVANYK, D.; MARCUS, A. Visualization of CVS Repository Information. **2006 13th Working Conference on Reverse Engineering**, p. 231-242, 2006. Disponível em: https://doi.org/10.1109/WCRE.2006.55.

YANG, H.; GRAHAM, H. Software Metrics and Visualisation. **Univ. of Auckland, Tech. Rep**, Citeseer, 2003.

ZOLKIFLI, N. N.; NGAH, A.; DERAMAN, A. Version Control System: A Review. **Procedia Computer Science**, v. 135, p. 408-415, 2018.

ZUSER, W.; GRECHENIG, T. Reflecting skills and personality internally as means for team performance improvement. **Proceedings 16th Conference on Software Engineering Education and Training, 2003. (CSEE&T 2003)**, p. 234-241, 2003. Disponível em: https://doi.org/10.1109/CSEE.2003.1191382.

ANEXO A

Tabela A.1 – Artigos selecionados no MSL.

#	Título	Autores	Ano	Base
A1	Reflecting skills and personality internally as means for team performance improvement	Zuser, W Grechenig, T	2003	IEEE
A2	Continuous productivity assessment and effort prediction based on Bayesian analysis	Yun, S. Simmons D.	2004	Ei-Compendex
A3	Visualization of CVS Repository Information	Xie, X Poshyvanyk, D Marcus, A	2006	IEEE
A4	A Visualization for Software Project Awareness and Evolution	Ripley, R. Sarma, A. van der Hoek, A.	2007	IEEE
A5	A 3-Dimensional Relevance Model for Collaborative Software Engineering	Omoronyia, I. Ferguson, J. Roper, M. Wood, M.	2007	IEEE
A6	Evaluating software project portfolio risks	Costa, H. Barros, M.	2007	Ei-Compendex
A7	SVNNAT: Measuring Collaboration in Software Development Networks	Travassos, G. Schwind, M. Wegmann, C.	2008	IEEE
A8	Mining Individual Performance Indicators in Collaborative Development Using Software Repositories	Zhang, S. Wang, Y. Xiao, J.	2008	IEEE
A9	Development of a project level performance measurement model for improving collaborative design team work	Yin, Y. Qin, S. Holland, R	2008	IEEE
A10	Measuring Developer Contribution from Software Repository Data	Gousios, G. Kalliamvakou, E. Spinellis, D.	2008	ACM
A11	Using transflow to analyze open source developers' evolution	Costa, J Santana Jr., F. De Souza, C	2009	Scopus
A12	Case study: Visual analytics in software product assessments	Telea, A. Voinea, L.	2009	Ei-Compendex
A13	Are Heroes Common in FLOSS Projects?	Ricca, F. Marchetto, A.	2010	ACM
A14	PIVoT: Project insights and Visualization Toolkit	Sharma, V. Kaulgud, V.	2012	IEEE
A15	Towards understanding how developers spend their effort during maintenance activities	Soh, Z. Khomh, F. Guéhéneuc, Y.	2013	IEEE
A16	Extracting, identifying and visualisation of the content, users and authors in software projects	Antoniol, G. Polášek, I. Uhlár, M	2013	Scopus
A17	Effect of personality type on structured tool comprehension performance	Gorla, N. Chiravuri, A. Meso P.	2013	Springer
A18	Estimating Development Effort in Free/Open Source Software Projects by Mining Software Repositories: A Case Study of OpenStack	Robles, G. González-Barahona, J. M Cervigón, C. Capiluppi, A.	2014	ACM
A19	Determining Developers' Expertise and Role: A Graph Hierarchy-Based Approach	Izquierdo-Cortázar, D. Bhattacharya, P. Neamtiu, I. Faloutsos, M.	2014	IEEE
A20	Extracting New Metrics from Version Control System for the Comparison of Software Developers	Moura, M. Nascimento, H. Rosa, T.	2014	IEEE

(continua)

Tabela A.1 – Artigos selecionados no MSL (continuação).

#	Título	Autores	Ano	Base
A21	A machine learning technique for predicting the productivity of practitioners from individually developed software projects	Lopez-Martin, C. Chavoya, A. Meda-Campana, M.	2014	IEEE
A22	Influence of Social and Technical Factors for Evaluating Contribution in GitHub	Tsay, J. Dabbish, L. Herbsleb, J.	2014	ACM
A23	Niche vs. Breadth: Calculating Expertise over Time through a Fine-Grained Analysis	da Silva, J. Clua, E. Murta, L. Sarma, A.	2015	IEEE
A24	Contributor's Performance, Participation Intentions, Its Influencers and Project Performance	Rastogi, A.	2015	IEEE
A25	Proposal for a Quantitative Skill Risk Evaluation Method Using Fault Tree Analysis	Liu, G. Yokoyama, S.	2015	IEEE
A26	Assessing developer contribution with repository mining-based metrics	Lima, J. Treude, C. Filho, F.	2015	IEEE
A27	Identifying wasted effort in the field via developer interaction data	Kulesza, U. Balogh, G. Antal, G. Beszedes, A. Vidacs, L. Gyimothy, L.	2015	IEEE
A28	TeamWATCH Demonstration: A Web-based 3D Software Source Code Visualization for Education	Vegh, T. Zoltan A. Gao, M. Liu, C.	2015	Scopus
A29	Knowledge discovery in software teams by means of evolutionary visual software analytics	González-Torres, A. García-Peñalvo, F. Therón-Sánchez, R.	2016	Scopus
A30	Using Temporal and Semantic Developer-Level Information to Predict Maintenance Activity Profiles	Colomo-Palacios, R. Levin, S. Yehudai, A.	2016	IEEE
A31	Open Source Resume (OSR): A Visualization Tool for Presenting OSS Biographies of Developers	Jaruchotrattanasakul, T. Yang, X. Makihara, E. Fujiwara, K. Iida, H.	2016	IEEE
A32	Software Project Managers' Perceptions of Productivity Factors: Findings from a Qualitative Study	Oliveira, E. Conte, T. Cristo, M.	2016	ACM
A33	A Comparative Study of Algorithms for Estimating Truck Factor	Mendes, E. Ferreira, M. Avelino, G. Valente, M.	2016	IEEE
A34	Quantifying and Mitigating Turnover-Induced Knowledge Loss: Case Studies of Chrome and a project at Avaya	Ferreira, K. Rigby, P. Zhu, Y. Donadelli, S. Mockus, A. Rigb, P. Zhu, Y.	2016	Scopus
A35	Collabcrew — An intelligent tool for dynamic task allocation within a software development team	Donadell, S. Mockus, A. Samath, S. Udalagama, D. Kurukulasooriya, H. Premarathne, D. Thelijjagoda, S.	2017	IEEE
A36	A Comparison of Three Algorithms for Computing Truck Factors	Ferreira, M. Valente, M. Ferreira, K.	2017	IEEE

(continua)

Tabela A.1 – Artigos selecionados no MSL (continuação).

