PODER EXECUTIVO
MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DO AMAZONAS
INSTITUTO DE COMPUTAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

FATORES DE INFLUÊNCIA NA PRODUTIVIDADE DOS DESENVOLVEDORES DE ORGANIZAÇÕES DE SOFTWARE

EDSON CÉSAR CUNHA DE OLIVEIRA

Manaus, Dezembro de 2017

EDSON CÉSAR CUNHA DE OLIVEIRA

FATORES DE INFLUÊNCIA NA PRODUTIVIDADE DE DESENVOLVEDORES DE ORGANIZAÇÕES DE SOFTWARE

Tese de Doutorado submetida ao corpo docente do Programa de Pós-Graduação em Informática da Universidade Federal do Amazonas (PPGI-UFAM).

Orientadora: Profª. Tayana Uchôa Conte, D.Sc.
Co-Orientador: Prof. Marco Cristo, D.Sc.

Manaus, Dezembro de 2017

Ficha Catalográfica

Ficha catalográfica elaborada automaticamente de acordo com os dados fornecidos pelo(a) autor(a).

O48f

Oliveira, Edson César Cunha de
Fatores de influência na produtividade dos desenvolvedores de organizações de software / Edson César Cunha de Oliveira. 2017
182 f.: il. color; 31 cm.

Orientadora: Tayana Uchôa Conte
Coorientador: Marco Antônio Pinheiro de Cristo
Tese (Doutorado em Informática) - Universidade Federal do Amazonas.

1. Engenharia de Software. 2. Produtividade de Software. 3. Produtividade do Desenvolvedor. 4. Fatores de Produtividade. I. Conte, Tayana Uchôa II. Universidade Federal do Amazonas III. Título

PODER EXECUTIVO
MINISTÉRIO DA EDUCAÇÃO
INSTITUTO DE COMPUTAÇÃO

PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

UFAM

FOLHA DE APROVAÇÃO

"Fatores de Influência na Produtividade de Desenvolvedores de Organizações de Software"

EDSON CÉSAR CUNHA DE OLIVEIRA

Tese de Doutorado defendida e aprovada pela banca examinadora constituída pelos Professores:

Profa. Tayana Uchôa Conte - PRESIDENTE

Prof. José Reginaldo Hughes Carvalho - MEMBRO INTERNO

Prof. José Luiz de Souza Pio - MEMBRO EXTERNO

Prof. Gleison dos Santos Souza - MEMBRO EXTERNO

Prof. Alessandro Fabrício Garcia - MEMBRO EXTERNO

Manaus, 15 de Dezembro de 2017

À Deus,

à minha esposa Nívea

e aos meus filhos Alexandre e Isabela.

AGRADECIMENTOS

À Deus, pois sem Ele nada teria sido possível!

A minha esposa, pelo apoio incondicional, mesmo nas horas mais difíceis.
Aos meus filhos, pela compreensão de minha ausência prolongada em suas vidas.

À minha orientadora Tayana Conte, por sempre acreditar em mim, mesmo quando nem eu mesmo acreditava.
Ao meu co-orientador Marco Cristo, por seus conselhos e pelo apoio desde o início dessa caminhada.

À Dra. Monalessa Barcellos, Dr. Alessandro Garcia, Dr. Gleison Santos, Dr. José Reginaldo Hughes Carvalho e Dr. José Luiz de Souza Pio por aceitarem participar de minhas bancas de qualificação e de defesa.

Ao grupo USES, que sempre incentivou e ajudou de diversas formas para que esse trabalho fosse concluído.

À todos os que participaram de alguma forma na realização desta tese.

À SEFAZ pela liberação para o desenvolvimento deste trabalho.
À UFAM pela oportunidade da realização de mais um sonho.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

RESUMO

A atividade de desenvolvimento de *software* tem sido caracterizada como uma atividade essencialmente humana e as organizações de *software* devem investir no aprimoramento da produtividade de seus desenvolvedores para se manterem competitivas. Na literatura científica, centenas de fatores de influência sobre a produtividade do desenvolvedor foram identificados. Apesar disso, as organizações de *software* ainda não conhecem quais são os fatores mais significantes. Além disso, cada organização deve considerar fatores de produtividade relacionados com seu próprio contexto. Assim, a escolha de práticas eficazes para o aprimoramento dos seus desenvolvedores deve fundamentar-se nos fatores preponderantes, ou seja, que exerçam maior influência sobre a produtividade do desenvolvedor dentro do contexto das organizações. Deste modo, nesta pesquisa foram investigadas métricas e fatores de influência na produtividade do desenvolvedor, categorizados em fatores técnicos, humanos e organizacionais. Foram realizados estudos qualitativos e quantitativos na busca das respostas com resultados práticos e aplicáveis em organizações reais de desenvolvimento de *software*. Após a execução da metodologia adotada, aplicada em dados da oriundos da indústria, foram produzidos uma série de resultados importantes sobre as métricas de avaliação da produtividade do desenvolvedor e sobre a influência dos fatores técnicos, humanos e organizacionais. Com isso, espera-se contribuir para o corpo de conhecimento da Engenharia de *Software* sobre métricas e fatores de influência da produtividade dos desenvolvedores das organizações de desenvolvimento de *software*.

Palavras-chave: Produtividade do Desenvolvedor e Engenharia de *Software*.

ABSTRACT

Software development has been characterized in essence as a human activity and software organizations must invest in improving the productivity of their developers to stay competitive. In the scientific literature, hundreds of influencing factors on developer productivity have been identified. Despite of this, software organizations still do not know what the most significant factors are. In addition, each organization must consider productivity factors related to its own context. Thus, the choice of effective practices for the improvement of its developers' productivity must be based on the preponderant factors, that exert greater influence, on the developers' productivity within the organization context. Therefore, in this research we investigated productivity metrics and factors of influence in the developers' productivity, categorized in technical, human and organizational factors. Qualitative and quantitative studies were carried out in the search for answers with practical and applicable results in real software development organizations. Following the accomplishment of the research adopted methodology, applied directly in the industry, a series of important results were produced on the metrics for evaluation of developer's productivity and on the influence of technical, human and organizational factors. With this, we hope to contribute to the body of knowledge in Software Engineering about the metrics and factors of developer's productivity located in software development organizations.

Keywords: Developer Productivity and Software Engineering.

Capítulo 1 – INTRODUÇÃO

Este capítulo apresenta a contextualização desta pesquisa de doutorado. Além disso, apresenta as motivações, a definição do problema, os objetivos de pesquisa, a estratégia de pesquisa e a metodologia adotada.

1.1 CONTEXTO

A atividade de desenvolvimento de *software* tem sido caracterizada como uma atividade essencialmente humana [Amrit *et al.* 2014] e, por isso, tem incertezas desde o seu início [Trendowicz e Münch 2009]. As organizações de sucesso serão aquelas que melhor se adaptarem no recrutamento, aprimoramento e retenção dos seus profissionais com as habilidades, perspectivas e experiências necessárias para a organização [Ulrich 1998]. Guthridge *et al*. (2008) afirmam que as organizações promovem a ideia de que seus profissionais são a sua maior fonte de vantagem competitiva, porém, a maioria das empresas ainda está despreparada para o desafio de encontrar, motivar e reter os seus profissionais competentes.

A necessidade de investimento das organizações de *software* em práticas que possam não somente recrutar, mas principalmente aprimorar os seus engenheiros de *software*, a fim de se manterem eficientes e competitivas, tem sido um tema recorrente nos últimos anos. Como exemplo, os modelos de processo *People Capability Maturity Model* (P-CMM) [Curtis *et al.* 2009] e Melhoria de Processo do *Software* Brasileiro – Gestão de Pessoas (MR-MPS-RH) [Softex 2016a] foram criados visando a esta necessidade de gerir a relação das organizações de *software* com os seus profissionais.

A organização de *software* deve estabelecer práticas eficazes ao implementar a recomendação de aprimorar continuamente seus engenheiros de *software*. Estas práticas devem ser baseadas nos fatores preponderantes que possam de fato influenciar e contribuir para o aprimoramento dos seus engenheiros. Na literatura científica é possível encontrar muitos diferentes fatores que influenciam a produtividade do desenvolvedor de *software*, tais como: o conhecimento técnico, a experiência, a personalidade, a motivação, os processos e a cultura organizacional. Para uma organização de *software*, a compreensão da relação entre os diversos

fatores que influenciam na produtividade do desenvolvedor pode servir de base para a definição de práticas eficazes de aprimoramento dos seus desenvolvedores.

1.2 MOTIVAÇÕES

A melhoria da produtividade no processo de desenvolvimento de *software* é o principal meio de tornar a organização de *software* mais competitiva [Aquino Junior e Meira 2009]. As organizações de *software* não consideram os fatores de produtividade em seus processos de melhoria da produtividade porque elas frequentemente não sabem com quais fatores elas devem começar, ou seja, quais são os fatores mais significativos [Trendowicz e Münch 2009]. Várias pesquisas têm sido direcionadas para a identificação de fatores que possuam um impacto significativo na produtividade do desenvolvimento de software [Trendowicz e Münch 2009]. Desde o fim dos anos 70, a academia e a indústria vêm despendendo esforço para catalogar esses fatores [Sampaio *et al.* 2010].

Apesar dos vários estudos existentes sobre os fatores de influência na produtividade de software, as organizações de software ainda não conhecem quais são os fatores mais significantes [Sampaio *et al.* 2010]. Além disso, cada organização deve considerar os fatores de produtividade de seu próprio ambiente, ao invés de adotar sem nenhuma avaliação os fatores utilizados em outros contextos [Trendowicz e Münch 2009]. Por fim, as organizações devem considerar a importância das pessoas no desenvolvimento de software [Amrit *et al.* 2014], uma vez que o sucesso dos projetos de software ainda depende essencialmente das pessoas envolvidas [Trendowicz e Münch 2009].

1.3 DEFINIÇÃO DO PROBLEMA

As organizações de *software* têm a necessidade identificar os fatores que exercem maior influência na produtividade dos seus desenvolvedores para basear as suas estratégias de melhoria da produtividade no desenvolvimento de *software*. Porém, a literatura científica não ajuda a guiar as organizações para identificar os fatores mais preponderantes. Apesar de existir uma grande quantidade de fatores de influência na produtividade publicados na literatura [Paiva *et al.* 2010; Sampaio *et al.* 2010; Trendowicz e Münch 2009; Wagner e Ruhe 2008], esses estudos não apresentam uma relação de preponderância entre os fatores por eles investigados.

Analisando o comportamento das organizações de *software* quanto aos seus processos de contratação de novos desenvolvedores é possível notar as seguintes observações: (i) as organizações estabelecem requisitos técnicos para a contratação de novos desenvolvedores em seus anúncios públicos; (ii) as organizações estabelecem para o desenvolvedor os projetos nos quais irão trabalhar, os processos de desenvolvimento que irão seguir, além das condições e do ambiente de trabalho. Além disso, é importante considerar que o desenvolvimento de projetos de *software* é uma atividade essencialmente humana [Trendowicz e Münch 2009], onde as pessoas são vistas como a maior fonte de variação do desempenho dos projetos de *software*. Por isso, nesta pesquisa, ao invés de investigar fatores individuais isoladamente, os fatores relacionados com a produtividade dos desenvolvedores foram classificados em três grupos: técnicos, humanos e organizacionais.

Os fatores técnicos referem-se à competência individual adquirida pela educação especializada e pela experiência na prática, necessários à realização de determinadas atribuições dentro de uma organização de *software*. Exemplos de tais fatores incluem o conhecimento e experiência nas ferramentas, tecnologias e métodos utilizados pela organização [Paiva *et al.* 2010; Trendowicz e Münch 2009]. Os fatores humanos referem-se às características pessoais do desenvolvedor não inerentes à sua função na organização, tais como a sua motivação [França *et al.* 2011] e a sua personalidade [Cruz *et al.* 2015]. Os fatores organizacionais referem-se a fatores existentes no ambiente da organização que impactam na produtividade do desenvolvedor, tais como o processo adotado [SEI 2010; Softex 2016b] e as ferramentas utilizadas [Boehm *et al.* 1995].

A identificação da relação de preponderância entre os fatores técnicos, humanos e organizacionais pode auxiliar as organizações de desenvolvimento de *software* a focar seus esforços de melhoria da produtividade dos desenvolvedores no grupo de maior influência e, por consequência, obter melhoria na produtividade da organização em geral. Portanto, a questão de pesquisa que este trabalho visa responder é: "*Qual a preponderância entre os fatores técnicos, humanos e organizacionais na influência sobre a produtividade do desenvolvedor de organizações de software*?".

1.4 OBJETIVOS DE PESQUISA

O objetivo principal desta pesquisa é buscar fatores preponderantes entre os fatores técnicos, humanos e organizacionais sobre a produtividade dos desenvolvedores de organizações de *software*. Os objetivos específicos deste trabalho são:

- Identificação de métricas de produtividade do desenvolvedor representativas em cada organização pesquisada, a partir da literatura e da percepção por parte da organização de *software*;
- Identificação de um conjunto de fatores técnicos, humanos e organizacionais, a partir da literatura e da percepção por parte da organização de *software*;
- Avaliação da preponderância de influência entre os fatores técnicos, humanos e organizacionais na produtividade do desenvolvedor de organizações de *software*.

1.5 PLANO DE PESQUISA

1.5.1 Estratégia de pesquisa

A comunidade de Engenharia de *Software* tem uma visão pragmática, ao invés de uma posição filosófica, e orientada a resultados quanto à sua metodologia de pesquisa [Runeson *et al.* 2012]. As pesquisas pragmáticas caracterizam-se pela busca da aplicabilidade daquilo que realmente funciona e de soluções práticas para os problemas [Patton 1990]. Nessa abordagem, os pesquisadores procuram pelo *o quê* e *como* pesquisar o fenômeno investigado [Creswell 2013], utilizando-se de todos os métodos de pesquisa disponíveis para a compreensão do problema [Rossman e Wilson 1985].

Nesta pesquisa, adotou-se uma visão pragmática quanto à sua abordagem de investigação, utilizando um método misto de investigação, caracterizado pelo uso de estudos quantitativos e qualitativos, orientados pela busca de respostas com resultados práticos e aplicáveis às organizações de *software*. Esta pesquisa realizou a investigação principalmente através de estudos de caso, pois “um estudo de caso investiga um fenômeno contemporâneo (o “caso”) em seu contexto no mundo real, especialmente quando as fronteiras entre o fenômeno e o contexto possam não estar

claramente evidentes" [Yin 2015]. Segundo Runeson *et al.* (2012), os estudos de caso em Engenharia de *Software*, dada a natureza prática da pesquisa na área, tendem a pesquisas explanatórias, onde as implicações práticas de uma certa prática são mais relevantes que questões sobre princípios filosóficos abstratos.

Uma pesquisa que utiliza estudos de caso pode investigar um único estudo de caso ou múltiplos casos. A utilização de um único estudo de caso é justificada principalmente na investigação de fenômenos raros ou de muito difícil acesso por parte dos pesquisadores; já a utilização de múltiplos casos é indicada para investigações de fenômenos mais comuns, pois "a evidência dos casos múltiplos é, muitas vezes, considerada mais vigorosa e o estudo, em geral, é, por essa razão, visto como mais robusto" [Yin 2015]. Por essa razão, esta pesquisa adota o modelo de múltiplos casos.

A seleção dos casos em um estudo de múltiplos casos não segue a lógica de amostra aleatória normalmente adotada em pesquisas quantitativas. A escolha dos casos deve ter um propósito, que pode ser para encontrar resultados similares, confirmando resultados anteriores, ou para encontrar resultados contrastantes por razões previsíveis [Runeson *et al.* 2012]. Nesta pesquisa, os casos foram escolhidos visando à identificação de resultados similares. Por isso, foram escolhidas três organizações de desenvolvimento de *software* reais representativas da indústria de *software*, que serão apresentadas na próxima subseção.

1.5.2 Organizações selecionadas

As três organizações de software selecionadas para esta pesquisa, que tiveram seus nomes omitidos por questões de confidencialidade, possuem as seguintes características:

- **Órgão público**: Órgão de governo que dentre seus objetivos estão a organização, gerenciamento e disciplina do processo de pagamento e arrecadação, além da coordenação e do controle da execução orçamentária. Seu departamento de informática tem como missão desenvolver soluções de *software* para apoiar os objetivos do órgão no domínio da administração fazendária. As tecnologias adotadas tem predominância da utilização de tecnologias para a *Mainframe* e *Web*.

- **Empresa Pública**: Organização pública do governo que atua como agente executor das políticas de tecnologia da informação. É responsável pela maioria dos sistemas de informação desenvolvidos nas diversas áreas de atuação do governo, tais como, educação, recursos humanos, segurança pública, administração, planejamento e saúde. A organização possui um quadro de 392 funcionários, sendo que a maioria desses é dedicada ao desenvolvimento de *software*. Desenvolve *software* para as plataformas de *Mainframe*, *Web* e *Mobile*.
- **Fundação Privada**: Organização sem fins lucrativos criada em 1998. Atua como um centro tecnológico de pesquisa e desenvolvimento de projetos nas áreas de *software* e *hardware*, capacitação tecnológica e responsabilidade social. Possui por volta de 260 empregados, sendo na maioria de desenvolvedores de *software*. Desenvolve soluções de *software* para automação industrial e sistemas de informação. Entre as tecnologias adotadas para o desenvolvimento de *software* estão as linguagens *Java*, *Objective-C* e *Delphi*, nas plataformas *Desktop*, *Web* e *Mobile*.

O resumo das principais características das organizações selecionadas estão detalhadas na Tabela 1. Nessa tabela podemos observar que as organizações selecionadas tem muitas similaridades a algumas diferenças, tais como as suas certificações, natureza social e domínio de aplicação. Ademais, essas organizações são representativas na indústria de *software* pelo impacto de suas contribuições para a sociedade.

1.5.3 Metodologia adotada

Neste trabalho foram utilizados múltiplos métodos de pesquisa visando buscar responder a questão de pesquisa proposta nesta tese. Mais especificamente, foram utilizadas revisões de literatura, exploratórias e sistemáticas, estudos de caso qualitativos, com análise qualitativa de entrevistas com gerentes e líderes das organizações selecionadas, e um estudo correlacional quantitativo, envolvendo as métricas de produtividade dos desenvolvedores com a percepção dos líderes de projeto das organizações selecionadas.

Tabela 1. Características das Organizações Selecionadas.

Características	Organização 1	Organização 2	Organização 3
Criada em	*1990*	*1970*	*1998*
Quantidade de Funcionários	*78*	*392*	*260*
Certificações e Avaliações de Qualidade	–	*ISO-9001*	*ISO-9001* *MPS.BR Nível F*
Natureza Social	*Órgão Público*	*Empresa Pública*	*Fundação Privada*
Tipo de Software Desenvolvido	*Sistemas de Informação*	*Sistemas de Informação*	*Sistemas de Informação* *Automação Industrial*
Domínio de Negócio	*Administração Fazendária*	*Educação, Finanças, Data Warehouse, etc.*	*Pesquisa e Inovação Tecnológica*
Tecnologias de Desenvolvimento	*Java*	*Java, Natural, PHP*	*Java, Delphi, Android, iOS*
Plataformas Tecnológicas	*Mainframe* e *Web*	*Mainframe*, *Web e Mobile*	*Mainframe*, *Desktop* e *Web*
Tipo de Processo de Desenvolvimento	*Tradicional*	*Tradicional*	*Tradicional e SCRUM*

A metodologia adotada foi dividida em três fases: Fase 1, revisões da literatura; Fase 2, investigação da percepção, fatores e métricas da produtividade; e Fase 3, investigação dos fatores preponderantes de influência da produtividade em desenvolvedores produtivos das organizações selecionadas. Porém, antes da aplicação metodológica dessas fases, em uma fase que chamamos de *warm-up*, foram realizados alguns estudos em parceria com outros autores que ajudaram na definição do escopo desta pesquisa. Nas próximas subseções, as fases e os estudos serão descritos.

1.5.3.1 *Fase Warm-Up*

Esta fase compreendeu estudos conduzidos durante a definição do escopo deste trabalho, que contribuíram significativamente na definição do contexto e do objetivo desta pesquisa. Os primeiros estudos envolveram o aprendizado e implementação de técnicas de extração e mineração de dados utilizando algoritmos

de *data mining* e aprendizagem de máquina. Tais estudos auxiliaram na implementação de algoritmos de extração da produtividade em repositórios de *software* fornecidos pelas organizações selecionadas.

Um estudo sobre a gestão do conhecimento e cultura organizacional em organizações de *software*, realizado em colaboração com outros pesquisadores, focou na relação entre as práticas de gestão de conhecimentos adotadas pelas cultura de cada organização de *software* [Rabelo *et al.* 2015]. Os resultados desse estudo auxiliaram na compreensão dos fatores técnicos e organizacionais de influência na produtividade do desenvolvedor de organizações de *software*.

Um estudo sobre a influência do gerente de projeto em projetos de *software*, também realizado em colaboração com outros pesquisadores, focou nas influências da personalidade, de acordo com o modelo proposto por Myers (1962), e do papel de equipe, conforme proposto por Belbin (2010), do gerente de projetos sobre o esforço utilizado nos projeto de *software* [Branco *et al.* 2015]. Os resultados desse estudo auxiliaram na compreensão da influência dos fatores humanos na produtividade do desenvolvedor de organizações de software.

1.5.3.2 *Fase 1 – Revisões da Literatura*

As revisões de literatura realizam um resumo do estado da arte na pesquisa de um determinado tópico a partir de dados constantes em publicações da literatura científica e podem ser classificadas como estudos secundários ou terciários. São estudos secundários quando revisam estudos primários, ou seja, novos estudos que respondem a suas questões de pesquisa coletando e analisando novos dados na academia ou na indústria. São estudos terciários quando revisam estudos secundários. Essas revisões são utilizadas frequentemente para encontrar lacunas em áreas de pesquisa ou áreas que requerem confirmação ou estudo mais aprofundado. Além disso, a utilização de revisões de literatura auxiliam os estudantes de graduação e pós-graduação a formar uma base de conhecimento necessária para o bom andamento de suas pesquisas [Kitchenham e Charters 2007].

Essa primeira fase iniciou-se com uma revisão exploratória sobre a produtividade de *software* (Capítulo 2), concluindo com duas revisões sistemáticas (Figura 1): um mapeamento sistemático sobre as métricas de produtividade (Capítulo 3) e uma revisão terciária sobre os fatores de influência da produtividade (Capítulo 4).

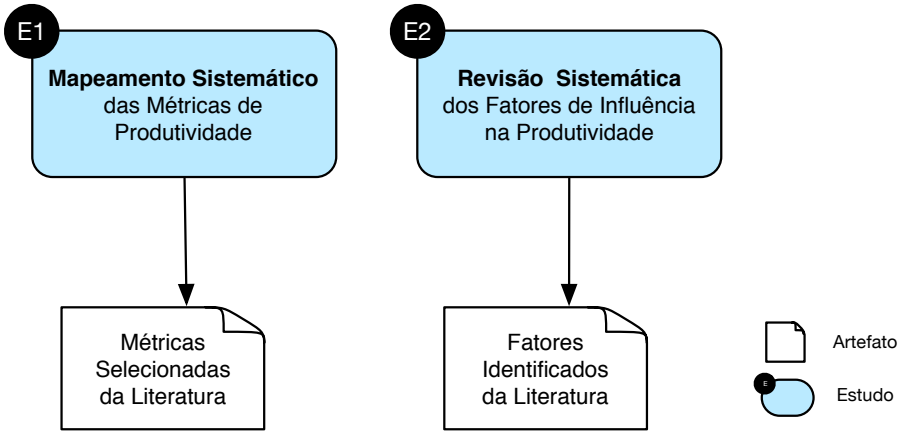

Figura 1. Fase 1 – Revisões sistemáticas da literatura.

O estudo *E1 – Mapeamento Sistemático das Métricas de Produtividade* teve como objetivo identificar na literatura como os pesquisadores da Engenharia de *Software* têm mensurado e analisado a produtividade. As métricas de produtividade identificadas foram extraídas e aplicadas, na Fase 2, para a avaliação da produtividade dos desenvolvedores de organizações de software, em projetos reais, a fim de identificarmos os desenvolvedores mais produtivos.

O estudo *E2 – Revisão Sistemática dos Fatores de Influência na Produtividade* teve como objetivo identificar os fatores de influência na produtividade. Além disso, esses fatores foram classificados como: técnicos, humanos e organizacionais. Essa revisão sistemática foi terciária, pois utilizou como base quatro revisões de literatura existentes, sendo que duas foram identificadas durante a revisão exploratória sobre a produtividade de *software*. Os fatores identificados nesta revisão auxiliaram na análise e codificação das entrevistas realizadas nos estudos qualitativos realizados na Fase 2.

1.5.3.3 *Fase 2 – Percepção, Fatores e Métricas de Produtividade*

A investigação sobre os fatores de influência na produtividade do desenvolvedor foi realizada através de estudos de casos, dentro de seu contexto real, com gerentes e líderes de *software* dentro de projetos reais das organizações selecionadas. Para isso, foram realizados dois estudos qualitativos, sendo um com gerentes de *software* (Capítulo 5) e outro com os líderes de projetos de *software* (Capítulo 6), e um estudo quantitativo sobre as métricas de produtividade (Capítulo 7), aplicando métricas de produtividade nos repositórios de *software* de projetos

reais, relacionados com os líderes entrevistados nas organizações selecionadas (Figura 2).

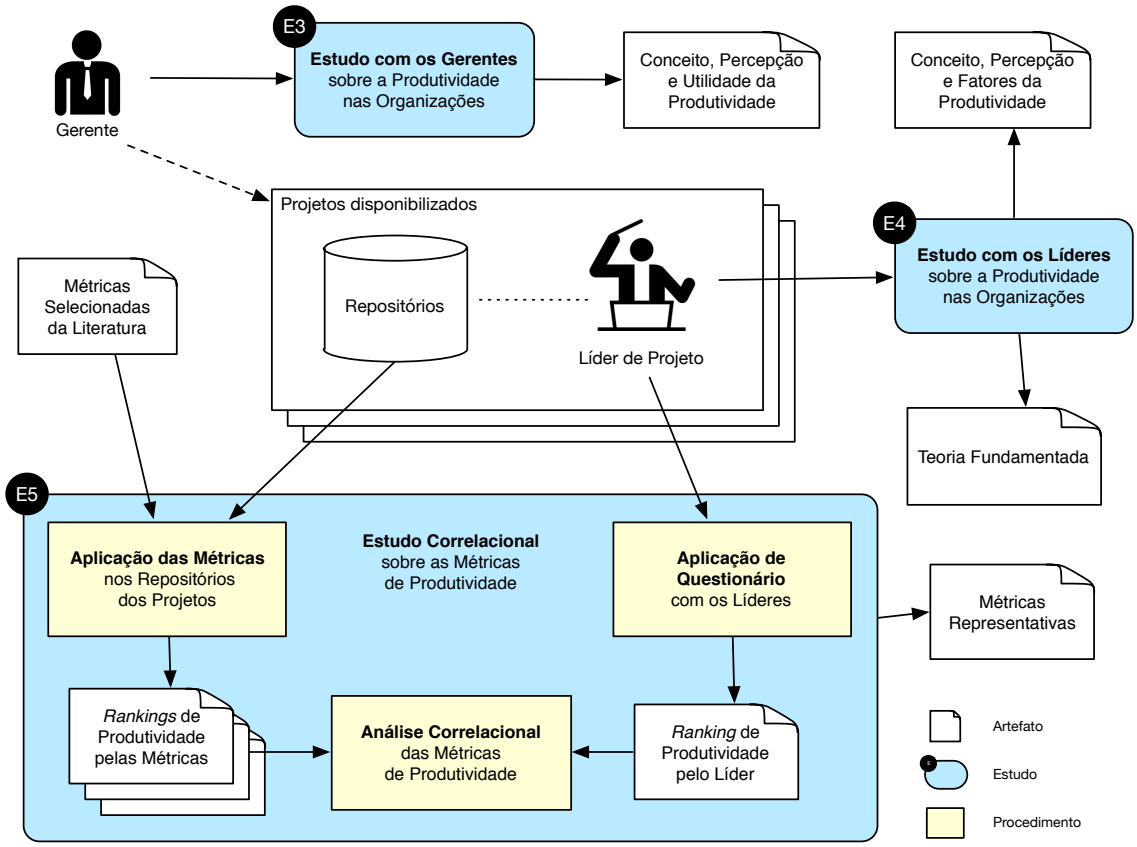

Figura 2. Fase 2 – Conceito, percepção, fatores e métricas de produtividade.

O estudo qualitativo *E3 – Estudo com os Gerentes sobre a Produtividade nas Organizações* teve como objetivo investigar a percepção dos gerentes de *software* das organizações quanto ao conceito, identificação e utilidade da produtividade dos desenvolvedores. Os resultados desse estudo apresentaram o contexto da produtividade dentro de organizações de desenvolvimento de software, esclarecendo qual o conceito de produtividade por eles compreendidos, como eles percebem a produtividade dos desenvolvedores e para o quê utilizam as informações de produtividade dos desenvolvedores. Além disso, a compreensão desses resultados auxiliou no planejamento dos próximos estudos.

O estudo qualitativo *E4 – Estudo com os Líderes sobre a Produtividade nas Organizações* teve como objetivo investigar a percepção dos líderes de projetos de *software* quanto ao conceito, identificação e os fatores de influência da produtividade dos desenvolvedores. Os resultados desse estudo permitiram a comparação entre os conceitos e percepções da produtividade entre os gerentes

(Estudo *E3*) e líderes das organizações. Outro resultado obtido, a partir da comparação entre os dados das diferentes organizações, foi uma teoria fundamentada preliminar sobre os fatores de influência na produtividade dos desenvolvedores de organizações de *software*. Essa teoria também serviu de base de comparação com resultados obtidos no estudo sobre os fatores preponderantes da Fase 3.

O estudo quantitativo *E5 – Estudo Correlacional sobre as Métricas de Produtividade* teve como objetivo principal a identificação dos desenvolvedores considerados mais produtivos nos projetos disponibilizados pelas organizações selecionadas. Utilizando as métricas selecionadas na literatura, aplicáveis aos desenvolvedores nos repositórios de *software* das organizações, foram estudadas as correlações entre os *rankings* gerados pelas métricas selecionadas e os *rankings* fornecidos pelos líderes dos projetos. Os resultados dessas correlações permitiram a identificação de métricas representativas das percepções dos líderes.

1.5.3.4 *Fase 3 – Fatores Preponderantes na Produtividade dos Desenvolvedores*

Esta fase final teve como propósito responder a questão principal desta pesquisa, identificando a preponderância entre os fatores técnicos, humanos e organizacionais. Para isso, um novo estudo qualitativo foi executado junto aos líderes de projeto visando identificar os fatores característicos dos desenvolvedores mais produtivos dos projetos disponibilizados pelas organizações selecionadas (Figura 3).

O estudo qualitativo *E6 – Estudo dos Fatores de Influência na Produtividade dos Desenvolvedores Produtivos* teve como objetivo identificar os fatores de influência da produtividade presentes, segundo a percepção dos líderes de projetos de *software*, nos desenvolvedores considerados produtivos. Os desenvolvedores produtivos foram selecionados a partir da triangulação de todos os *rankings* obtidos pelas métricas e pelo *ranking* informado por cada líder de projeto. Os resultados desse estudo apresentaram um conjunto de fatores característicos dos desenvolvedores produtivos e também outro conjunto de fatores que os distinguia. Além disso, combinando esses resultados com a teoria fundamentada, e considerando ainda o contexto das organizações, foi realizada uma análise comparativa entre os fatores identificados na teoria fundamentada com os fatores

identificados nos desenvolvedores produtivos. Essa comparação fundamentou a resposta para a questão principal de pesquisa desta tese.

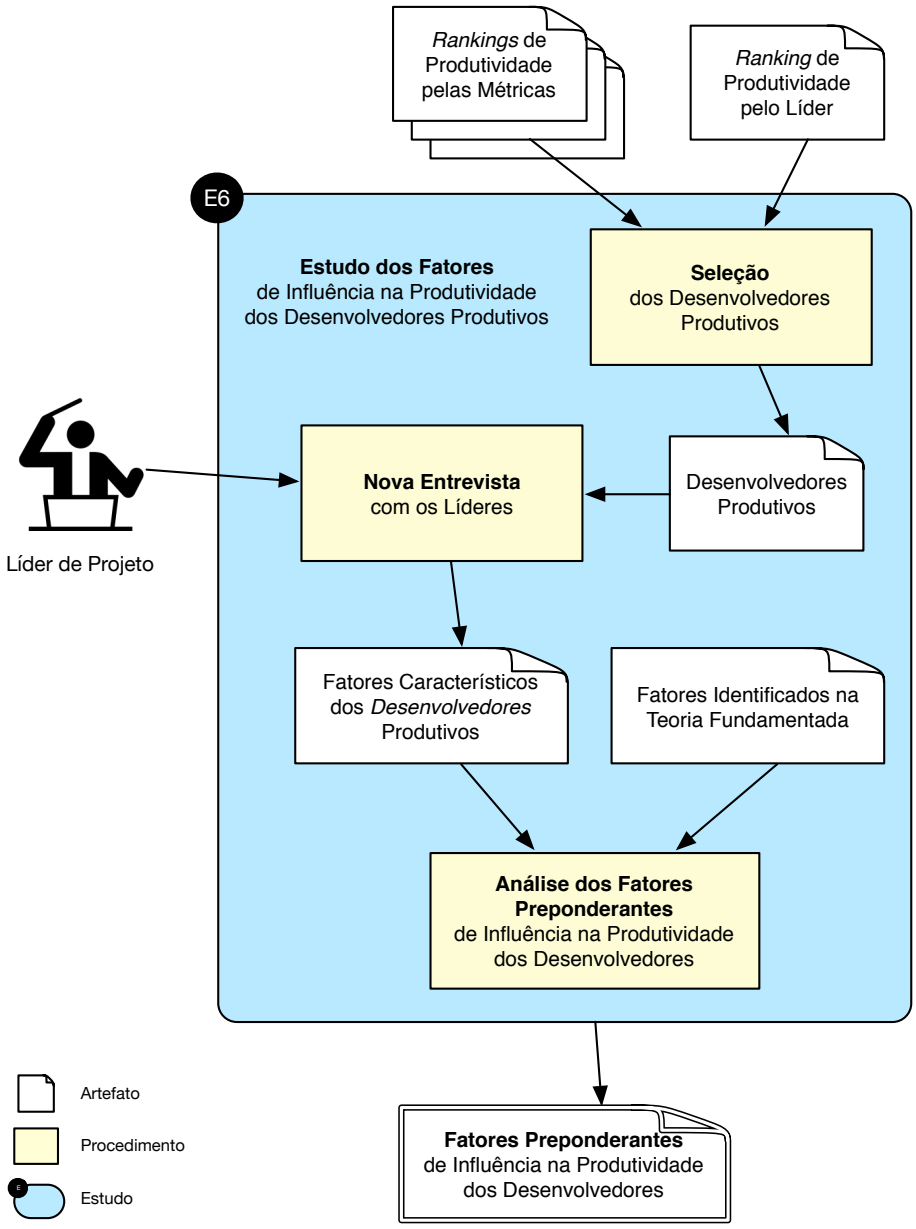

Figura 3. Fase 3 – Fatores preponderantes na produtividade dos desenvolvedores.

A *Análise dos Fatores Preponderantes de Influência na Produtividade dos Desenvolvedores* teve como objetivo realizar uma análise comparativa entre a expectativa dos fatores de influência na produtividade, identificados na teoria fundamentada no estudo com os líderes de projeto, com a realidade, a partir dos fatores de influência identificados nos desenvolvedores considerados produtivos em seus projetos. Essa comparação permitiu avaliar se a expectativa dos líderes quanto aos fatores de influência estavam presentes ou não nos desenvolvedores produtivos

de suas equipes, ou se novos fatores caracterizavam esses desenvolvedores. A partir dessa análise comparativa, avaliando todos os casos investigados, identificou-se os fatores que preponderaram tanto nas expectativas dos líderes, quanto nos desenvolvedores mais produtivos dos projetos de cada organização, com suporte em estudos realizados sobre fatores a partir da literatura científica.