#	Título	Autores	Ano	Base
A37	Revisiting Turnover-Induced Knowledge Loss in Software Projects	Nassif, M. Robillard, M P	2017	Scopus
A38	RepoVis: Visual Overviews and Full-Text Search in Software Repositories	Feiner, J. Andrews, K.	2018	IEEE
A39	Measuring Developers' Contribution in Source Code using Quality Metrics	de Bassi, P. Wanderley, G. Banali, P.	2018	IEEE
A40	git2net - Mining Time-Stamped Co-Editing Networks from Large git Repositories	Paraiso, E. Gote, c. Scholtes, I. Schweitzer, F.	2019	IEEE
A41	Selecting Project Team Members through MBTI Method: An Investigation with Homophily and Behavioural Analysis	Kollipara, P. Regalla, L.; Ghosh, G. Kasturi, N.	2019	IEEE
A42	Knowledge-Oriented Models Based on Developer-Artifact and Developer-Developer Interactions	Lucas, E. M. Oliveira, T. C. Schneider, D.	2020	IEEE
A43	Productivity, Turnover, and Team Stability of Agile Teams in Open-Source Software Projects	Alencar, P. S. C. Scott, E. Charkie, K. N. Pfahl, D.	2020	IEEE

Fonte: Do Autor (2021).

ANEXO B

Tabela B.1 – Medidas encontradas nos artigos selecionados no MSL.

#	Medida	Grupo	Referência
1	Autoria por culpa (blame)	Con	A36
2	CDI	Qua	A14
3	Change code documentation (CAD)	Con	A10, A12
4	Close a bug (BCL)	ImP	A8, A19, A10
5	Close a bug that is then reopened (BCR)	ImP	A10
6	Close a lingering thread (MCT)	Pro	A10
7	Colaboração (arquivos)	TrC, Con	A14, A3, A7, A11, A29, A42
8	Colaboração (interação)	TrC, CoP	A9
9	Colaboration (CodeChurn)	TrC, Con	A20, A40
10	Comment on a bug report (BCR)	CoP	A19, A10
11	Commit binary files (CBF)	Con	A10
12	Commit code that closes a bug (CCB)	Con	A38, A10
13	Commit comment that includes a bug report num (CBN)	Con, ImP	A15, A19, A10
14	Commit documentation/translation files (CDF)	Con	A38, A10
15	Commit fixes to code style (CSF)	Qua, Con	A38, A10
16	Commit more than X files in a single commit (CMF)	Con	A19, A22, A28, A10
17	Commit new source file or directory (CNS)	Con	A10
18	Commit with empty commit comment (CEC)	Con	A10
19	Commits Versatility	Con	A30
20	Complexidade	Qua	A8, A26, A39, A12
21	Comprometimento	ImP, CoP	A9, A32, A17
22	Contribuition start	Con, ImP	A30
23	Contribution Duration	ImP, Con	A30
24	CQI	Qua	A14
25	Custo	Pro	A2
26	DOA	Con	A33, A36
27	Domínio das tecnologias do projeto	ImP, Con	A25, A32
28	Duplicação de código	Qua	A12
29	Entrega de tarefas	Pro	A24, A32, A35, A42, A43
30	Esforço mensal do desenvolvedor	Pro	A18, A2
31	Esforço no commit	Pro, Con	A15
32	Esforço por modificação	Pro, Con	A27, A2
33	Expected Shortfall (ES)	Con, ImP	A37
34	Experiência do desenvolvedor	ImP	A21, A31
35	Expertise Breadth of a Developer (EBD)	Con, ImP	A23
36	Expertise of a Developer (ED)	Con, ImP	A20, A23
37	Fator de Contribuição de código	Con	A10
38	First reply to thread (MRT)	CoP	A10
39	Fragmentação do programador	Con	A14, A22, A24
40	Heroi	Con, ImP	A13
41	Knowledge at Risk (KaR)	Con, ImP	A37
42	knowledge loss	Con, ImP	A34, A37
43	Link a wiki page from documentation/mail file (WLP)	Qua	A10
44	Métricas CK	Qua	A8, A39
45	Métricas de Martin	Qua	A39, A12
46	Métricas de Qmood	Qua	A39
47	MTBC	Pro	A30
48	Quantidade de code churn	Con	A10, A14
49	Participate in a flamewar (MFW)	CoP	A22, A10
50	Percentagem de commits	Con	A19, A36
51	Percentagem de linhas de código	ImP, Con	A4, A5, A14, A36
52	QCTE	Qua	A14
53	Quantidade de commits	Con	A3, A8, A11, A18, A19, A29, A30, A33, A38, A42
54	Quantidade de dias ativo	Con, ImP	A18, A30

(continua)

Tabela B.1 – Medidas encontradas nos artigos selecionados no MSL (continuação).

#	Medida	Grupo	Referência
55	Quantidade de linhas de código (NLOC)	Con, TrC	A16, A19, A3, A8, A14, A20, A21, A22, A26, A38, A10, A40
56	Report a confirmed/invalid bug (BRP)	ImP	A10
57	Retrabalho	Qua	A27, A32
58	Source abandoned	Con, ImP	A34, A37
59	Start a new thread (MST)	Con	A10
60	Start a new wiki page (WSP)	Con	A10
61	Status	ImP	A22
62	Truck Factor	Con, ImP	A13, A33, A36
63	Último commit "leva tudo"	Con	A36
64	Update a wiki page (WUP)	Con	A22
65	Velocity	Pro	A43

Fonte: Do Autor (2021).

ANEXO C

Como parte deste trabalho, foi elaborada uma técnica de visualização exclusiva da abordagem *Developer Tracker*. Seu objetivo é permitir a visualização conjunta das medidas **NLOC, Quantidade de *Commits*, Colaboração (em arquivos) e *Truck Factor***. As três primeiras referem-se à atuação de cada desenvolvedor em cada artefato do projeto, enquanto a última indica se determinado(s) desenvolvedor(es) está(ão) ou não concentrando em si o conhecimento (e a importância) sobre o código, considerando todos os artefatos do projeto. É importante ressaltar que os artefatos estão organizados de forma hierárquica em um conjunto de arquivos e pastas. Respeitando essa hierarquia, os artefatos podem ser classificados em: i) **projeto:** pasta raiz do projeto que contém o conjunto total de arquivos e pastas de uma versão do sistema de software; ii) **diretório:** pasta do projeto que contém arquivos e outras pastas; e iii) **arquivo:** documento do projeto que contém linhas de código ou de texto. Os arquivos são a base da hierarquia e seu conteúdo é o insumo para o cálculo das medidas. Por isso, são calculadas as medidas referentes aos arquivos de forma imediata. Por outro lado, o cálculo das medidas referentes ao projeto e aos diretórios é resultante do somatório das medidas calculadas para os artefatos neles contidos.

Para visualizar as medidas, é preciso considerar suas relações com os desenvolvedores e os diferentes tipos de artefatos, exigindo esforço cognitivo para interpretar essas informações de forma combinada. Para facilitar o entendimento das informações, foi elaborada uma metáfora visual. Nessa metáfora, é definida a existência de um espaço onde trabalha uma equipe de pessoas (**desenvolvedores**). Nesse espaço, existe uma maleta (**artefato do tipo projeto**) que armazena todos os itens de trabalho da equipe. Um item de trabalho pode ser um documento (**artefato do tipo arquivo**) ou um envelope (**artefato do tipo diretório**). Os documentos podem estar localizados dentro de envelopes e/ou arranjados diretamente na maleta. Os envelopes, por sua vez, podem conter documentos e outros envelopes. As pessoas trabalham nos itens da maleta. Trabalhar em um item significa criá-lo, descartá-lo e/ou modificá-lo. Quando uma pessoa trabalha em um item, há uma relação de trabalho entre a pessoa e o item (medidas **NLOC e Quantidade de *Commits***). As pessoas trabalham de forma colaborativa nos itens (medida **Colaboração (em arquivos)**). Assim, várias pessoas podem possuir relação de trabalho com um mesmo item, bem como uma pessoa pode ter relação de trabalho com mais de um item. Há a possibilidade de que as relações de trabalho nos itens estejam concentradas (medida ***Truck Factor***) em uma ou algumas pessoas da equipe, então significa que essa(s) pessoa(s) (**desenvolvedores incluídos**

na medida *Truck Factor*) é(são) considerada(s) importante(s) para o estado atual de trabalho dos itens da maleta.

Desse modo, o espaço definido na metáfora visual contém cinco tipos de elementos: i) pessoa da equipe; ii) maleta; iii) envelope; iv) documento; e v) relação de trabalho. Entretanto, os elementos não são visíveis no espaço durante todo o tempo. De forma geral, a visibilidade dos elementos é limitada pela hierarquia de arquivos e pastas e pelas relações de trabalho dos desenvolvedores com os artefatos. Esse limite de visibilidade permite visualizar os elementos em diferentes níveis de granularidade, respeitando as seguintes situações:

a) **Maleta Fechada.** Nessa situação, a maleta, no nível mais alto da hierarquia de arquivos e pastas, encontra-se "fechada". Além disso, a maleta é visível no espaço. Da mesma forma, as pessoas da equipe e suas respectivas relações de trabalho com a maleta são visíveis no espaço. Por outro lado, nenhum item de trabalho (documentos e/ou envelopes) é visível no espaço;
b) **Maleta Aberta.** Nessa situação, a maleta encontra-se "aberta" e os seus itens de trabalho (documentos e/ou envelopes) são visíveis no espaço. Da mesma forma, as pessoas da equipe que trabalharam nos itens visíveis e suas relações de trabalho com esses itens são visíveis no espaço. Por outro lado, a maleta não é visível no espaço. Além disso, os envelopes visíveis, permanecem "fechados", logo, seus possíveis documentos e envelopes internos não são visíveis no espaço;
c) **Envelope Aberto.** Nessa situação, um envelope, localizado em algum nível abaixo da maleta na hierarquia de arquivos e pastas, encontra-se "aberto". Os itens internos (documentos e/ou envelopes) desse envelope são visíveis no espaço. As pessoas da equipe que trabalharam nos itens visíveis e suas relações de trabalho com esses itens são visíveis no espaço. Por outro lado, o envelope aberto não é visível no espaço. Além disso, os elementos externos ao envelope aberto, localizados em níveis superiores na hierarquia, também não são visíveis.

Os elementos visíveis são representados no espaço com a utilização de ícones. A representação de elementos de interesse utilizando ícones é uma característica do paradigma visual conhecido como paradigma Iconográfico (KEIM; KRIEGEL, 1996). Uma preocupação ao utilizar esse paradigma é escolher ícones representativos, que permitam uma interpretação eficiente dos dados visualizados. Ao considerar os tipos de elementos da metáfora e suas respectivas características, foram definidos cinco ícones para a técnica visual:

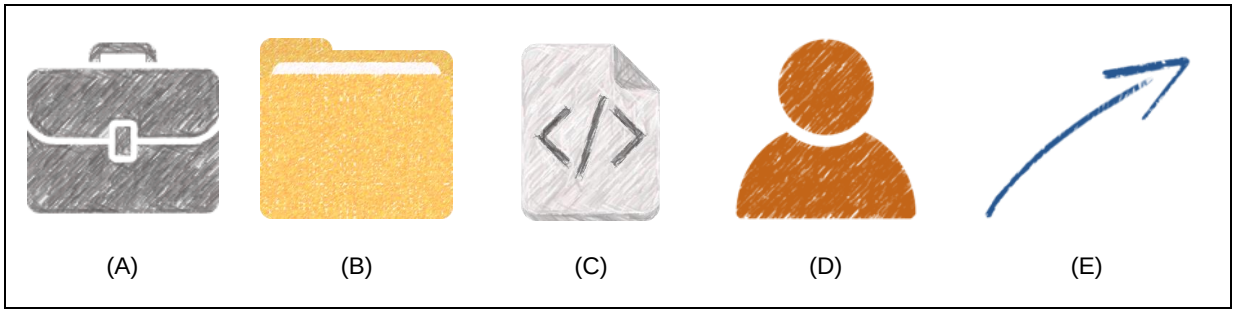
a) **Ícone Maleta.** Deve ser atribuído ao elemento do tipo maleta (artefato do tipo projeto). O ícone é definido pela ilustração de uma maleta (Figura C.1-A), representando a ideia desse elemento ser o recipiente que contém todos os itens de trabalho (artefatos do projeto);
b) **Ícone Envelope.** Deve ser atribuído a cada elemento do tipo envelope (artefato do tipo diretório). O ícone é definido pela ilustração de um envelope de papel (Figura C.1-B), representando a ideia desse elemento ser um recipiente localizado na maleta e que tem a capacidade de armazenar documentos (artefatos do tipo arquivo) e outros envelopes (artefatos do tipo diretório);
c) **Ícone Documento.** Deve ser atribuído a cada elemento do tipo documento (artefato do tipo arquivo). O ícone é definido pela ilustração de um documento de texto (Figura C.1-C), representando a ideia desse elemento armazenar texto (ou código) e está contido na maleta (artefato do tipo projeto) ou em algum envelope (artefato do tipo diretório);
d) **Ícone Pessoa.** Deve ser atribuído a cada elemento do tipo pessoa da equipe (desenvolvedor do projeto). O ícone é definido pela ilustração de uma pessoa (Figura C.1-D), representando a ideia desse elemento ser referente a uma pessoa (desenvolvedor) que trabalhou em itens (artefatos). Além disso, deve haver uma variação de coloração desse ícone, diferenciando os desenvolvedores incluídos na medida *Truck Factor* dos não incluídos;
e) **Ícone Relação de Trabalho.** Deve ser atribuído a cada elemento do tipo relação de trabalho (medidas NLOC, Quantidade de *Commits* e Colaboração (em arquivos)). O ícone é definido pela ilustração de uma seta (Figura C.1-E), representando a ideia desse elemento representar a relação entre uma pessoa (desenvolvedor) e o item (artefato) que ela trabalhou. Além disso, a seta deve partir da pessoa e apontar para o item que ela trabalhou (artefato do projeto), conectando esse par de elementos.