1.6 PRINCIPAIS CONTRIBUIÇÕES

As principais contribuições desta tese de doutorado são descritas a seguir:

- Identificação, classificação e análise das métricas de produtividade de *software* existentes na literatura, utilizadas pelos pesquisadores na Engenharia de *Software*, conforme mapeamento sistemático realizado;
- Identificação e classificação em técnicos, humanos ou organizacionais de fatores de influência na produtividade dos desenvolvedores de *software*, conforme revisão terciária realizada;
- Identificação dos conceitos, percepções e utilidade da produtividade dos desenvolvedores de organizações de *software*, conforme estudo qualitativo realizado com gerentes de organizações de *software*;
- Identificação de uma teoria fundamentada sobre os fatores de influência na produtividade dos desenvolvedores de organizações de *software*, conforme estudo qualitativo realizado com líderes de projeto de *software*;
- Identificação de métricas de produtividade do desenvolvedor representativa das percepções da organização, conforme estudo quantitativo com métricas de produtividade nos repositórios de software, disponibilizados pelas organizações, com os líderes de projeto;
- Identificação de fatores característicos em desenvolvedores produtivos das organizações de *software*, conforme um segundo estudo qualitativo realizado com líderes de projeto;
- Identificação de fatores preponderantes na produtividade dos desenvolvedores, conforme análise de todos os estudos realizados, respondendo à questão de pesquisa desta tese.

1.7 ORGANIZAÇÃO DA TESE

Esta tese de Doutorado está organizada em mais oito capítulos, além deste primeiro capítulo de introdução, que apresentou o contexto, a motivação e o problema investigado nesta pesquisa de Doutorado. A organização do restante do texto desta qualificação segue a estrutura abaixo:

- **Capítulo 2 – Produtividade de Software:** Descreve os principais conceitos relacionados à produtividade em geral, seu sentido econômico e sua aplicação na Engenharia de *Software*. Este capítulo apresenta também conceitos sobre medição e métricas de produtividade e a definição dos fatores técnicos, humanos e organizacionais utilizados nesta pesquisa.
- **Capítulo 3 – Mapeamento Sistemático sobre Métricas de Produtividade de *Software*:** Apresenta o estudo *E1 – Mapeamento Sistemático das Métricas de Produtividade* realizado com o objetivo de identificar métricas de produtividade utilizadas pelos pesquisadores na literatura. Este capítulo apresenta também a seleção das métricas de produtividade dos desenvolvedores que foram utilizadas na segunda fase desta pesquisa.
- **Capítulo 4 – Revisão Terciária sobre Fatores de Influência na Produtividade de *Software*:** Apresenta o estudo *E2 – Revisão Sistemática dos Fatores de Influência na Produtividade* realizado com o objetivo de identificar os fatores de influência da produtividade, a partir da extração de quatro revisões sistemáticas encontradas na literatura. Este capítulo apresenta também a classificação desses fatores em técnicos, humanos e organizacionais.
- **Capítulo 5 – Percepção da Produtividade do Desenvolvedor pelos Gerentes de *Software*:** Descreve o *E3 – Estudo com os Gerentes sobre a Produtividade nas Organizações* nas organizações selecionadas com o objetivo de investigar a percepção dos gerentes de *software* das organizações quanto ao conceito, identificação e utilidade da produtividade dos desenvolvedores.

- **Capítulo 6 – Percepção da Produtividade do Desenvolvedor pelos Líderes de Projetos de *Software*:** Descreve o estudo *E4 – Estudo com os Líderes sobre a Produtividade nas Organizações* com o objetivo de investigar a percepção dos líderes de projetos de *software* quanto ao conceito, identificação e os fatores de influência da produtividade dos desenvolvedores.
- **Capítulo 7 – Métricas de Produtividade do Desenvolvedor de Organizações de *Software*:** Descreve o estudo *E5 – Estudo Correlacional sobre as Métricas de Produtividade* com o objetivo principal a identificação dos desenvolvedores considerados mais produtivos nos projetos disponibilizados pelas organizações selecionadas. Este capítulo apresenta também a representatividade do conjunto de métricas selecionadas no mapeamento sistemático, utilizando correlações estatísticas, avaliadas a partir da percepção dos líderes de projeto das organizações.
- **Capítulo 8 – Fatores de Influência dos Desenvolvedores Produtivos de *Software*:** Descreve o estudo *E6 – Estudo dos Fatores de Influência dos Desenvolvedores Produtivos* com o objetivo de identificar os fatores de influência da produtividade presente, segundo a percepção dos líderes de projetos de *software*, nos desenvolvedores considerados produtivos, segundo os *rankings* fornecidos pelos líderes e gerado pelas métricas de produtividade aplicadas. Além disso, também está descrito a *Análise dos Fatores Preponderantes de Influência na Produtividade dos Desenvolvedores*, comparando a expectativa dos líderes, utilizando os fatores identificados na teoria fundamentada, com as realidade, nos projetos em que lideram, utilizando os fatores identificados a partir dos desenvolvedores considerados produtivos em seus projetos.
- **Capítulo 9 – Conclusões e Perspectivas Futuras:** Por fim, contém as conclusões e contribuições desta pesquisa, além de indicar possibilidades de trabalhos futuros.

Capítulo 2 – PRODUTIVIDADE DE *SOFTWARE*

Este capítulo apresenta um referencial teórico desta pesquisa, mostrando os principais conceitos relacionados à produtividade de software, seu sentido econômico e sua aplicação na Engenharia de Software. Além disso, são apresentadas também os conceitos sobre medição e métricas de produtividade e a definição dos fatores técnicos, humanos e organizacionais utilizados nesta pesquisa.

2.1 INTRODUÇÃO

A palavra produtividade[1] é um substantivo derivado do adjetivo produtivo[2]. Produtivo é aquele que produz, que dá ou produz lucro. Esse termo tem origem no latim *productivus*, e que significa fértil, rendoso, proveitoso, profícuo [Rezende 2005] e também significa a ideia de produto trazido à luz[3]. Portanto, nesse sentido original, o termo produtividade traz a ideia de alguém que tem a capacidade em construir algo que dá ou produz lucro. Essa capacidade remete à ideia da virtude de tornar algo real, que, por sua vez, é um sentido figurado da palavra eficácia[4].

A produtividade, em seu sentido econômico, que pode ser creditado a *François Quesnay* [Quesnay 1766] de acordo com Hernández-López et al. (2013), é a relação entre o que é produzido e os meios ou recursos aplicados nessa produção. Essa relação é medida em termos da razão de saída (montante produzido) por unidade de entrada (recursos aplicados). Esse sentido econômico da produtividade remete a uma ideia mais associada com o significado de eficiência[5], que em uma de suas acepções é a qualidade de algo ou alguém que produz com o mínimo de erros ou de meios.

O sentido econômico de produtividade é largamente utilizado em diversas áreas, desde a agricultura até a economia [Aquino Junior e Meira 2009] e, como será apresentado mais adiante, também muito utilizado na Engenharia de *Software*.

---

[1] https://www.priberam.pt/DLPO/produtividade
[2] https://www.priberam.pt/DLPO/produtivo
[3] http://www.oxforddictionaries.com/definition/english/productive
[4] https://www.priberam.pt/DLPO/eficácia
[5] https://www.priberam.pt/DLPO/eficiência

Esse sentido econômico é bem aplicável na indústria baseada no paradigma de manufatura pois, nesse caso, o montante produzido é baseado em quantidades padronizadas e unidades de medida claramente definidas [Hernández-López et al. 2013], com métodos projetados e testados para determinação da produtividade [Dalcher 2006].

Para a indústria baseada em serviços, como a Engenharia de *Software*, onde os produtos são normalmente intangíveis, esse conceito já não se aplica tão adequadamente [Hernández-López et al. 2013], já que as quantidades produzidas e as medidas do que foi produzido não estão claramente definidas. Além disso, o maior custo na Engenharia de *Software* está no processo de desenvolvimento do *software* [Tomaszewski e Lundberg 2005] do que no processo de manufatura, pois as organizações de *software* normalmente desenvolvem novos produtos de *software* ao invés de replicá-los muitas vezes [Trendowicz e Münch 2009]. Ademais, os processos de desenvolvimento de *software* parecem ser significativamente mais difíceis do que os processos de produção em outras áreas [Abdel-Hamid 1996; Briand e Wieczorek 2002]. Como exemplo, um dos desafios clássicos da indústria de *software* é a entrega dos projetos de *software* dentro do prazo acordado com o cliente [Dale e Van der Zee 1992].

O ambiente competitivo existente no mercado atualmente requer que as organizações aumentem o seu nível de qualidade e reduzam os seus custos de produção, sendo que e a principal forma de redução de custos no desenvolvimento de *software* é através do aumento da produtividade [Aquino Junior e Meira 2009]. Por isso, o uso da medição da produtividade para a avaliação da eficiência da entrega dos projetos não deve ser considerada uma surpresa [Gummesson 1992]. Porém, a complexidade do desenvolvimento de *software* implica na dificuldade de medição da produtividade [Anselmo et al. 2003] e, apesar da produtividade ter sido estudada intensamente [Trendowicz e Münch 2009], ela ainda continua a ser um desafio para toda a indústria de *software* [Hernández-López et al. 2011].

2.2 PRODUTIVIDADE NA ENGENHARIA DE *SOFTWARE*

As primeiras pesquisas científicas envolvendo o conceito de produtividade na Engenharia de *Software* foram publicadas por volta do início da década de 80 [Albrecht 1979; Basili e Freburger 1981; Chrysler 1978; Jeffery e Lawrence 1981].

O foco dessas pesquisas pioneiras foi voltado para as atividades de programação, envolvendo temas como medição da produtividade [Albrecht 1979; Jeffery e Lawrence 1981] e da busca por fatores que influenciassem a produtividade em projetos de *software* [Basili e Freburger 1981; Chrysler 1978].

Foi durante a década de 90 que houve um aumento significativo na quantidade de pesquisas sobre a produtividade de *software*. As pesquisas relativas à produtividade na programação continuaram [Humphrey e Singpurwalla 1991; Moser e Nierstrasz 1996], mas houve também novos focos de pesquisa, como estudos sobre a produtividade da manutenção de *software* e seus fatores [Gill e Kemerer 1991] e a avaliação do impacto da reutilização de *software* na produtividade [Deng-Jyi e Lee 1993; Lim 1994]. Outras pesquisas não mensuraram a produtividade diretamente, mas utilizaram banco de dados de medições de produtividade coletadas na indústria para avaliação da produtividade de projetos [Maxwell *et al.* 1996; Morisio *et al.* 1999]. Por fim, houve pesquisas com o foco voltado especificamente para métricas da produtividade de *software* [Chatman 1995; Rothenberger e Dooley 1999].

A partir do ano 2000, as pesquisas envolvendo a produtividade exploraram os novos métodos e técnicas de desenvolvimento de *software*, tais como a avaliação da produtividade na programação em pares [Lui e Chan 2006] e o impacto do *refactoring*[6] na qualidade e produtividade de um time ágil [Moser *et al.* 2008]. Outras pesquisas investigaram a produtividade e seus fatores em diferentes contextos, tais como a produtividade em projetos de *Enterprise Resource Planning (ERP)* [Stensrud e Myrtveit 2003], a produtividade no desenvolvimento *open-source* [Wray e Mathieu 2008] e a produtividade de equipes trabalhando continuamente em fusos horários distintos [Colazo 2008]. Além disso, diversas pesquisas focaram em novas métricas da produtividade de *software*, seja utilizando combinações de múltiplas medições do *software* [Kitchenham e Mendes 2004], avaliando a relação entre as entradas e saídas do desenvolvimento de *software* utilizando uma técnica de análise envoltória de dados (*DEA*) [Yang e Paradi 2009] ou minerando informações dos repositórios de *software* das organizações [Huang *et al.* 2011].

[6] *Refactoring* é uma técnica para melhoria do projeto de uma base de código já existente.

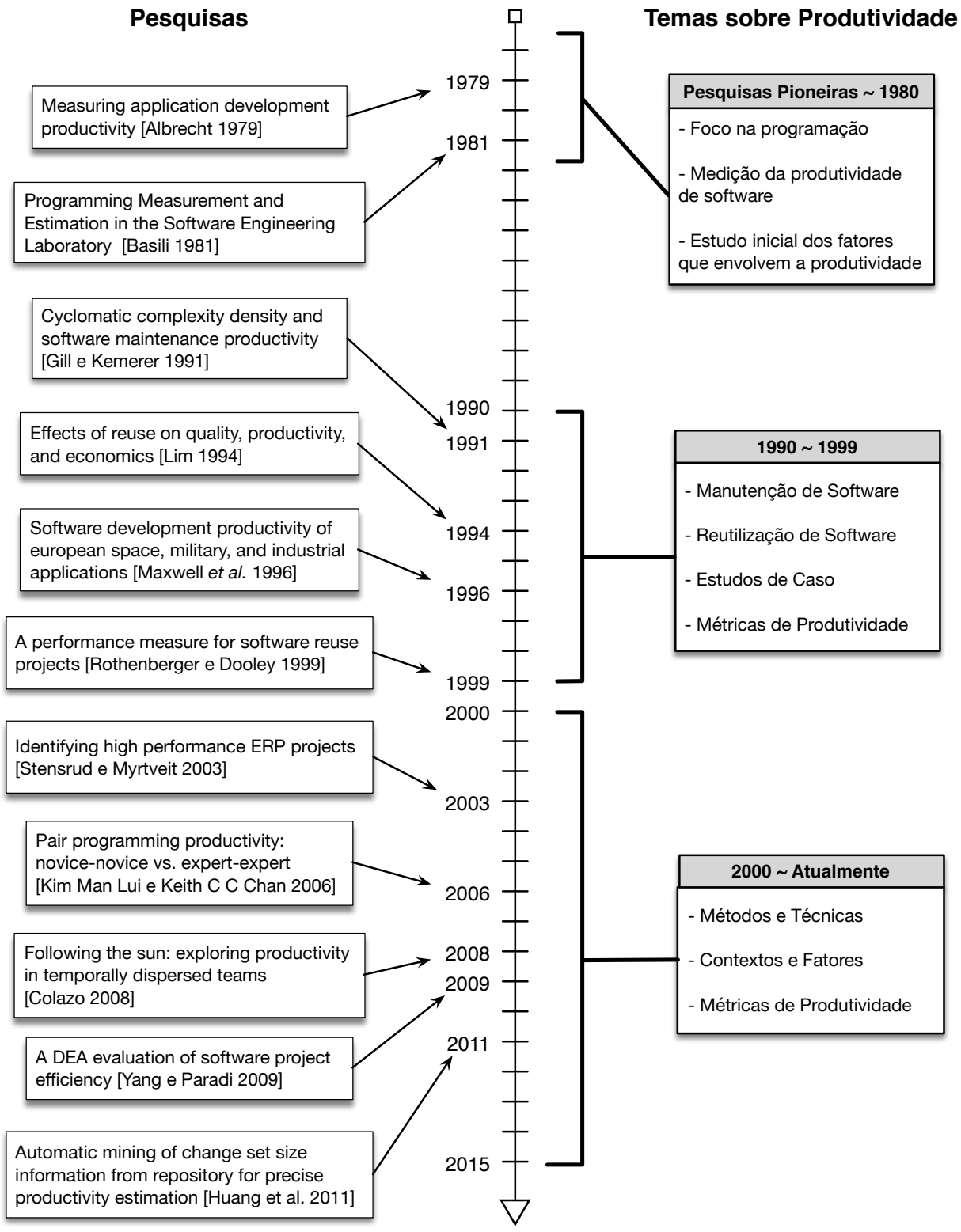

Figura 4. Linha do tempo das pesquisa sobre a produtividade.

A Figura 4 apresenta uma linha do tempo com as pesquisas mencionadas e um resumo dos principais temas citados. Cheikhi *et al.* (2012) afirmam que a pesquisa científica sobre a produtividade de *software* está direcionada para dois

tópicos principais: (i) a proposição de métricas para medição da produtividade e (ii) a identificação do contexto e os fatores que impactam a produtividade. Estes dois tópicos são detalhados a seguir.

2.3 MÉTRICAS DE PRODUTIVIDADE

A medição de *software* permite obter informações sobre os objetos e eventos escolhidos, tornando-os compreensíveis e controláveis [Fenton e Pfleeger 1998]. As medições também são essenciais para a realização de melhorias em processos de desenvolvimento de *software*, porque fornecem dados objetivos que permitem conhecer o seu desempenho [Rocha *et al.* 2012]. Portanto, a medição de *software* torna possível a compreensão de qualquer aspecto do *software* para um controle mais efetivo e um acompanhamento mais eficaz.

A medição de *software* tem duas finalidades básicas: estimar ou avaliar. Medir para estimar é prever o valor de certas variáveis antes do desenvolvimento do *software*, tais como o custo, o tempo e o esforço necessário, o que permite definir e alocar os recursos necessários para a sua execução [Fenton e Pfleeger 1998]. Medir para avaliar é coletar variáveis durante e após o desenvolvimento de um *software*. Isto possibilita entender, controlar e melhorar o que acontece durante e após o processo desenvolvimento de *software*. Em qualquer uma das finalidades, medir é necessário e o corolário proposto por Fenton e Pfleeger (1998), baseado em DeMarco (1986), continua válido – "você não pode nem estimar nem controlar aquilo que você não consegue medir".

A medição da produtividade é, portanto, necessária para avaliar a eficiência do seu processo produtivo ou de seus desenvolvedores [Hernández-López *et al.* 2012] e uma ferramenta efetiva de comparação do desempenho entre diferentes projetos e desenvolvedores [Cheikhi *et al.* 2012]. Para que a medição da produtividade seja realmente efetiva, ela deve estar alinhada com o objetivos da organização [Rocha *et al.* 2012]. Dessa forma, dependendo do objetivo que se tenha com a medição da produtividade, diferentes métricas podem ser utilizadas.

Uma organização pode definir sua própria métrica de produtividade, utilizando métodos de apoio como o GQM (*Goal-Question-Metric*) [Basili *et al.* 1994] ou o PSM (*Practical Software Measurement*) [McGarry *et al.* 2002]. Ela ainda pode utilizar uma das diferentes métricas de produtividade propostas em vários estudos

presentes na literatura científica. Órgãos internacionais como a *International Organization for Standardization* (ISO) e o *Institute of Electrical and Electronics Engineers* (*IEEE*) definiram padrões para a medição da produtividade aplicado ao desenvolvimento de *software*. A ISO publicou o padrão ISO 9126, onde define a produtividade como um fator de qualidade do produto de *software*. Em uma de suas métricas, a produtividade das pessoas é definida como a eficácia pelo esforço, onde a eficácia pode ser medida pelas tarefas executadas e o esforço pelo tempo empregado nessas tarefas. O *Institute of Electrical and Electronics Engineers* (*IEEE*) publicou o padrão IEEE Std. 1045/1992 para métricas de produtividade de *software*, onde define a produtividade de *software* como o relacionamento de uma saída primitiva (*source statements, function points* ou documentos) com a sua entrada primitiva correspondente (*effort*, p.e., homem-hora). Segundo Card (2006), os relatórios técnicos do *Software Engineering Institute* (*SEI*) discutem como definir as métricas base de esforço e tamanho para o *software*, enquanto o padrão IEEE sugere métodos de combinação destas métricas base para a derivação da métrica de produtividade.

Fenton e Pfleeger (1998), em seu livro *Software Metrics*, afirmam que a métrica de produtividade de *software* é uma medida indireta dos recursos utilizados no desenvolvimento de *software*, baseando-se em atributos do processo adotado e do produto resultante. Os recursos utilizados no processo são as entradas e o montante produzido são as saídas do processo. Cheikhi *et al.* (2012) afirmam que a métrica *de facto* da produtividade *software* é a razão entre o tamanho do *software* pelo esforço despendido. Nesse caso, o tamanho do *software* é o montante produzido e o esforço despendido são os recursos aplicados.

De forma geral, tanto na literatura sobre produtividade de *software* quanto nos padrões internacionais, existem diferentes pontos de vista sobre as métricas de produtividade desenvolvidas pelos praticantes e pesquisadores de diversas organizações [Cheikhi *et al.* 2012]. As definições acima apresentadas são derivações baseadas no sentido econômico de produtividade, diferindo apenas na escolha dos atributos que representam os recursos aplicados e o montante produzido. Porém, além dessas escolhas, existem outras igualmente pertinentes que devem ser consideradas para a definição de uma métrica de produtividade.

2.3.1 Estrutura das métricas de produtividade

A equação tradicionalmente utilizada [Melo *et al.* 2011] em diversos estudos sobre produtividade na literatura científica é a razão entre o tamanho do *software* produzido e o esforço empregado:

$$Produtividade = Tamanho/Esforço \quad (1)$$

Para aplicar esta equação (1) é preciso decidir como será medido o tamanho do *software* e também como será medido o esforço. A escolha de uma unidade baseada no tempo para o esforço, como um recurso aplicado de entrada para a produtividade, é bem frequente [Meyer *et al.* 2014], pois o mesmo é considerado como um fator crucial para a definição da produtividade de *software* [Hernández-López *et al.* 2013]. Porém, outros recursos também podem ser importantes, já que não se sabe com certeza qual é o conjunto total dos recursos empregados a serem considerados para a produtividade de *software*, nem como as mudanças do processo afetam o relacionamento entre estes recursos e o *software* produzido [Fenton e Pfleeger 1998].

A definição da saída para a métrica de produtividade ainda é de mais difícil solução [Aquino Junior e Meira 2009], pois o significado e a quantificação do montante produzido do desenvolvimento de *software* em termos de tamanho, complexidade ou valor para o cliente é um problema muito complexo [Boehm 1987]. Hernández-López *et al.* (2013) afirmam ainda que a escolha do tamanho é problemática devido à falta de alguns componentes, já que a métrica da produtividade deve distinguir os fatores, as entradas e as saídas que impactam na produtividade de *software*.

Portanto, apesar dessa métrica tradicional de produtividade ser simples, ela pode não contemplar componentes que possam ser importantes para a medição da produtividade. Card (2006) discutiu o desafio da medição da produtividade de *software* e apresentou os componentes básicos a serem considerados e as suas relações para a definição de uma métrica de produtividade (Figura 5). Devem ser considerados como componentes básicos para um processo, ou subprocesso, os requisitos, os recursos aplicados e seu custo, como entradas; para as saídas devem

ser considerados o próprio *software*, ou algum artefato intermediário, e o valor, que segundo o autor é normalmente o custo para o cliente.

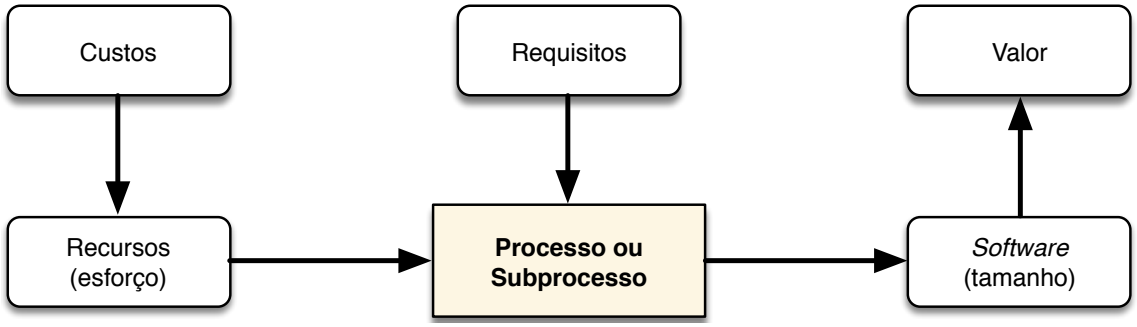

Figura 5. Componentes básicos de uma métrica de produtividade [Card 2006].

Segundo Fenton e Pfleeger (1998), para realizar a escolha desses atributos, é preciso distinguir se o interesse está, por exemplo, em medir a produtividade do processo ou dos recursos envolvidos, tais como a equipe ou o desenvolvedor. Além disso, é necessário identificar os fatores (entradas e saídas) que serão incluídos na métrica de produtividade [Wagner e Ruhe 2008] e qual abordagem matemática será utilizada na combinação desses fatores.

$$Produtividade_{abstração} = f_{abordagem\_quantitativa}(entradas, saídas) \quad (2)$$

A equação (2) acima apresenta a produtividade de *software* como uma função baseada em uma *abordagem quantitativa*, seja como uma razão simples ou ponderada da relação entre as *entradas* e *saídas* e definida para uma *abstração,* também chamado de unidade de análise ou nível de medição. As entradas e saídas estão no plural porque podem existir mais de uma entrada ou saída na definição da métrica de produtividade. Neste caso, chamamos de dimensionalidade a quantidade de variáveis utilizadas para medir a entrada e a quantidade de variáveis utilizadas para medir a saída.

A abstração (Tabela 2) descreve a entidade e em qual nível as medições da produtividade serão realizadas [Petersen 2011]. Dependendo do objetivo, a abstração pode ser a organização, projeto e desenvolvedor, de acordo com Hernández-López *et al.* (2012). Outro nível seria a equipe de desenvolvimento, mas este nível é equivalente ao nível de projeto. Petersen (2011) descreve também essas

abstrações, acrescentando a tarefa e o processo. Por fim, também incluímos o módulo, pois ele é citado em nosso mapeamento sistemático (Capítulo 3).

Tabela 2. Abstrações mensuradas na produtividade de *software*.

Abstração	Descrição
Organização	A produtividade de toda a organização, ou de um local de desenvolvimento da organização, é mensurada
Projeto	A produtividade de um projeto de *software* é mensurada
Módulo	A produtividade de uma parte do *software* desenvolvido ou em manutenção é mensurado
Processo	A produtividade do processo de desenvolvimento ou manutenção de *software* é mensurado
Desenvolvedor	A produtividade de um engenheiro de *software* que participa do desenvolvimento ou manutenção do *software* é mensurada
Tarefa	A produtividade de uma das tarefas específicas de um engenheiro de *software* dentro do desenvolvimento de *software* é mensurada

A abordagem quantitativa (Tabela 3) de uma métrica de produtividade diz respeito aos diferentes métodos que relacionam as entradas e saídas a fim de capturar a produtividade [Petersen 2011]. O método mais simples e mais frequente encontrado é a razão matemática entre as entradas e as saídas, embora outras relações sejam propostas nos estudos sobre a produtividade na literatura científica.

Tabela 3. Abordagens quantitativas utilizadas nas métricas de produtividade.

Abordagem quantitativa	Descrição
Razão simples	Esta abordagem utiliza a razão matemática entre as entradas e saídas, podendo ser aplicada com métricas unidimensionais ou multidimensionais. $Produtividade_{abstração} = \frac{Saída_1 + Saída_2 + \cdots + Saída_n}{Entrada_1 + Entrada_2 + \cdots + Entrada_m}$
Razão ponderada	É uma variação da razão simples onde para cada variável de entrada e saída é atribuído um peso. $Produtividade_{abstração} = \frac{a_1 \cdot Saída_1 + b_2 \cdot Saída_2 + \cdots + c_n \cdot Saída_n}{x_1 \cdot Entrada_1 + y_1 \cdot Entrada_2 + \cdots + z_m \cdot Entrada_m}$

Análise envoltória de dados (DEA)	É uma abordagem capaz de determinar uma fronteira de produção entre um conjunto de unidades de decisão, tais como os projetos, os desenvolvedores e os processos. Colocando as unidades mais produtivas como referência para as outras unidades é possível estabelecer a distancia entre elas, mostrando, por exemplo, o quanto uma unidade de decisão pode melhorar.
Padrão Estatístico	É uma abordagem baseada em padrões estatísticos, utilizando, por exemplo, algoritmos de aprendizagem de máquina. Esses algoritmos são utilizados para classificar a produtividade de uma nova avaliação com base em exemplos disponíveis em uma base de dados histórica. Dessa forma, ao se apresentar um novo exemplo para o algoritmo, o mesmo é capaz de classificar a categoria de produtividade ao que o novo exemplo pertence.

Os recursos aplicados, ou entradas, no processo de desenvolvimento de *software* são amplamente reconhecidos e relativamente fáceis de serem determinados [Card 2006]. Frequentemente a unidade da métrica de entrada é baseada no tempo [Hernández-López *et al.* 2013; Meyer *et al.* 2014]. De acordo com Boehm (1987), o uso do tempo como a única entrada pode ser devido a duas considerações: (i) a entrega do projeto no prazo combinado é considerada um fator de sucesso do projeto, e (ii) o maior custo para as organizações que desenvolvem projetos de *software* está relacionado com o tempo de esforço dedicado pela equipe de desenvolvimento. A unidade da métrica de entrada baseada no tempo depende do nível de medição desejado pois, para um único indivíduo, ela é uma unidade do tempo como anos, dias, meses, horas, etc.; para uma equipe de desenvolvedores, a métrica baseada no tempo normalmente utilizada é a de esforço, tais como homem-hora, homem-mês e demais variações.

Trendowicz e Münch (2009) consideram os custos de desenvolvimento como uma derivação da produtividade de *software* e, por isso, não o tratam de forma diferente do esforço. Os custos como métrica têm problemas com relação a questões econômicas, tais como a inflação ou reajustes salariais, que não envolvem o desenvolvimento de *software*. Por isso, o custo não é adequado para ser utilizado como métrica de entrada, especialmente em determinadas situações onde tais efeitos externos sejam um forte fator relevante.

A abordagem mais frequente para medir a saída dos processos de desenvolvimento de *software* é o tamanho do *software* [Card 2006]. Os dois métodos mais comuns para medição do tamanho do *software* são o baseado em pontos por

função (*Function Points – FP*) e o baseado em linhas de código (*Lines of Code – LOC*) [Card 2006; Hernández-López *et al.* 2013].

A análise por pontos por função (*FP*) é uma métrica baseada na quantidade de funcionalidades a serem desenvolvidas para o *software*. Essa técnica foi originalmente desenvolvida para fazer estimativas com base em informações disponíveis no início do ciclo de desenvolvimento de *software* [Card 2006]. A partir da análise da complexidade dos artefatos, uma medida em pontos é definida. Diferentes algoritmos para contagem de pontos foram desenvolvidos e esta abordagem continua a apresentar um alto grau de subjetividade [Card 2006].

A medição do tamanho do *software* baseado na quantidade de linhas de código (*LOC*) é a métrica mais amplamente utilizada em estudos envolvendo a produtividade de *software* [Aquino Junior e Meira 2009]. Uma de suas limitações é que só pode ser medida com confiança após o término do projeto [Card 2006]. Não há uma definição precisa sobre o que deve ser considerado ou desconsiderado para contagem de linhas nessa métrica (ex. comentários e linhas em branco) e, por isso, uma posterior análise e comparação pode se tornar mais difícil [Aquino Junior e Meira 2009].

As medidas baseadas em *FP* e *LOC* utilizam diferentes estratégias para medir o tamanho do *software* e, apesar disso, tendem a ser altamente correlacionadas [Banker *et al.* 1991; Laranjeira 1990]. Porém, Gencel e Demirors (2008) afirmam que essa suposição da correlação, ou relação linear entre essas medidas, é imprecisa. As medições do tamanho do *software* baseadas em *FP* e *SLOC* apresentam diversas limitações e, mesmo assim, são muito populares nas organizações [Asmild *et al.* 2006].

2.3.2 Métricas de produtividade do desenvolvedor de *software*

Com relação às métricas de produtividade do desenvolvedor de *software*, é possível destacar informações de duas revisões sistemáticas recentes. A primeira, realizada por Petersen (2011) sobre métricas de produtividade de *software*, que somente considerou artigos nos quais as métricas de produtividade foram avaliadas por experimento, estudo de caso ou até prova de conceito. Dos 38 artigos selecionados, somente cinco possuíam métricas de produtividade do desenvolvedor. Destes cinco, três eram métricas de estimativa e somente dois estudos eram métricas

de avaliação da produtividade. Sendo ainda que, nesses dois últimos estudos, as métricas foram avaliadas através de uma prova de conceito.

A revisão sistemática de Hernández-López *et al.* (2013) focou em métricas para a produtividade no nível individual, considerando os diversos papéis desempenhados no processo de desenvolvimento de *software*. Dos seis estudos selecionados, as métricas de produtividade encontradas foram *linhas de código (LOC)/Tempo* e *Quantidade de Tarefas/Tempo*. Nesses estudos, os autores relataram que tiveram dificuldade de determinar para quais papéis dentro do desenvolvimento as métricas estavam se referindo, pois os papéis não estavam claramente descritos no texto dos artigos. Mesmo assim, os autores concluíram que a métrica baseada na quantidade de tarefas executadas pode ser aplicada a qualquer papel dentro do processo de desenvolvimento de *software*, enquanto que a métrica baseada em linhas de código somente pode ser aplicável a papéis com tarefas de programação.

Portanto, essas duas revisões sistemáticas recentes demonstram que existem poucos estudos focados na produtividade do desenvolvedor. Do total de 11 estudos selecionados com foco no indivíduo, somente duas métricas foram extraídas, relacionadas com avaliação de produtividade.

2.4 FATORES DE INFLUÊNCIA NA PRODUTIVIDADE

A melhoria da produtividade no processo de desenvolvimento de *software* é o principal meio de tornar a organização de *software* mais competitiva [Aquino Junior e Meira 2009]. Portanto, a escolha das práticas efetivas para o aumento da produtividade deve ser baseada nos fatores que tenham maior influencia sobre a produtividade. Os gestores estão mais conscientes da importância dos fatores que influenciam a produtividade da equipe envolvida nos projetos de *software* [Melo *et al.* 2011]. Porém, o impacto na produtividade de *software* desses fatores pode ser diferente dependendo do contexto e das características do desenvolvedor, da equipe, do projeto e de toda a organização [Trendowicz e Münch 2009].

Segundo Hernández-López et al. (2013), muitos dos fatores que influenciam a produtividade de *software* são conhecidos e utilizados em modelos de estimativas como o COCOMO [Boehm 1981] e o COCOMO II [Boehm *et al.* 1995]. Contudo, não está claro se a importância dos fatores identificados se modificou com o tempo, já que os processos e as ferramentas evoluíram consideravelmente desde os estudos

iniciais [Hernández-López et al. 2013]. Outro problema é que existem possivelmente centenas de fatores que influenciem a produtividade e considerá-los todos para uma análise não faria sentido do ponto de vista econômico [Trendowicz e Münch 2009]. Dessa forma, deve-se focar em um número limitado de fatores, que possuam maior impacto na produtividade da organização de *software*.

Wagner e Ruhe (2008) realizaram uma revisão sistemática e agruparam 51 fatores em 9 categorias. Os autores tinham como objetivo identificar na literatura os principais fatores que influenciam a produtividade de *software*, com uma especial consideração para os fatores humanos. Trendowicz e Münch (2009) encontraram 246 fatores diferentes em sua revisão sistemática de literatura. Essa revisão avaliou publicações científicas assim como também avaliou a experiência da indústria sobre a produtividade de *software*. Sampaio *et al.* (2010), em outra revisão de fatores de produtividade, encontraram 39 fatores, mas somente relataram 20 fatores que tiveram pelo menos 4 citações. O objetivo dos autores foi fornecer uma visão consolidada dos fatores de produtividade mais citados ao longo dos anos, junto com as estratégias para conduzir esses fatores.

Portanto, por essas revisões sistemáticas e pela quantidade de fatores encontrados, é possível perceber que uma considerável quantidade de pesquisa foi direcionada para a identificação de fatores que tenham um impacto significativo na produtividade do desenvolvimento de *software*. Apesar dessa quantidade de estudos sobre os fatores de influência na produtividade, as organizações de *software* ainda não conhecem quais são os fatores mais significantes [Sampaio *et al.* 2010]. Além disso, cada organização deve considerar os fatores de produtividade de seu próprio ambiente, ao invés de adotar, sem nenhuma avaliação, os fatores utilizados em outros contextos [Trendowicz e Münch 2009]. Assim sendo, as organizações podem selecionar os fatores existentes na literatura, além de identificar alguns próprios, e avaliar a influência desses fatores dentro de seu próprio contexto.

2.4.1 Fatores da produtividade do desenvolvedor de *software*

O desenvolvedor de software é peça fundamental para todo projeto de desenvolvimento, pois assume um papel crítico [Amrit *et al.* 2014]. As pessoas são o que fazem as organizações tão complexas e diferentes. Assim, ao não considerar o fator humano, as metodologias convencionais de desenvolvimento de *software*, não

se dirigem a organizações reais [Avison *et al.* 1999]. Além disso, o desenvolvimento de *software* requer um alto nível de habilidades cognitivas dos seus desenvolvedores, o que aumenta ainda mais a sua complexidade [Hernández-López *et al.* 2013].

Os desenvolvedores de *software* não são máquinas, ou seja, recursos materiais com iguais níveis de produtividade; são seres humanos, pessoas diferentes e a diferença de sua produtividade pode ser grande [Hernández-López *et al.* 2013]. Essa diferença entre a produtividade dos desenvolvedores pode inclusive causar danos à produtividade da equipe de desenvolvimento, a ponto de DeMarco (1986) sugerir que "tirar um desenvolvedor de baixo desempenho da equipe pode ser frequentemente mais produtivo para a equipe do que adicionar um bom desenvolvedor". Portanto, compreender os fatores que influenciam a produtividade dos desenvolvedores de *software* é um foco importante a ser pesquisado dentro da Engenharia de *Software*. Apesar dessa importância, a maioria dos trabalhos sobre produtividade de *software* focam mais na organização de *software* e nos projetos de *software*, do que no desenvolvedor de *software* [Meyer *et al.* 2014].

Trendowicz e Münch (2009) afirmam que entre as muitas decisões gerenciais que precisam ser tomadas em um projeto de *software*, os fatores pessoais estão entre os que mais afetam a produtividade de *software*. Wagner e Ruhe (2008), em sua revisão sistemática sobre os fatores de produtividade no desenvolvimento de *software*, consideraram a classificação dos fatores em dois grupos: os fatores técnicos e humanos (*soft*). Hernández-López *et al.* (2013), nas conclusões de sua revisão sistemática sobre fatores na produtividade de *software*, afirmam que o seu grande resultado é afirmar que o sucesso dos projetos de desenvolvimento de *software* ainda depende das pessoas.

Contudo, do ponto de vista da organização de *software*, que tem o intuito de aprimorar seu processo de desenvolvimento de *software*, elas podem tomar atitudes e implementar práticas que influenciem direta ou indiretamente nos fatores de produtividade de seus desenvolvedores. Dessa forma, para facilitar uma melhor compreensão da grande quantidade de fatores de influência sobre a produtividade dos desenvolvedores identificados na literatura, uma boa classificação desses fatores é importante. Para definir uma boa classificação, é igualmente importante conhecer as classificações existentes na literatura.

O modelo COCOMO (*COnstructive COst MOdel*), desenvolvido por Boehm (1981) e depois atualizado para o COCOMO II [Boehm *et al.* 1995], é um modelo de estimativa de custos para o processo de desenvolvimento de *software*. As fórmulas de estimativa desses modelos relacionam as entradas e saídas do processo de desenvolvimento com um conjunto de fatores que possuem influência significativa encontradas pelos seus autores. Esses diversos fatores utilizados no modelo estão classificados em cinco categorias: fatores relacionados a características do produto, como a complexidade; fatores relacionados a plataforma, como restrições de espaço de armazenamento; fatores relacionados ao projeto, como o uso de ferramentas de desenvolvimento; fatores relacionados com a escala, como a maturidade do processo de desenvolvimento; e fatores relacionados ao pessoal, como a capacidade e experiências dos desenvolvedores.

Wagner e Ruhe (2008) classificaram os fatores de produtividade encontrados em dois grandes grupos: fatores técnicos e fatores não técnicos (*soft*). Os fatores técnicos compreendem os fatores relacionados ao produto, ao processo e ao ambiente de desenvolvimento. Os fatores não técnicos são vistos pelos autores como presentes de forma intangível dentro da equipe de desenvolvimento e do ambiente de trabalho. Esses fatores não técnicos são considerados pelos autores como fatores humanos e entre eles estão fatores relacionados à cultura organizacional, à cultura da equipe, às capacidades e experiências, ao ambiente e ao projeto.

Trendowicz e Münch (2009) dividiram os fatores de produtividade de *software* em dois grupos maiores, os fatores de contexto e os fatores de influência. Segundo os autores, dado um modelo de produtividade, os fatores considerados parte integrante do modelo são chamados de fatores de influência; e os outros fatores, considerados fora do modelo, e por isso sem variação dentro do modelo, são chamados de fatores do contexto para esse modelo. Os fatores de influência foram subdivididos em quatro grupos de fatores: relacionados ao produto, relacionados ao pessoal; relacionados ao projeto; e relacionados ao processo.

Hernández-López *et al.* (2013) utilizaram uma classificação baseada no trabalhos de Wagner e Ruhe (2008) e Boehm (1981). Porém, Hernández-López *et al.* (2013), citando Boehm (1981), afirmam que é bem clara a existência de uma outra classificação geral dos fatores de produtividade no processo de

desenvolvimento de *software*: os fatores que dependem das pessoas, chamados fatores humanos e os fatores tecnológicos.

As organizações têm historicamente contratado seus desenvolvedores utilizando características técnicas, o que é evidenciado pelos requisitos descritos em anúncios de vagas de emprego. Além disso, essas organizações são responsáveis por determinar o ambiente de trabalho, tais como: a seleção dos gerentes e líderes de projeto, a composição da equipe, a definição dos processos e metodologias, o local e os equipamentos de trabalho. Mais recentemente, novas práticas e métodos surgiram dentro da área de desenvolvimento de *software* considerando a importância de compreender que os engenheiros de *software* são seres humanos, com características individuais próprias. Dessa forma, os fatores de influência da produtividade do desenvolvedor foram classificados considerando esses três escopos (Tabela 4): fatores técnicos, ligados à experiência e competência técnica do desenvolvedor; fatores humanos, ligados à individualidade do desenvolvedor; e fatores organizacionais, ligados ao ambiente definidos pela organização.

Tabela 4. Classificação dos fatores da produtividade do desenvolvedor.

Grupo de Fatores	Descrição
Fatores Técnicos	Fatores que referem-se à competência individual adquirida pela educação especializada e à experiência na prática que um indivíduo possua e deva desenvolver para realizar um tipo determinado de atribuição na organização.
Fatores Humanos	Fatores que referem-se às características individuais do desenvolvedor, que não são inerentes à sua função na organização.
Fatores Organizacionais	Fatores que referem-se a características existentes no ambiente da organização que impactam no trabalho do desenvolvedor.

A investigação da preponderância entre esses três grupos de fatores é muito importante para as organizações de *software*. Isso se dá porque uma relação de preponderância pode direcionar os esforços de melhoria da produtividade da organização de *software* através (i) da melhoria de seus desenvolvedores, para práticas mais relacionados com o desenvolvedor, considerando o seu lado humano (fatores humanos) e sua capacitação técnica (fatores técnicos); ou (ii) de melhorias

no ambiente da organização (fatores organizacionais), que impactem de modo positivo no trabalho de seus desenvolvedores .

2.5 CONSIDERAÇÕES FINAIS

Este capítulo apresentou um resumo sobre a produtividade na Engenharia de *Software*. A produtividade na Engenharia de *Software* frequentemente adota o sentido econômico do termo. As pesquisas sobre a produtividade de *software* podem ser classificadas em dois tópicos principais [Cheikhi *et al.* 2012]: métricas de produtividade e fatores de influência. Na literatura científica existem diversas métricas de produtividade definidas, ainda sem um consenso geral; e também se encontram muitos trabalhos sobre fatores de influência sobre a produtividade de *software*. Além disso, a relevância do desenvolvedor dentro do desenvolvimento de *software* foi apresentada, mostrando que existem poucos trabalhos de pesquisas focados na produtividade do desenvolvedor. Por fim, foi apresentada a classificação dos fatores de influência na produtividade utilizada nesta pesquisa.

Capítulo 3 – MAPEAMENTO SISTEMÁTICO SOBRE MÉTRICAS DE PRODUTIVIDADE DE *SOFTWARE*

Este capítulo apresenta um estudo secundário (Mapeamento Sistemático da Literatura) realizado com o objetivo de identificar métricas de produtividade utilizadas pelos pesquisadores na literatura. Além disso, é apresentada também a seleção das métricas de produtividade de desenvolvedores utilizadas em estudos posteriores.

3.1 INTRODUÇÃO

A melhoria da produtividade no desenvolvimento de software é o principal meio de tornar a organização de software mais competitiva [Aquino Junior e Meira 2009]. Como não se pode controlar aquilo que não se pode medir [DeMarco 1986], é essencial para a organização a utilização de métricas para avaliação da produtividade do desenvolvedor, e para a definição de práticas que visem aprimorar essa produtividade. Porém, existem diferentes pontos de vista sobre as métricas de produtividade desenvolvidas pelos praticantes e pesquisadores de diversas organizações [Cheikhi *et al.* 2012]. Por isso, nesta pesquisa optou-se por identificar na literatura científica as métricas de produtividade utilizadas pelos pesquisadores da Engenharia de *Software*.

Um mapeamento sistemático de literatura (MSL) é um tipo de revisão sistemática que visa identificar e classificar os estudos científicos existentes na literatura relacionada de forma ampla de uma área de pesquisa [Kitchenham *et al.* 2011]. Este capítulo descreve o mapeamento sistemático realizado nesta pesquisa, apresentando os resultados obtidos e discutindo as métricas encontradas, suas definições e seu contexto de aplicação. Por fim, com base nos resultados desse mapeamento sistemático, um conjunto de métricas de produtividade do desenvolvedor foi selecionado para ser utilizado em estudos posteriores desta pesquisa.

3.2 PROTOCOLO DO MAPEAMENTO SISTEMÁTICO

O principal objetivo de um mapeamento sistemático é prover uma visão geral de uma área de pesquisa, identificando a quantidade, o tipo de pesquisa e os resultados disponíveis dentro dessa área [Petersen *et al.* 2008]. O objetivo deste mapeamento sistemático está estruturado de acordo com o paradigma GQM (*Goal-Question-Metric*) proposto por Basili e Rombach (1988) e definido na Tabela 5.

Tabela 5. Objetivo do MSL segundo paradigma GQM.

Analisar as	as métricas de produtividade
com o propósito de	caracterizar
com relação a	definição e utilização
do ponto de vista dos	Pesquisadores
no contexto do	desenvolvimento e manutenção de *software*

Segundo Kitchenham e Charters (2007), a especificação da questão de pesquisa é a parte mais importante de qualquer revisão sistemática de literatura. De acordo com o objetivo apresentado, as questões de pesquisa desse mapeamento sistemático estão apresentadas na Tabela 6.

Tabela 6. Questão de pesquisa principal e secundárias do MSL.

QP	Como os pesquisadores da Engenharia de *Software* têm mensurado a produtividade do desenvolvimento e manutenção de *software*?
SQ1	Com qual abstração a métrica de produtividade foi associada?
SQ2	Como a métrica de produtividade foi definida?
SQ3	Para qual contexto a métrica de produtividade foi definida?

As questões secundárias são derivadas a partir da questão principal de pesquisa e suas respostas ajudam a compor a resposta da questão principal. A questão secundária *SQ1* visa identificar para qual abstração foram definidas as métricas de produtividade. A definição da métrica de produtividade, subquestão *SQ2*, inclui a identificação das entradas e saídas utilizadas pela métrica assim como

a abordagem quantitativa utilizada. Por fim, o contexto para o qual a métrica foi definida é o foco da subquestão *SQ3*.

3.2.1 Estratégia de busca dos estudos

Em um mapeamento sistemático nem todas as publicações são relevantes, de acordo com os objetivos propostos e as questões de pesquisa definidas. Por isso, faz-se necessário a adoção de uma estratégia de busca e seleção para incluir as publicações relevantes e excluir aquelas que não forem relevantes, isto porque frequentemente os resultados retornados pelas máquinas de busca em revisões sistemáticas apresentam um percentual grande de publicações não relevantes [Jalali e Wohlin 2012]. Portanto, a definição de uma estratégia de busca visa reduzir o escopo da busca a fim de eliminar o esforço desnecessário com publicações não relevantes. A estratégia de busca deste mapeamento incluiu os seguintes itens:

- **Fontes de busca:** As bibliotecas digitais *Elsevier Scopus*[7] e *ISI Web of Science*[8] foram selecionadas para a busca das publicações científicas. Essas duas meta-bibliotecas digitais além de indexar outras bibliotecas digitais, também permitem estabelecer filtros para o idioma, tipo de documento e a área de conhecimento;
- **Tipo de documento:** Somente as publicações científicas, artigos de conferências e periódicos, foram considerados para esse mapeamento pois possuem seu conteúdo revisado por outros pesquisadores independentes utilizando o método de análise por pares (*peer review*);
- **Idioma da busca:** Somente os artigos em inglês, devido à sua adoção pela grande maioria das conferências e periódicos internacionais;
- **Área do conhecimento:** Somente publicações das áreas de Engenharia de *Software* e de negócios. A primeira área é evidente dado a ser a área de pesquisa deste trabalho, da mesma forma que outra área também é importante, pois a produtividade é um tema de interesse relevante para a indústria de *software*.

[7] http://www.scopus.com
[8] http://webofknowledge.com

A estratégia de busca também incluiu a definição da *string* de busca a ser utilizada nas fontes de buscas selecionadas. Essa *string* de busca foi elaborada a partir da extração da questão de pesquisa principal utilizando o critério *PICOC* (População, Intervenção, Comparação, resultadO e Contexto), sugerido por Petticrew e Roberts (2006):

- **População:** A população é onde as métricas de produtividade são aplicadas. Para esse mapeamento, são o desenvolvimento e a manutenção de *software*, derivando termos como: *software development, software maintenance, software process* e *software engineering*;
- **Intervenção:** A intervenção são as métricas de produtividade aplicadas ao desenvolvimento e manutenção de *software*. As palavras *metric*, *measure*, *measurement* e *measuring* foram utilizadas como sinônimos para métricas. Além disso, para a produtividade utilizamos o termo *productivity* e os sinônimos *performance, efficiency e effectiveness*. Porém, esses sinônimos isoladamente trouxeram muitas publicações fora do escopo, decidiu-se, portanto, por restringir utilizando-os em conjunto com qualificadores. Esses qualificadores foram as abstrações utilizadas nas métricas de produtividade. Os termos utilizados para as abstrações foram: *organization, process, project, individual, programmer, developer,* onde *programmer* e *developer* são sinônimos para *individual*. Dessa forma, foram utilizadas combinações como: *process performance* e *individual effectiveness*;
- **Comparação:** A *string* de busca em mapeamentos sistemáticos não limita a seleção de estudos somente às publicações que avaliem seus resultados comparando com outros resultados considerados como "controle". Por esse motivo, esse critério não é aplicável para mapeamentos sistemáticos;
- **Resultado:** A *string* de busca nesta pesquisa não limitou a seleção das publicações por algum tipo de específico de avaliação da métrica. Por esse motivo, esse critério também não foi aplicado;
- **Contexto:** A *string* de busca nesta pesquisa também não limitou a seleção das publicações por algum contexto em particular onde a métrica foi aplicada. Por esse motivo também esse critério não foi aplicado.

A *string* de busca, utilizando os critérios acima apresentados, ficou definida conforme apresentado na Tabela 7.

Tabela 7. String de busca utilizada neste MSL.

Critério PICOC	*String* de busca
População	("software process" **OR** "software development" OR "software engineering" **OR** "software maintenance") **AND** (measure **OR** measurement **OR** measuring **OR** metric) **AND**
Intervenção	(productivity **OR** "organization efficiency" **OR** "process efficiency" **OR** "development efficiency" **OR** "project efficiency" **OR** "programmer efficiency" **OR** "individual efficiency" **OR** "developer efficiency" **OR** "task efficiency" **OR** "organization effectiveness" **OR** "process effectiveness" **OR** "development effectiveness" **OR** "project effectiveness" **OR** "programmer effectiveness" **OR** "individual effectiveness" **OR** "developer effectiveness" **OR** "task effectiveness" **OR** "organization performance" **OR** "process performance" **OR** "development performance" **OR** "project performance" **OR** "programmer performance" **OR** "individual performance" **OR** "developer performance" **OR** "task performance")

3.2.2 Critérios para seleção dos estudos

Os critérios de seleção dos estudos objetivam garantir a relevância desses estudos para o mapeamento sistemático. Os critérios de seleção servem para definir se um estudo será incluído ou excluído do mapeamento sistemático. Neste mapeamento, esses critérios foram divididos em dois filtros. Os critérios de seleção do primeiro filtro estão apresentados na Tabela 8. Os critérios definidos para o primeiro filtro foram simplificados, utilizando somente um critério de inclusão e um de exclusão, uma vez que muitas publicações investigam a produtividade no desenvolvimento de *software*, mas nem todas têm por foco principal o estudo de uma métrica de produtividade. Dessa forma, identificar se a publicação define e utiliza uma métrica de produtividade somente a partir do título e resumo (*abstract*) da publicação pode levar a muitos erros. Por isso, optou-se por simplificar este primeiro filtro para incluir todas as publicações que possam ter utilizado alguma métrica de produtividade de *software*.

Tabela 8. Critérios de inclusão (CI) e exclusão (CE) para o primeiro filtro.

Critério	Descrição
CI.1	O título e/ou resumo da publicação descreve um estudo que possa fazer uso de alguma métrica de produtividade no desenvolvimento ou manutenção de *software*.
CE.1	A publicação não atende ao critério de inclusão.

O segundo filtro foi executado a partir da leitura completa da publicação. Os critérios de seleção adotados nesse segundo filtro estão apresentados na Tabela 9.

Tabela 9. Critérios de inclusão (CI) e exclusão (CE) para o segundo filtro.

Critério	Descrição
CI.1	A publicação define explicitamente e utiliza uma métrica de produtividade em um estudo para avaliação de um desenvolvimento ou manutenção de *software*.
CE.1	A métrica de produtividade não foi definida ou foi definida como uma medida direta (ex. o tempo) e não como uma relação entre pelo menos duas medidas diretas diferentes, descaracterizando assim a ideia (constructo) de produtividade no sentido econômico do termo.
CE.2	A métrica de produtividade foi definida para um contexto muito específico, ou seja, ela não pode ser aplicada em outro contexto de desenvolvimento ou manutenção de *software*.
CE.3	A métrica de produtividade não foi utilizada no estudo, ou seja, a métrica não foi coletada e nem aplicada em um estudo dentro da publicação científica.
CE.4	A publicação não é um artigo científico, por exemplo, é um capítulo de um livro, portanto, não garantindo que houve revisão por pares (*peer review*).
CE.5	A publicação não está em inglês ou não está disponível *online*.

Estes critérios foram baseados no objetivo proposto deste mapeamento sistemático. As publicações relevantes são aquelas que definem e utilizam pelo menos uma métrica de produtividade para avaliação de um desenvolvimento ou manutenção de software. Portanto, não são considerados estudos relevantes aqueles que utilizam métricas somente para estimativa da produtividade. Nesta

mapeamento sistemático, a exigência de utilização da métrica de produtividade dentro do estudo serve como um indicativo de validade e confiabilidade da métrica.

Em relação ao procedimento de seleção final, os critérios do segundo filtro foram utilizados após a leitura completa dos artigos que restaram da seleção preliminar segundo os critérios do primeiro filtro.

3.2.3 Estratégia para extração dos dados

Após a seleção das publicações com aplicação da estratégia de seleção descrita anteriormente, os dados dessas publicações foram extraídos. Os dados extraídos neste mapeamento foram classificados em dados da publicação, dados da métrica de produtividade e dados específicos contexto da publicação.

Os dados da publicação ajudam a mapear com que frequência e em quais veículos da comunidade científica as métricas de produtividade foram utilizadas nas pesquisas sobre a produtividade dentro do desenvolvimento e manutenção de *software*. Os dados específicos da publicação extraídos estão descritos na Tabela 10.

Tabela 10. Dados específicos da publicação.

Código	Número que identifica a publicação extraída
Ano de Publicação	Ano em que foi publicado
Fonte	Veículo em que foi publicado
Título	Título da publicação
Autores	Os autores da publicação

Os dados de definição das métricas de produtividade utilizados na publicação ajudam a compreender quais foram as entradas, saídas e abordagem quantitativa utilizadas. Os dados extraídos sobre a definição das métricas de produtividade estão apresentados na Tabela 11.

Tabela 11. Dados específicos das métricas de produtividade.

Abstração da Métrica	Abstração para a qual as métricas de produtividade foram definidas
Definição da Métrica	Identificação de cada métrica de produtividade existente na publicação, incluindo a descrição, a fórmula de cálculo e a descrição de cada uma das variáveis utilizadas

Na Tabela 12 estão listados os dados específicos extraídos em relação a utilização da métrica de produtividade por parte do pesquisador na publicação científica. Os dados de contexto são importantes, pois as métricas definidas para um contexto normalmente não são aplicáveis a outros contextos [Petersen e Wohlin 2009].

Tabela 12. Dados específicos do contexto da publicação.

Fonte dos dados	**Origem de onde vieram os dados utilizados na publicação: indústria ou academia**
Tipo de Desenvolvimento	A métrica foi aplicada em um desenvolvimento ou manutenção de software
Domínio da aplicação	O domínio de aplicação do desenvolvimento/manutenção: científica, negócio, tempo real, etc.
Linguagem de programação	Linguagem de programação utilizada no desenvolvimento/manutenção

A análise dos dados extraídos neste mapeamento sistemático foi realizada utilizando a técnica *content analysis* (análise de conteúdo). A análise de conteúdo é uma técnica para categorização e determinação da frequência destas categorias, possibilitando fazer a tabulação dos dados facilitando a análise das evidências [Dixon-Woods *et al.* 2005]. A categorização utilizada neste mapeamento sistemático foi realizada com o esquema de classificação apresentado anteriormente, no Capítulo 2, Subseção 2.3.1.

3.2.4 Estudos selecionados após execução do Mapeamento Sistemático

A execução deste mapeamento envolveu o trabalho de três pesquisadores com o fim de evitar o viés de um único pesquisador na aplicação da estratégia de busca e seleção. A aplicação dos critérios de seleção foi realizada por um pesquisador, enquanto outro pesquisador, de forma independente, também aplicou o mesmo filtro em uma amostra de 50 publicações, selecionadas aleatoriamente, para avaliar a confiança da classificação realizada [Siegel e Castellan 1988]. O resultado da concordância entre os dois pesquisadores foi avaliado utilizando-se do teste estatístico *Kappa* [Cohen 1960]. O resultado dessa avaliação de concordância, utilizando dados do primeiro filtro, entre a aplicação dos critérios por parte dos pesquisadores foi significativa (*kappa* = 0.699), segundo a sugestão de interpretação proposta por Landis e Koch (1977).

Ao final do processo foram encontradas 595 publicações na biblioteca digital *Scopus* e 100 publicações na biblioteca digital *Web of Science*, gerando um total de 695 publicações encontradas. Após a remoção das publicações duplicadas, o total de publicações selecionadas para filtragem ficou em 625. Dessas 625 publicações, 401 não atendiam ao critério de inclusão do primeiro filtro e, por isso, foram excluídas. As 224 publicações restantes foram lidas integralmente e, no final da seleção, somente 71 publicações atenderam aos critérios do segundo filtro (Figura 6).

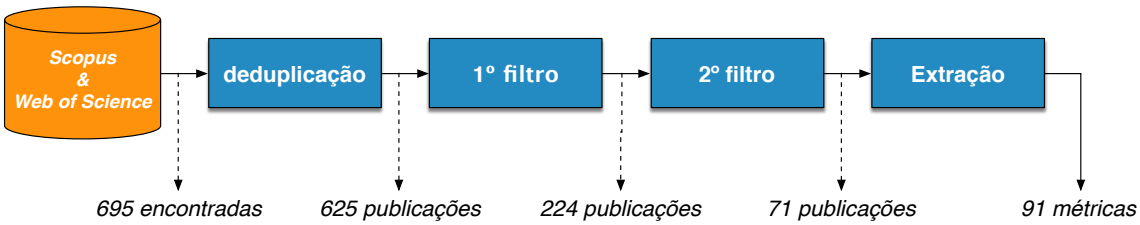

Figura 6. Processo de seleção das publicações deste MSL.

A frequência das publicações por ano envolvendo as métricas de produtividade de *software* vão desde o início da década de 80 até o ano de 2015, ano de conclusão deste mapeamento sistemático (Figura 7). É possível observar que o interesse dos pesquisadores em estudos que avaliaram a produtividade de *software* utilizando-se de métricas de produtividade se intensificou a partir do início da década de 90 pois, a partir dessa década, todos os anos houve dois (mediana da frequência) ou mais estudos sobre o tema, exceção aos ano de 1991 e 2013.

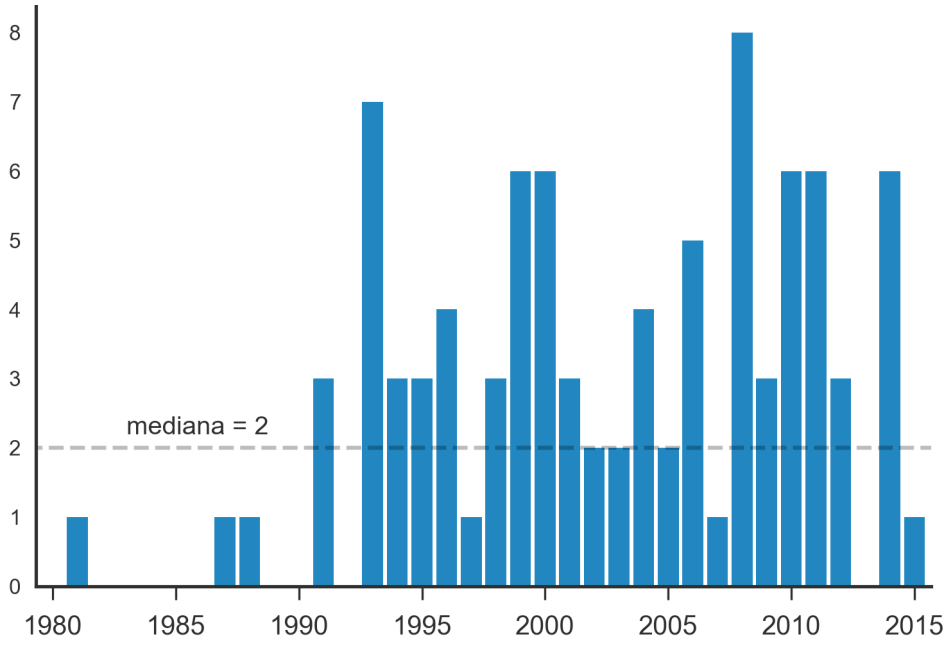

Figura 7. Frequência de publicações por ano.

3.3 RESULTADOS OBTIDOS

3.3.1 Análise dos fóruns de publicação

Um dos objetivos do mapeamento sistemático também é identificar quais os fóruns de publicação são mais frequentemente utilizados pelos pesquisadores de acordo com o objeto de estudo. A Tabela 13 apresenta a lista dos fóruns mais frequentes, com pelo menos três ou mais publicações.

Tabela 13. Fóruns com mais publicações sobre produtividade.

FÓRUM DE PUBLICAÇÃO	ABREVIAÇÃO	QUANTIDADE
IEEE TRANSACTIONS ON SOFTWARE ENGINEERING	*IEEE TSE*	8
IEEE SOFTWARE	*IEEE Softw.*	6
INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING	*ICSE*	6
JOURNAL OF SYSTEMS AND SOFTWARE	*JSS*	4
INTERNATIONAL SOFTWARE METRICS SYMPOSIUM	*METRICS/ESEM*	4
JOURNAL OF INFORMATION AND SOFTWARE TECHNOLOGY	*IST*	3

A lista dos fóruns presente na Tabela 13 representa mais de 43% dos fóruns encontrados nas publicações selecionadas. Não houve a predominância de um fórum em particular. O fórum com o maior número de publicações, com mais de 11% das publicações, foi o *IEEE Transactions on Software Engineering* (IEEE TSE), seguido por *IEEE Software* e pelo *International Conference on Software Engineering* (ICSE), ambos com mais de 8% das publicações selecionadas.

3.3.2 SQ1. Com qual abstração a métrica de produtividade foi associada?

Foram extraídas um total de 91 métricas de produtividade de *software* nas 71 publicações selecionadas. A maioria dos estudos envolvendo as métricas de produtividade foram definidas para avaliar a produtividade dos projetos e desenvolvedores de *software* (Figura 8). Essas duas abstrações destacam-se em relação às demais abstrações encontradas, pois as abstrações de processo, organização, tarefa e módulo tiveram uma frequência bem inferior.

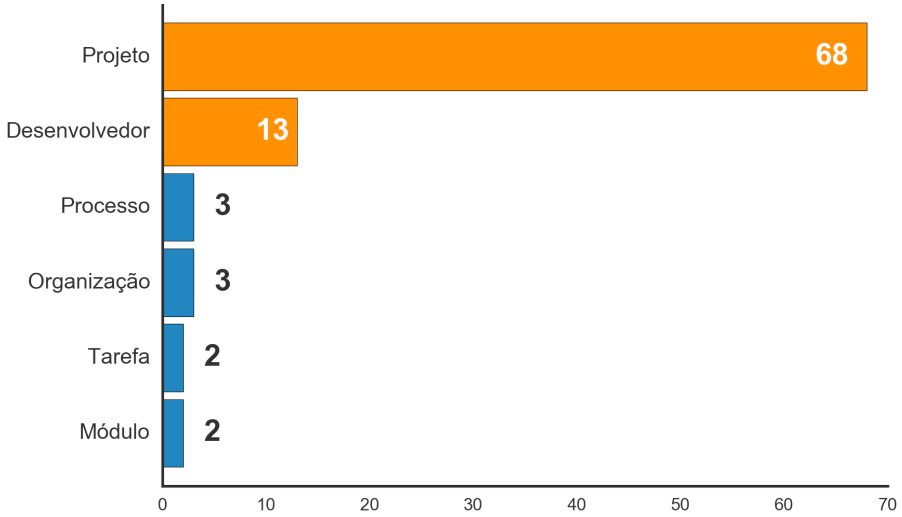

Figura 8. Quantidade de métricas extraídas por abstração.

Analisando essas abstrações ao longo dos anos (Figura 9), é possível observar que desde do início da década de 90, em quase todos os anos, houve pelo menos um estudo sobre a produtividade em projetos de *software*. Além do projeto, é possível também destacar que a produtividade do desenvolvedor, mesmo que de forma esparsa, também obteve uma frequência maior do que as demais abstrações.

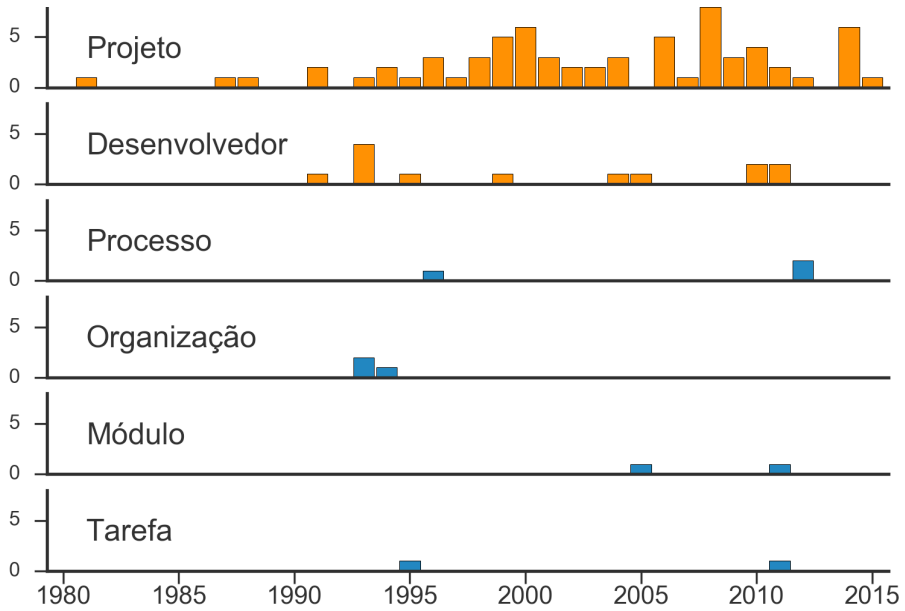

Figura 9. Métricas por abstração ao longo do tempo.

Portanto, respondendo à pergunta de pesquisa *SQ1*, pode-se afirmar que as métricas de produtividade foram principalmente definidas para avaliar os projetos de *software*. A avaliação da produtividade dos desenvolvedores também teve um pequeno destaque, porém bem distante da avaliação da produtividade dos projetos

de *software*. As outras abstrações identificadas para as métricas de produtividade podem ser consideradas como estudos isolados.

3.3.3 SQ2. Como a métrica de produtividade foi definida?

Para melhor analisar os resultados para essa subquestão, as métricas de produtividade extraídas das publicações selecionadas foram categorizadas de acordo com a sua estrutura, utilizando as definições apresentadas no Capítulo 2, Subseção 2.3.1. De acordo com essa estrutura, a definição das métricas de produtividade foram decompostas em: (i) métricas de entrada e saída e (ii) abordagem quantitativa. Essa decomposição permitiu uma melhor análise e compreensão da definição das métricas de produtividade extraídas.

Com relação às abordagens quantitativas, percebeu-se que a razão simples foi a abordagem mais adotada (em 79 de 91 métricas) e a única que abrangeu todas as abstrações identificadas neste mapeamento sistemático (Figura 10). Essa abordagem foi muito utilizada na medição da produtividade de projetos de *software* e desenvolvedores. As outras abordagens quantitativas tinham pequenas frequências (12 de 91 métricas) e não abrangeram todas as abstrações. É possível destacar a abordagem de Análise Envoltória de Dados (*Data Envelopment Analysis – DEA*) para a avaliação da produtividade em projetos de *software*, com seis métricas.

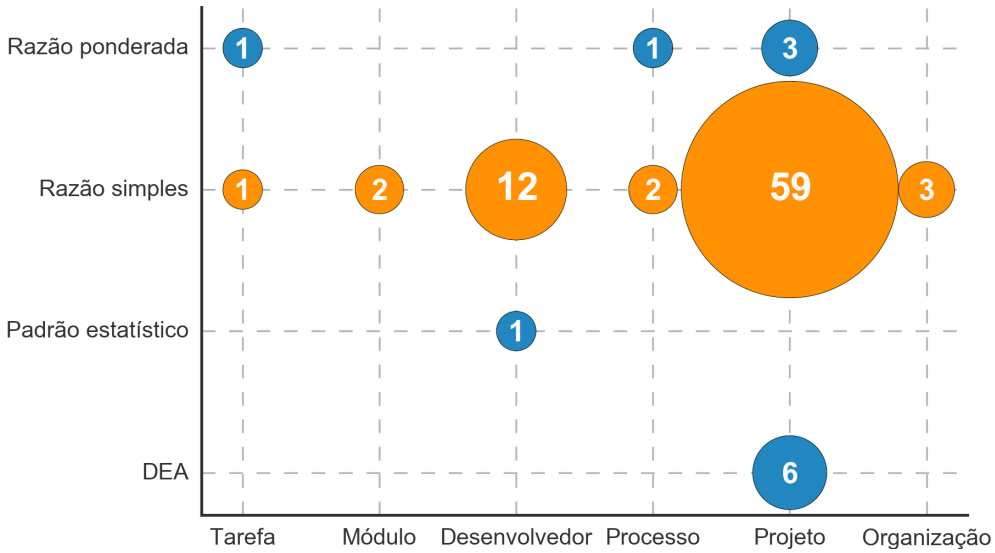

Figura 10. Abordagem quantitativa das métricas de produtividade por abstração.

Durante a decomposição das métricas de produtividade em suas variáveis de entrada e saída, foi observado que mesmo utilizando as mesmas medidas de entrada

e/ou saídas, os pesquisadores definiam-nas de formas diferenciadas. Eles as definiam com diferentes nomes de variáveis, mas com a mesma unidade. Por exemplo, o esforço teve algumas definições, tais como *man-month*, *person-day*, *developer-year*, todas medidas utilizando a unidade de pessoa por tempo. Outro exemplo é o tamanho de saída medido em linhas de código (*LOC*), que teve definições como *NCLOC* (linhas de código de não-comentário), *KLOC* (linhas de código Kilo) e *SLOC* (linhas de código fonte). Para melhorar a análise das métricas de produtividade, essas medidas de entrada e saída foram agrupadas sob o nome de uma medida mais geral. Os mapeamentos adotados para medidas de entrada são mostrados na Tabela 14, para medidas de saída são mostrados na Tabela 15.

Tabela 14. Mapeamento das medidas de entrada.

Medida	Nome da variável	Unidade utilizada
Pessoa	*Developer*	*person*
Custo	*C, Man-Cost*	*person-cost*
Tempo	*Hour, T, Time, Minute, DevTime, TimeMonth, Month, CycleTime, Year*	*hour, minute, month, year*
Esforço	*Developer-Hour, Developer-Quarter, Developer-Year, EngineeringMonth, E, Eft, H, Man-Day, Man-Hour, Man-Month, Effort, PD, PH, Man-Quarter, PM, Man-ProjectTime, SM, Person-Days, Person-Month, Staff-Hour, Staff-Month*	*person-hour, person-day, person-month, person-quarter, person-year*

Essas tabelas mostram o nome da variável e a unidade utilizada na métrica de produtividade dentro da definição descrita pela publicação. A primeira coluna apresenta o nome da medida que será utilizada daqui por diante.