A representação de artefatos de software utilizando ícones como maleta, envelope e documento não é uma inovação da técnica visual usada em *Developer Tracker*. Existem sistemas operacionais (*e.g. Windows*[22]) e IDE's (*e.g. Eclipse*[23]) que utilizam esses ícones ou ícones parecidos para representar artefatos de software. Um exemplo, no *Eclipse*, é o uso do ícone pacote (semelhante ao Ícone Envelope) para representar um diretório contendo arquivos *Java*[24]. Assim, uma das justificativas para a escolha desses ícones é provocar a sensação de

[22] https://www.microsoft.com/pt-br/windows/
[23] https://www.eclipse.org/
[24] https://www.java.com/pt-BR/

familiaridade, diminuindo a carga cognitiva para interpretar as informações. Além de ícones, o uso de rótulos e/ou de legendas é sugerido para facilitar o entendimento das informações. Uma sugestão é adicionar um rótulo ao Ícone Pessoa contendo o nome do desenvolvedor representado. Outra sugestão é adicionar rótulos com nome dos diretórios aos ícones de envelopes, e nome dos arquivos aos ícones de documentos. A última sugestão é adicionar as medidas NLOC e Quantidade de *Commits* como rótulo de seus respectivos Ícones de Relação de Trabalho.

Figura C.1 - Exemplos de ícones da técnica visual da abordagem *Developer Tracker*.



Fonte: Do Autor (2021).

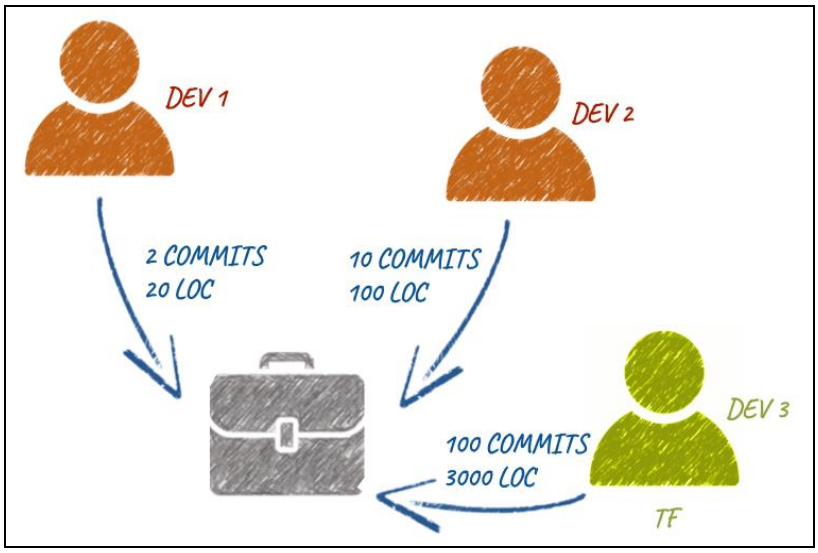
As definições apresentadas para a metáfora visual permitem a visualização de medidas quantitativas, relacionando-as com desenvolvedores e artefatos de um projeto software. Contudo, o software evolui ao longo do tempo de projeto. Durante o processo de evolução do sistema de software, são geradas diferentes versões e, em cada versão, há artefatos em diferentes estados de trabalho. Além disso, ao comparar duas versões do projeto de software, podem existir desenvolvedores, artefatos e valores diferentes para as medidas. Nesse sentido, na metáfora visual, possui a definição de uma variação do espaço, permitindo a visualização simultânea de duas versões do projeto de software. Nessa variação, são consideradas as seguintes regras:

a) É possível visualizar no espaço, pessoas da equipe e itens de trabalho referentes a duas versões da maleta (projeto). Por exemplo, se em uma das versões havia duas pessoas na equipe e, na outra versão, havia as mesmas duas pessoas e uma pessoa adicional, então será possível visualizar as três pessoas no espaço. A visibilidade dos elementos segue as mesmas regras definidas nas situações: Maleta Fechada, Maleta Aberta, Envelope Aberto;
b) É possível visualizar no espaço, a diferença das relações de trabalho entre pessoas e itens, considerando duas versões da maleta (projeto). Desse modo, as relações de trabalho são caracterizadas de três maneiras: i) **relação de trabalho mantida ou modificada** (existe nas duas versões e pode possuir ou não medidas diferentes); ii) **relação de trabalho nova**

(não existe na primeira versão, mas existe na segunda versão); e iii) **relação de trabalho removida** (existe na primeira versão, mas não existe na segunda versão).

É importante ressaltar que, ao comparar duas versões, o Ícone Relação de Trabalho possui três variações de coloração para diferenciar as relações de trabalho: i) mantidas ou modificadas; ii) novas; e iii) removidas. Além disso, o Ícone Pessoa possui um rótulo para diferenciar desenvolvedores incluídos na medida *Truck Factor* na primeira versão, na segunda versão ou em ambas as versões da comparação. São apresentados exemplos de aplicação da técnica visual na Figura C.2, na Figura C.3 e na Figura C.4. A primeira figura contém a ilustração de uma equipe composta por três desenvolvedores. Cada desenvolvedor possui relações de trabalho com a maleta (projeto). O *Dev 3* é o desenvolvedor incluído no *Truck Factor*, pois possui uma coloração diferente dos outros desenvolvedores e um rótulo "*TF*". A situação do espaço nessa figura é "Maleta Fechada".

Figura C.2 - Exemplo da aplicação da técnica visual da abordagem *Developer Tracker* na situação "Maleta Fechada"



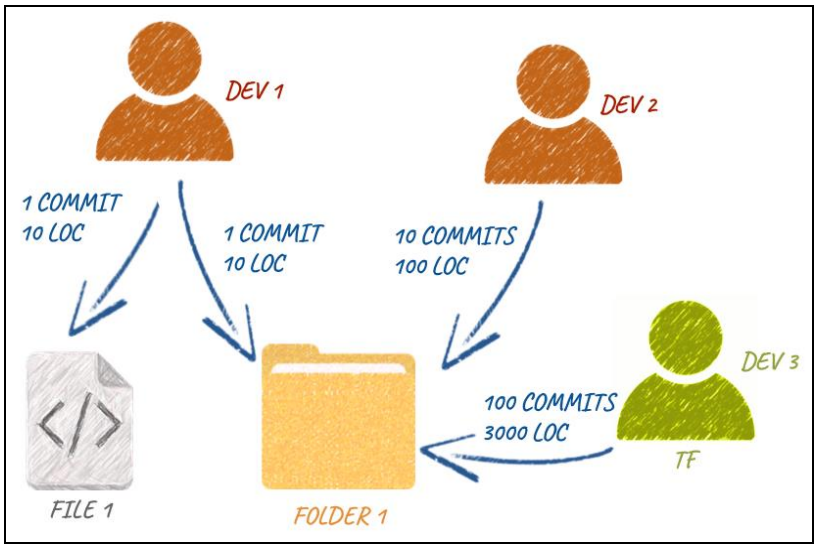
Fonte: Do Autor (2021).

Na Figura C.3, é apresentado o mesmo exemplo da Figura C.2, porém, na situação "Maleta Aberta". A maleta não é mais visível, mas os itens internos (*File 1* e *Folder 1*) sim. Além disso, todos os desenvolvedores da equipe colaboraram com o estado de trabalho dos itens contidos no envelope *Folder 1*. Por fim, o *Dev 1* foi o único a trabalhar no arquivo *File 1*.

Continuando com o exemplo anterior, na Figura C.4, é apresentada a visão comparativa de duas versões do projeto. A situação do espaço é "Maleta Fechada". A primeira versão de comparação é *V1* e a segunda versão é *V2*, logo, as medidas são calculadas

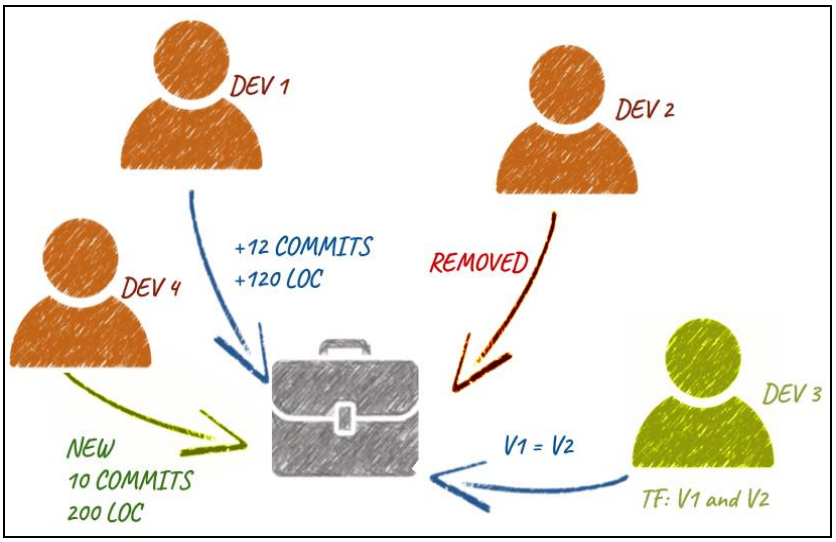
considerando que *V1* evolui para *V2*. Observando a relação de trabalho do *Dev 1* com a maleta, ele incrementou novos *commits* e é autor de mais linhas do código na *V2* do que na *V1* (indicado pela coloração da seta e o rótulo que descreve quanto os valores das medidas foram incrementados). O *Dev 2* não realizou *commit* na evolução de *V1* para *V2*, bem como todas as suas linhas de código foram excluídas ou substituídas. Assim, a relação de trabalho que existia entre *Dev 2* e a maleta foi extinta em *V2* (indicado pela coloração da seta e o rótulo *Removed*). O *Dev 3* é o desenvolvedor incluído no *Truck Factor* em *V1* e em *V2* (indicado pela coloração do seu ícone e pelo rótulo inferior *TF*: *V1 and V2*). Além disso, *Dev 3* não realizou novo *commit* durante a evolução de *V1* para *V2*, bem como nenhuma de suas linhas de código foi excluída ou modificada por outro desenvolvedor (indicado pela coloração da seta e pelo rótulo *V1 = V2*). Por fim, o *Dev 4* não contribuiu com o desenvolvimento de *V1*, mas contribuiu com a evolução de *V1* para *V2* (indicado pela coloração da seta e pelo rótulo *NEW*). É importante ressaltar que não necessariamente *V1* precisa ser uma versão anterior a *V2*. Para fins de comparação, considera-se que são versões diferentes e o cálculo das medidas será direcionado pela seguinte pergunta: o que mudou de *V1* para *V2*?

Figura C.3 - Exemplo da aplicação da técnica visual da abordagem *Developer Tracker* na situação “Maleta Aberta”



Fonte: Do Autor (2021).

Figura C.4 - Exemplo da aplicação da técnica visual da abordagem *Developer Tracker* na situação “Maleta Fechada” da visualização comparativa.



Fonte: Do Autor (2021).

ANEXO D

TERMO DE CONSENTIMENTO LIVRE E ESCLARECIDO

Esta pesquisa tem o objetivo de caracterizar a utilização da abordagem *Developer Tracker* para apoiar a realização de atividades de um Gerente de Projetos (GP) e está sendo conduzida por pesquisadores da Universidade Federal de Lavras (UFLA).

Se você é um Gerente de Projetos, gostaríamos de convidá-lo a participar executando um conjunto de tarefas e respondendo este roteiro de entrevista. O tempo estimado para participação é em torno de 90 minutos. Ao responder a pesquisa, você concorda em permitir que os pesquisadores obtenham, usem e divulguem as informações geradas a partir dos dados conforme descrito abaixo com finalidade acadêmica.

Agradecemos a colaboração.

CONDIÇÕES E ESTIPULAÇÕES

1. Você está sendo convidado para participar da pesquisa "Uma Abordagem Visual para Apoiar o Gerente de Projetos a Compreender o Trabalho dos Desenvolvedores".

2. Você foi selecionado para ser voluntário e sua participação não é obrigatória.

3. A qualquer momento você pode desistir de participar e retirar seu consentimento.

4. Sua recusa não trará nenhum prejuízo em sua relação com o pesquisador ou com a instituição.

5. Sua participação nesta pesquisa consistirá em realizar um conjunto de tarefas e em responder uma entrevista com questões objetivas e de texto curto.

6. A sua participação nesta pesquisa pode envolver algum desconforto, ainda que pequeno, relacionado ao tempo despendido para realização das tarefas e resposta à entrevista. Faremos o possível para minimizar possíveis desconfortos e não ocupar seu tempo desnecessariamente.

7. As informações obtidas serão confidenciais e asseguramos o sigilo sobre sua participação. Os dados publicados não permitirão a sua identificação. Os dados publicados não permitirão a identificação da sua empresa/organização. Os dados publicados não permitirão a identificação do seu projeto.