Esses mapeamentos permitiram uma melhor organização dos resultados e facilitaram a análise de conteúdo das métricas de produtividade extraídas. Nessas tabelas, é possível observar que existem muitos nomes de variáveis para medir o tempo, o esforço, a tarefa, pontos por função e linhas de código. A diversidade entre tempo e esforço ocorreu porque foram medidas com diferentes unidades de tempo. Os pontos por função e as linhas de código tiveram nomes diferentes devido aos diferentes métodos ou estratégias utilizados pelos pesquisadores. As medidas que eram composta pela soma de vários artefatos de *software* foram agrupadas sob o

nome de tarefa, pois que representavam o montante produzido de uma determinada tarefa trabalhada.

Tabela 15. Mapeamento das medidas de saída.

Medida	Nome da variável	Unidade utilizada
Halstead Effort	*Halstead Effort*	*halstead effort*
Tarefa	Number of: *Classes, Modifications, ModificationsRequest, ModuleModifications, Modules, WorkItems, Pages, Requirements*	*#Classes, #Modifications, #Modules, #WorkItems, #Pages, #Requirements*
Pontos por Função (FP)	*FP, CFP, EFP, S, CodeSize, OOmFPWeb, UFP, OOFP, SM*	*function points*
Linhas de Código (LOC)	*LOC, KLOC, KSLOC, SLOC, ELOC, NLOC, AvgLOC, WSDI, SLC, KNCSS, LOC added, S, SL L, CP, Size, TotalChurn, NCLOC, CodeContribution*	*lines of code*

3.3.3.1 *Métricas de produtividade definidas com uma razão simples*

Analisando apenas as métricas de produtividade definidas como uma razão simples, contando com 79 métricas do total de 91 métricas extraídas, utilizando a estratégia de mapeamento explicada anteriormente, foram obtidas nove métricas de produtividade (Figura 11).

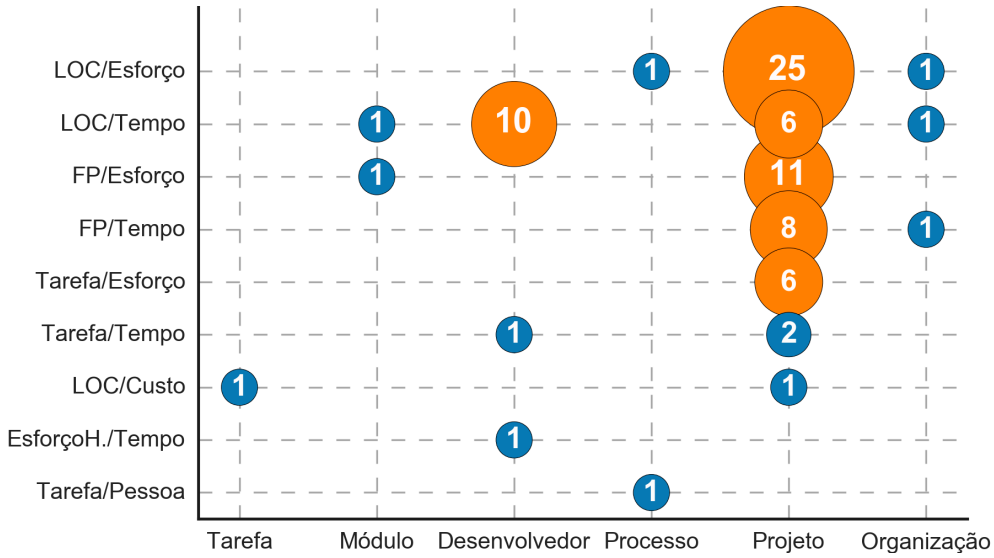

Figura 11. Métricas de produtividade com uma razão simples por abstração.

Na Figura 11, foram destacadas (em laranja) as métricas de produtividade utilizadas por pelo menos mais de três publicações. Todas as métricas destacadas foram definidas para a abstração do projeto de software, exceto uma que foi definida para o desenvolvedor de *software*. Essas métricas utilizaram apenas cinco medidas: *LOC*, *FP* e Tarefa para saída; Esforço e Tempo para a entrada. Essas medidas foram no total utilizadas por 72 publicações. A métrica *LOC/Esforço* sozinha representa mais de um terço de todas as abordagens quantitativas das métricas de produtividade definidas com uma razão simples.

Para explorar ainda mais a definição dessas métricas, isolamos as medidas de entrada e saída da definição da métricas de produtividade extraídas (Figura 12). Claramente, linhas de código (*LOC*) e esforço foram as medidas mais utilizadas, sendo que *LOC* foi utilizada para medir todas as abstrações encontradas neste mapeamento. Em segundo lugar, pontos por função (*FP*) e tempo foram as medidas de entrada mais utilizadas.

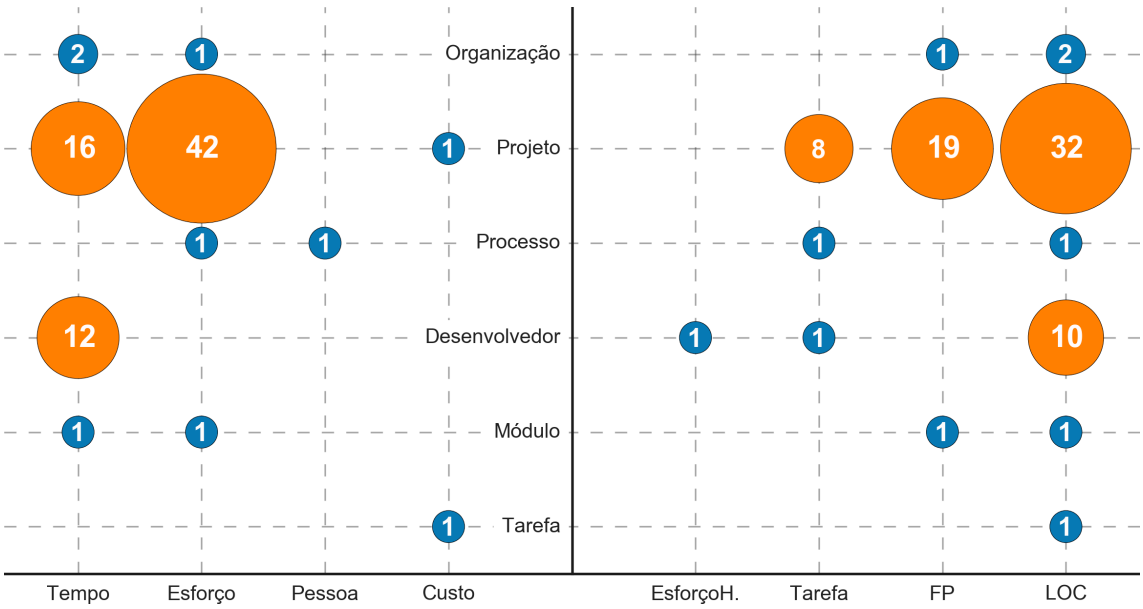

Figura 12. Medidas de entrada e saída das métricas por abstração.

Na Figura 12, destacamos (em laranja) as medidas com uma frequência acima de três. A medida de tarefa, além das outras medidas já mencionadas, foi a única outra destacada com uma frequência acima de três. A medição de tarefas foi usada para avaliar o desenvolvedor, o processo e o projeto. É possível observar também que as medidas mais frequentes utilizadas para compor a definição de produtividade do desenvolvedor de *software* foram *LOC* e tempo.

3.3.3.2 *Métricas de produtividade definidas com outras abordagens quantitativas*

Considerando as outras abordagens quantitativas, a Tabela 16 mostra a lista de onze métricas de produtividade não baseadas em uma abordagem quantitativa utilizando um razão simples. A maioria dessas métricas (oito métricas) foram definidas para a abstração de projeto de *software*. A análise envoltória de dados (*DEA*) foi a abordagem quantitativa mais utilizada (com seis métricas), sendo todas também para avaliar projetos de *software*. As métricas baseadas em razões ponderadas foram definidas para as abstrações de projeto, processo e tarefa; enquanto a métrica baseada em padrão estatístico foi usada apenas em uma única publicação para avaliar a produtividade do desenvolvedor, utilizando um algoritmo de aprendizado de máquina não supervisionados (*clustering*).

Tabela 16. Métricas de produtividade com outras abordagens quantitativas.

Abstração	Abordagem quantitativa	Medida de entrada	Medida de saída
Desenvolvedor	Padrão estatístico	Pessoa-Tempo	Complexidade/LOC #Comentários/LOC
Projeto	DEA	1 (*constante*)	LOC/Custo LOC/Tempo
Projeto	DEA	Pessoa-Tempo	#Usuários #Interfaces #Programas-de-Conversão
Projeto	DEA	Pessoa-Tempo	FP #Defeitos
Projeto	DEA	Pessoa-Tempo	FP #Usuários #Localidades #Unidades-de-Negócio
Projeto	DEA	#Desenvolvedores #Relatores-de-Bug	Rank-*SourceForge* #Downloads #Kb-Baixados
Projeto	DEA	Tempo Pessoa-Custo Terceiros-Custo	LOC

Abstração	Abordagem quantitativa	Medida de entrada	Medida de saída
Projeto	Razão ponderada	Tempo %Reuso	#Páginas-Web #Novas-Imagens #Funções-de-Alto-Esforço
Projeto	Razão ponderada	Tempo %Custo-Reuso	LOC-Código-Novo LOC-Código-Reutilizado
Processo	Razão ponderada	Tempo Pessoa-Tempo	LOC
Tarefa	Razão ponderada	Tempo #Tarefas	LOC

Considerando as unidades de entrada e saída utilizadas nessas métricas, é possível observar na Tabela 16 que as medidas baseadas no tempo (tempo e esforço) também foram predominantes como medidas de entrada. As medidas de saída têm uma grande variação de medidas utilizadas, embora a maioria delas inclua o tamanho do software utilizando ou *LOC*, ou *FP*, ou uma forma alternativa de contagem, como o número de páginas da *web*, o número de funções, etc.

Respondendo a subquestão de pesquisa *SQ2*, é possível afirmar que as métricas de produtividade de software foram definidas pelos pesquisadores principalmente por meio de uma abordagem quantitativa de razão simples. As medidas de entrada mais utilizadas nessa abordagem foram o tempo e esforço para projetos de *software* e o tempo para avaliar os desenvolvedores. Linhas de código (*LOC*) e pontos por função (*FP*) foram as medidas de saída mais utilizadas, sendo *FP* apenas utilizada para projetos de *software*. Outras abordagens foram encontradas, mas a maioria delas foi usada em apenas um estudo.

3.3.4 SQ3. Para qual contexto a métrica de produtividade foi definida?

O contexto é o conjunto de fatores do ambiente onde a métrica de produtividade foi aplicada. No entanto, nem todas as publicações deixaram de forma clara a descrição do contexto para o qual a métrica de produtividade foi definida. As características do contexto mais descritas nas publicações selecionadas foram o tipo de desenvolvimento (*desenvolvimento* ou *manutenção*) e a linguagem

de programação utilizada. Esses são duas características mais descritas em estudos sobre a produtividade de software [Trendowicz e Münch 2009].

A maioria das publicações selecionadas utilizou dados da indústria, incluindo todas as abstrações (Figura 13). Nesse sentido, é claro o interesse em avaliar a produtividade de projetos de *software* na indústria. Apenas alguns estudos utilizaram dados da academia e, nesse contexto, abrangendo apenas projetos de *software* e também a produtividade do desenvolvedor.

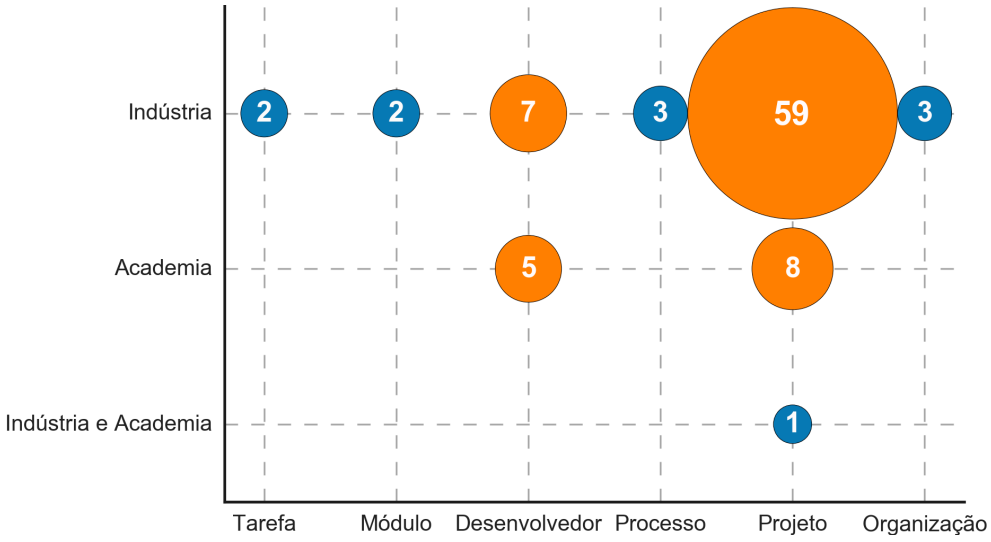

Figura 13. Fontes de dados por abstração.

Considerando o tipo de desenvolvimento, quase todas as métricas de produtividade foram definidas e utilizadas no contexto do desenvolvimento de um novo *software* (Figura 14). Apenas quatro métricas foram definidas exclusivamente no contexto da manutenção de *software*, avaliando projetos de *software*.

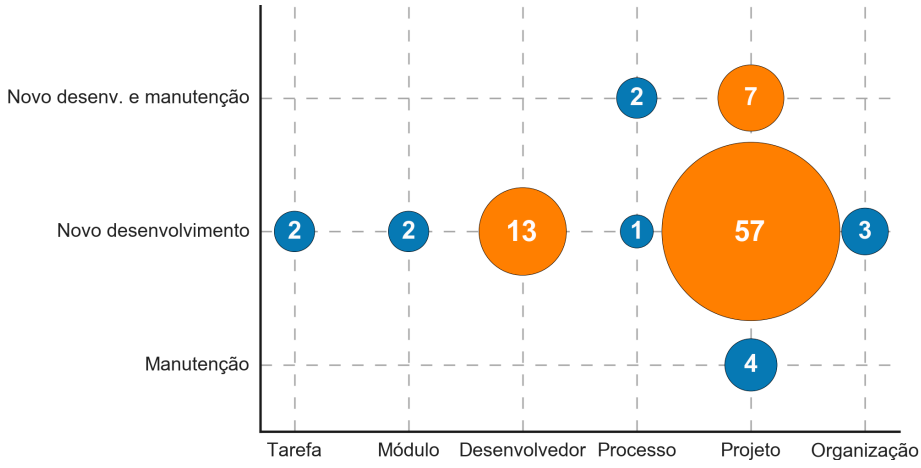

Figura 14. Tipos de desenvolvimento por abstração.

Por fim, a Figura 15 mostra a lista de linguagens de programação extraídas das publicações selecionadas, apresentando somente as linguagens que apareceram em mais de um estudo. Um total de 35 linguagens de programação diferentes foram encontradas nas publicações selecionadas.

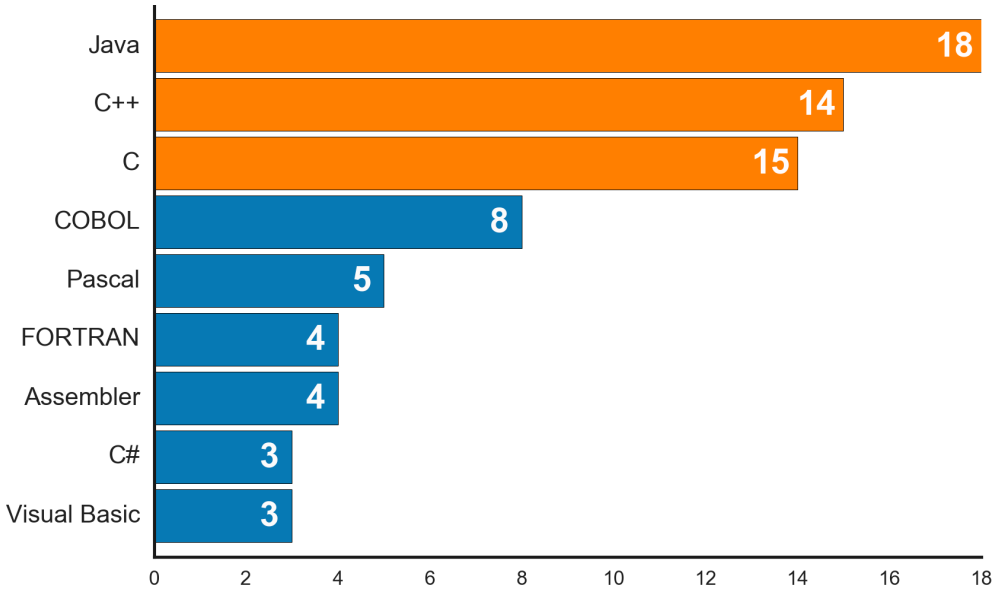

Figura 15. Linguagens de programação mais utilizadas.

Assim, respondendo à questão de pesquisa *SQ3*, o contexto mais frequente em que as métricas de produtividade do *software* foram definidas e utilizadas por pesquisadores estavam usando dados provenientes da indústria, em novos desenvolvimentos, avaliando projetos de *software* ou desenvolvedores, usando principalmente as linguagens de programação C, C ++ ou Java.

3.4 DISCUSSÃO DOS RESULTADOS

3.4.1 QP. Como os pesquisadores da Engenharia de *Software* têm mensurado a produtividade do desenvolvimento e manutenção de *software*?

Este mapeamento sistemático teve como objetivo investigar as métricas de produtividade do software aplicadas para avaliar o desenvolvimento e manutenção de software do ponto de vista dos pesquisadores da Engenharia de *Software*. A maioria dos pesquisadores investigou a produtividade de projetos e desenvolvedores em novos desenvolvimentos com dados da indústria e utilizando uma métrica baseada em uma razão simples. Eles usaram principalmente o tempo e o esforço como medidas de entrada e linhas de código (*LOC*) como medida de saída. Esse

resultado é interessante, pois linhas de código recebeu muitas críticas em sua utilização [Barb *et al.* 2014], especialmente de pesquisadores que defendem o uso de pontos por função.

Neste mapeamento, foi mostrado que, de forma mais intensa nos últimos quinze anos, os pesquisadores da Engenharia de *Software* se focaram na compreensão da produtividade dos projeto de *software*. Eles utilizaram diferentes abordagens quantitativas, com uma variedade de medidas de entrada e saída do desenvolvimento de *software*. No entanto, a razão simples foi a principal escolha da maioria das publicações selecionadas. Esse resultado era de uma certa forma esperado, uma vez que a sobrevivência das organizações de *software* depende, até certo ponto, do sucesso de seus projetos de *software*.

Em menor grau, os desenvolvedores foram outro foco de pesquisa na Engenharia de *Software*. A quantidade de métricas de produtividade definida para desenvolvedores apareceu em segundo lugar, destacado acima das outras abstrações encontradas. Esse fato também não é surpreendente, considerando que os fatores humanos alcançaram maior importância no campo da pesquisa de Engenharia de *Software* nos últimos anos [Amrit *et al.* 2014]. No entanto, o interesse na produtividade do desenvolvedor ainda é pequeno, especialmente quando comparado ao número de publicações para investigar a produtividade dos projetos de *software*. Esta é uma questão mesmo relevante se considerarmos o papel do desenvolvedor no desenvolvimento de software, pois o desenvolvedor é o elemento que traz mais incerteza aos projetos de *software* [Trendowicz e Münch 2009], contribuindo fortemente para o sucesso ou falha de qualquer projeto de *software*.

Finalmente, podemos responder a nossa principal questão de pesquisa associada a este capítulo: pesquisadores da Engenharia de *Software* têm mensurado principalmente a produtividade de projetos de *software* e desenvolvedores de *software* na indústria. As medições da produtividade do projetos de software e desenvolvedores de software são predominantemente utilizando as medidas de linhas de código (*LOC*) por esforço e linhas de código (*LOC*) por tempo, respectivamente. A escolha dessas métricas pode devido ao fato dessas métricas serem obtidas com razoável facilidade, como apontado por outros pesquisadores [Boehm 1987; Hernández-López *et al.* 2013; Mockus 2009].

3.4.2 Relacionamento com as evidências existentes

Neste estudo, também foi analisado as métricas de produtividade de acordo com sua estrutura de definição. Foram considerados alguns aspectos da estrutura das métricas, tais como a abstração escolhida, as medidas de entrada e saída adotadas e a abordagem quantitativa usada nas métricas. Conforme já mencionado, a maioria das métricas de produtividade foram definidas para projetos de *software* e desenvolvedores, utilizando principalmente tempo e esforço como medida de entrada e quantidade de linhas de código (*LOC*) como medida de saída, integradas predominantemente por uma abordagem quantitativa de razão simples.

Petersen (2011) também investigou a estrutura das métricas de produtividade do software. Em seu trabalho, o esquema de classificação também considerou os aspectos da abstração e da abordagem quantitativa, mas não incluiu as medidas de entrada e saída usadas pelas métricas identificadas. Comparando os resultados deste mapeamento com os resultados obtidos por Petersen (2011), é possível confirmar que o projeto de *software* foi o foco mais frequente dos pesquisadores, superando em muito, pela quantidade maior número de estudos, outras abstrações como a dos desenvolvedores de *software*. Comparando as abordagens quantitativas, é possível identificar importantes diferenças entre os resultados. Enquanto os resultados de Petersen (2011) indicam que a análise envoltória de dados (*DEA*), razão ponderada, simulação baseada em eventos e razão simples foram as abordagens quantitativas mais adotadas, os resultados deste mapeamento indicam que a razão simples sozinha foi amplamente a abordagem mais frequente. Essas diferenças são devidas aos diferentes enfoques dos estudos e, consequentemente, a diferentes estratégias de busca adotadas.

Hernández-López *et al.* (2013) trabalharam as métricas de produtividade, em sua revisão sistemática, também de acordo com as medidas de entrada e saída. Eles se concentraram nos indivíduos, incluindo o desenvolvedor de *software*. Os resultados desta pesquisa, ao considerar apenas as métricas de produtividade do desenvolvedor, confirmam os resultados encontrados por Hernández-López *et al.* (2013). Segundo os autores, as medidas de tempo e linhas de código (*LOC*) foram as medidas de entrada e saída mais frequentes usadas em métricas de produtividade para desenvolvedores de *software*. Esses achados são consistentes também com outros estudos [Boehm 1987; Meyer *et al.* 2014].

3.5 SELEÇÃO DAS MÉTRICAS DE PRODUTIVIDADE DO DESENVOLVEDOR

Este mapeamento sistemático extraiu de 71 publicações selecionadas na literatura 91 métricas de produtividade de *software*. A maioria dessas métricas foram definidas para avaliar projetos de *software*, com 68 métricas. As métricas para avalição da produtividade do desenvolvedor apareceram em segundo lugar, com 12 métricas. As outras métricas de produtividade restantes foram definidas para outras abstrações.

A métrica de produtividade mais frequentemente utilizada para avaliar os desenvolvedores foi a quantidade de linhas de código pelo tempo (*LOC/Tempo*), sendo seguida pela quantidade de tarefas pelo tempo (*Tarefas/Tempo*) e o esforço de Halstead (1977) pelo tempo (*Halstead/Tempo)*. Além dessas métricas, também foi identificada uma única outra métrica de avalição da produtividade do desenvolvedor: uma métrica utilizando padrões estatísticos, através de um algoritmo de aprendizagem de máquina, considerando medidas como a complexidade, a quantidade de linhas de código, a quantidade de comentários no código e a quantidade de desenvolvedores.

O propósito com este mapeamento para a pesquisa de doutorado foi de mapear e depois selecionar métricas de produtividade do desenvolvedor a partir das métricas existentes na literatura. Dentre as métricas identificadas na literatura, somente a métrica utilizando padrões estatísticos foi excluída da seleção para o restante desta pesquisa. Essa métrica, para ser implementada, precisa de informações sobre outros projetos da organização para determinar alguns internos, inerentes à implementação do algoritmo utilizado. Porém, não era factível a coleta dessas informações necessárias junto à organização, pois não seria possível ter acesso a todos os fontes de todos os projetos das organizações selecionadas nesta pesquisa. Por isso, essa métrica não foi incluída no conjunto final das métricas de produtividade do desenvolvedor a serem utilizadas em estudos posteriores.

Portanto, o conjunto final de métricas de produtividade selecionadas para a avaliação do desenvolvedor, a serem utilizadas nos estudos posteriores desta pesquisa, é composto por: (i) quantidade de linhas de código pelo tempo (*LOC/Tempo*), (ii) quantidade de tarefas executadas pelo tempo (*Tarefas/Tempo*) e (iii) esforço de *Hasltead* pelo tempo (*HalsteadE./Tempo*).

3.6 AMEAÇAS À VALIDADE

Todo estudo possui ameaças que podem afetar a validade dos seus resultados [Wohlin *et al.* 2012]. Dentre as ameaças à validade deste estudo, destacamos duas principais. A primeira ameaça está relacionada à validade de conclusão desse mapeamento sistemático é quanto a generalização dos resultados, mitigada pela escolha de duas meta-bibliotecas digitais, que indexam outras bibliotecas digitais de diferentes áreas do conhecimento, incluindo duas áreas onde a produtividade é um tópico recorrente. A outra principal ameaça à validade dos resultados é a possibilidade do autor deste estudo ter introduzido seu viés durante a execução do protocolo de revisão, mitigada pelo acompanhamento e revisão por outros pesquisadores mais experientes durante o processo de execução.

3.7 CONSIDERAÇÕES FINAIS

Esse capítulo apresentou um mapeamento sistemático sobre as métricas de produtividade utilizadas pelos pesquisadores dentro da literatura científica. Foram apresentadas as métricas de produtividade mais frequentemente utilizadas, as abstrações medidas, a estrutura de sua definição e seu contexto de utilização. Por fim, um conjunto de métricas de produtividade do desenvolvedor foi selecionado a partir desse mapeamento.

Capítulo 4 – REVISÃO TERCIÁRIA SOBRE FATORES DE INFLUÊNCIA NA PRODUTIVIDADE DE *SOFTWARE*

Este capítulo apresenta um estudo terciário (Revisão Terciária de Literatura) realizado com o objetivo de identificar os fatores de influência na produtividade de software. Além disso, são apresentados também esses fatores classificados em fatores técnicos, humanos e organizacionais.

4.1. INTRODUÇÃO

Para reduzir os custos de produção através da melhoria da produtividade [Aquino Junior e Meira 2009], é necessário que a organização selecione e implemente práticas eficazes visando à melhoria da produtividade. Essas práticas, por sua vez, devem ser baseadas nos fatores de produtividade mais relevantes para a melhoria da produtividade da organização [Sampaio *et al.* 2010]. Os gestores das organizações estão mais conscientes da importância dos fatores que influenciam a produtividade da equipe envolvida nos projetos de software [Melo *et al.* 2011]. O problema é que a produtividade de software é influenciada por muitos fatores e as organizações frequentemente não sabem quais são esses fatores e nem por onde começar.

Durante a realização do estudo anterior, o mapeamento sistemático de literatura (MSL) sobre as métricas de produtividade, foram encontrados alguns estudos secundários sobre fatores de influência da produtividade. Por isso, optou-se por realizar uma revisão terciária para identificar na literatura científica os fatores de influência da produtividade. Revisões terciárias são revisões sistemáticas realizadas sobre estudos secundários. Este capítulo descreve a revisão terciária realizada nesta pesquisa, apresentando os resultados obtidos e discutindo e classificando os fatores de influência encontrados.

4.1 PROTOCOLO DA REVISÃO TERCIÁRIA

O objetivo principal deste estudo é encontrar os fatores de influência na produtividade de *software* nos estudos secundários publicados na literatura

científica. Assim como no estudo anterior, o objetivo desta revisão terciária está estruturado de acordo com o paradigma GQM (*Goal-Question-Metric*) proposto por Basili e Rombach (1988) e definido na Tabela 17.

Tabela 17. Objetivo da revisão terciária segundo paradigma GQM.

Analisar as	os fatores de influência na produtividade de *software*
com o propósito de	caracterizar
com relação a	definição e classificação
do ponto de vista dos	Pesquisadores
no contexto do	desenvolvimento e manutenção de *software*

As questões de pesquisa desta revisão terciária estão apresentadas na Tabela 18, de acordo com o objetivo apresentado. As questão principal de pesquisa serve de base para derivação das questões secundárias que, por sua vez, servem para compor a resposta da questão principal. A questão secundária *SQ1* visa identificar as classificações utilizadas nos estudos secundários para organizar os fatores de influência da produtividade. Os fatores de influência na produtividade são o foco da subquestão *SQ2*. Por fim, a subquestão *SQ3* visa identificar, através da quantidade de referências, quais os fatores mais investigados pelos pesquisadores da produtividade.

Tabela 18. Questão de pesquisa principal e secundárias da revisão terciária.

QP	Quais são os fatores de produtividade encontrados pelos estudos secundários existentes sobre fatores de produtividade no desenvolvimento de *software*?
SQ1	Qual foi a classificação adotada para organizar os fatores encontrados?
SQ2	Quais foram os fatores de influência na produtividade pesquisados?
SQ3	Quais foram os fatores de influência mais frequentemente pesquisados?

4.1.1 Estratégia de busca dos estudos

A definição de uma estratégia de busca visa reduzir o escopo da busca a fim de eliminar o esforço desnecessário com publicações não relevantes, frequentemente representando um percentual grande das publicações retornadas pelas máquinas de

busca [Jalali e Wohlin 2012]. A estratégia de busca deste mapeamento incluiu os seguintes itens:

- **Fontes de busca:** As bibliotecas digitais *ACM Digital Library*[9], *Engineering Village*[10], *IEEE Xplore Digital Library*[11], *Scopus*[12] e *Web of Science*[13]. A escolha dessas bibliotecas foi baseada na experiência relatada por Dybå *et al.* (2007);
- **Tipo de documento:** Somente as revisões de literatura publicadas em veículos científicos, como anais de conferências e periódicos, foram considerados para essa revisão terciária, pois possuem seu conteúdo revisado por outros pesquisadores independentes utilizando o método de análise por pares (*peer review*);
- **Idioma da busca:** Somente os artigos em inglês, devido à sua adoção pela maioria das conferências e periódicos internacionais de Engenharia de Software.

A geração da *string* de busca foi baseada nos termos utilizados em uma revisão terciária realizada por Kitchenham *et al.* (2010) e também nos termos selecionados a partir de uma lista de referência composta por quatro estudos secundários sobre fatores de produtividade literatura [Paiva *et al.* 2010; Trendowicz e Münch 2009; Wagner e Ruhe 2008]. Essas revisões de literatura foram encontradas durante a realização do mapeamento sistemático no estudo descrito no capítulo anterior e formam uma lista de referência utilizada nesta revisão terciária. Esses termos foram classificados em três conjuntos: termos associados ao desenvolvimento de *software*, termos associados aos fatores de produtividade e termos associados a estudos secundários; Os conjuntos e seus termos são detalhados a seguir.

- **Termos associados ao desenvolvimento de software**. São termos relacionados com o contexto desta revisão terciária, baseados nas palavras

[9] http://dl.acm.org
[10] https://www.engineeringvillage.com
[11] http://ieeexplore.ieee.org
[12] https://www.scopus.com
[13] http://apps.webofknowledge.com

descritas no título e resumo da lista de referência utilizada. Os termos selecionados foram: "*software development*" e "*software engineering*";

- **Termos associados aos fatores de produtividade do desenvolvimento de *software***. São termos relacionados com fatores de produtividade, baseados nas *strings* de busca utilizadas pelos estudos da lista de referência. Os termos utilizados foram: *"factor", "indicator", "driver"* e *"productivity"*, "*development efficiency*", "*development effectiveness*", "*development performance*";
- **Termos associados a estudos secundários**. São termos relacionados com os estudos secundários, selecionados a partir dos termos de busca por estudos secundários da revisão terciária de Kitchenham *et al.* (2010). Os termos selecionados foram: *"review"*, *"overview"*, *"literature"*, *"meta-analysis"*, "*past studies"*, "*in-depth survey*", "*subject matter expert*", "*analysis of research*", "*empirical body of knowledge*", "*overview of existing research*" e "*body of published research*".

A *string* de busca utilizada nas máquinas de busca das bibliotecas digitais, utilizando os termos acima apresentados, ficou definida conforme apresentado na Tabela 19.

Tabela 19. *String* de busca utilizada nesta revisão terciária.

Critério	*String* de busca
Desenvolvimento de *Software*	("software development" **OR** "software engineering") **AND**
Fatores de Produtividade	("factor" **OR** "indicator" **OR** "driver") **AND** ("productivity" **OR** "development efficiency" **OR** "development effectiveness" **OR** "development performance") **AND**
Estudos Secundários	("review" **OR** "overview" **OR** "literature" **OR** "meta-analysis" **OR** "past studies" **OR** "in-depth survey" **OR** "subject matter expert" **OR** "analysis of research" **OR** "empirical body of knowledge" **OR** "overview of existing research" **OR** "body of published research")

A análise dos dados extraídos nesta revisão terciária foi realizada utilizando a técnica *content analysis* (análise de conteúdo), utilizada para categorização e

determinação da frequência destas categorias, facilitando a análise das evidências [Dixon-Woods *et al.* 2005]. Em um primeiro nível, a categorização adotada utilizou uma combinação das categorias encontradas nos estudos secundários e, em segundo nível, essas categorias foram classificadas em fatores técnicos, humanos ou organizacionais, de acordo com as definições apresentadas no Capítulo 2, Subseção 2.4.1.

4.1.2 Critérios para seleção dos estudos

Nem todas as publicações retornadas pelas máquinas de busca das bibliotecas digitais são úteis para responder as questões de pesquisa de uma revisão sistemática. Um dos motivos pode ser a expansão dos termos utilizados por parte da máquina de busca, fazendo com que as publicações encontradas não estejam relacionadas com a questão de pesquisa. Como não é possível conhecer antecipadamente todas as características das máquinas de busca utilizadas, é importante realizar um processo de filtragem das publicações retornadas. Esse processo visa restringir as publicações retornadas a um subconjunto que esteja relacionado com a questão de pesquisa. Por isso, alguns critérios e procedimentos são realizados antes da extração dos dados das publicações encontradas. Esses critérios foram divididos em dois filtros, apresentados na Tabela 20 e Tabela 21.

Tabela 20. Critérios de inclusão (CI) e exclusão (CE) para o primeiro filtro.

Critério	Descrição
CI.1	A publicação descreve uma revisão de literatura sobre fatores de produtividade no desenvolvimento de software.
CE.1	A publicação não atende ao critério de inclusão.

Os critérios de seleção do primeiro filtro (Tabela 20) foram utilizados para selecionar as publicações a partir da leitura de seu título e resumo (*abstract*). Da mesma forma que no estudo anterior, os critérios definidos para o primeiro filtro também foram simplificados, utilizando somente um critério de inclusão e um de exclusão. Dessa forma, as publicações que descreveram uma revisão de literatura sobre fatores de produtividade atenderam a esse primeiro filtro.

A *string* de busca utilizada retornou estudos secundários que não se caracterizavam como uma revisão sistemática de literatura. Por isso, os estudos secundários que não apresentaram em seu conteúdo, pelo menos a descrição de um processo definido de busca foram excluídos pelo segundo filtro. Esse critério também adotado por Kitchenham *et al.* (2010), em sua revisão terciária. Esse critério e os outros critérios do segundo filtro (Tabela 21) foram aplicados a partir da leitura completa das publicação selecionadas.

Tabela 21. Critérios de inclusão (CI) e exclusão (CE) para o segundo filtro.

Critério	Descrição
CI.1	A publicação é uma revisão sistemática de literatura, com pelo menos um processo de busca definido, sobre fatores de produtividade no desenvolvimento de software
CE.1	A publicação não é um estudo secundário ou não tem um processo definido de busca.
CE.2	A publicação não é um estudo secundário sobre os fatores de produtividade no desenvolvimento de software.
CE.3	A publicação não possui a lista de fatores de produtividade extraídos.
CE.4	A publicação não é um artigo científico, por exemplo, é um capítulo de um livro, portanto, não garantindo que houve uma revisão de outro pesquisador (*peer review*).
CE.5	A publicação não está em inglês ou não está disponível *online*.

4.1.3 Estratégia para extração dos dados

Os dados das publicações selecionadas serão extraídos para que possam ser analisados, sintetizados e interpretados afim de responder cada uma das subquestões de pesquisa. Assim como também responder à questão principal de pesquisa desta revisão terciária. Os dados extraídos nesta revisão terciária foram classificados em dados da publicação, dados da classificação dos fatores e dados específicos dos fatores de produtividade.