8. As informações obtidas serão utilizadas única e exclusivamente para os fins específicos deste estudo.

9. Os benefícios relacionados à sua participação estão em contribuir com a pesquisa científica. Será permitido o acesso aos resultados desta pesquisa por meio de publicações científicas realizadas a partir desse estudo.

10. Esta entrevista utiliza o pacote de aplicativo Google Docs, portanto a coleta e o uso de informações do Google estão sujeitos à Política de privacidade do Google (https://www.google.co.uk/policies/privacy/).

11. Ao responder "Concordo", você concorda com as informações aqui descritas, porém a qualquer momento você pode interromper a pesquisa sem ônus algum.

12. Abaixo seguem os dados de contato dos responsáveis por esta pesquisa, com os quais você pode tirar suas dúvidas sobre sua participação.

Matheus Ferreira (matheus.ferreira5@estudante.ufla.br)

Heitor Costa (heitor@ufla.br) - Orientador

Universidade Federal de Lavras (UFLA) - Instituto de Ciências Exatas e Tecnológicas - Departamento de Ciência da Computação - Caixa Postal 3037, CEP 37200-900 - Lavras/MG

ANEXO E

Tabela E.1 – Questões do roteiro de entrevista.

Grupo	#	Tipo	Questão	Tarefa	Alternativas
1	**1.1**	**Objetiva**	1.1 - Qual(is) das seguintes atividades você executa/executou em projetos de software?	-	() Planejamento do projeto () Geração de relatórios () Gerenciamento de riscos () Gerenciamento de pessoas () Elaboração de propostas
	1.2	**Objetiva**	1.2 - Quanto tempo você possui de experiência em executar as atividades selecionadas na questão anterior?	-	() Menos de 1 ano () Entre 1 e 3 anos () Entre 3 e 8 anos () Mais de 8 anos
	1.3	**Objetiva**	1.3 - Você participou de quantos projetos durante o tempo informado na questão anterior?	-	() 1 projeto () Entre 2 e 3 projetos () Entre 4 e 5 projetos () Entre 6 e 8 projetos () Mais de 8 projetos
	1.4	**Objetiva**	1.4 - Em paralelo ao projeto disponibilizado, quantos projetos simultâneos você gerencia/gerenciou?	-	() Gerencio apenas 1 projeto () 2 projetos () 3 projetos () 4 projetos () Mais de 4 projetos
	1.5	**Descritiva**	1.5 - Identificador da empresa.	-	-
2	**2.1**	**Objetiva**	2.1 - Qual a duração do projeto (tempo do contrato)?	-	() Menos de 1 ano () Entre 1 e 2 anos () Entre 2 e 4 anos () Entre 4 e 5 anos () Mais de 5 anos
	2.2	**Objetiva**	2.2 - Quantos desenvolvedores fazem/fizeram parte do time que executa/executou o projeto?	-	() 1 desenvolvedor () Entre 2 e 3 desenvolvedores () Entre 4 e 5 desenvolvedores () Entre 6 e 8 desenvolvedores () Mais de 8 desenvolvedores
3	**3.1**	**Objetiva**	3.1 - Você sabe informar qual é a tecnologia (linguagem de programação) mais demandada no projeto?	Tarefa 1	() Sim () Não
	3.2	**Objetiva**	3.2 - Quando pequena parcela de desenvolvedores concentra o conhecimento sobre a implementação, há risco de dependência dessas pessoas. Você sabe informar se há concentração de conhecimento sobre a implementação?	Tarefa 2	() Sim () Não
	3.3	**Objetiva**	3.3 - Você sabe informar qual módulo/parte do projeto tem demandado mais esforço do time de desenvolvedores?	Tarefa 3	() Sim () Não
	3.4	**Objetiva**	3.4 - Você sabe informar em qual módulo/ parte do projeto cada desenvolvedor tem maior atuação?	Tarefa 4	() Sim () Não
	3.5	**Objetiva**	3.5 - Considerando quaisquer duas versões do software, você sabe informar se houve alguma mudança significativa na atuação dos desenvolvedores nos módulos/partes do projeto entre essas duas versões? (exemplos: grau de concentração de conhecimento mudou; algum desenvolvedor atuou em módulos diferentes em cada uma das versões)	Tarefa 5	() Sim () Não
	3.6	**Descritiva**	3.6 - Como você obtém as informações para responder as questões anteriores (3.1 até 3.5)? (método, métrica, ferramenta, observação - conhecimento empírico, etc.)	-	-

(continua)

Tabela E.1 – Questões do roteiro de entrevista (continuação).

Grupo	#	Tipo	Questão	Tarefa	Alternativas
4	**4.1**	**Objetiva**	4.1 - Como o uso da abordagem *Developer Tracker* impactou em sua compreensão anterior sobre a questão "3.1 - Você sabe informar qual é a tecnologia (linguagem de programação) mais demandada no projeto?"?	Tarefa 1	() Aprimorou minha compreensão () Confirmou minha compreensão () Não interfere na minha percepção () Piorou minha compreensão
	4.2	**Objetiva**	4.2 - Como o uso da abordagem *Developer Tracker* impactou em sua compreensão anterior sobre a questão "3.2 - Quando uma pequena parcela de desenvolvedores do time concentra o conhecimento sobre a implementação, há um risco de dependência dessas pessoas no projeto. Você sabe informar se há concentração de conhecimento sobre a implementação?"?	Tarefa 2	() Aprimorou minha compreensão () Confirmou minha compreensão () Não interfere na minha percepção () Piorou minha compreensão
	4.3	**Objetiva**	4.3 - Como o uso da abordagem *Developer Tracker* impactou em sua compreensão anterior sobre a questão "3.3 - Você sabe informar qual módulo/parte do projeto tem demandado mais esforço do time de desenvolvedores?"?	Tarefa 3	() Aprimorou minha compreensão () Confirmou minha compreensão () Não interfere na minha percepção () Piorou minha compreensão
	4.4	**Objetiva**	4.4 - Como o uso da abordagem *Developer Tracker* impactou em sua compreensão anterior sobre a questão "3.4 - Você sabe informar em qual módulo/parte do projeto cada desenvolvedor tem maior atuação?"?	Tarefa 4	() Aprimorou minha compreensão () Confirmou minha compreensão () Não interfere na minha percepção () Piorou minha compreensão
	4.5	**Objetiva**	4.5 - Como o uso da abordagem *Developer Tracker* impactou em sua compreensão anterior sobre a questão "3.5 - Considerando quaisquer duas versões do software, você sabe informar se houve alguma mudança significativa na atuação dos desenvolvedores nos módulos/partes do projeto entre essas duas versões? (exemplos de mudança: o desenvolvedor principal mudou; a concentração de conhecimento diminuiu; algum desenvolvedor atuou em módulos diferentes em cada uma das versões)"?	Tarefa 5	() Aprimorou minha compreensão () Confirmou minha compreensão () Não interfere na minha percepção () Piorou minha compreensão
5	**5.1**	**Descritiva**	4.6 - Em quais atividades do seu trabalho você acha que o uso a abordagem pode auxiliar?	-	-
	5.2	**Descritiva**	4.7 - Em quais atividades do seu trabalho você acha que o uso da abordagem pode atrapalhar?	-	-
	5.3	**Objetiva**	4.8 - De forma geral, quão útil você considera o uso dessa abordagem para auxiliar suas ações?	-	() Extremamente útil () Útil () Inútil () Inútil e prejudicial (pode atrapalhar minha tomada de decisão)

Fonte: Do Autor (2021).

ANEXO F

Tabela F.1 – Opinião dos participantes sobre o uso da abordagem.

Participante	Como a abordagem pode ajudar?	Como a abordagem pode atrapalhar?	Opinião sobre a utilidade da abordagem
P1	- Identificação dos gargalos de equipe (pessoas que concentram conhecimento, realizar a distribuição para permitir férias, por exemplo) - Identificar as referências em Tecnologias (podendo olhar em vários projetos inclusive para formação de equipes de projeto). Uma grande vantagem é capturar os dados de uma forma mais fácil e prática.	- Essas métricas são mais um insumo do que decisões finais, pois podem ser enviesadas por LOC e *commits*. Usá-las como julgamento final pode ser um erro.	Extremamente útil
P2	- Alocação/gerenciamento de equipe (observando tecnologia e módulos/partes do projeto) - Contribuir para medir (NÃO DEFINIR) performance de maneira objetiva (ex. quais são as pessoas que mais contribuem no time). Funciona como um indicador a mais. - Gestão de risco relacionada ao conhecimento concentrado em poucas pessoas. Fornece uma visão boa dentro dos módulos e do projeto como um todo de quem são essas pessoas. - Quando é preciso realizar trocas na liderança ou passagens de conhecimento é possível utilizar a ferramenta para transferir "sem viés".	- Impressão de que performance por LOC, mesmo isso não representando valor e qualidade do que foi implementado.	Extremamente útil
P3	- Saber a concentração de conhecimento sobre a implementação, ajudando na análise de riscos - Ver muitos *commits* em alguma região do código em espaço pequeno de tempo pode significar que as pessoas não entenderam o que deveria ser implementado, ou que era uma implementação complexa e/ou que pode ter gerado uma grande qtd de bugs.	- Fazer análise sem conhecimento prévio do projeto poderia levar a conclusões equivocadas. Por exemplo, pessoas podem aparecer como concentradoras do conhecimento trabalhando em arquivos gerados/compilados ou arquivos de testes que representem grande quantidade de linhas.	Extremamente útil
P4	- Identificar concentração de conhecimento e ajudar na tomada de ações para mitigá-la. - Acompanhar evolução do projeto utilizando a comparação de versões.	-	Útil
P5	Associar com métricas de estimativa/sprint. Mensurar se ações de descentralização tiveram efeito.	Fornece um falso-positivo uma vez que métricas baseadas em linhas de código não representam complexidade da solução implantada	Útil

(continua)

Tabela F.1 – Opinião dos participantes sobre o uso da abordagem (continuação).

Participante	Como a abordagem pode ajudar?	Como a abordagem pode atrapalhar?	Opinião sobre a utilidade da abordagem
P6	- Identificar pessoas chaves do projeto (mapear pessoas que não posso perder no projeto). - Identificar necessidade de incorporar pessoas com habilidades em tecnologias específicas, combinando com meu conhecimento empírico. - Planejamento da rotina. Se identifico que somente uma pessoa trabalha em determinado módulo, eu preciso estar com isso em mente para dar continuidade quando a pessoa entrar de férias, por exemplo.	- No caso específico desse projeto, que tem código compartilhado com outro projeto, a análise pode ficar distorcida. Poque quando analiso como um todo, tem-se nos relatórios a participação de pessoas externas projeto.	Útil
P7	Saber quem pode ser o foco para resolver ou atuar em determinada parte do sistema, como por exemplo, se um recurso conhece mais aquela parte do código, ele mesmo pode ser designado para resolver um problema daquele trecho.	-	Útil
P8	- Avaliar pessoas novas no time. - Descobrir pontos de atenção (se há um arquivo com muito desenvolvimento, talvez seja retrabalho, talvez não estejamos testando o suficiente) - Verificar se alguma parte do código concentrada em uma só pessoa	- Avaliar a produtividade observando LOC e *commit*, tirando o foco de outras atividades.	Extremamente útil
P9	- Para ambientação de pessoa nova no projeto. - Planejamento de equipe. - Avaliação de desempenho (em promoções, por exemplo, é possível aproveitar os insumos da abordagem par a tomada de decisão)	- Possibilidade de falso positivo dos influenciadores/maiores contribuidores do projeto, principalmente em projetos de longo prazo (devido a visão de todo histórico do projeto, mudanças recentes podem não ficar tão visíveis).	Útil
P10	- Direcionar tarefas para especialistas, pois consigo ver as linguagens que ele domina e onde ele mais trabalhou - Ter um ponto de referência para atender certas demandas - Aplicar métricas para verificar pontos do código que está necessitando de mais correções/evoluções	-	Extremamente útil
P11	- Alocação de pessoas com base em proficiência em linguagem de programação - Medir a produtividade de uma pessoa, com ressalvas para qualidade do código. Identificar momentos em que a pessoa não atuou (exemplos: algum problema? Férias? Ocupado com outras atividades?). Isso é possível ver com o git também. A vantagem da ferramenta é a praticidade em consultar os dados do projeto.	Por falta de configurações de tempo, a visualização mostra desenvolvedores que participaram somente no início do projeto e que atualmente são irrelevantes para as análises. Estaria por exemplo vendo 50 pessoas e atualmente a equipe só tem 10 pessoas.	Útil

(continua)

Tabela F.1 – Opinião dos participantes sobre o uso da abordagem (continuação).