Os dados da publicação ajudam a mapear com qual frequência e em quais veículos foram publicadas as revisões de literatura sobre fatores de produtividade no desenvolvimento de software. Os dados específicos da publicação extraídos estão descritos na Tabela 22.

Tabela 22. Dados específicos da publicação.

Código	Número que identifica a publicação extraída
Ano de Publicação	Ano em que foi publicado
Fonte	Veículo em que foi publicado
Título	Título da publicação
Autores	Os autores da publicação

Os dados sobre a classificação utilizada, com nomes e descrição das categorias utilizadas para os fatores de produtividade, são importantes para responder a subquestão *SQ1*. Além da classificação utilizada, também será extraído o autor da classificação, pois diferentes revisões podem ter utilizado uma classificação baseadas ou derivadas de um mesmo autor. Esses dados podem indicar se existe uma classificação comum adotada pelos pesquisadores. Os dados extraídos sobre a definição das métricas de produtividade estão apresentados na Tabela 23.

Tabela 23. Dados da classificação dos fatores de produtividade.

Classificação dos fatores	A classificação adotada
Autor original da classificação	Extrair o autor original da classificação

Por fim, os dados sobre como os fatores de produtividade foram nomeados e descritos, para responder a subquestão *SQ2*, e a quantidade de referências, para responder a subquestão *SQ3*, serão extraídos. Esses dados podem não estar presentes na publicação selecionada, uma vez que a sua ausência não constitui um critério de exclusão para as publicações. Os dados específicos dos fatores de produtividade extraídos estão apresentados na Tabela 24.

Tabela 24. Dados dos fatores de influência da produtividade.

Nome do fator	O nome utilizado para o fator de produtividade
Descrição do fator	A descrição utilizada para o fator de produtividade
Quantidade de referências do fator	A quantidade de referências identificadas na revisão de literatura para cada fator ou grupo de fatores encontrados

4.1.4 Estudos selecionados após execução da revisão terciária

A execução da estratégia de busca e seleção definida nesta revisão terciária envolveu o trabalho de dois pesquisadores. A busca nas bibliotecas digitais foi realizado por um pesquisador e a aplicação dos critérios contou com o apoio de outro pesquisador. Para avaliar a confiança da aplicação dos critérios definidos [Siegel e Castellan 1988], os dois pesquisadores aplicaram os critérios de seleção de forma independente em uma amostra aleatória de 30 publicações, utilizando o teste estatístico *Kappa* [Cohen 1960] para avaliar a concordância. O resultado dessa avaliação de concordância, utilizando dados do primeiro filtro, foi significativa ($kappa = 0.783$), de acordo com a sugestão de interpretação desse valor proposta por Landis e Koch (1977).

Foram encontradas 353 publicações após a busca nas bibliotecas digitais selecionadas. Removendo as publicações duplicadas, o total de publicações selecionadas para filtragem ficou em 240. Dessas publicações, 221 não atenderam ao critério de inclusão do primeiro filtro e, por isso, foram excluídas. As 19 publicações restantes foram lidas integralmente e, no final da seleção, somente 3 publicações atenderam aos critérios do segundo filtro. Uma das publicações de referência utilizada para a definição da *string* de busca não apareceu em nenhuma máquina de busca das bibliotecas digitais. Após a busca direta pela publicação, notou-se que a mesma não estava indexada em nenhuma delas, portanto decidiu-se incluí-la manualmente entre as publicações selecionadas (Figura 16).

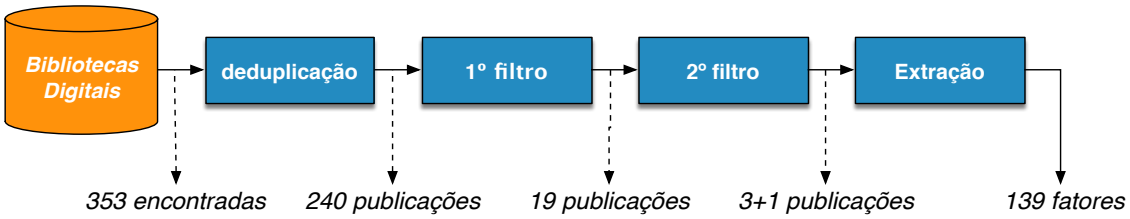

Figura 16. Processo de seleção das publicações desta revisão terciária.

As quatro publicações selecionadas com estudos secundários envolvendo os fatores de influência da produtividade continham um total de 139 grupos de fatores diferentes. Cada autor, em sua revisão, realizou uma classificação dos fatores encontrados em grupos de fatores. O período dessas publicações é bem recente (2008 a 2015) e reflete o aumento do interesse pelos fatores de influência da

produtividade por parte dos pesquisadores. Os detalhes das publicações selecionadas, contendo o ano de publicação, os autores e o título de cada publicação estão apresentadas na Tabela 25.

Tabela 25. Publicações com estudos secundários sobre fatores de produtividade.

Revisão	Ano	Autores	Publicação
R1	2008	Wagner, S., Ruhe, M.	*A Systematic Review of Productivity Factors in Software Development*
R2	2009	Trendowicz, A., Münch, J.	*Factors Influencing Software Development Productivity – State-of-the-Art and Industrial Experiences. Advances in Computers*
R3	2010	Paiva, E., Barbosa, D., Lima, R., Albuquerque, A.	*Factors that Influence the Productivity of Software Developers in a Developer View*
R4	2015	Dutra, A. C. S., Prikladnicki, R., Franca, C.	*Measuring productivity in agile software development process: a scoping study*

4.2 RESULTADOS OBTIDOS

4.2.1 SQ1. Qual foi a classificação adotada para organizar os fatores encontrados?

Os autores das revisões selecionadas agruparam os fatores extraídos em fatores unificados com significado similar, visando facilitar a análise dos fatores extraídos dos estudos primários. Esses fatores foram posteriormente agrupados em categorias. Em duas revisões, as categorias foram novamente agrupadas, criando uma classificação hierárquica dos fatores de influência extraídos. Somente a revisão de Paiva *et al.* (2010) (revisão *R3*) não adotou uma classificação além da unificação dos fatores. Das classificações adotadas, uma a revisão *R1* definiu e utilizou uma classificação própria, enquanto as outras duas (*R2* e *R4*) basearam sua classificação em outros estudos. As classificações adotadas nas revisões de literatura identificadas estão apresentadas na Tabela 26.

Wagner e Ruhe (2008) (revisão *R1*) classificaram os fatores encontrados em dois grandes grupos: (i) fatores técnicos, que são os fatores relativos ao produto, processo e ao ambiente de desenvolvimento; e (ii) fatores não técnicos (*soft*), ou seja, os fatores presentes de forma intangível na equipe de desenvolvimento e no

ambiente de trabalho. Esses fatores não técnicos foram considerados pelos autores também como fatores humanos.

Tabela 26. Classificações utilizadas nas revisões de literatura sobre fatores.

Revisão	Classificação
R1	**Fatores Técnicos:** • Fatores de Produto • Fatores de Processo • Fatores do Ambiente de Desenvolvimento **Fatores não-Técnicos:** • Fatores da Cultura Organizacional • Fatores da Cultura da Equipe • Fatores de Capacidades e Experiência • Fatores do Ambiente de Trabalho • Fatores do Projeto
R2	**Fatores de Influência:** • Fatores de Produto • Fatores de Pessoal • Fatores de Projeto • Fatores de Processo **Fatores de Contexto**
R3	*Sem classificação definida*
R4	**Fatores de Estados Emergentes de Equipe** **Fatores de Características Individuais** **Fatores de Tarefas de Suporte**

Trendowicz e Münch (2009) (revisão *R2*) dividiram primeiramente os fatores extraídos em fatores de contexto e fatores de influência. Segundo os autores, dado um modelo de produtividade, os fatores considerados pelo modelo são os fatores de influência; enquanto que os fatores ausentes do modelo são os fatores de contexto, fatores presentes no contexto e considerados constantes para o modelo definido. Os autores ainda subdividiram os fatores de influência em quatro grupos: fatores do produto, fatores de pessoal, fatores do projeto e fatores do processo. Essa classificação foi baseada em classificações de outros autores: Fenton e Pfleeger (1998), Jones (2005) e Ruhe *et al.* (2003).

Dutra *et al.* (2015) (revisão *R4*) categorizaram os fatores extraídos de acordo com a unidade de análise indicada no estudo primário e classificaram em três grupos: (i) fatores de estados emergentes de equipe, (ii) fatores de características individuais e (iii) fatores de tarefas de suporte. Essa classificação foi baseada no trabalho de Marks *et al.* (2001), visando estudar os fatores de influência das equipes de alto desempenho dentro do desenvolvimento de *software*.

Portanto, respondendo à subquestão *SQ1*, apesar de ser possível observar algumas semelhanças entre as classificações adotadas, especialmente entre as revisões *R1* e *R2*, elas possuem estruturas diferentes. Dessa forma, não é possível afirmar que existe uma classificação comum adotada pelas revisões sistemáticas sobre os fatores de influência da produtividade na literatura científica.

4.2.2 SQ2. Quais são os fatores de influência na produtividade pesquisados?

A descrição dos fatores nos estudos primários é frequentemente incompleta e limitada somente ao nome do fator, segundo Trendowicz e Münch (2009) (revisão *R2*). Dessa forma, para uma análise dos fatores encontrados, os autores das publicações selecionadas utilizaram estratégias de unificação dos fatores extraídos dos estudos primários. Somente depois dessa unificação, é que esses fatores unificados foram agrupados de acordo a classificação hierárquica adotada por cada revisão sistemática (vide subquestão *SQ1*).

Wagner e Ruhe (2008) extraíram 51 fatores, agrupados através da utilização de termos sinônimos. Trendowicz e Münch (2009) extraíram 246 fatores, agrupando-os de acordo com o nome e descrição utilizados pelos estudos primários. Paiva *et al.* (2010) realizaram a extração de 32 fatores, mas sem explicar o processo utilizado. Dutra *et al.* (2015) obtiveram 15 fatores e agruparam os fatores de acordo com a sua similaridade semântica.

Da mesma forma que nas revisões selecionadas, também foi necessário unificar os fatores extraídos dessas revisões, pois fatores similares apresentavam pequenas diferenças no nome, descrição ou na forma de medição. Para tanto, a estratégia utilizada foi unificar fatores com nome ou descrição igual ou similar. Após isso, os fatores que não se unificaram com nenhum outro foram agrupados em fatores chamados *outras características do indivíduo, da equipe, do projeto e da organização*. Por fim, os fatores unificados foram classificados em técnicos, humanos

e organizacionais. A Tabela 27, Tabela 28 e Tabela 29 apresentam o resultado dessa extração e classificação.

Tabela 27. Fatores técnicos extraídos das revisões sistemáticas.

Fator Unificado	Fator Extraído
Capacidade e Experiência	*Experience (R3), Programming language experience (R2), Teamwork capabilities (R2), Project manager experience & skills (R2), Application Experience & Familiarity (R2), Overall personnel experience (R2), Tool experience (R2), Applications Experience (R1), Language and Tool Experience (R1), Manager Application Experience (R1), Platform Experience (R1), Analyst Capability (R1), Manager Capability (R1), Programmer Capability (R1)*
Conhecimento	*Knowledge (R4), Domain of the Application (R3), Task-specific expertise (R2)*

Os fatores técnicos (Tabela 27) identificados foram divididos em capacidade e experiência e em conhecimento. A capacidade e experiência representam a familiaridade e as habilidades individuais adquiridas no exercício das funções durante um projeto de desenvolvimento *software*; enquanto que o conhecimento representa os conhecimentos necessários para realização das tarefas demandadas ao engenheiro de *software* pela organização.

Tabela 28. Fatores humanos extraídos das revisões sistemáticas.

Fator Unificado	Fator Extraído
Objetivo Claro	*Clear Goals (R1), Goal Setting (R4)*
Diversidade	*Diversity (R4), Developer Temperaments (R1)*
Motivação	*Motivation (R3), Motivation (R4)*
Coesão e Comunicação na Equipe	*Communication (R4), Cohesion (R4), Communication (R3), Interpersonal Relationship (R3), Team Cohesion/communication (R2), Communication (R1), Team Cohesion (R1)*
Outras Características Individuais	*Attitudes (R4), Intelligence (R4), Learning ability (R4), Personality (R4), Emotional Intelligence (R4), Empathy (R4), Leadership Style (R4), Work satisfaction (R4), Commitment (R3)*
Outras Características da Equipe	*Mutual respect (R4), Self-efficacy (R4), Trust (R4), Autonomy (R4), Sense of Eliteness (R1), Team Identity (R1), Fairness (R1)*

Os fatores humanos (Tabela 28) identificados foram relacionados à equipe e aos indivíduos. A definição de objetivos claros, a diversidade de temperamentos, a

coesão e comunicação dentro da equipe são fatores de influência na produtividade. Outras características da equipe também influenciam, como o respeito mútuo e a confiança entre os membros da equipe. As características individuais como a motivação, atitudes, compromisso e relacionamento pessoal estão presente na lista dos fatores identificados.

Tabela 29. Fatores organizacionais extraídos das revisões sistemáticas.

Fator Unificado	Fator Extraído
Arquitetura	*Architecture (R3), Architecture Complexity (R2), Architecture Risk Resolution (R1)*
Complexidade	*Code complexity (R2), Complexity of interface to other systems (R2), Product Complexity (R1), User Interface (R1), Required Software Reliability (R1)*
Requisitos Consistentes	*Consistent Requirements (R3), Requirements Management (R2), Requirements Stability (R1)*
Complexidade e Tamanho do Banco de Dados	*Database Size & Complexity (R2), Database Size (R1)*
Desenvolvimento Descentralizado	*Decentralized development (R2), Physical Separation (R1)*
Restrições no Desenvolvimento	*Development Flexibility (R1), Execution Time Constraints (R1), Main Storage Constraint (R1)*
Ferramentas de Desenvolvimento	*Development Tool (R3), CASE tools (R2), Testing tools (R2), Use of Software Tools (R1)*
Tipo do Desenvolvimento	*Agile Methodology (R3), Type of Project (R3), Methodology (R3), Development Type (R2), Life cycle model (R2), Domain (R2)*
Documentação	*Documentation (R3), Documentation match to life-cycle needs (R1)*
Gestão do Conhecimento	*Shared Information (R4), Knowledge Management (R3)*
Modernidade	*Modernity (R3), Technological Gap (R3), Use of Modern Development Practices (R3)*
Maturidade do Processo	*Maturity Level (R4), Process maturity & stability (R2), Process Maturity (R1)*
Linguagem de Programação	*Programming Language (R3), Programming Language (R2), Programming Language (R1)*
Gestão do Projeto	*Managerial Involvement (R4), Project Management (R3)*
Tamanho do Projeto	*Project Size (R3), Project Duration (R1), Software Size (R1)*
Prototipação	*Prototyping (R3), Early Prototyping (R1)*

Fator Unificado	Fator Extraído
Reutilização de Código	*Code Reuse (R3), Quality of reused assets (R2), Reuse level (R2), Developed for Reusability (R1), Reuse (R1)*
Pressão do Cronograma	*Schedule pressure (R2), Schedule (R1)*
Tamanho da Equipe	*Team Size (R4), Team Size (R3), Team Size (R2), Average Team Size (R1)*
Facilidades de Telecomunicação	*Home Office (R3), Telecommunication Facilities (R1)*
Teste	*Test (R3), Testing (R2), Effective and Efficient V&V (R1)*
Fragmentação do Tempo	*E-Factor (R1), Time Fragmentation (R1)*
Treinamento	*Training (R3), Training level (R2)*
Troca de Pessoal	*Turnover (R4), Staff turnover (R2), Turnover (R1)*
Ambiente de Trabalho	*Work Environment (R4), Workstation (R4), Proper Workplace (R1), Camaraderie (R1)*
Outros Fatores do Projeto	*Guard Activities (R4), Work breakdown (R4), Target plataform (R2), Reviews & inspections (R2), Team structure (R2), Precedentedness (R1), Completeness of Design (R1), Platform Volatility (R1), Hardware Concurrent Development (R1), Product Quality (R1)*
Outros Fatores da Organização	*Organizational Commitment (R4), Benefits (R3), Internet Access (R3), Physical Location (R3), Salary (R3), Credibility (R1), Respect (R1), Support for Innovation (R1)*

Por fim, os fatores organizacionais (Tabela 29) foram os que apareceram em maior quantidade. Esses fatores envolvem características diretas da organização, tais como a localização física e o ambiente de trabalho, mas também características dos processos adotados e projetos desenvolvidos pela organização, tais como a maturidade do processo e o tamanho do projeto.

Respondendo à subquestão *SQ2*, é possível afirmar que existem pelo menos 35 grupos de fatores de influência na produtividade de *software*, extraídos a partir de quatro revisões sistemáticas, comumente investigados na literatura científica. Os fatores foram ainda classificados em fatores técnicos, humanos e organizacionais. Os fatores organizacionais foram, de longe, o grupo de fatores mais investigado pelos pesquisadores.

4.2.3 SQ3. Quais foram os fatores de influência mais frequentemente pesquisados?

As revisões sistemáticas selecionadas informaram a quantidade de referência por fator identificado, sendo exceção a revisão de Paiva *et al.* (2010) (revisão *R3*). Essa quantidade de referências por fator pode fornecer uma indicação da relevância do fator dentro da comunidade científica. Obviamente, a linha divisória entre algumas categorias é difusa e, por isso, alguns fatores podem alocar-se em múltiplas categorias [Wagner e Ruhe 2008]. Além disso, nem todas as revisões sistemáticas disponibilizaram a lista contendo todas as referências selecionadas, não permitindo assim uma contagem mais precisa das referências por fator unificado. Por isso, preferiu-se adotar, como representação da quantidade de referências, o maior valor da quantidade de referências entre as revisões sistemáticas selecionadas para um mesmo fator unificado.

Dentre os fatores técnicos (Tabela 30), a capacidade e a experiência foram o fator com mais referências. Na revisão de Trendowicz e Münch (2009) (revisão *R2*) foram encontradas 16 referências sobre a experiência na linguagem de programação como fator de influência na produtividade. Na mesma revisão sistemática, a perícia em tarefas específicas apresentaram as 5 referências de conhecimento que influenciam a produtividade.

Tabela 30. Quantidade de referências dos fatores técnicos extraídos.

Fator Unificado	Quantidade de Referências
Capacidade e Experiência	16
Conhecimento	5

Os fatores humanos (Tabela 31) tiveram predominância de referências em relação a coesão e comunicação da equipe de desenvolvimento, ainda na revisão de Trendowicz e Münch (2009) (revisão *R2*). Foram 18 referências sobre a organização da equipe de desenvolvimento de *software*. A motivação foi o fator humano individual com maior número de referências (4).

Tabela 31. Quantidade de referências dos fatores humanos extraídos.

Fator Unificado	Quantidade de Referências
Coesão e Comunicação na Equipe	18
Outras Características da Equipe	5
Motivação	4
Objetivo Claro	3
Diversidade	2
Outras Características Individuais	2

Nos fatores organizacionais (Tabela 32) muitas foram as referências encontradas por fator. Segundo a revisão Trendowicz e Münch (2009) (revisão *R2*), a pressão do cronograma é o fator de influência da produtividade com a maior quantidade de estudos (43). Além dele, destacam-se em quantidade de referências a linguagem de programação (29) e o tamanho da equipe de desenvolvimento (28) como fatores de influência na produtividade.

Tabela 32. Quantidade de referências dos fatores organizacionais extraídos.

Fator Unificado	Quantidade de Referências
Pressão do Cronograma	43
Linguagem de Programação	29
Tamanho da Equipe	28
Reutilização de Código	17
Troca de Pessoal	15
Tipo do Desenvolvimento	14
Maturidade do Processo	13
Ferramentas de Desenvolvimento	12
Outros Fatores do Projeto	11
Arquitetura	9
Complexidade e Tamanho do Banco de Dados	9
Desenvolvimento Descentralizado	9
Complexidade	8

Fator Unificado	Quantidade de Referências
Restrições no Desenvolvimento	7
Modernidade	7
Treinamento	7
Requisitos Consistentes	6
Teste	6
Outros Fatores da Organização	6
Tamanho do Projeto	4
Ambiente de Trabalho	3
Documentação	2
Facilidades de Telecomunicação	2
Fragmentação do Tempo	2
Gestão do Conhecimento	1
Gestão do Projeto	1
Prototipação	1

Além disso, é importante ressaltar que a quantidade de referências depende da abrangência de cada revisão sistemática. A revisão de Wagner e Ruhe (2008) (revisão *R1*) avaliou 50 estudos primários; já a revisão Trendowicz e Münch (2009) (revisão *R2*) extraiu seus fatores de 142 estudos primários; enquanto a revisão de Paiva *et al.* (2010) (revisão *R3*) avaliou 11 estudos primários; por fim, a revisão de Dutra *et al.* (2015) (revisão *R4*) reviu 15 estudos primários.

Finalmente, respondendo a subquestão *SQ3*, é possível afirmar que os fatores organizacionais foram os mais pesquisados, especialmente com relação à pressão do cronograma, linguagem de programação e o tamanho da equipe de desenvolvimento. Entre os fatores técnicos, a coesão e comunicação da equipe foi o fator com mais estudos primários. Por fim, a capacidade e experiência dos membros da equipe de desenvolvimento foram os fatores humanos mais investigados pelos pesquisadores da produtividade de *software*.

4.3 DISCUSSÃO DOS RESULTADOS

4.3.1 QP. Quais são os fatores de produtividade encontrados pelos estudos secundários existentes sobre fatores de produtividade no desenvolvimento de software?

Esta revisão terciária teve como objetivo identificar os fatores de influência na produtividade e, além disso, classificar os fatores extraídos em fatores técnicos, humanos e organizacionais, de acordo com a classificação proposta (Capítulo 2, subseção 2.4.1). Foram identificados e extraídos 35 fatores de quatro revisões sistemáticas sobre fatores de influência na produtividade de *software*. A partir da classificação desses fatores em técnicos, humanos e organizacionais, é possível afirmar que os pesquisadores investigaram mais fatores organizacionais do que fatores técnicos ou fatores humanos. Considerando a quantidade de estudos, essa observação é fortalecida, pois somente a soma da quantidade de referências de dois fatores organizacionais é superior à soma de todas as referências dos estudos primários sobre fatores técnicos e humanos. Por fim, apesar de ser possível observar algumas semelhanças entre as classificações adotadas, não é possível afirmar que existe uma classificação comum adotada pelas revisões sistemáticas sobre os fatores de influência da produtividade na literatura científica.

Esta revisão terciária, até o melhor de nosso conhecimento, é o primeiro estudo terciários sobre os fatores de influência da produtividade de *software*. Foram identificados na literatura científica quatro estudos secundários sobre fatores de produtividade: Wagner e Ruhe (2008) (revisão *R1*), Trendowicz e Münch (2009) (revisão *R2*), Paiva *et al.* (2010) (revisão *R3*) e Dutra *et al.* (2015) (revisão *R4*).

Wagner e Ruhe (2008) (revisão *R1*) realizaram uma revisão sistemática sobre os fatores de produtividade no desenvolvimento de software. O objetivo principal foi identificar os principais fatores que influenciam a produtividade de software, com uma especial consideração para os fatores humanos. Os autores observaram que, embora o esforço de comunicação tenha um impacto positivo na produtividade, a dificuldade de comunicação aumenta significativamente com o aumento da quantidade de pessoas envolvidas. Além disso, observam que a experiência, amplamente considerada como importante nas entrevistas de emprego, teve uma quantidade irrelevante de estudos na literatura científica. Dessas

conclusões, a comunicação ainda continua como forte fator pesquisado, pois além de formar um grupo à parte, juntamente com a coesão com 18 referências, também pode ser encontrada quando consideramos outros fatores, de uma certa forma relacionados, como a empatia e o relacionamento interpessoal. A experiência, considerando todas as revisões, aparece em vários estudos, relacionada com diferentes aspectos tais como a experiência na linguagem de programação, nas ferramentas, no negócios, etc.

Trendowicz e Münch (2009) (revisão *R2*) fizeram uma revisão sistemática sobre os fatores que têm os maiores impactos na produtividade de *software*. Essa revisão avaliou publicações científicas assim como também avaliou a experiência da indústria através dos estudos realizados pelo *Fraunhofer Institute* for *Experimental Software Engineering* (*IESE*)[14] sobre a produtividade de *software*. Os resultados obtidos através da experiência na indústria mostram que a qualidade dos requisitos e as capacidades e experiências da equipe de desenvolvimento são os principais fatores da produtividade de *software*. Segundo os autores, seu maior resultado é que o sucesso do projeto de *software* ainda depende das pessoas envolvidas, o principal fator de influência na produtividade. Essa conclusão diverge das observações realizadas nesta revisão terciária em dois pontos: (i) a quantidade de fatores diferentes identificados nas quatro revisões e (ii) a quantidade de referências de todos os fatores extraídos, pois ficou claro que a pesquisa de fatores de influência esteve muito mais voltada para os fatores organizacionais. Depois das pessoas, os autores apresentam as ferramentas e métodos como os outros maiores fatores de influência, resultado em maior concordância com esta revisão terciária.

Paiva *et al.* (2010) (revisão *R3*) fizeram um estudo sobre os fatores de influência da produtividade na visão dos desenvolvedores de *software*. Os fatores a serem avaliados pelos desenvolvedores foram extraídos a partir de uma revisão sistemática realizada pelos autores. A partir dos 32 fatores extraídos, os autores realizaram um *survey* com 77 desenvolvedores profissionais de 11 organizações desenvolvedoras de *software*. Os resultados da revisão sistemática, apesar de não informar a quantidade de referências, apontaram a experiência, a linguagem de programação, os requisitos consistentes, a reutilização de código e o tamanho de

[14] http://www.iese.fraunhofer.de/en.html

projeto como os fatores mais pesquisados. Nesta revisão terciária, esses fatores tiveram uma grande quantidade de referências (superiores a 15 referências), exceção aos fatores tamanho do projeto e requisitos consistentes. Além disso, os autores apontaram a motivação e o comprometimento como fatores fundamentais para qualquer profissional desempenhar suas tarefas de forma produtiva, baseado na visão dos próprios desenvolvedores de *software*.

Dutra *et al.* (2015) (revisão *R4*) realizaram uma revisão sistemática sobre os fatores de influência das equipes de alto desempenho no desenvolvimento de software. Os resultados, a partir de 15 estudos primários, demonstraram que a comunicação na equipe e a motivação do indivíduo como os fatores mais pesquisados dentro do contexto de equipes de alto desempenho. Além desses fatores, somente tiveram mais de um estudo os fatores de autonomia, diversidade, tamanho da equipe, respeito mútuo e personalidade. Apesar do escopo mais fechado, os resultados replicam resultados obtidos anteriormente. A comunicação como um fator relevante está em concordância com o trabalho de Wagner e Ruhe (2008), assim como a motivação como fator relevante está em concordância com a opinião dos desenvolvedores, segundo o *survey* de Paiva *et al.* (2010).

As conclusões destas revisões sistemáticas tendem a considerar a comunicação (*R1, R4*) na equipe e a motivação (*R3, R4*) do membro da equipe como os fatores mais relevantes para a produtividade de *software*. Esses fatores dependem basicamente do comportamento das pessoa (*R2*). Porém, considerando a quantidade de fatores diferentes e a quantidade de estudos primários sobre esses diferentes fatores, podemos concluir que são os fatores organizacionais os mais investigados pelos pesquisadores. Essa divergência, porém, está de acordo com a afirmação de Meyer *et al.* (2014), que diz que a maioria dos trabalhos sobre produtividade de *software* focam mais na organização e nos projetos de *software*, do que no desenvolvedor, apesar de sua importância fundamental ao processo de desenvolvimento de *software*. Fica em aberto, portanto, qual dos fatores, se os técnicos, humanos ou organizacionais, tem maior importância para a produtividade de *software*.

4.4 AMEAÇAS À VALIDADE

Todo estudo possui ameaças que podem afetar a validade dos seus resultados [Wohlin *et al.* 2012] e assim, como no mapeamento sistemático apresentado no capítulo anterior, dentre as ameaças à validade deste estudo, podemos novamente destacar as principais. A primeira principal ameaça está relacionada à validade de conclusão dessa revisão terciária quanto a generalização dos seus resultados, pois a estratégia de busca pode não ter coletado alguns artigos relevantes, afetando os resultados. Para isso foram utilizadas cinco diferentes bibliotecas digitais, baseado na experiência relatada por Dybå *et al.* (2007) . A outra principal ameaça à validade dos resultados é a possibilidade do autor deste estudo ter introduzido seu viés durante a execução do protocolo de revisão. Para mitigar essa ameaça, o processo de execução do protocolo de revisão foi revisado por outro pesquisador mais experiente.

Além dessas, uma outra ameaça foi relacionada com as frequências encontradas. Como nem todas as revisões sistemáticas disponibilizaram a lista contendo todas as referências selecionadas, as frequências apresentadas dos fatores pesquisados podem não corresponder a realidade. Ainda assim, preferiu-se utilizar as quantidades disponíveis como uma forma de aproximar-se da resposta a questão investigada. Por fim, a falta de detalhes constante nas revisões sistemáticas selecionadas podem ter influenciado nas interpretações adotadas. Todas as respostas foram construídas a partir do maior nível de detalhe apresentados, recorrendo aos relatórios técnicos quando os mesmos estavam disponíveis.

4.5 CONSIDERAÇÕES FINAIS

Esse capítulo apresentou uma revisão terciária sobre os fatores de influência na produtividade de *software*. Quatro estudos secundários foram identificados na literatura contendo revisões sistemáticas sobre fatores de influência na produtividade de *software*. Diversos fatores foram identificados, extraídos, unificados e classificados a partir dessas revisões sistemáticas. Os fatores organizacionais apresentaram-se em maior quantidade e com mais estudos primários. Apesar disso, alguns autores das revisões sistemáticas destacaram a importância das pessoas com a sua motivação como fator primordial para a produtividade no desenvolvimento de *software*. Por fim, essa divergência aumenta a

motivação desta pesquisa de doutorado na busca pelos fatores preponderantes dentro do processo de desenvolvimento de *software*.

Capítulo 5 – PERCEPÇÃO DA PRODUTIVIDADE DO DESENVOLVEDOR PELOS GERENTES DE *SOFTWARE*

Este capítulo apresenta um estudo qualitativo para investigar a percepção dos gerentes de software das organizações quanto ao conceito, identificação e utilidade da produtividade dos desenvolvedores.

5.1 INTRODUÇÃO

Medir a produtividade do *software* é um elemento central para os gerentes de *software*, pois é esta medição que possibilita comparar a produtividade de seus projetos e desenvolvedores. Enquanto não existem métricas confiáveis ou precisas, os gerentes de *software* utilizam suas percepções para a tomada de decisões gerenciais. Por isso é importante entender como essas percepções são definidas, identificadas e utilizadas dentro da organização.

Este capítulo apresenta um estudo para investigar como os gerentes das organizações de *software* conceituam, identificam e como utilizam as suas percepções sobre a produtividade dos seus desenvolvedores. Os resultados desse estudo esclarecem o contexto da produtividade dentro de organizações, auxiliando na análise e compreensão não só desse, mas dos outros estudos.

5.2 PLANO DE ESTUDO

O objetivo desse estudo foi investigar como a organização, sob o ponto de vista de seus gerentes de *software*, entende, identifica e utiliza as percepções sobre a produtividade dos seus desenvolvedores. Os gerentes de *software* foram escolhidos para representar o ponto de vista organizacional, pois são os responsáveis por determinar os rumos dos projetos e dos desenvolvedores de *software* dentro da organização.

As questões de pesquisa desse estudo estão apresentadas na Tabela 33, todas utilizando o ponto de vista do gerente de *software*. Para facilitar a investigação da questão principal (*QP*), ela foi desdobrada em três subquestões. A questão secundária *SQ1* visou identificar qual o conceito de produtividade os gerentes de

software possuem. Como os gerentes de software identificam a produtividade dos desenvolvedores, foi o propósito da subquestão *SQ2*. Por fim, a subquestão *SQ3* teve como propósito esclarecer a utilidade da informação da produtividade dos desenvolvedores para a organização, identificando quais ações são tomadas pelos seus gerentes de *software* com essas informações.

Tabela 33. Questões de pesquisa para os gerentes de *software*.

QP	Como os gerentes de *software* conceituam, identificam e utilizam a produtividade dos desenvolvedores de *software* da organização?
SQ1	O que é produtividade do desenvolvedor?
SQ2	Como é identificada a produtividade do desenvolvedor?
SQ3	Qual a utilidade das percepções sobre a produtividade do desenvolvedor para a organização de *software*?

5.2.1 Procedimentos de coleta de dados

A coleta de dados foi realizada em cada uma das organizações selecionadas utilizando entrevistas semiestruturadas com seus gerentes de *software*. Para realização destas coletas, foi assinado um acordo de confidencialidade com todas as organizações, resguardando desse modo as organizações participantes. Todos os gerentes de *software* entrevistados também assinaram um termo de consentimento livre e esclarecido (TCLE).

Todos os gerentes da área de desenvolvimento de *software* das organizações selecionadas foram entrevistados, totalizando doze gerentes de *software* nas três organizações, sendo cinco na Organização 1 (Org. 1), três na Organização 2 (Org. 2) e quatro na Organização 3 (Org. 3). Em cada organização, os gerentes de *software* de maior posto hierárquico foram entrevistados primeiro. A Tabela 34 apresenta os gerentes de *software* participantes deste estudo, onde cada participante é relacionado com sua organização, o título do seu cargo, nível de educação e a quantidade de anos trabalhados na organização. É possível observar que a maioria dos gerentes têm mais de 10 anos (exceção ao participante *g8*) em suas organizações, sugerindo que os gerentes já possuem uma experiência considerável em suas organizações.

Tabela 34. Gerentes de *software* entrevistados.

Id.	Org.	Título do Cargo	Educação	Anos na Org.
g1	1	Diretor do Departamento	Bacharel	23
g2	1	Gerente de Desenvolvimento de SW	Bacharel	10
g3	1	Coordenador de Desenvolvimento de SW	Bacharel	20
g4	2	Diretor Técnico	Mestre	14
g5	2	Chefe do Departamento de Software	Bacharel	32
g6	2	Gerente de Desenvolvimento de Software	Bacharel	10
g7	2	Gerente de Qualidade de Software	Bacharel	18
g8	2	Gerente de Negócios de Software	Bacharel	5
g9	3	Gerente de Portfólio	Bacharel	16
g10	3	Gerente de Desenvolvimento de Software	Bacharel	13
g11	3	Gerente de Qualidade de Software	Mestre	13
g12	3	Coordenador de Desenvolvimento de SW	Bacharel	10

Além disso, todos os gerentes possuem bacharelado, sendo dois com mestrado em informática, exceção aos gerentes *g1* e *g5*, que respectivamente possuem bacharelado em economia e contabilidade. Os participantes *g1*, *g4* e *g9*, com a posição hierárquica mais elevada em suas respectivas organizações, possuem responsabilidades com os projetos de desenvolvimento de *software* e também serviços relacionados à infraestrutura tecnológica, à gestão de demandas do cliente e, em alguns casos, à gestão comercial. O participante *g5* é responsável pela gestão da divisão de desenvolvimento de *software* em sua organização (Org. 2), ainda abaixo hierarquicamente do participante *g4*, porém tem outros serviços relacionados à tecnologia da informação, como por exemplo, a gestão da infraestrutura de rede dentro e fora da organização. Dessa forma, o participante *g5* é comparável aos participantes *g1*, *g4* e *g9*.

O participante *g8* não tem similaridades com esse grupo, mas lida com as relações de negócio com os clientes, dessa forma ele traz a percepção do cliente sobre os produtos de *software* desenvolvidos de sua organização. Os participantes *g7* e *g11* são gerentes de qualidade de *software* em suas organizações e são responsáveis por equipes que avaliam a qualidade dos produtos de *software*, coordenando suas equipes para realizar atividades como a revisão de código (*code*

review) e testes de caixa preta (*black box testing*). Por fim, os participantes *g2*, *g6* e *g10* são gerentes de *software* que lidam somente com o desenvolvimento de *software*, mas não diretamente com as equipes de desenvolvimento, responsabilidade dos líderes de projeto de *software*.

Todas as entrevistas usaram um roteiro semiestruturado e foram organizadas em duas partes (Tabela 35). A primeira parte consistiu em algumas questões introdutórias sobre os antecedentes dos participantes, ambientando-os na entrevista, de acordo com as diretrizes propostas por Runeson *et al.* (2012); já a segunda parte compreendeu as principais questões da entrevista. Para o *SQ1*, exploramos variações da pergunta com pontos de vista diferentes, visando melhor distinguir se as crenças variavam de acordo com as convicções pessoais, com práticas recorrentes ou pelas diretrizes de organização. Para o *SQ2*, também exploramos diferentes pontos de vista, questionando sobre como a organização percebe que um desenvolvedor é produtivo e como eles comparam a produtividade de desenvolvedores da mesma equipe. Por fim, para a *SQ3*, questionamos diretamente a utilidade da informação da produtividade para a organização.

Tabela 35. Roteiro semiestruturado utilizado nas entrevistas com os gerentes de *software*.

Subquestão de Pesquisa	Descrição
SQ1	Para você, de um modo geral, o que é produtividade? Para você, o que é produtividade de software? Para você, o que é produtividade do desenvolvedor de software? Para a organização, o que é produtividade do desenvolvedor de software?
SQ2	Como a organização percebe que um desenvolvedor é produtivo? Como a organização compara a produtividade entre desenvolvedores de uma mesma equipe?
SQ3	Em que pode ser útil a informação sobre a produtividade do desenvolvedor para a organização?

As entrevistas foram realizadas simultaneamente com o processo de análise. Foram realizados ciclos compostos de entrevistas, transcrições e análise. Dois pesquisadores realizaram as transcrições, conforme também sugerido por Runeson *et al.* (2012), o que contribuiu para uma melhor compreensão dos dados coletados.

5.2.2 Procedimentos de análise de dados

As transcrições das entrevistas foram importadas e analisadas com apoio do software Atlas.ti[15]. A análise qualitativa realizada neste estudo utilizou procedimentos da Teoria Fundamentada (*Grounded Theory – GT*). A Teoria Fundamentada emprega um conjunto de procedimentos sistemáticos de coleta e análise de dados para gerar, preparar e validar teorias sobre os fenômenos essencialmente sociais [Glaser e Strauss 1968]. Dos procedimentos de GT, foram utilizadas a codificação aberta, a codificação axial, mas não a codificação seletiva. A codificação aberta envolve a repartição, análise, comparação, conceituação, e a categorização dos dados. A codificação axial examina as relações entre as categorias identificadas. A codificação seletiva realiza o refinamento de todo esse processo, identificando uma categoria central com o qual todos as outras categorias são relacionadas. Finalmente, optou-se por não eleger uma categoria central, pois que nenhuma categoria emergiu naturalmente dos dados, sugerindo que uma possível saturação teórica não havia sido atingida, o que é uma regra central de GT [Strauss e Corbin 2008]. Por esta razão, não é possível afirmar que o método GT foi aplicado de forma completa, mas que utilizou-se alguns procedimentos específicos de GT.

O autor desta tese fez a codificação aberta, associando códigos com citações de transcrições, e a codificação axial, onde os códigos foram mesclados e agrupados em categorias mais abstratas. Uma vez preparados, os códigos e as redes identificadas nas categorias, incluindo breves *memos* descrevendo as relações, foram revistos e analisados por outros pesquisadores. Após esse procedimento, um novo turno de entrevistas, transcrições e análise foi iniciado. Este ciclo de processo foi repetido até não haver mais participantes disponíveis entre o conjunto de participantes disponibilizados pelas organizações.

5.3 RESULTADOS OBTIDOS

5.3.1 SQ1. O que é produtividade do desenvolvedor?

> *"[O desenvolvedor para ser produtivo,] é ele ter qualificação suficiente, tanto técnica quanto comportamental para desenvolver sua atividade de forma efetiva" – g11*

[15] http://atlasti.com

> *"[A produtividade do desenvolvedor é] entregar no prazo, com qualidade, garantindo a satisfação do cliente. Eu acho que é essa visão de produtividade que a pessoa tem que ter" – g12*

> *"Então ser produtivo é isso: fazer [de forma] que não dê retrabalho. Entregou e não fica dando retrabalho" – g9*

A produtividade do desenvolvedor segundo os gerentes de *software* das organizações é o desenvolvedor que possui **qualificação técnica e comportamental**, **entregando suas tarefas no prazo**, *atendendo* **as expectativas das partes interessadas** e sem que haja **necessidade de retrabalho**.

A **qualificação técnica e comportamental** foi considerada um dos componentes da definição da produtividade dos desenvolvedores de organizações de *software*. Os gerentes de *software* têm uma expectativa diferente de produtividade de cada desenvolvedor da organização, de acordo com a sua qualificação técnica e comportamental. Os desenvolvedores com maior qualificação técnica estão associados com a capacidade de serem mais produtivos, provavelmente porque essa qualificação técnica habilita aos desenvolvedores maior flexibilidade nas tarefas que podem desempenhar. Como exemplo, um desenvolvedor que conhece mais linguagens de programação pode assumir mais tarefas de programação que um outro que só sabe uma única linguagem. Além disso, a qualificação comportamental também foi citada, considerando comportamentos como foco, concentração, tranquilidade, comprometimento.

> *"O primeiro ponto importante é a qualificação das pessoas, esse é o ponto central. Você vai investir nessa qualificação até para que você possa cobrar a produtividade [dos desenvolvedores]" – g5*

> *"O comportamental vai muito pro lado do foco do desenvolvedor de ele saber que existe um cliente que esta esperando aquele produto e que ele precisa cumprir aquele cronograma, então ele precisa ser comprometido, tudo isso eu acho que impacta na produtividade dele como desenvolvedor" – g11*

> *"[A produtividade do desenvolvedor] é a capacidade que ele tem de se concentrar, ficar tranquilo e produzir aquilo que foi pedido num tempo aceitável" – g1*

Entregar a tarefa no prazo foi o componente da definição de produtividade dos desenvolvedores mais citado pelos gerentes de *software*. Embora alguns gerentes tenham mencionado a realização da tarefa no menor tempo possível, para a maioria deles a entrega no prazo é suficiente. Porém, também é levado em consideração a complexidade das tarefas para a produtividade do desenvolvedor, pois, para o gerentes, quanto mais complexa for a tarefa, mais produtivo será o desenvolvedor que complete essa tarefa.

> *"Considera-se produtividade conseguir fazer as tarefas da programação que foram acertadas no prazo" – g3*
>
> *"O desenvolvedor produtivo é o aquele que consegue entregar bem, dentro do prazo" – g9*
>
> *"[Produtividade do desenvolvedor] é entregar um pedido, dentro do menor tempo possível" – g1*
>
> *"A única coisa que vai estar mudando seria a nossa percepção [da produtividade do desenvolvedor] diz respeito à complexidade do que foi passado para ele." – g7*

Os desenvolvedores que entregam as tarefas **atendendo as expectativas das partes interessadas** preenchem o perfil de produtividade considerado pelos gerentes de *software*. Quanto mais tarefas entregues no prazo, com qualidade e sem custos desnecessários para a organização, mais o desenvolvedor é considerado produtivo pelos gerentes de *software* de sua organização. Sob esse componente, a qualidade da tarefa entregue pelo desenvolvedor é parte essencial para garantir a satisfação do cliente.

> *"É entregar um pedido, [...] com a melhor qualidade que atenda às necessidades dos nossos clientes." – g1*
>
> *"Tem que ser de alta qualidade, [pois] não adianta eu produzir um software que [quando] eu colocar em produção, ele estará cheio de bugs, parando, atrapalhando o serviço do cliente, gerando a insatisfação do cliente" – g5*
>
> *"Desenvolvedor produtivo é aquele que consegue entregar num tempo bom, com uma qualidade boa, e pensando no conceito de*

> *economicidade. Não só [economicidade] de prazo, mas de custo mesmo" – g4*

Por fim, **sem necessidade de retrabalho** é um aspecto importante para a definição de produtividade dos desenvolvedores de acordo com os gerentes de *software*. Um desenvolvedor produtivo é aquele que realiza suas tarefas sem que elas precisem ser retrabalhadas mais tarde, por isso todo retrabalho é um sinal de falta de produtividade do desenvolvedor. Às vezes, esse mesmo sinal também é reconhecido durante um *code review*, atividade do desenvolvimento onde um outro desenvolvedor revisa o códigos desenvolvidos em busca de falhas ou de oportunidades de melhoria.

> *"É uma demanda que está dentro daqueles padrões de não ter tanto problema, [de] não ter tanto retrabalho dentro dessa tarefa" – g3*

> *"A minha percepção de produtividade é basicamente se o tempo que ele estimou para fazer alguma coisa foi feito e não teve muito review no código" – g10*

> *"Se aquilo que é entregue sempre precisa voltar para um redesenvolvimento, isso pesa negativamente na produtividade daquele desenvolvedor" – g2*

5.3.2 SQ2. Como é identificada a produtividade do desenvolvedor?

> *"Na maioria das vezes, a gente percebe isso mais claramente é pelas atitudes [do desenvolvedor]" – g5*

> *"A gente percebe pelas demandas que são entregues" – g2*

> *"Basicamente em cima do feedback dos times" – g10*

De acordo com os gerentes de *software* entrevistados, a produtividade do desenvolvedores da organização podem ser identificadas pelo seu **comportamento e atitudes**, pela **características das entregas** de suas tarefas e também pelo **feedback** dado pelas equipes de desenvolvimento.

O **comportamento e atitudes do desenvolvedor** ajuda aos gerentes de *software* perceberem os desenvolvedores produtivos da organização. Comportamentos como a proatividade e o comprometimento com os projetos da

organização são indícios de desenvolvedores produtivos. Para alguns gerentes, a postura e as atitudes do desenvolvedor são mais importantes que a sua produção, possivelmente por saberem que tem em sua organização pessoas com que possam contar para atividades extras em benefício da própria organização.

> *"São os fatores mais comportamentais e técnicos mesmo que nos levam a identificar se uma pessoa é produtiva ou não" – g11*
>
> *"A percepção de produtividade [do desenvolvedor] está muito voltada por ser percepções de questões não técnicas, mas comportamentais" – g4*
>
> *"Ele diz: [eu] acabei e estou pronto para fazer outra coisa, então a percepção que se tem é de que as pessoas que são produtivas são aquelas que tem esse tipo de postura" – g4*
>
> *"Aquela pessoa é produtiva porque ela mostra trabalho, mostra que está sempre em busca de coisa nova, inovação" – g11*

Além do comportamento, as características das **entregas de tarefas** dos desenvolvedores conseguem fornecer indícios da sua produtividade. As entregas frequentes dentro ou antes do prazo, com qualidade, sem a necessidade de retrabalho indicam o trabalho de um desenvolvedor produtivo. Ademais, a recorrência de entregas com essas características criam nos gerentes de *software* da organização uma confiança em relação ao trabalho do desenvolvedor. Dessa forma, quanto melhor forem executadas as tarefas pelo desenvolvedor, mais produtivo o desenvolvedor será para o gerente de *software*.

> *"A gente identifica e percebe produtividade com base em demandas atendidas, é basicamente o conceito" – g2*
>
> *"São os desenvolvedores que entregam mais, com mais frequência. Eles conseguem entregar e o trabalho deles não fica tendo retrabalho. Esses são os mais produtivos" – g9*
>
> *"É a questão de entregar a tarefa dentro do prazo, de entregar em menos tempo do que o previsto, é de garantir que aquele código está com qualidade" – g11*

> *"Pode passar tarefa para ele, que ele vai entregar no prazo que foi estipulado. Eu cruzo os braços e lavo as mãos e ele entrega" – g12*

Por fim, os gerentes de *software* têm também como fonte de informações o ***feedback* da equipe de desenvolvimento** do desenvolvedor. Os gerentes adquirem informação sobre a produtividade do desenvolvedor através de reuniões com os gerentes intermediários e líderes de projeto. Os líderes de projeto obtêm essas percepções através da sua própria percepção, de acordo com o seu comportamento e suas entregas e também pelo *feedback* dos outros desenvolvedores.

> *"O líder da equipe que ao entregar as demandas para os programadores e ao receber de volta a produção deles acaba construindo na sua cabeça quem ele enxerga como mais produtivo ou menos produtivo" – g7*

> *"O desenvolvedor está dentro de uma equipe e a própria equipe elogia o cara: não esse cara aqui é bom" – g12*

> *"Nas reuniões de final de ciclo a gente consegue identificar qual o desenvolvedor que foi mais produtivo ou não" – g9*

5.3.3 SQ3. Qual a utilidade das percepções sobre a produtividade do desenvolvedor para a organização de *software*?

> *"A gente usa isso para tentar qualificar ou equilibrar as equipes [de desenvolvimento] dentro dos projetos" – g3*

> *"Se a gente tivesse uma informação dessa, por exemplo, conseguiria montar um plano de treinamento" – g4*

> *"É uma medida importante nas questões das promoções" – g2*

> *"Influencia quando eu tenho que desligar alguém, pois não vou desligar alguém que é altamente produtivo" – g9*

As informação sobre a produtividade dos desenvolvedores das organização de *software* são úteis aos gerentes para apoiar os líderes na **alocação dos desenvolvedores** entre os projetos da organização, para o **aprimoramento dos desenvolvedores**, através de planos de treinamento, e também para **apoiar decisões gerenciais**, como promoções e desligamentos.

O emprego mais citado das informações sobre a produtividade dos desenvolvedores pelos gerentes de *software* foi na **alocação dos desenvolvedores para tarefas ou projetos**. Os desenvolvedores mais produtivos são alocados para tarefas críticas e projetos estratégicos a fim de assegurar que sejam executados no prazo acordado e com a qualidade esperada pelos clientes. Essa alocação também pode ter a finalidade de criar equipes de desenvolvimento equilibradas entre os projetos das organização. A organização sem esse equilíbrio entre suas equipes podem criar riscos para certos projetos da organização e sobrecarregar desenvolvedores produtivos, que mais tarde podem por desmotivação se desligar da organização.

> *"É alocar os desenvolvedores para aquelas atividades que a gente considera fundamentais e importantes para organização" – g2*
>
> *"A gente usa isso para tocar aqueles projetos estratégicos, que são críticos, que a gente não pode errar e que a gente tem que apresentar um bom produto" – g1*
>
> *"Muitas vezes um projeto é muito crítico, ele tem uma importância estratégica para a organização que é alto, então a gente acaba direcionando essas pessoas mais produtivas para projetos mais críticos com maior prioridade" – g6*
>
> *"Na hora de montar um time, você conseguir balancear quem é mais produtivo e garantir que vai conseguir fazer um time que vai atender a necessidade do cliente" – g9*
>
> *"Fazer com que a gente tenha uma equipe mais nivelada, porque se a gente não tiver isso, eu vou ficar sobrecarregando dois ou três e os outros nem tanto" – g5*

Outra utilidade mencionada é o **aprimoramento dos desenvolvedores da organização**. Um plano de treinamento mais adequado pode ser construído a fim de resolver as dificuldades encontradas pelos desenvolvedores da organização. Outra forma mencionada de aprimorar um desenvolvedor é fornecendo um *feedback* para eles mesmos sobre a sua produtividade. A informação da produtividade de certos desenvolvedores produtivos pode também ser divulgada dentro da

organização como forma de exemplo, para servir de inspiração e referência para os outros desenvolvedores da organização.

> *"Fazer todo o nosso [plano de] treinamento baseado nas dificuldades que a gente tem no dia-a-dia, [a partir] da falta de produtividade de alguns colaboradores" – g6*

> *"Para passar um feedback para o colaborador, para ele mesmo começar a melhorar essa questão de produtividade dele" – g11*

> *"E para os outros que não foram 'tão produtivos', enxergarem que tem um norte, tem uma pessoa de referência para seguir" – g12*

Finalmente, alguns gerentes de *software* mencionaram que as informações sobre a produtividade desenvolvedores são úteis para **apoiar decisões gerenciais sobre os desenvolvedores**. As decisões sobre promoções e desligamentos de desenvolvedores são apoiadas pela percepção da produtividade na organização.

> *"É uma medida importante nas questões das promoções" – g2*

> *"Em nível institucional, por exemplo, se acontece da gente precisar demitir colaboradores, essa é uma métrica que eu vejo que poderia ser utilizada para embasar essas decisões" – g11*

5.4 DISCUSSÃO DOS RESULTADOS

5.4.1 QP. Como os gerentes de *software* conceituam, percebem e utilizam a produtividade dos desenvolvedores de *software* da organização?

Este estudo teve como objetivo investigar como a organização entende, identifica e utiliza as percepções sobre a produtividade dos seus desenvolvedores. Os gerentes de *software* foram escolhidos para representar o ponto de vista organizacional, uma vez que são os responsáveis por realizar a gestão dos projetos e desenvolvedores dentro da organização.

Os gerentes de *software* entendem que a produtividade do desenvolvedor é possuir qualificação técnica e comportamental, entregando as tarefas no prazo, atendendo as expectativas das partes interessadas e sem que haja necessidade de retrabalho. Por isso, os gerentes identificam a produtividade de seus desenvolvedores a partir de seus comportamentos e atitudes, pela características das

entregas de suas tarefas, além do *feedback* dado pelas equipes de desenvolvimento. Eles utilizam as informação da produtividade obtidas de seus desenvolvedores para apoiar seus líderes na alocação dos desenvolvedores entre os projetos da organização, para apoiar o aprimoramento dos desenvolvedores através de planos de treinamento e também para apoiar decisões gerenciais sobre os desenvolvedores, como promoções e desligamentos.

Esses resultados encontrados guardam uma similaridade com o conceito básico de produtividade utilizado na Engenharia de Software, onde a produtividade é a relação entre a quantidade produzida pelo esforço despendido [Mockus 2009]. Nesse caso, a quantidade produzida são as tarefas entregues do desenvolvedor e o esforço despendido é o tempo utilizado para realizá-las. Porém, possuem uma divergência na sua interpretação, pois enquanto a produtividade baseada no sentido econômico é mais orientada à ideia de eficiência, os gerentes de *software* consideram a produtividade mais orientada à ideia de eficácia. Por essa razão, para os gerentes, é necessário que as tarefas dos desenvolvedores atendam no prazo as expectativas de todas partes interessadas no projeto (*stakeholders*). Em consequência, se alguma tarefa do desenvolvedor tiver alguma necessidade de retrabalho, esse desenvolvedor não foi eficaz, logo, não foi produtivo.

Além disso, a qualificação técnica e comportamental dos desenvolvedores também é considerada pelos gerentes de *software*. Quanto maior for a qualificação técnica do desenvolvedor, mais habilitado ele estará para realizar as mais diversas tarefas técnicas e, por esse motivo, a percepção de ser mais eficaz para a organização. Ademais, o comportamento e as atitudes do desenvolvedor moderam essa definição de produtividade, pois se os gerentes perceberem a ausência de, por exemplo, foco ou comprometimento por parte dos desenvolvedores, é sinal de que ele ser mais eficiente, mais produtivo.

Essas percepções do comportamento e atitudes do desenvolvedor não fazem parte do conceito básico de produtividade baseado no sentido econômico, mas são devidamente consideradas em estudos mais recentes sobre a importância dos fatores humanos dentro do processo de desenvolvimento de *software* [Amrit *et al.* 2014]. Essas percepções de comportamento e atitudes também são complementadas pelo *feedback* de seus líderes de projeto equipes de desenvolvimento, que informam, por exemplo, quando um desenvolvedor serve de referência para o restante da equipe

de desenvolvimento, influenciando positivamente a percepção de produtividade dos gerentes de *software*.

Esses resultados também apresentam as aplicações práticas das percepções sobre a produtividade dos gerentes das organizações de *software*. Utilizando o histórico dessas percepções de produtividade dos desenvolvedores, os gerentes constroem um *ranking* de produtividade de seus desenvolvedores. Esse *ranking* é útil para algumas de suas ações gerencias. Entre essas ações gerenciais estão a alocação de equipes de desenvolvimento, aprimoramento dos desenvolvedores e decisões sobre promoções e demissões dentro da organização.

5.4.2 Relacionamento com as evidências existentes

Trabalhos recentes da literatura também focaram na produtividade do desenvolvedor de *software*. Os trabalhos mais relacionados com esse estudo são o de Melo *et al.* (2011), Hernández-López *et al.* (2012) e Meyer *et al.* (2014). Melo *et al.* (2011) investigaram as percepções das equipes ágeis sobre os fatores que impactam a produtividade. Hernández-López *et al.* (2012) investigaram o significado de produtividade da perspectiva dos desenvolvedores. Por fim, Meyer *et al.* (2014) investigaram as percepções do desenvolvedor de software sobre a sua própria produtividade dentro do desenvolvimento de software.

Melo *et al.* (2011) também utilizaram um abordagem qualitativa em seu trabalho. O método principal de coleta de dados foi a entrevista semiestruturada, analisando os dados coletados através de uma análise temática. Foram entrevistados um total de 13 participantes, dos quais dois eram líderes de projeto, dois *product owners*, oito desenvolvedores e um *scrum master*. Os autores não conseguiram uma definição clara e objetiva da produtividade do desenvolvedor em equipes ágeis, mas concluíram que possui alguns termos como a pontualidade, a quantidade, a qualidade e a satisfação do cliente. Esses termos também foram encontrados nesta pesquisa, apesar de não considerar a qualificação técnica e comportamental dos desenvolvedores. O trabalho de Melo *et al.* (2011) difere desta pesquisa principalmente com relação ao contexto, no qual focaram em equipes utilizando a metodologia ágil, e aos participantes, incluindo líderes de projeto, *product owners*, desenvolvedores e *scrum master*.

Hernández-López *et al.* (2012) também realizaram uma pesquisa qualitativa em sua investigação. Os autores entrevistaram um total de 15 participantes, dos quais quatro líderes de projeto, nove engenheiros sênior e dois engenheiros júnior. Os participantes tinham uma média de 2,5 anos na organização em que trabalhavam. Nesse trabalho, os autores investigaram as definições de produtividade para os engenheiros de *software*, líderes de projeto e para a organização. Considerando somente a definição de produtividade do engenheiro de *software*, o resultado obtido pelos autores define a produtividade com tarefas completadas com a qualidade requerida, resolvendo problemas e defeitos dentro do prazo especificado. A definição encontrada por Hernández-López *et al.* (2012) é consistente com o resultado obtido nesta pesquisa, apesar de também não considerar a qualificação técnica e comportamental dos desenvolvedores. O trabalho de Hernández-López *et al.* (2012) difere desta pesquisa principalmente com relação aos participantes entrevistados. Hernández-López *et al.* (2012) utilizaram diversos tipos de participante, enquanto este estudo focou somente no ponto de vista dos gerentes de *software*.

Meyer *et al.* (2014) realizaram um *survey* e um estudo observacional sobre a produtividade do desenvolvedor. O *survey* foi realizado com 379 desenvolvedores, enquanto o estudo observacional foi realizado somente com 11 desenvolvedores. Os resultados obtidos com o *survey* indicaram que os desenvolvedores avaliam sua produtividade através das tarefas que eles completam e dos artefatos de *software* que entregam. Os resultados do estudo observacional identificaram que o desenvolvedor gosta de organizar seu trabalho de forma a entrar no que os autores chamara de “o fluxo” (*the flow*). Esse fluxo acontece quando o desenvolvedor tem menos interrupções naquilo que ele está fazendo e poucas mudanças de atividade. Esses resultados também estão de acordo com os resultados desta pesquisa, pois o foco também foi citado neste estudo pelos gerentes de *software* como uma característica do desenvolvedor produtivo. Esse trabalho difere desta pesquisa pois os participantes desse estudo foram somente desenvolvedores, enquanto os desta pesquisa foram somente gerentes de *software*.

Todos esses estudos relacionados descritos focaram na produtividade do desenvolvedor de *software*, investigando principalmente o conceito de produtividade do desenvolvedor. Ao contrário desses estudos, nesta pesquisa o conceito de

produtividade do desenvolvedor foi investigada apenas pela perspectiva dos gerentes de *software*, representante das gestão de projetos das organizações de *software*. Os componentes do conceito encontrados por esses estudos estavam também presente nos resultados desta pesquisa, porém além desses componentes, os gerentes de *software* também consideraram outros componentes, como a qualificação técnica e comportamental do desenvolvedor. Além disso, essa pesquisa também investigou como os gerenciadores de *software* identificam e utilizam a percepção da produtividade de seus desenvolvedores.

5.5 AMEAÇAS À VALIDADE

Diversas ameaças afetam a validade dos resultados de todo estudo científico [Wohlin *et al.* 2012] e, por isso, devem ser relatadas. Nesta seção são apresentadas as principais as ameaças à validade deste estudo.

A primeira ameaça à validade deste estudo é quanto a generalização dos resultados aqui obtidos para todas as outras organizações de *software*. Estudos qualitativos não podem usar nada semelhante a um argumento estatístico para reivindicar a generalização de seus resultados. Independente disso, para mitigar essa ameaça, foram entrevistados 12 gerentes de *software* de três diferentes organizações representativas de desenvolvimento de *software*.

Outra ameaça é que as percepções dos entrevistados podem ser tendenciosas em relação a suas próprias crenças. Essas crenças podem causar algumas distorções na interpretação da realidade, causando distorções nos resultados obtidos. Para reduzir essa ameaça, os gerentes de *software* escolhidos para as entrevistas foram aqueles que tinham mais experiência dentro de suas organizações, ou seja, gerentes que tenham vivenciado uma maior variedade de situações envolvendo o desenvolvimento de projetos de *software* dentro de suas organizações.

Ainda outra ameaça à validade dos resultados é a possibilidade do autor do estudo ter introduzido seu viés no processo de análise de dados. A este respeito, o processo de análise dos dados coletados foi realizado junto com outros pesquisadores mais experientes. Esses pesquisadores revisaram e analisaram todos os resultados intermediários, tais como os códigos escolhidos. Este processo iterativo foi repetido até o final da coleta e análise de dados. Também foram realizadas reuniões com os gerentes de projetos de *software* para validar os resultados obtidos.

5.6 CONSIDERAÇÕES FINAIS

Este capítulo descreveu um estudo utilizando uma investigação qualitativa sobre a produtividade do desenvolvedor a partir da perspectiva dos gerentes das organizações de *software*. O estudo foi conduzido nas três organizações de *software* selecionadas. Um total de 12 gerentes de *software* foram entrevistados incluindo diretores de departamento, gerentes de qualidade de *software*, gerentes de desenvolvimento de *software*. Os resultados obtidos a partir da percepção dos gerentes de *software* indicam um conceito de produtividade dos desenvolvedores baseado na entrega das tarefas dentro do prazo, atendendo as expectativas do cliente, sem a necessidade de retrabalho e possuindo qualificação técnica e comportamental. É a partir dos componentes desse conceito que os gerentes identificam a produtividade de seus desenvolvedores e utilizam essa informação para alocar e aprimorar seus desenvolvedores, além de apoiar decisões gerenciais da organização.

Capítulo 6 – PERCEPÇÃO DA PRODUTIVIDADE DO DESENVOLVEDOR PELOS LÍDERES DE PROJETO DE *SOFTWARE*

Este capítulo apresenta um estudo qualitativo para investigar a percepção dos líderes de projetos de software quanto ao conceito de produtividade. Além disso também são investigados, segundo a percepção dos líderes de projeto, os fatores de influência na produtividade dos desenvolvedores.

6.1 INTRODUÇÃO

Compreender os fatores que influenciam a produtividade dos desenvolvedores de *software* é um foco importante a ser pesquisado dentro da Engenharia de *Software*, pois que o sucesso dos projetos de desenvolvimento de *software* ainda depende das pessoas [Hernández-López *et al.* 2013],. Apesar dessa importância, a maioria dos trabalhos sobre produtividade de *software* foca mais na organização de *software* e nos projetos de *software*, do que no desenvolvedor de *software* [Meyer *et al.* 2014].

Este capítulo apresenta um estudo para investigar como os líderes de projeto das organizações de *software* conceituam e identificam a produtividade, além de esclarecer quais são os fatores de influência sobre a produtividade de seus desenvolvedores. Os resultados do conceito e identificação da produtividade são comparados com os resultados do estudo anterior e a investigação sobre os fatores de influência gerou uma teoria fundamentada preliminar.

6.2 PLANO DE ESTUDO

O objetivo desse estudo foi investigar como os líderes de projetos de *software* das organizações selecionadas entendem e identificam a produtividade dos seus desenvolvedores. Além disso, outro objetivo é investigar, sobre o ponto de vista dos líderes de projeto, quais são os fatores relevantes de influência sobre a produtividade dos desenvolvedores de *software* das organizações. Os líderes de projeto de *software* foram escolhidos para essa investigação, pois são os responsáveis diretos pelos desenvolvedores em um projeto de *software*, desde a montagem da

equipe de desenvolvedores e seus papéis, passando pelo acompanhamento das atividades e avaliando os resultados parciais e final do projeto de *software*.

A duas questões principais (*QP1* e *QP2*) dessa investigação e as subquestões derivadas estão apresentadas na Tabela 36. A primeira questão principal foi derivada em duas subquestões: a subquestão *SQ1*, que visou identificar o conceito de produtividade utilizado pelos líderes de projeto da organização e a subquestão *SQ2*, que focou em como os líderes de projeto identificam a produtividade dos seus desenvolvedores. A segunda questão principal (*QP2)* também foi derivada em duas subquestões: a subquestão *SQ3*, que procurou identificar os fatores de influência sobre a produtividade dos desenvolvedores e a subquestão *SQ4*, que visou identificar se dentre os fatores citados pelos líderes de projeto existem alguns com maior relevância em relação aos outros.

Tabela 36. Questões de pesquisa para os líderes de projeto.

QP1	Como os líderes de projeto conceituam, identificam a produtividade dos desenvolvedores da organização?
SQ1	O que é produtividade do desenvolvedor?
SQ2	Como é identificada a produtividade do desenvolvedor?
QP2	Quais são os fatores relevantes de influência sobre a produtividade dos desenvolvedores de *software* da organização?
SQ3	Quais são os fatores de influência sobre a produtividade dos desenvolvedores da organização?
SQ4	Quais dos fatores de influência sobre a produtividade dos desenvolvedores citados são consideradas mais relevantes?

6.2.1 Procedimentos de coleta de dados

Da mesma forma que no estudo anterior, a coleta de dados foi realizada em cada uma das organização selecionadas utilizando entrevistas semiestruturadas com seus líderes de projeto de *software*. Todos os líderes de projeto entrevistados também assinaram um termo de consentimento livre e esclarecido (TCLE), assim como no estudo anterior nessas organizações.

As organizações disponibilizaram um total de doze líderes de projeto de *software* para este estudo, sendo quatro na Organização 1 (Org. 1), três na Organização 2 (Org. 2), e cinco na Organização 3 (Org. 3). A Tabela 37 apresenta

os líderes de projeto de *software* participantes deste estudo, onde cada participante é relacionado com sua organização, nível de educação, a quantidade de anos como líder de projeto e a quantidade de anos trabalhados na organização.

Tabela 37. Líderes de projeto de *software* entrevistados.

Id.	Org.	Nível de educação	Anos como Líder	Anos na Org.
l1	1	Especialização em Gestão Empresarial	9	10
l2	1	Especialização em Desenvolvimento de SW	3	6
l3	1	Especialização em Banco de Dados	7	8
l4	1	Especialização em Desenvolvimento de SW	21	26
l5	2	Especialização em Gestão de Projetos	5	26
l6	2	Especialização em Engenharia de SW	1	6
l7	2	Especialização em Engenharia de SW	5	10
l8	3	Bacharel em Ciência da Computação	4	7
L9	3	Especialização em Gestão de Projetos	8	8
l10	3	Bacharel em Análise de Sistemas	4	8
l11	3	Especialização em Engenharia de SW	2	3
l12	3	Bacharel em Análise de Sistemas	3	6

Na Tabela 37, observa-se que a maioria dos líderes de projeto tem especialização dentro da área de informática. É possível também observar que a maioria dos líderes tem pelo menos cinco anos trabalhando nas organizações, exceção ao líder *l11*, com mais da metade do tempo trabalhando como líder de projeto de *software*. Outra observação a se fazer quanto ao tempo de trabalho é que os líderes *l5* e *l6*, ambos da Org. 2, possuem pouco tempo atuando como líderes, comparado ao tempo dentro da organização. Isso sugere que a maioria dos líderes de projeto da organização possui conhecimentos suficientes sobre projetos de desenvolvimento de *software* e dos processos organizacionais para atuar como líder de projeto.

Assim como no estudo anterior, todas as entrevistas também utilizaram um roteiro semiestruturado. O roteiro também seguiu as diretrizes propostas por Runeson *et al.* (2012), realizando uma primeira parte com questões introdutórias sobre os antecedentes dos participantes, permitindo a ambientação com a entrevista. A segunda parte compreendeu as principais questões da entrevista. Diferente do

estudo anterior, e de acordo com a experiência do estudo anterior e das sugestões colhidas com outros pesquisadores, o roteiro de entrevista foi simplificado (Tabela 38) para ficar somente com perguntas chaves. Desta forma, os entrevistados puderam falar mais à vontade, sem delimitar por demais as perguntas. As questões do roteiro foram praticamente as mesmas das questões de pesquisa para este estudo, porém com o cuidado de que cada questão contivesse questões secundárias para o caso do entrevistado não falar sobre o assunto investigado.

Tabela 38. Roteiro semiestruturado utilizado nas entrevistas com os líderes de projeto.

Subquestão	Descrição
SQ1	• O que é a produtividade do desenvolvedor de *software* para a organização? • Os desenvolvedores têm o mesmo nível produtividade?
SQ2	• Como é identificada a produtividade do desenvolvedor de *software* na organização? • Como é percebido que um desenvolvedor é produtivo? • Como é comparada a produtividade entre desenvolvedores de uma mesma equipe?
SQ3	• Quais são os fatores de influência que impactam na produtividade do desenvolvedor de *software*?
SQ4	• Dos fatores de influência citados, existem alguns que são mais relevantes?

Assim como no procedimento do estudo anterior, as entrevistas foram realizadas simultaneamente com o processo de análise, realizando ciclos compostos de entrevistas, transcrições e análise.

6.2.2 Procedimentos de análise de dados

As transcrições das entrevistas realizadas neste estudo também foram importadas e analisadas com o software Atlas.ti[16]. A análise qualitativa realizada também utilizou procedimentos da Teoria Fundamentada (*Grounded Theory – GT*) [Glaser e Strauss 1968]. Foram utilizados todos os procedimentos de GT, incluindo a codificação aberta, a codificação axial e a codificação seletiva. Para as duas primeiras subquestões (*SQ1* e *SQ2*) não foi obtida uma categoria central emersa naturalmente dos dados. Porém, para a subquestão *SQ3* dos fatores de influência,

[16] http://atlasti.com

uma categoria central emergiu a partir dos dados. Essa categoria central aponta para uma teoria fundamentada sobre os fatores de influência da produtividade dos desenvolvedores de organização de *software*, de acordo com a percepção dos líderes de projeto. Além disso, ainda para a subquestão *SQ3*, a partir das últimas entrevistas codificadas e analisadas, a identificação de novos códigos reduziu-se a zero, sugerindo que uma possível saturação teórica fora atingida [Strauss e Corbin 2008]. Porém, como será visto no Capítulo 1Capítulo 8, novas informações foram extraídas, a partir de novas entrevistas, evoluindo a teoria fundamentada preliminar aqui identificada.

6.3 RESULTADOS OBTIDOS

6.3.1 SQ1. O que é produtividade do desenvolvedor?

> *"A produtividade seria o desenvolvedor entregar o trabalho que foi especificado no prazo, com qualidade" – l5*
>
> *"Se ele fez um trabalho mal feito e vai precisar refazer depois, ele não está sendo produtivo" – l3*
>
> *"Quanto mais foco você tem em uma atividade, mais produtivo você é" – l8*

Os líderes de projeto consideram que a produtividade de um desenvolvedor é **entregar suas tarefas no prazo**, com qualidade, ou seja, **sem a necessidade de retrabalho** dessas atividades. Além disso, o **comportamento e as atitudes** do desenvolvedor também são consideradas, avaliando o foco e a proatividade dos desenvolvedores durante o trabalho.

Entregar a tarefa no prazo foi um dos componentes da produtividade do desenvolvedor mais citado pelos líderes de projeto das organizações selecionadas. Os líderes de projetos de desenvolvimento de *software* têm um plano a cumprir com a sua equipe e, por isso, veem nos desenvolvedores produtivos aqueles que entregam as demandas a eles confiadas dentro do prazo estimado. Quando o desenvolvedor consegue entregar antes do prazo estipulado, o líder de projeto vê nessa atitude um forte indício de produtividade; enquanto a entrega com atraso traz a impressão oposta, a falta de produtividade. Além disso, os líderes também consideram a quantidade e a complexidade das tarefas realizadas.