Participante	Como a abordagem pode ajudar?	Como a abordagem pode atrapalhar?	Opinião sobre a utilidade da abordagem
P12	- Realizar a gestão do conhecimento (observar onde as pessoas estão atuando para direcionar atividades para distribuir conhecimento)	- Viés de commit e LOC que pode não representar o domínio da pessoa sobre o "assunto"	Útil
P13	- Bom para liderança técnica (gerentes ligados a parte técnica podem aproveitar melhor por exemplo, informação de que tem mais domínio de determinadas tecnologias)	-	Extremamente útil
P14	- O TF ajudaria a gerenciar o risco de pessoas concentrando o conhecimento (gestão de riscos) - Identificação de pessoas indicadas para tarefas (gestão de pessoas)	Pode não trazer uma ajuda tão efetiva. Pois, na prática do dia a dia a pessoa indicada será preterida pela pessoa disponível (em muitos casos). Com relação à concentração, é algo sabido pela empresa/projeto, mas não há muitos movimentos para tratar esse risco. Geralmente a atuação é mais reativa do que preventiva. Além disso, apesar da abordagem aprimorar algumas convicções, muitas delas eu preferiria não usar a abordagem para tomar minha decisão.	Útil
P15	- Se um projeto está parado há algum tempo ou irá trocar de equipe. A abordagem ajuda a buscar informações sobre alguns módulos/partes com pessoas que mais trabalham nesses módulos/partes (algumas pessoas podem ter saído da empresa, outras podem estar em outros projetos/equipes - ajuda a contatar as pessoas certas)	- Alguns *commits* são em arquivos gerados por bibliotecas e contabilizam como conexão do desenvolvedor no artefato. Se há muitos *commits* nesses arquivos pode parecer que houve um grande esforço para gerar aquele arquivo, mas na verdade é um arquivo gerado (demanda conhecer bem a estrutura de arquivos do projeto).	Extremamente útil
P16	- Identificar locais do código onde há necessidade de colocar outra pessoa para ajudar (melhorar a alocação de recursos em projetos). - Identificar o alto grau de concentração de conhecimento (ver o valor do truckfactor) -> um alerta para tomar ações de distribuição do conhecimento. Ajuda a planejar férias de pessoas, por exemplo, o que é um problema crítico da empresa.	- Viés da quantificação por LOC e *Commit*. A pessoa add muita linha de código, mas será que ela tá escrevendo bom código. Commitar mais não é sinônimo de trabalhar mais.	Útil

Fonte: Do Autor (2021).

ANEXO G

Tabela G.1 – Relação de similaridade entre opinião dos participantes e atividades do GP

Atividade	Respostas Agrupadas
Identificação das habilidades e perfil do desenvolvedor	P1 - Identificar as referências em Tecnologias (podendo olhar em vários projetos inclusive para formação de equipes de projeto). P2 - Alocação/gerenciamento de equipe (observando tecnologia e módulos/partes do projeto) P6 - identificar necessidade de incorporar pessoas com habilidades em tecnologias específicas, combinando com meu conhecimento empírico. - Planejamento da rotina. Se identifico que somente uma pessoa trabalha em determinado módulo, eu preciso estar com isso em mente para dar continuidade quando a pessoa entrar de férias, por exemplo. P7 - Saber quem pode ser o foco para resolver ou atuar em determinada parte do sistema, como por exemplo, se um recurso conhece mais aquela parte do código, ele mesmo pode ser designado para resolver um problema daquele trecho. P9 - Planejamento de equipe. P10 - Direcionar tarefas para especialistas, pois consigo ver as linguagens que ele domina e onde ele mais trabalhou P10 - Ter um ponto de referência para atender certas demandas P11 - Alocação de pessoas com base em proficiência em linguagem de programação P13 - Bom para liderança técnica (gerentes ligados a parte técnica podem aproveitar melhor por exemplo, informação de que tem mais domínio de determinadas tecnologias) P14 - Identificação de pessoas indicadas para tarefas (gestão de pessoas) P15 - se um projeto está parado há algum tempo ou irá trocar de equipe. A abordagem ajuda a buscar informações sobre alguns módulos/partes com pessoas que mais trabalham nesses módulos/partes (algumas pessoas podem ter saído da empresa, outras podem estar em outros projetos/equipes - ajuda a contatar as pessoas certas) P16 - identificar locais do código onde há necessidade de colocar outra pessoa para ajudar (melhorar a alocação de recursos em projetos).
Estimativa de custos e prazos de projetos e identificação de anomalias no desempenho do desenvolvedor	P4 - Acompanhar evolução do projeto utilizando a comparação de versões. P8 - Avaliar pessoas novas no time. P9 - Para ambientação de pessoa nova no projeto. P3 - Ver muitos commits em alguma região do código em espaço pequeno de tempo pode significar que as pessoas não entenderam o que deveria ser implementado, ou que era uma implementação complexa e/ou que pode ter gerado uma grande qtd de bugs. P8 - Descobrir pontos de atenção (se há um arquivo com muito desenvolvimento, talvez seja retrabalho, talvez não estejamos testando o suficiente) P8 - Verificar se alguma parte do código concentrada em uma só pessoa P10 - Aplicar métricas para verificar pontos do código que está necessitando de mais correções/evoluções P11 - Medir a produtividade de uma pessoa, com ressalvas para qualidade do código. Identificar momentos em que a pessoa não atuou (exemplos: algum problema? Férias? Ocupado com outras atividades?). Isso é possível ver com o git também.
Melhoria no desempenho das equipes	P2 - Contribuir para medir (NÃO DEFINIR) performance de maneira objetiva (ex. quais são as pessoas que mais contribuem no time). Funciona como um indicador a mais. P5 - Associar com métricas de estimativa/sprint.

(continua)

Tabela G.1 – Relação de similaridade entre opinião dos participantes e atividades do GP (continuação).

Atividade	Respostas Agrupadas
Compreensão e controle da distribuição do conhecimento	P1 - Identificação dos gargalos de equipe (pessoas que concentram conhecimento, realizar a distribuição para permitir férias, por exemplo) P2 - Gestão de risco relacionada ao conhecimento concentrado em poucas pessoas. Fornece uma visão boa dentro dos módulos e do projeto como um todo de quem são essas pessoas. P2 - Quando é preciso realizar trocas na liderança ou passagens de conhecimento é possível utilizar a ferramenta para transferir "sem viés". P3 - Saber a concentração de conhecimento sobre a implementação, ajudando na análise de riscos P4 - Identificar concentração de conhecimento e ajudar na tomada de ações para mitigá-la. P5 - Mensurar se ações de descentralização tiveram efeito. P6 - identificar pessoas chaves do projeto (mapear pessoas que não posso perder no projeto); P12 - Realizar a gestão do conhecimento (observar onde as pessoas estão atuando para direcionar atividades para distribuir conhecimento) P14 - O TF ajudaria a gerenciar o risco de pessoas concentrando o conhecimento (gestão de riscos) P16 - identificar o alto grau de concentração de conhecimento (ver o valor do truckfactor) -> um alerta para tomar ações de distribuição do conhecimento. Ajuda a planejar férias de pessoas, por exemplo, o que é um problema crítico da empresa.
Reajustes de remuneração	P9 - Avaliação de desempenho (em promoções, por exemplo, é possível aproveitar os insumos da abordagem par a tomada de decisão)
Identificação de necessidade de investimento (treinamento ou equipamentos)	-
Planejamento de melhoria da qualidade de código	-

Fonte: Do Autor (2021).