> *"Ele é produtivo se ele entregar as coisas no prazo" – l2*
>
> *"É um trabalho bem feito dentro do prazo estipulado, a produtividade dele" – l3*
>
> *"Eu vejo que a produtividade do desenvolvedor seria: é se ele conseguiu realizar aquela atividade dentro do que foi planejado. Se ele conseguir em menos tempo do que foi [estimado]: ele foi produtivo; se ele demorou muito mais: não foi tão produtivo assim" – l9*
>
> *"A gente vê a quantidade de atividades que fechou, não só a quantidade, mas também a complexidade das atividades que esse desenvolvedor está fechando" – l12*

Entregar com qualidade, sem necessidade de retrabalho foi o outro componente da produtividade mais citado pelos líderes de projeto. O retrabalho foi um conceito que ficou muito relacionado com a qualidade, praticamente um sinônimo, conforme o entendimento dos líderes de projeto. Isso porque a identificação de retrabalho significa atraso no cronograma e, por conseguinte, pode significar atraso na entrega para o cliente. Práticas de qualidade são realizadas em algumas organizações, utilizando uma equipe dedicada ao teste do *software*, realizando *code review*, porém é com a identificação da necessidade do retrabalho que a produtividade do desenvolvedor é negativamente identificada. Para o líder *l10*, o desenvolvedor pode até entregar um pouco além do planejado, sendo sem retrabalho será ainda considerado produtivo.

> *"A produtividade está mais relacionada com a qualidade" – l11*
>
> *"Eu não posso fazer uma coisa rápida e entregar com má qualidade, então teríamos que fazer uma ponderação de todos esses fatores" – l5*
>
> *"Produtividade é trazer uma funcionalidade sem defeitos, sem precisar fazer retrabalho, sem precisar corrigir" – l6*
>
> *"Uma vez que eu entrego, por mais que eu tenha levado um pouco mais de tempo, mas eu não tenho um "tempo perdido futuramente" para ter que refazer, com excesso de bugs" – l10*

> *"A produtividade de um desenvolvedor é fazer um código, que além de ser rápido na entrega, tenha o menor retrabalho possível, ou seja, eu entrego no prazo e o código não me dê retrabalho depois, para mim isso é produtividade" – l7*

Por fim, o **comportamento e as atitudes do desenvolvedor** também foram considerados como um componente da produtividade. Os desenvolvedores não são máquinas e possuem, por isso, produtividade inerentemente diferentes. Assim sendo, o comportamento é avaliado pelos líderes como uma forma de identificar se o desenvolvedor está em seu nível máximo de produtividade. Para isso, os líderes de projeto avaliam se o desenvolvedor está focado em seu trabalho e/ou sendo proativo, especialmente quando utiliza de seu tempo na implementação de automatizações que tragam melhorias para a produtividade da equipe de desenvolvimento.

> *"Se ele conseguir ali se manter focado, a gente consegue chegar mais próximo do que foi planejado, ou até fazer mais rápido" – l8*

> *"Ele pode ser produtivo também criando meios que ajudem os outros a produzir mais, por exemplo: automatizando uma coisa que está, de repente, bloqueando a equipe e fazendo com que o processo ande mais devagar" – l12*

> *"Acredito que deva ser considerado um cara produtivo o fato dele estar trazendo melhorias que colaborem com o desenvolvimento do time" – l9*

6.3.2 SQ2. Como é identificada a produtividade do desenvolvedor?

> *"Eu observo muito o foco do desenvolvedor na atividade" – l8*

> *"Eu percebo que o desenvolvedor tem uma boa produção é quando ele te dá o feedback" – l7*

> *"Como é identificada a produtividade do desenvolvedor de software na organização? – Isso daí é nas entregas" – l3*

> *"A gente vai pelo histórico daquele desenvolvedor" – l7*

Os líderes de projeto identificam a produtividade de um desenvolvedor pelo seu **comportamento e atitudes**, pelas **características da entrega de suas tarefas**,

pelo **feedback** dado por ele e por membros de sua equipe, além do *histórico* de seu trabalho dentro da organização.

Os líderes acompanham o **comportamento e atitudes** do desenvolvedor para ter um conceito de sua produtividade. Isto porque nenhum desenvolvedor tem um mesmo nível de produtividade igual a outro desenvolvedor. Uma tarefa que foi difícil para um desenvolvedor, pode ser fácil para outro e o que definirá se o desenvolvedor foi produtivo ou não está justamente no seu comportamento e atitude frente as tarefas que lhe são designadas. O desenvolvedor que foca no seu trabalho e que tem atitudes proativas é considerado como produtivo pelos líderes de projeto.

> *"E no dia a dia também, você está ali do lado, você está vendo o que o desenvolvedor está fazendo" – l3*
>
> *"Do que eu vi até agora, normalmente, a pessoa que é mais produtiva é aquela que é mais focada também" – l12*
>
> *"Eu ando muito nas equipes para saber se os desenvolvedores não estão enrolando" – l2*

Outro momento onde o líder identifica a produtividade do desenvolvedor é na entrega de tarefas, ao avaliar as **características das tarefas realizadas**. Uma das tarefas do líder de projeto é demandar tarefas aos desenvolvedores, estimando o tempo necessário de execução de acordo com a capacidade de cada desenvolvedor. Ao entregar suas tarefas, o desenvolvedor é avaliado pelo líder em relação ao tempo gasto. Se entregou no prazo, conforme planejado, ele é considerado produtivo pelo líder. Esse acompanhamento pode ocorrer em reuniões ou até mesmo pelo uso de ferramentas de acompanhamento como o cronograma. A qualidade é um ponto importante também para o líder, descrita no próximo item.

> *"A gente avalia a pessoa por esse aspecto: a gente passou uma missão, uma atividade e ela correspondeu com resultados" – l9*
>
> *"Pela quantidade de atividades que são passadas, demandas de requisitos e o tempo que ele leva para retornar isso e também a qualidade, não é só o tempo" – l1*
>
> *"O líder também fica monitorando o time para saber se uma pessoa está ou não entregando conforme o planejado, na qualidade*

> *certa. Tem as reuniões diárias, [em] que a gente consegue monitorar isso de uma forma mais fácil" – l8*

O ***feedback*** dado pelo próprio desenvolvedor ou pelos membros de sua equipe fazem com que o líder de projeto consiga identificar a produtividade deles. Quando o desenvolvedor atua junto ao líder para atualizá-lo de suas tarefas, por exemplo, informando metas atingidas e/ou desafios conquistados, faz com que o seu líder possa acompanhar a sua produtividade. Além disso, o *feedback* passado pelos outros membros da equipe, como companheiros de *pair programming* ou os desenvolvedores que realizaram o *code review*, podem informar ao líder características do seu trabalho, especialmente com relação à qualidade.

> *"Ou passando feedbacks que nos permitiram concluir que esse cara foi realmente produtivo" – l9*

> *"Eu tenho que esperar o retorno do trabalho dele, até mesmo porque, às vezes, outros desenvolvedores podem comentar sobre a qualidade do código dele" – l10*

Por fim, o líder através das percepções anteriores vai montando ao longo do tempo um **histórico** do desenvolvedor. Desenvolvedores antigos, cujo o líder já possui um histórico de seu trabalho, facilitam o planejamento das tarefas, pois a expectativa do líder já está com informações do histórico de desempenhos do desenvolvedor. Para novos desenvolvedores, o líder vai avaliando o resultado das tarefas demandadas, criando, dessa forma, a ideia da produtividade do desenvolvedor dentro dos trabalhos da organização.

> *"À medida que um desenvolvedor vai desempenhado atividades com relativa complexidade é que se tem uma percepção individual da produtividade dele, [...] um histórico de bons desempenhos" – l9*

> *"Quando a gente não o conhece, a gente começa a trabalhar e tem que esperar um pouco esse retorno do trabalho dele. Não analisa simplesmente o primeiro ponto que ele me entregou em um dia ou em meio dia" – l10*

> *"Geralmente, os desenvolvedores mais experientes, a gente diz realmente que aquele cara vai entregar naquele prazo. Quando tem um desenvolvedor novo, aí a gente tem que jogar isso para ele,*

> *seja pela funcionalidade que ele consegue terminar, seja pela expertise dele, então ele consegue entregar em X dias. A partir disso daí, a gente consegue tirar as nossas métricas" – l7*

6.3.3 SQ3. Quais são os fatores de influência sobre a produtividade dos desenvolvedores da organização?

A resposta para essa subquestão de pesquisa resultou em mais de 20 fatores citados pelos líderes de projeto. Para uma melhor compreensão desse resultado (ver Tabela 39), eles foram inicialmente categorizados durante a codificação axial, e ainda agrupados em fatores **técnicos**, **humanos** e **organizacionais**, segundo a classificação de fatores de influência da produtividade do desenvolvedor proposta no Capítulo 2, Subseção 2.4.1. As subseções a seguir descrevem os fatores citados em maiores detalhes.

Tabela 39. Fatores de influência segundo os Líderes de Projeto.

Classificação	Categorias	Fatores citados
Fatores Técnicos	Conhecimento técnico	*Conhecimento na tecnologia* *Nível de escolaridade*
	Domínio e experiência	*Domínio do negócio* *Experiência na tecnologia* *Familiaridade com o projeto*
Fatores Humanos	Comportamento e atitudes	*Foco* *Comprometimento* *Motivação* *Proatividade*
	Relacionamento interpessoal	*Relacionamento com o líder* *Relacionamento com a equipe* *Relacionamento com a gerência* *Relacionamento com o cliente*
Fatores Organizacionais	Políticas Organizacionais	*Processo de desenvolvimento* *Definição dos requisitos* *Gestão do relacionamento com o Cliente*
	Gestão dos Desenvolvedores	*Falta de reconhecimento* *Tamanho da equipe de desenvolvimento* *Alocação de desenvolvedores nas equipes*
	Ambiente de trabalho	*Ambiente salubre* *Ambiente tranquilos* *Equipamentos adequados*

6.3.3.1 *Fatores Técnicos*

> *"Posso dizer que o que pode estar impactando seria o conhecimento técnico para o desenvolvedor exercer as suas atividades em um determinado projeto" – l9*
>
> *"Acho que [a experiência] na tecnologia faz muita diferença e no conhecimento do projeto em si" – l8*

Os fatores técnicos são aqueles que referem-se à competência individual adquirida pela educação especializada e à experiência na prática de um desenvolvedor. Esses fatores foram categorizados em **conhecimento técnico** e **domínio e experiência**. O conhecimento técnico diz respeito a sua formação básica especializada para estar apto ao desenvolvimento de *software*; enquanto o domínio e experiência dizem respeito ao domínio e a experiência adquiridas na aplicação prática dos conhecimentos técnicos adquiridos em projetos reais.

O **conhecimento técnico** de um desenvolvedor é um fator básico que habilita o desenvolvedor a contribuir com um projeto de desenvolvimento de *software* dentro de sua organização. No início de carreira, o conhecimento do desenvolvedor é basicamente formado pela sua formação acadêmica, proporcional ao seu nível de escolaridade, e pelos cursos técnicos realizados. O conhecimento do desenvolvedor o habilita ao desenvolvimento de software, mas sem a experiência e domínio das tecnologias, não são raras as vezes que ele tem que parar seu trabalho para adquirir os conhecimentos necessários a fim de completar sua tarefa. Por isso, os líderes de projeto consideram a formação e o conhecimento técnico como fatores de produtividade do desenvolvedor.

> *"O nível de escolaridade, o nível de formação do desenvolvedor também ajuda a incrementar. Então se ele é formado em uma escola melhor, em geral ele tem maior produtividade" – l1*
>
> *"No início de carreira o conhecimento técnico faz com que o técnico se torne muito mais produtivo" – l4*
>
> *"A produtividade [do desenvolvedor] depende de muitos fatores: o conhecimento técnico dele, lógico" – l5*

> *"Às vezes, é um cara bom, mas ele ainda não conhece tanto aquela tecnologia, mas a gente sabe que ele vai se virar. Aí, a gente tem que levar em consideração a curva de aprendizagem dele" – l8*

A medida que adquire maior **domínio e experiência** dos conhecimentos técnicos, o desenvolvedor torna-se mais produtivo na realização de suas tarefas. Os líderes de projeto citaram a experiência na tecnologia, obtida através do exercício prático repetitivo dos conhecimentos técnicos adquiridos; o conhecimento do negócio, entendido aqui como o domínio do negócio do projeto de *software* na qual o desenvolvedor participa; e a familiaridade do projeto, entendida aqui como o conhecimento de como o projeto está organizado, quais ferramentas são utilizadas, facilitando o trabalho em equipe do desenvolvedor.

> *"A experiência na tecnologia é um fator importante" – l7*

> *"Bem, tem a experiência técnica, tem o conhecimento do negócio, que ajuda muito quando ele conhece e é passado de forma clara para ele como funciona o negócio, para que ele codifique" – l5*

> *"Quem domina bastante tecnologia tende a produzir mais, a gerar mais código, gerar mais telas. Mas precisa de alguém com visão mais ampla de negócio, para que esses códigos saiam e que essas telas apareçam com a qualidade e atendendo as expectativas do cliente" – l4*

> *"Se coloca uma pessoa que ainda não está familiarizada com o contexto além da tecnologia, ela vai ter que se familiarizar [com o projeto]. E tudo isso influencia na produtividade" –l8*

6.3.3.2 *Fatores Humanos*

> *"Se o cara não quiser, ele pode ser o melhor técnico do mundo, mas se não for do perfil dele de querer ser produtivo, de querer entregar, de querer fazer, ele não vai fazer. Eu não levei em consideração o conhecimento técnico, justamente por isso" – l3*

> *"Tem que ter um bom relacionamento com as lideranças, tem que ter um bom relacionamento com os colegas de trabalho" – l4*

Conforme definido anteriormente, os fatores humanos referem-se às características individuais do desenvolvedor, que não são inerentes à sua função dentro da organização. De acordo com os fatores citados pelos líderes de projeto, os fatores humanos de influência na produtividade podem ainda ser agrupados em fatores relativos ao **comportamento e atitudes** individuais e fatores relativos ao **relacionamento interpessoal,** com a equipe, com as lideranças e com os clientes.

Segundo os líderes de projeto, a produtividade do desenvolvedor é influenciada pelo seu **comportamento e atitudes**. A motivação, o comprometimento, a proatividade e o foco foram exemplos de comportamentos e atitudes citadas. Um desenvolvedor pode até ter conhecimento técnico e experiência para desenvolver uma atividade, mas se ele não utilizar da sua vontade para trabalhar de forma produtiva, ele simplesmente será improdutivo. Essa conclusão, por parte de alguns líderes de projeto, coloca esse grupo de fatores com um dos principais fatores na produtividade dos desenvolvedores das organizações de *software*.

> *"A motivação é um dos principais [fatores]" – l7*
>
> *"O compromisso leva a produtividade" – l6*
>
> *"Influencia a postura dele de proatividade na produtividade. Por mais que eu não saiba fazer, mas que eu consiga me levantar logo e tirar dúvidas e estar questionando e procurando entender" – l10*
>
> *"Ele acaba aprendendo mais por não se dispersar. Ele aprende mais sobre a tecnologia, sobre o projeto, se foca na atividade para fazer. Ele usa esse foco para aumentar a própria produtividade. – l1*
>
> *"Eu tenho exemplo num projeto agora, um desenvolvedor domina muito a tecnologia desse projeto atual, só que ele constantemente perde o foco. Acaba que outro que domina menos está produzindo absurdamente mais do que ele, que deveria ser mais produtivo. – l12*

Além do comportamento e atitudes individuais, o **relacionamento interpessoal**, ou seja, relacionamento com a equipe, com o líder, com a gerência e até com o cliente pelo desenvolvedor é também considerado um fator de influência na sua produtividade. Alguns líderes consideram que o trabalho em equipe é um

fator essencial para a produtividade em um projeto e, por isso, um desenvolvedor que não se relaciona bem com os demais membros da equipe não está sendo produtivo, pois ou poderia estar ajudando outros ou sendo ajudado a resolver problemas de tarefas de desenvolvimento do *software*. O relacionamento com o líder também é importante, pois, de acordo com o tipo de relacionamento, o desenvolvedor pode se sentir motivado ou desmotivado. Além disso, o relacionamento com o cliente também pode afetar negativamente a produtividade, especialmente quando o desenvolvedor é pressionado em situações de contato com o cliente.

> *"Porque o nosso cliente, ele pressiona bastante" – l3*
>
> *"Eu procuro acompanhá-los em reuniões com o cliente para não deixar eles pegarem a 'porrada' de frente" – l2*
>
> *"Eu ligo muito a produtividade ao time. Além da qualidade, é um todo, é uma entrega. E acho que isso influencia bastante" – l10*
>
> *"Então volto a dizer, eu acho importantíssimo haver o relacionamento, as pessoas conversarem e dialogarem" – l4*
>
> *"Eu imagino que eu possa ser a pessoa mais esperta de uma sala, mas eu não sou mais esperto do que a sala inteira. Então, se a gente conseguir fazer essa ligação, a gente tende a ser mais esperto, mais produtivo" – l10*
>
> *"Acho que o relacionamento com o líder é um fator importante para estar motivando e impactando no desempenho do desenvolvedor" – l9*

6.3.3.3 *Fatores Organizacionais*

> *"Algumas políticas da organização também podem estar gerando impacto" – l9*
>
> *"Tem algo do ambiente [de trabalho] que atrapalha a produtividade? – Talvez um ambiente tranquilo, mais silencioso" – l11*

Os fatores organizacionais são aqueles que se referem às características existentes no ambiente da organização que impactam no trabalho do desenvolvedor,

onde esses fatores são normalmente controlados pela própria organização. Entre as categorias de fatores que impactam na produtividade do desenvolvedor estão as **políticas organizacionais**, a **gestão de desenvolvedores** pela organização e o **ambiente de trabalho** oferecido aos desenvolvedores.

As **políticas organizacionais** adotadas têm impacto nas produtividade do desenvolvedor de *software*. A adoção de um processo de desenvolvimento com muitas regras pode onerar o trabalho do desenvolvedor, atrapalhando a sua produtividade. As políticas organizacionais quanto ao relacionamento com os clientes, relacionadas ao estabelecimento de prazos e as mudanças de requisitos permitidas durante o desenvolvimento, podem afetar a produtividade do desenvolvedor, pois eles perdem o estímulo ao trabalho. Além disso, os desenvolvedores também perdem produtividade quando os requisitos não são bem definidos, atrasando a entrega de suas tarefas, pois necessitam primeiro completar a especificação dos requisitos.

> *"A gente segue um processo [de desenvolvimento]. Esse processo tem uma série de regras que a gente tem que estar cumprindo e, naturalmente, essas regras geram um impacto na produtividade dos desenvolvedores" – l9*

> *"Ocorre bastante de o desenvolvedor ficar chateado porque já tinha feito algo que o cliente que pediu para mudar em cima da hora e que não estava planejado. Eu acho que tudo isso afeta o desenvolvedor em termos de produtividade, porque ele fica desestimulado" – l8*

> *"A gente consegue saber se ele atrasou, se não atrasou. Depois, a gente tenta entender o porquê aquilo aconteceu, nem sempre foi por falta de foco, às vezes é falta de detalhe do requisito" – l11*

Os líderes de projeto também colocaram que a **gestão de desenvolvedores** influencia na produtividade dos desenvolvedores. Falta de reconhecimento, equipes muito grandes e a realocação de desenvolvedores entre as equipes, possivelmente por causa de priorizações entre os projetos da organização, não contribuem para a produtividade do desenvolvedor. A falta de reconhecimento pode gerar um impacto negativo para o desenvolvedor em relação à organização. Além disso, equipe muito

grandes têm maior a necessidade de sintonia entre os desenvolvedores para se obter uma boa produtividade. Por fim, a mudança da alocação de desenvolvedores entre equipes causa para esses desenvolvedores uma necessidade de readaptação na nova equipe, diminuindo a sua produtividade.

> *"Também pode existir impacto individual da falta de reconhecimento. Também lembrando que o reconhecimento não é só do dinheiro, pode vir por outros meios" – l4*
>
> *"Às vezes você tem uma equipe muito grande e tem dificuldade de manter todo mundo sintonizado [...] Numa equipe muito grande você também tem uma queda de produtividade" – l1*
>
> *"Então às vezes quando você tem projetos que flutuam muito recurso, quando você vai tirar alguém, isso é um problema para a equipe, problema de produtividade, não só para o colega que sai, quanto para o colega que entra em outro grupo." – l6*

Por fim, o **ambiente de trabalho** oferecido aos desenvolvedores por parte da organização também foi considerado como fator de influência na produtividade. Um ambiente de trabalho insalubre ou com muita dispersão influencia negativamente na produtividade do desenvolvedor, uma vez que impacta na sua saúde e no seu foco no trabalho. Além disso, equipamentos defasados podem atrasar pequenas tarefas realizadas pelo desenvolvedor que, somadas ao longo do tempo refletirão em atrasos, influenciando na percepção de produtividade dos líderes de projeto.

> *"Um ambiente salubre influencia. Eu tenho o conhecimento de alguns desenvolvedores aqui que não estão conseguindo produzir porque um tem problema de rinite" – l7*
>
> *"Se o ambiente causa dispersão, tem outras equipes desocupadas ou outras pessoas que acabam tirando a atenção daquela pessoa, é também um dos itens que derruba a produtividade" – l1*
>
> *"É evidente que o equipamento ajuda, de você ter um equipamento bom. É burrice não dar um equipamento bom para o técnico" – l4*

6.3.4 SQ4. Quais dos fatores de influência sobre a produtividade dos desenvolvedores citados são consideradas mais relevantes?

A maioria dos líderes de projeto consideraram **fatores humanos** os mais relevantes sobre a produtividade dos desenvolvedores (Tabela 40). Somente o líder de projeto *l9* considerou o conhecimento técnico, um **fator técnico,** como o mais relevante para a produtividade do desenvolvedor de organizações de *software*.

Os fatores humanos citados entre os fatores mais relevantes foram o foco, a proatividade, o comprometimento e a motivação. A relação desse conjunto de fatores traz a ideia de aquele desenvolvedor que age com foco, maximizando o tempo dedicado ao trabalho; que age com proatividade, buscando por melhorias além do que lhe foi demandado; que age com comprometimento, cooperando com a organização para a finalização do projeto; e motivado, trabalhando ultrapassando os obstáculos que apareçam; além de possuir o conhecimento técnico para desempenhar suas tarefas; esse desenvolvedor, que possua esse conjunto de atributos é o desenvolvedor capaz de superar ou atingir a produtividade esperada pelos líderes de projeto.

É importante ainda observar que os fatores humanos citados podem ter relações de influencia entre si. O líder de projeto *l7* citou uma relação entre o foco e a motivação, dizendo que não é possível ter foco sem a motivação necessária.

Tabela 40. Fatores mais relevantes da produtividade do desenvolvedor.

Id.	Fatores mais relevantes	Citação
l1	Foco, Proatividade	*"Eu tenho percebido que a proatividade juntamente com o foco são as coisas que tornam os desenvolvedores mais produtivos"*
l2	Foco	*"O indivíduo que tem foco sempre faz mais com menos"*
l3	Foco	*"O foco, se você já conhece o negócio e a arquitetura, é só manter o foco"*
l4	Foco, Proatividade	*"Eu acho que o foco, a proatividade e a vontade de querer dar certo dentro da empresa"*
l5	Comprometimento	*"Eu acho que o primordial é a pessoa está envolvida com o que terá que ser feito, é ser comprometida"*

Id.	Fatores mais relevantes	Citação
l6	Comprometimento	*"Acho que o compromisso. Porque eu posso estar aqui, mas posso estar pensando em outra coisa, [...] até familiar mesmo e isso impacta bastante"*
l7	Motivação	*"A motivação é o principal fator, e o foco você acaba perdendo por conta da motivação"*
l8	Foco	*"Quanto mais foco você tem em uma atividade, mais produtivo você é"*
l9	Conhecimento Técnico	*"Posso dizer que o que pode estar impactando seria o conhecimento técnico para o desenvolvedor exercer as suas atividades em um determinado projeto*
l10	Proatividade	*Proatividade, para mim, pesa mais.*
l11	Proatividade	*"Quando o desenvolvedor consegue ir além de apenas só o código, quando eles fazem melhorias..."*
l12	Foco, Motivação	*"É a perda de foco que impacta negativamente aí. [...] Acho que falta de motivação também é um problema ruim"*

6.4 TEORIA FUNDAMENTADA PRELIMINAR SOBRE FATORES DE INFLUÊNCIA NA PRODUTIVIDADE DO DESENVOLVEDOR DE ORGANIZAÇÕES DE *SOFTWARE*

"Você tem que ter motivação para trabalhar" – l5 (a)

"O fato da gente estar programando, trabalhando com uma máquina, não vai eliminar o fator humano" – l4 (a)

Durante o processo de codificação e análise qualitativa desse estudo, especificamente das subquestões *SQ3* e *SQ4*, foram identificados relacionamentos de influência entre os fatores de produtividade do desenvolvedor, citados pelos líderes de projeto das organizações. Esses relacionamentos entre os fatores conduziram nossa análise até a geração de uma teoria sobre os fatores de influência na produtividade do desenvolvedor de organizações de software.

Os fatores técnicos não são os mais determinantes para a produtividade do desenvolvedor (Figura 17) – Eles servem de base para que o desenvolvedor consiga executar suas tarefas de acordo com os requisitos demandados. A falta de conhecimento, incluindo o domínio e a experiência na tecnologia e no negócio, impedem que o desenvolvedor produza de forma efetiva, pois ele precisa parar sua

produção para aprender como realizar sua tarefa. Uma vez que o desenvolvedor obtém o conhecimento necessário, a produtividade dele passa a depender somente de seu comportamento e de suas atitudes. Se o desenvolvedor, por exemplo, não focar no trabalho, de nada adiantará seus conhecimentos e sua experiência para a sua produtividade, segundo os líderes de projeto.

> *"Se você conhece mais, produz mais. Conhece bem tecnologia e conhece bem a área de negócio vai produzir muito, se tiver interesse e estiver motivado é claro" – l4 (a)*
>
> *"– Quer dizer que o conhecimento técnico dele ajuda até a pessoa dominar a tecnologia? – Isso, dominar. – E depois que ele domina? – Aí depende dele, aí não é mais questão técnica, é fator comportamental" – l3 (b)*
>
> *"A questão do conhecimento técnico. Apesar de ser uma coisa que também influencia, dependendo do nível de criticidade da entrega, eu acho que não é um fator tão importante" – l7 (b)*
>
> *"Tem muito a questão de perca de foco. Não adianta o cara dominar mais [a tecnologia] se ele perde constantemente o foco" – l12 (b)*
>
> *"Se ela não entregar as coisas no prazo, por conta de que perdeu o foco, ela não está sendo produtiva para mim. O desenvolvedor pode ser bem experiente, mas não está aplicando a experiência dele para atingir os objetivos alocados para a semana, então não está sendo produtivo" – l3 (b)*

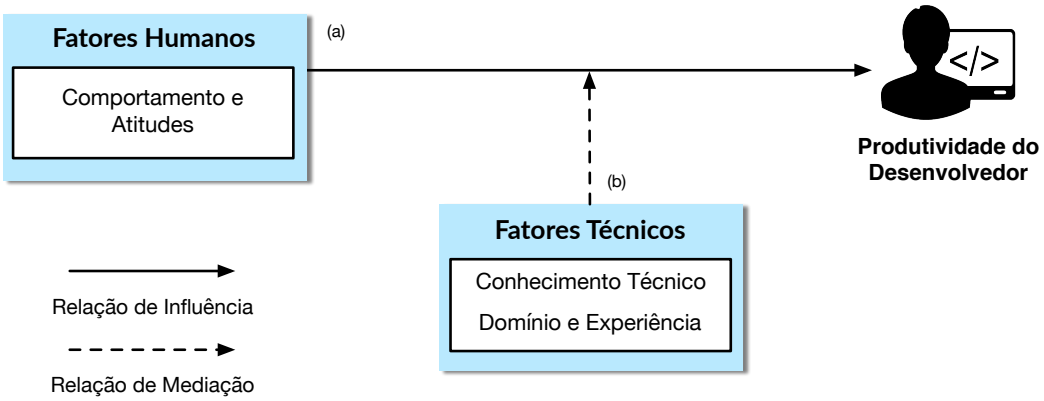

Figura 17. Fatores técnicos e a produtividade do desenvolvedor.

Os fatores organizacionais podem ter impacto relevante no comportamento e atitudes do desenvolvedor (Figura 18) – A organização pode influenciar positivamente ou negativamente seus desenvolvedores através das suas políticas organizacionais, do ambiente de trabalho oferecido e do reconhecimento por seu trabalho.

As políticas organizacionais adotadas pela organização durante a execução dos seus processos, especialmente considerando os prazos acordados com o cliente, influenciam na motivação dos desenvolvedores. Os desenvolvedores se sentem desmotivados quando a organização estabelece prazos muito curtos para um projeto. Uma explicação para isso é que os desenvolvedores acreditam que seja bem possível a necessidade deles trabalharem além do seu horário normal de expediente, para que consigam cumprir os prazos acordados pela organização.

> *"A gente já sabe que aquele cliente é complicado, que os prazos são muito curtos. Acho que isso leva algumas pessoas, não todas, a já entrar no projeto desestimulado porque é o cliente X. (...) Tem gente que já chega desmotivado" – l8 (c)*

> *"Às vezes, pode ser porque os desenvolvedores fazem uma estimativa e chega na coordenação e na gerência e acabam por remover várias horas da estimativa. No final das contas, acabou que o time já começa com o sentimento ruim de que não vai dar tempo, [que] está tudo muito apertado. Então, isso impacta bastante na motivação deles" – l12 (c)*

Quando a organização adota práticas de reconhecimento de seus desenvolvedores, colabora para que o desenvolvedor se sinta com mais motivação dentro de seu trabalho e consequentemente, será mais produtivo. Por fim, o ambiente de trabalho oferecido pela organização aos seus desenvolvedores influencia na sua motivação.

> *"O ambiente de trabalho que a organização oferece realmente motiva muitas pessoas a ficar aqui" – l9 (d)*

> *"Outra coisa positiva que a organização faz é que, quando alcança a meta, a gente tem uma premiação que é o projeto destaque, onde [...] tem um reconhecimento público dentro da instituição. E isso*

colabora bastante para a autoestima e para a motivação dos colaboradores, o que afeta também a produtividade deles" – l12 (e)

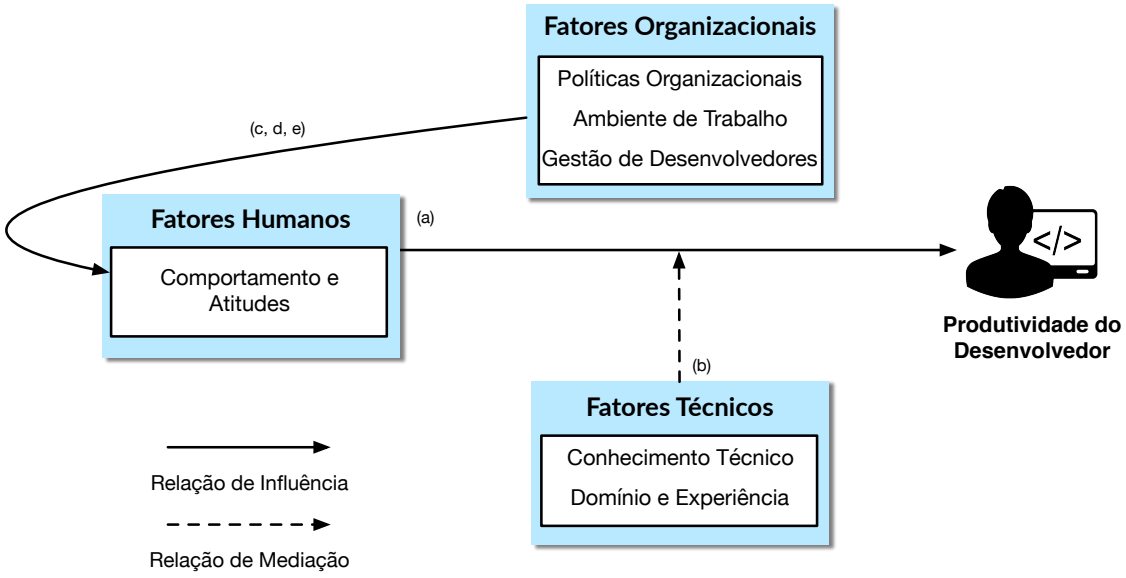

Figura 18. Fatores organizacionais e a produtividade do desenvolvedor.

Os fatores humanos têm como ponto central a motivação – o relacionamento interpessoal do desenvolvedor com o restante da equipe e das lideranças da organização influenciam na sua motivação para o trabalho. Além disso, a motivação do desenvolvedor é um fator de influência sobre seus outros comportamentos e atitudes.

O relacionamento interpessoal do desenvolvedor com o restante da organização afeta sua motivação para o trabalho (Figura 19). O líder de projeto pode exercer um papel de motivador dos seus desenvolvedores, criando um ambiente amistoso e agradável para o trabalho; no entanto, quando o desenvolvedor sente-se abandonado dentro da organização, ou pressionado de forma dura pelas lideranças, a sua motivação é influenciada negativamente.

"Acho que o relacionamento com o líder é um fator importante para estar motivando e impactando no desempenho do desenvolvedor" – l9 (f)

"A motivação sem dúvida, porque quando você tem um líder muito ausente ou que não está ali ouvindo as pessoas, ouvir os problemas, saber do outro lado que tem alguém, que está tentando

entender o que está acontecendo, não só impondo metas, não só cobrando, mas também fazendo parte da solução, isso ajuda na motivação" – l6 (f)

"Cara a motivação. Eu já trabalhei em empresas em que o chefe gritava com a gente" – l2 (f)

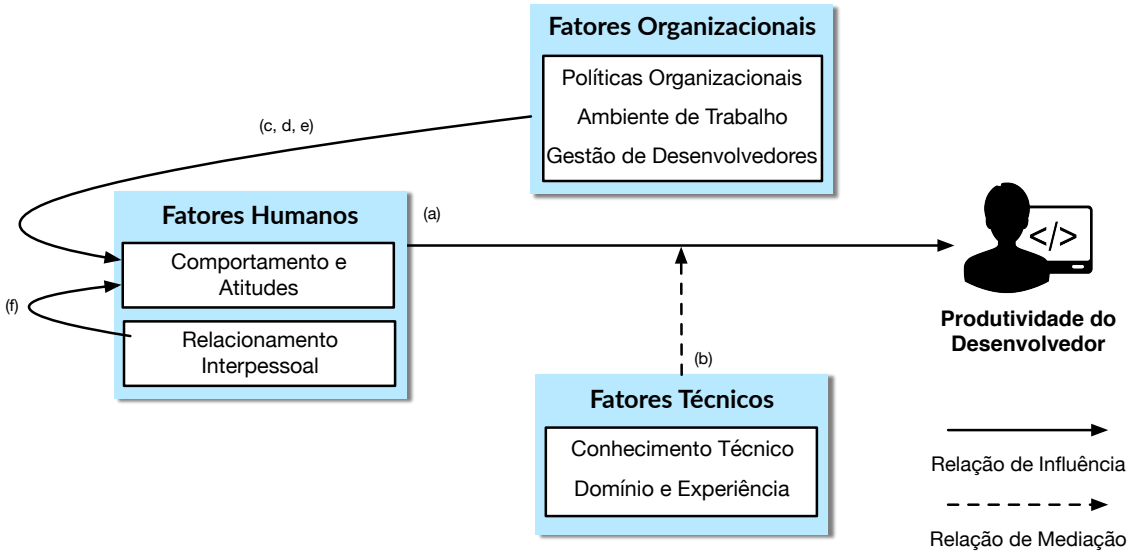

Figura 19. Fatores humanos (relacionamentos) e a produtividade do desenvolvedor.

Por fim, a motivação parece possuir um papel central dentro do comportamento e atitudes do desenvolvedor. De acordo com os líderes de projeto, a motivação tem influência sobre o comprometimento, a proatividade e o foco do desenvolvedor.

"A motivação interfere no comprometimento" – l7 (g)

"A motivação é a pior de todas, porque você não vê perspectiva, então por isso, por não ter perspectiva, você acaba perdendo a iniciativa" – l7 (h)

"O foco você acaba perdendo, por conta da motivação. Se eu estou fazendo uma atividade que para mim não é interessante, eu desfoco muitas vezes, e aí eu acabo atrasando também. Então, eu acho que a motivação influencia, com certeza" – l7 (i)

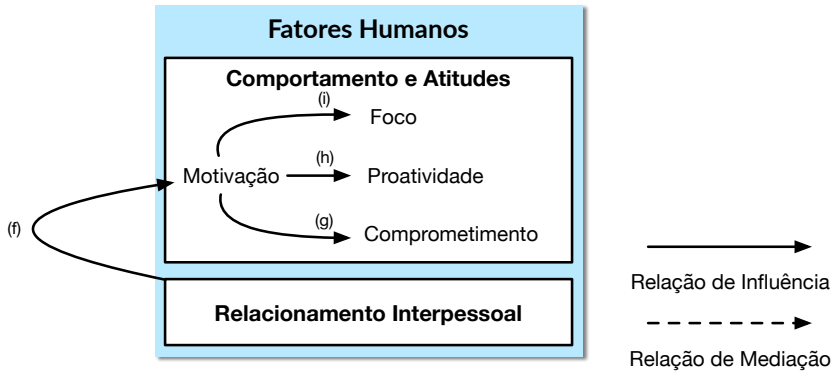

Figura 20. Fatores humanos e a motivação como ponto central.

A partir de todos esses achados, foi possível desenvolver uma teoria fundamentada preliminar conforme o ponto de vista dos líderes de projeto, sobre os fatores de influência na produtividade do desenvolvedor de organizações de *software* (Figura 21). Todas as citações encontradas estão representadas nas figuras anteriores e na Figura 21 pelas letras entre parênteses.

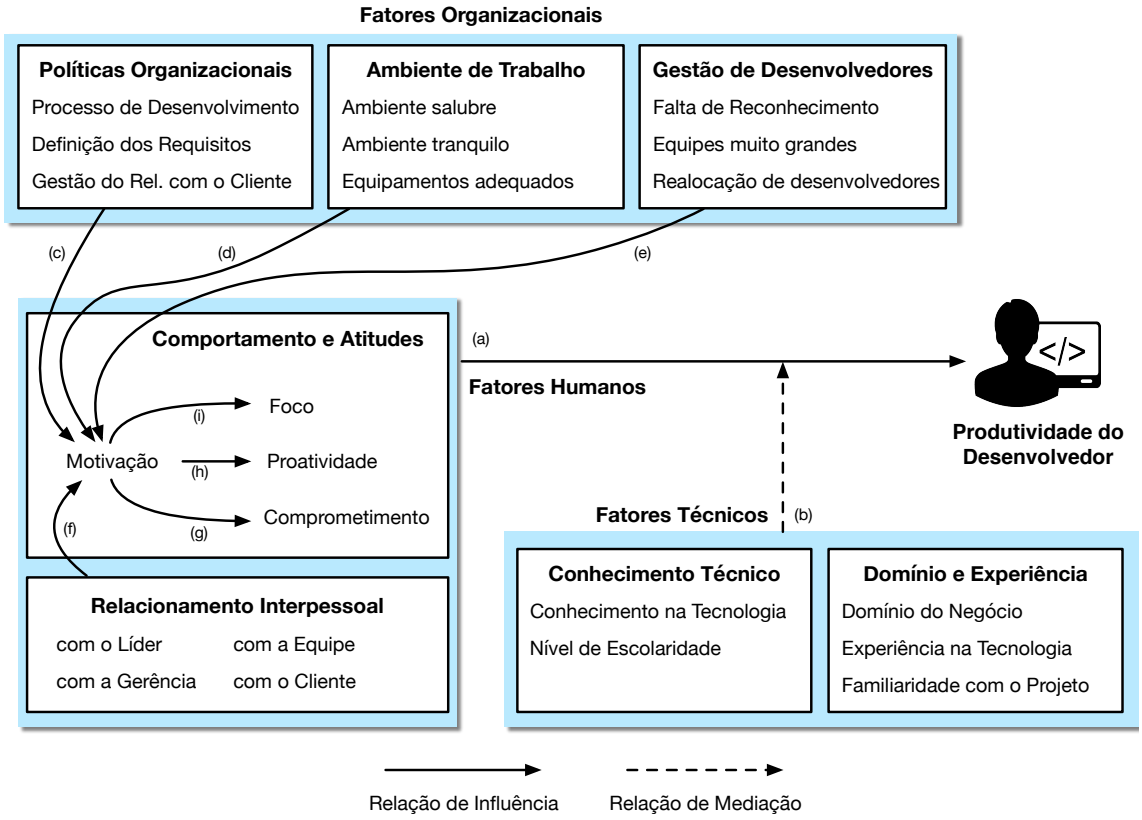

Figura 21. Fatores de Influência sobre a Produtividade dos Desenvolvedores.

A Figura 21 apresenta todas as relações de influência entre os fatores de acordo com as citações dos líderes de projeto. As relações de influência apresentadas

estão conectadas com as citações dos líderes entrevistados de acordo com a identificação alfabética utilizada entre parênteses. Com esta figura é possível observar que o comportamento e atitudes são o ponto central das observações realizada pelo líder com relação a produtividade. Além disso, os demais fatores citados influenciam o comportamento e as atitudes do desenvolvedor. Já os fatores técnicos tem uma influência de mediação na produtividade, pois uma vez que o desenvolvedor possua os conhecimentos técnicos e a experiência necessária para execução de suas tarefas, esse fator não impacta mais em sua produtividade.

6.5 DISCUSSÃO DOS RESULTADOS

6.5.1 QP1. Como os líderes de projeto conceituam, identificam a produtividade dos desenvolvedores da organização?

Este estudo teve como um dos seus objetivos investigar como os líderes de projetos de *software* das organizações selecionadas entendem e identificam a produtividade dos seus desenvolvedores. Os líderes de projeto das organizações são os responsáveis por fazer com que o projeto alcance o sucesso e, para isso, alocar e acompanhar a tarefa dos desenvolvedores de *software*.

Foi mostrado que os líderes de projeto consideram que a produtividade de um desenvolvedor é entregar suas tarefas no prazo, com qualidade, ou seja, sem a necessidade de retrabalho dessas atividades, com comportamento e atitudes coerentes com a execução de suas tarefas. Baseado nesses componentes, os líderes de projeto identificam a produtividade de um desenvolvedor pelas características da entrega de suas tarefas, pela observação de seu comportamento e atitudes, pelo feedback dado por ele e por membros de sua equipe, além do *histórico* de seu trabalho dentro da organização.

Esses resultados são similares e consistentes com os resultados obtidos com os gerentes de *software* das organizações. Em ambos os estudos, o conceito de produtividade do desenvolvedor envolve a entrega das tarefas no prazo, sem necessidade de retrabalho, considerando o comportamento e atitudes do desenvolvedor. A diferença só está em que os gerentes de *software* também consideram como elemento importante deste conceito o atendimento às expectativas das partes interessadas (*stakeholders*). Essa diferença é explicada dada

a diferença de cargos na organização, pois enquanto o líder tem foco no andamento do projeto, o gerente tem o foco no atendimento ao cliente. Da mesma forma, os resultados são similares quanto à forma de identificação da produtividade. Baseados nesse conceito de produtividade, os gerentes e os líderes identificam a produtividade de seus desenvolvedores a partir de seus comportamentos e atitudes, pelas características das entregas de suas tarefas, além do *feedback* recebido. Os líderes de projeto ainda citaram que, ao longo do tempo, criam um histórico do desenvolvedor.

As similaridades de conceito e identificação entre gerentes e líderes mostram o alinhamento existente nas organizações de *software* pesquisadas quanto ao conceito e identificação da produtividade dos desenvolvedores. Esses resultados combinados sugerem que a eficácia do desenvolvedor, o atendimento ao cliente no prazo com qualidade, tem maior valor do que a eficiência, o atendimento ao cliente antes do prazo. A identificação de uma necessidade de retrabalho é sinal importante de falta de produtividade do desenvolvedor, significando ainda mais para a organização, que pode passar a ser vista como possuindo processos de baixa qualidade. Ademais, assim como o retrabalho identifica uma falha de qualidade, o comportamento e atitudes podem indicar se o desenvolvedor está produzindo aquilo que pode, ou se está abaixo da expectativa. Esses dois sinais são registrados nas percepções dos líderes e gerentes da organização, gerando um histórico da percepção de produtividade dos seus desenvolvedores, obtido mesmo com a ausência de métricas formais de medição da produtividade.

6.5.2 QP2. Quais são os fatores relevantes de influência sobre a produtividade dos desenvolvedores de software da organização?

Este estudo teve como outro objetivo investigar, sobre o ponto de vista dos líderes de projeto, quais são os fatores relevantes de influência sobre a produtividade dos desenvolvedores de *software* das organizações. Mais de 20 fatores foram citados pelos líderes de projeto. Esses fatores foram categorizados e agrupados em fatores técnicos, humanos e organizacionais, visando uma melhor compreensão. Os fatores técnicos identificados foram o conhecimento técnico e o domínio e experiência; os fatores humanos foram o comportamento e atitudes dos desenvolvedores, e o relacionamento interpessoal, com a equipe, com as lideranças

e com os clientes; os fatores organizacionais foram as políticas organizacionais, a gestão de desenvolvedores pela organização e o ambiente de trabalho oferecido aos desenvolvedores. A maioria dos líderes de projeto consideraram os fatores humanos como os mais relevantes sobre a produtividade dos desenvolvedores, exceção somente a um líder de projeto que considerou um fator técnico como o mais relevante.

Existem muitos estudos sobre os fatores de influência na produtividade, conforme apresentado na revisão sobre os fatores de influência da produtividade (Capítulo 4). Todos os fatores citados pelos líderes de projeto estão presentes na literatura científica, variando somente na nomenclatura utilizada, problema similar encontrado nas revisões sistemáticas sobre o assunto [Trendowicz e Münch 2009]. Por outro lado, e até o melhor do nosso conhecimento, estudos comparativos da relevância entre fatores de influência da produtividade são raros. O trabalho de Paiva *et al.* (2010) é o único estudo encontrado com objetivo mais similar ao realizado nesta pesquisa.

Paiva *et al.* (2010) investigaram os fatores de influência da produtividade pela visão dos desenvolvedores de *software*. Para isso, os autores realizaram uma revisão sistemática e um *survey* com desenvolvedores de *software*. A revisão sistemática extraiu 32 fatores de influência da literatura, que foram utilizados em um *survey* com 77 desenvolvedores, para classificação desses fatores por ordem de relevância na influência da produtividade dos desenvolvedores. Os resultados apresentaram a motivação como o fator de maior relevância para os desenvolvedores, seguido, em ordem de relevância, pelo salário, o ambiente de trabalho, o comprometimento, a experiência, o relacionamento interpessoal, os requisitos consistentes, a gestão do projeto, a máquina e o domínio da aplicação, completando os dez fatores mais relevantes na influência da produtividade dos desenvolvedores. Além disso, para os cinco fatores mais relevantes, os desenvolvedores foram questionados sobre quais fatores exerciam influência sobre esses cinco mais relevantes. Dessa forma, os autores encontraram que a motivação é influenciada pelo ambiente de trabalho e influencia, por sua vez, o comprometimento.

Os resultados obtidos por Paiva *et al.* (2010) são bem similares aos obtidos nesta pesquisa. A motivação também foi identificada como um fator central de

influência na produtividade dos desenvolvedores. Outra similaridade é que, em ambos os estudos, a motivação é influenciada pelo ambiente de trabalho e influencia o comprometimento. Apesar dessas similaridades, também foram encontradas algumas diferenças, como a presença do salário como um fator importante para os desenvolvedores. Nesta pesquisa, o salário não foi considerado como um fator de influência, uma vez que, de acordo com os líderes de projeto desta pesquisa, o salário ajuda na produtividade por pouquíssimo tempo e, "*daqui há pouco, o salário não está bom [de novo]...*" (*l1*). Ademais, a gestão de projetos não foi considerada explicitamente nem pelos líderes de projeto nem pelos gerentes de *software*, porém esse fator indicado pelos desenvolvedores parece pode ter uma relação com as políticas organizacionais e gestão dos desenvolvedores, considerados fatores organizacionais. Uma possível explicação para essa relação está no fato de que mudanças não planejadas de requisitos ou a realocação dos desenvolvedores podem ter sido consideradas como uma falta de planejamento adequado da organização, sob o ponto de vista dos desenvolvedores.

Os resultados do processo de codificação e análise qualitativa identificaram diversos relacionamentos de influência entre os fatores de produtividade do desenvolvedor, de acordo com os líderes de projeto das organizações. Os líderes consideram os fatores técnicos, como o conhecimento e a experiência na tecnologia, como habilitadores para a produtividade. A ausência desses fatores pode diminuir a produtividade pela necessidade imediata, durante o desenvolvimento, de aprender e utilizar corretamente uma nova tecnologia. Porém, mesmo habilitados, não são os fatores técnicos que impulsionam a produtividade do desenvolvedor só por ter maior conhecimento ou experiência. Esse resultado está consistente com resultados de estudos [Jeffery e Lawrence 1985; Kitchenham e Mendes 2004; Lawrence 1981; Maxwell *et al.* 1996] que ao longo do tempo identificaram que o conhecimento e a experiência influenciam na produtividade somente nos primeiros anos de trabalho com a tecnologia.

Se consideramos que o papel dos fatores técnicos em relação à produtividade é habilitar, o papel dos fatores organizacionais é de ditar as normas e formas do trabalho ao desenvolvedor. As organizações de software realizam a gestão das equipes de desenvolvimento, organizam e mantêm o ambiente de trabalho dos desenvolvedores e estabelecem suas políticas organizacionais, como a definição do

processo de desenvolvimento. O ambiente de trabalho é um fator pouco registrado na literatura, pois Wagner e Ruhe (2008) encontraram somente três referências, sendo dois livros. A gestão dos desenvolvedores desta pesquisa encontram suporte na literatura, segundo as revisões de Wagner e Ruhe (2008) e Trendowicz e Münch (2009), em pelo menos dez estudos na literatura. Por fim, as políticas organizacionais também possuem suporte na literatura. Considerando somente o processo de desenvolvimento, por exemplo, existem mais de 15 estudos sobre esse fator na década de 2000 a 2009 [Sampaio *et al.* 2010]. Todos esses fatores organizacionais agem como mediadores da produtividade, dando suporte ou, em alguns casos, como em um processo cheio de regras, bloqueando a produtividade dos desenvolvedores da organização.

Os fatores humanos são, segundo esta pesquisa, aqueles que de fato impulsionam a produtividade dos desenvolvedores de *software*. Esse resultado está consistente com a afirmação de Hernández-López *et al.* (2013), afirmando que o sucesso dos projetos de desenvolvimento de *software* ainda depende das pessoas, e de Amrit *et al.* (2014), afirmando que os fatores humanos têm um papel crítico dentro do desenvolvimento de *software*. A teoria preliminar aqui identificada coloca os fatores humanos como parte central na influência da produtividade. Dessa forma, os líderes de projeto percebem como fatores humanos determinantes o foco, o comprometimento e a proatividade, como características perceptíveis para identificar um desenvolvedor produtivo. Essas características possuem como combustível a motivação do desenvolvedor. Dessa forma, os desenvolvedores motivados têm maior chance de atingir bons patamares de produtividade do que aqueles que são percebidos como desmotivados [Beecham *et al.* 2008; França *et al.* 2011].

O desenvolvedor de software é peça fundamental para todo projeto de desenvolvimento. Além disso, dentre as muitas decisões gerenciais que precisam ser tomadas em um projeto de *software*, os fatores humanos estão entre os que mais afetam a produtividade de *software* [Trendowicz e Münch 2009]. Esses resultados obtidos podem ajudar as organizações de *software* a decidir onde fazer os investimentos necessários para o aprimoramento da produtividade dos seus desenvolvedores, além de também ajudar aos pesquisadores da Engenharia de *Software* a decidir para quais fatores devem investir mais esforços de pesquisa.

6.6 AMEAÇAS À VALIDADE

Nesta seção são apresentadas as principais as ameaças à validade deste estudo. A primeira ameaça à validade deste estudo é quanto a generalização dos resultados obtidos para todas as outras organizações de *software*. Estudos qualitativos não utilizam a amostragem estatística para reivindicar a generalização de seus resultados. Ainda assim, para mitigar essa ameaça, foram entrevistados 12 líderes de projetos de três organizações de desenvolvimento de *software*.

Outra ameaça é que as percepções dos líderes de projeto podem ser tendenciosas em relação a suas próprias crenças, causando assim distorções na interpretação da realidade e consequente distorções nos resultados obtidos. Para reduzir essa ameaça, foi solicitado às organizações a participação na pesquisa dos seus líderes de projetos de *software* com maior experiência.

A possibilidade do autor do estudo ter introduzido seu viés no processo de análise de dados é outra ameaça à validade dos resultados. Para mitigar essa ameaça o processo de análise dos dados coletados foi realizado junto com outro pesquisador mais experiente. Esses pesquisadores revisaram e analisaram os resultados intermediários. Este processo de revisão foi iterativo e foi repetido até o final da coleta e análise de dados.

6.7 CONSIDERAÇÕES FINAIS

Este capítulo descreveu um estudo utilizando uma investigação qualitativa sobre a produtividade do desenvolvedor a partir da perspectiva dos líderes de projeto das organizações de *software*. O estudo foi conduzido nas três organizações de software selecionadas, com um total de 12 líderes de projeto entrevistados. Os resultados obtidos a partir da percepção dos líderes de projeto indicam um conceito de produtividade dos desenvolvedores baseado na entrega das tarefas dentro do prazo, sem a necessidade de retrabalho e possuindo qualificação técnica e comportamental. É a partir dos componentes desse conceito que os líderes de projeto também identificam a produtividade de seus desenvolvedores. Por fim, os fatores humanos são um ponto central na produtividade dos desenvolvedores, enquanto os fatores técnicos e organizacionais possuem papéis diferentes na produtividade do desenvolvedor, ambos podendo gerar obstáculos à produtividade.

Capítulo 7 – MÉTRICAS DE PRODUTIVIDADE DO DESENVOLVEDOR DE ORGANIZAÇÕES DE *SOFTWARE*

Este capítulo apresenta um estudo quantitativo sobre a representatividade, a partir da percepção dos líderes de projeto, de um conjunto de métricas selecionadas no mapeamento sistemático.

7.1 INTRODUÇÃO

Este capítulo apresenta um estudo quantitativo para investigar as métricas de produtividade do desenvolvedor segundo a percepção dos líderes de projeto das organizações de *software*. Após investigar com os gerentes (Estudo *E3*, Capítulo 5) e líderes de projeto (Estudo *E4*, Capítulo 6) sobre suas percepções da produtividade, objetivou-se investigar se as métricas de produtividade utilizada pelos pesquisadores tinham relação com as percepções de produtividade dos líderes de projeto de cada organização.

Um conjunto de métricas de produtividade foi extraída da literatura (Estudo *E3*, Capítulo 3). Esse conjunto foi dividido em dois grupos: métricas baseadas nos arquivos de código fonte e métricas baseadas na frequência de *commits* realizados nos repositórios de código. Os resultados deste estudo foram obtidos através da correlação entre os *rankings* gerados pelas métricas e pelos *rankings* fornecidos pelos líderes dos projetos, permitindo identificar quais métricas de produtividade melhor correspondem às percepções da organização. Este estudo foi realizado apenas em duas das três organizações selecionadas, pois os repositórios de código fornecidos por uma dessas organizações não possuíam dados suficientes para realização do estudo correlacional.

7.2 PLANO DE ESTUDO

O principal objetivo deste estudo é investigar a relação entre as métricas de produtividade e a percepção das organizações de *software* sobre a produtividade do desenvolvedor. As questão principal de pesquisa (*QP*) deste estudo está apresentada na Tabela 41. Porém, para responder a esta pergunta principal, é necessário antes

estabelecer as métricas de produtividade que serão utilizadas, como seu relacionamento será avaliado e de quem e como as percepções da organização do *software* serão coletadas. As métricas escolhidas serão detalhadas na Subseção 7.2.1, enquanto a forma de análise será a correlação estatística detalhada na Subseção 7.2.3 e os líderes de projeto serão os representantes das percepções da organização.

Tabela 41. Questões de pesquisa para o estudo sobre as métricas de produtividade.

QP	As métricas de produtividade do desenvolvedor de *software* correspondem às percepções das organizações?
SQ1	Qual a correlação entre os *rankings* de produtividade obtidos pelas métricas de produtividade e pelas percepções do líderes do projeto?
SQ2	Qual o impacto do contexto da organização na percepção dos líderes de projeto sobre a produtividade de seus desenvolvedores?

A subquestão *SQ1* investiga como as métricas de produtividade do desenvolvedor se comparam às percepções dos líderes de projeto sobre a produtividade de seus desenvolvedores, conforme descrito no capítulo anterior (Capítulo 6). Neste estudo, foi decidido se concentrar nas percepções dos líderes de projeto porque suas tarefas de gerenciamento têm uma influência direta na produtividade do desenvolvedor, conforme discutido em estudos anteriores [Sampaio *et al.* 2010; Trendowicz e Münch 2009; Wagner e Ruhe 2008]. Além disso, os líderes de projeto são frequentemente responsáveis pela coordenação das tarefas da equipe de desenvolvimento, gozando da confiança por parte da organização de *software* [Trendowicz e Münch 2009].

A subquestão *SQ2* investiga como o contexto organizacional influencia a percepção dos líderes de equipe sobre a produtividade do desenvolvedor. O acesso a duas organizações de *software* diferentes permitiu com que fosse possível analisar se suas diferentes características de contexto influenciam a relação entre as percepções dos líderes de equipe sobre a produtividade do desenvolvedor e as métricas de produtividade do desenvolvedor.

7.2.1 Métricas de produtividade do desenvolvedor

A avaliação da produtividade é essencial para apoiar a tomada de decisões dos líderes das equipes, pois geralmente esses líderes coordenam as tarefas de desenvolvimento. No entanto, as percepções dos líderes sobre a produtividade do

desenvolvedor podem variar devido à natureza subjetiva do desenvolvimento de software [DeMarco e Lister 1987], tornando difícil avaliar sistematicamente a produtividade do desenvolvedor [Bessen e Hunt 2007]. No entanto, várias métricas de produtividade do desenvolvedor foram propostas, apesar de que nenhum consenso sobre as métricas de produtividade foi até agora alcançado [Oliveira *et al.* 2017; Petersen *et al.* 2008].

O conjunto de métricas selecionadas para este estudo baseou-se na revisão sistemática sobre métricas de produtividade apresentada no Capítulo 3 e também na revisão sistemática realiza por Petersen (2011). Essas métricas foram classificadas nesta pesquisa em dois grupos: métricas baseadas em código e em métricas baseadas na frequência de *commits*.

As métricas selecionadas baseadas em código foram mais frequentemente utilizadas em estudos anteriores [Chand e Gowda 1993; Jeffery e Lawrence 1985; Lawrence 1981; Scacchi 1995] para avaliar a produtividade do desenvolvedor (Capítulo 3). Essas métricas baseadas em código avaliam a produtividade do desenvolvedor ao analisar as características dos arquivos de código fonte por unidade de tempo. Uma dessas características é o tamanho do arquivo do código fonte, significando que quanto maior for o código produzido por um desenvolvedor, mais possibilidade desse desenvolvedor ser mais produtivo. Essa característica é frequentemente medida usando a quantidade de linhas de código fonte (*SLOC/Time*) [Lawrence 1981]. Outra característica é a complexidade do código fonte, significando que códigos mais complexos podem indicar maior produtividade do desenvolvedor. Apenas uma métrica de produtividade (*HalsteadEff./Time*) foi encontrada na literatura que utiliza essa abordagem para avaliar a produtividade do desenvolvedor [Chand e Gowda 1993]. Para calcular a produtividade do desenvolvedor também é necessário identificar o proprietário principal de cada arquivo de código. Dessa forma, utilizamos o conceito de propriedade código proposto por [Bird *et al.* 2011] para identificar e calcular a produtividade do desenvolvedor. Além disso, apresentamos uma nova métrica baseada em código usando apenas essa característica, avaliando a produtividade do desenvolvedor através da quantidade de arquivos de código de propriedade dele (*CodeOwned/Time*).

Estudos recentes [Mockus *et al.* 2000, 2002; Munson e Elbaum 1998; Scholtes *et al.* 2016] têm explorado os repositórios de código fonte para avaliar a produtividade dos desenvolvedores. Os autores desses estudos utilizam métricas de produtividade baseadas na frequência de *commits*, capturando propriedades de produtividade do desenvolvedor refletidas pelas atividades registradas no repositório de código. Uma abordagem utilizada para avaliar a produtividade do desenvolvedor é contar a quantidade de *commits* realizadas por cada desenvolvedor, significando que quanto mais *commits* realizado por um desenvolvedor, mais produtivo ele será considerado (*Commits/Time*) [Mockus *et al.* 2000, 2002]. Outra abordagem utilizou a quantidade de linhas de código modificadas em cada *commit* realizado. Essa métrica (*CommitsLM/Time*) também é conhecida como *Code Churn* [Munson e Elbaum 1998]. Finalmente, outra abordagem utilizou a quantidade de edições por um desenvolvedor realizadas nos arquivos de código em *commits* realizados. Esta última abordagem avaliou as alterações no código-fonte ao nível de caracteres individuais através da distância de edição Levenshtein (*CommitsED/Time*) [Levenshtein 1966; Scholtes *et al.* 2016].

A Tabela 42 apresenta as métricas selecionadas dos dois grupos para este estudo, assim como a formula que as definem e sua descrição.

Tabela 42. Métricas de produtividade do desenvolvedor.

Grupo	Métrica	Descrição
Métricas baseadas em Código	*CodeOwned/Time*	Número de arquivos de código fonte de propriedade do desenvolvedor pelo tempo de projeto
	SLOC/Time	Quantidade de linhas de código fonte de propriedade do desenvolvedor pelo tempo de projeto
	HalsteadEff/Time	Esforço de *Halstead* dos arquivos de código fonte de propriedade do desenvolvedor pelo tempo de projeto
Métricas baseadas em *Commits*	*Commits/Time*	Número de *commits* realizados pelo desenvolvedor pelo tempo de projeto
	CommitsLM/Time	Quantidade de linhas modificadas por *commit* do desenvolvedor pelo tempo de projeto
	CommitsED/Time	Distância de edição de Levenshtein dos *commits* realizados pelo desenvolvedor pelo tempo de projeto

7.2.2 Procedimentos de coleta de dados

Para realizar a investigação e para responder às questões de pesquisa propostas neste estudo, os procedimentos adotados para coleta e análise dos dados estão apresentados na Figura 22. A coleta dos dados foi realizada durante a execução do Estudo *E4* (Capítulo 6). Após as entrevistas realizadas com os líderes projeto no estudo anterior, os mesmos respondiam a um questionário. O questionário aplicado perguntava o *ranking* dos desenvolvedores do projeto, do mais produtivo para o menos produtivo, de acordo com suas percepções sobre a produtividade dos desenvolvedores em seu projeto. Durante a aplicação do questionário, os líderes do projeto não tiveram acesso a nenhuma outra fonte de informação do projeto, recorrendo apenas à sua memória.

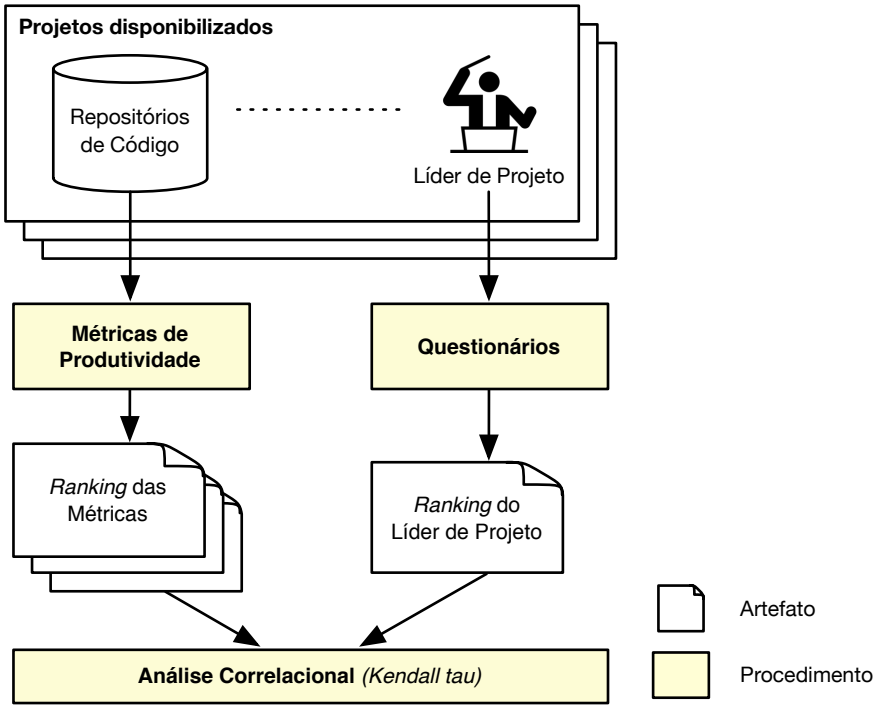

Figura 22. Procedimentos adotados na execução do estudo.

Os repositórios de código foram coletados durante e após a execução do estudo anterior. Todas as organizações selecionadas forneceram repositórios de seus projetos de *software*, correspondentes aos líderes entrevistados, selecionados de acordo com a conveniência dos gerentes de *software*. Dos projetos coletados somente nove projetos de *software*, das organizações 2 e 3, puderam ser utilizados neste estudo. Foram excluídos os projetos que não possuíam uma quantidade mínima de desenvolvedores para realização de um estudo correlacional, incluindo

todos os projetos da Organização 1. Relacionados a estes nove projetos estavam oito líderes, já que dois projetos possuíram o mesmo líder. Esses líderes das organizações 2 e 3 foram os mesmos entrevistados no Estudo *E4*, apresentados na Tabela 37.

As métricas de produtividade do desenvolvedor selecionadas foram calculadas usando dados coletados diretamente dos repositórios de *software* das organizações. Esses cálculos foram realizados através de programas desenvolvidos especificamente para essa pesquisa. As métricas de produtividade foram calculadas considerando apenas os arquivos de código fonte da linguagem de programação principal de cada projeto.

Para as métricas baseadas na frequência de *commits*, a informação do autor do *commit* dos repositórios foi utilizada para definir a autoria do *commit*. Para calcular as métricas baseadas em código fonte, foi utilizado o conceito de propriedade do código (*Code Ownership*), conforme proposto por Bird *et al.* (2011). Essa estratégia foi necessária, pois é comum que vários desenvolvedores tenham contribuído para os mesmos arquivos de código fonte. Utilizando a métrica de *Code Ownership*, os maiores contribuidores para um determinado código fonte foram identificados como os donos (*owner*) do referido código fonte. Com essa estratégia, um conjunto de arquivos de código fonte foi associado com cada um dos desenvolvedores do projeto. As métricas baseadas em arquivos de código fonte de cada desenvolvedor foram calculadas utilizando seu conjunto de arquivos de código fonte. Finalmente, a partir de cada métrica calculada, um *ranking* dos desenvolvedores foi gerado.

7.2.3 Procedimentos de análise de dados

A avaliação da relação entre as métricas de produtividade do desenvolvedor e as percepções dos líderes dos projetos foi realizada através da correlação dos *rankings* obtidos. A decisão pela utilização da correlação entre os *rankings* obtidos foi baseado em duas principais razões: (i) os *rankings* fornecem uma maneira de comparar as métricas de produtividade, mesmo que essas métricas não estejam nas mesmas unidades e (ii) seria difícil para os líderes de projeto estimar a produtividade dos desenvolvedores através de valores nas unidades específicas das métricas de produtividade.

A análise de correlação, para avaliar a relação entre os *rankings*, foi realizada utilizando o coeficiente de correlação Kendall tau (τ) [Kendall 1955]. O coeficiente de Kendall tau é um teste não-paramétrico apropriado para este estudo, porque não assume como requisito que os dados sejam de uma distribuição normal. Além disso, esse coeficiente de correlação funciona bem com amostras pequenas, o que é o caso para os dados coletados, devido ao pequeno número de desenvolvedores identificados na maioria dos projetos coletados. O coeficiente de correlação tau de Kendall também calcula o significado da correlação obtida, o que é útil para a interpretação dos resultados e, para isso, será utilizado como guia a proposta de interpretação sugerida por Cohen (1988).

7.3 RESULTADOS OBTIDOS

7.3.1 Repositórios das Organizações de *Software*

Após a coleta e extração dos dados dos nove repositórios de *software* fornecidos pelas organizações, foram processados um total de 34,856 *commits* contribuídos por 187 diferentes *commiters* (Tabela 43). Alguns dos *commits* encontrados nos repositórios foram realizados por pessoas que não eram desenvolvedores do projeto, tais como *commits* de equipes de documentação e *commits* de ferramentas automatizadas. Nenhum desses *commits* realizaram alterações em arquivo de código fonte do projeto em desenvolvimento. Dessa forma, esses *commits* e seus respectivos autores não foram computados nos cálculos das métricas de produtividade. Somente os *commits* relacionados com os arquivos de código fonte foram considerados.

Um total de 68 desenvolvedores foram identificados em todos os projetos investigados. Ao comparar a lista desses desenvolvedores com a lista de desenvolvedores presentes nos *rankings* fornecidos pelos líderes do projeto, apenas três desenvolvedores de dois projetos (P3 e P5) foram identificados nos repositórios, mas não foram mencionados pelos líderes do projeto em seus *rankings*. Após um novo contato com os líderes desses projetos para entender porque esses desenvolvedores não foram mencionados, o líder do projeto P3 informou que ele não incluiu um nome de propósito, uma vez que esse desenvolvedor havia recentemente deixado o projeto. Depois de explicado que a presença desse desenvolvedor era importante para a pesquisa, o líder refez o seu *ranking*, incluindo

o desenvolvedor que estava faltando. O líder do projeto P5 informou que ele simplesmente esqueceu dois nomes ao preencher o questionário. Considerou-se o esquecimento como uma explicação razoável, uma vez que de um projeto com 18 desenvolvedores, somente dois não foram mencionados. Da mesma forma que o líder anterior, esse líder também atualizou seu *ranking*, incluindo os dois desenvolvedores esquecidos.

Tabela 43. Lista dos projetos fornecidos pelas organizações.

Projs.	Org.	Quantidade de Desenvolvedores	Quantidade de *Commits*	Quantidade de Arquivos	Tempo (meses)
P1	2	7	7011	Java 1788 (100%)	73
P2	2	5	780	Java 512 (100%)	6
P3	2	4	777	PHP 241 (98.8%)	76
P4	2	4	196	Java 512 (95%)	11
P5	3	18	7894	Java 964 (100%)	8
P6	3	10	4052	Java 1494 (99.9%)	6
P7	3	8	7127	Java 1463 (99.1%)	12
P8	3	7	4474	Java 512 (94.6%)	11
P9	3	5	2308	Java 681 (99.8%)	7

Conforme é possível ver na Tabela 43, quase todos os projetos de *software* investigados usaram *Java* como a principal linguagem de programação, exceto o projeto P3 que utilizou *PHP*. Apesar dessa diferença, o projeto foi analisado em conjunto com os outros projetos coletados. A linguagem de programação principal de cada projeto é usada em mais de 94% de todos os arquivos de código fonte existentes em seus respectivos projetos. Os arquivos de código fonte existentes nesses projetos com uma linguagem diferente da principal linguagem do projeto eram arquivos de *shell script*, utilizados para compilar ou testar o projeto, além de bibliotecas *javascript* não desenvolvidas pela equipe do projeto. Para esses casos, esses arquivos também foram excluídos do cálculo das métricas de produtividade.

Também é possível verificar que sete de nove projetos tiveram um tempo de desenvolvimento variando de 6 a 12 meses. Os outros dois projetos restantes estavam com mais de 70 meses de desenvolvimento, de acordo com os dados dos repositórios. Esses projetos já estavam, na verdade, em manutenção evolutiva, diferente dos outros projetos que estavam no estágio de desenvolvimento e que ainda não haviam sido implantados em produção.

Os projetos da organização 3 (P5 a P9) apresentaram maior quantidade de *commits* por mês, totalizando 589*,* quando comparados aos projetos da organização 2 (P1 a P4), totalizando 64. A organização 3 adotava o método SCRUM de desenvolvimento em seus projetos, realizando *Sprints* semanais, enquanto que a Organização 2 adotava *Sprints* de duas ou três semanas. Além disso, o número de desenvolvedores por projeto na organização 2 era bem menor do que os da organização 3. A combinação dessas diferenças pode explicar a diferença no número de *commits* observados nos repositórios dos projetos coletados.

7.3.2 Qual a correlação entre os rankings de produtividade obtidos pelas métricas de produtividade e pelas percepções do líderes do projeto?

Para responder a essa questão, foram avaliadas as percepções dos líderes de projeto sobre a produtividade do desenvolvedor utilizando os dados das duas organizações. A Figura 23 apresenta todos os resultados da correlação de Kendall tau dos projetos analisados, através da correlação entre os *rankings* dos líderes de projeto com os *rankings* de cada métrica de produtividade. Com base na proposta de intepretação sugerida por Cohen (1988), é possível observar que, ao considerar a mediana das correlações, a maioria das métricas baseadas em código têm uma forte correlação com a percepção dos líderes de projeto. Por outro lado, apenas uma métrica baseada em *commit* (*Commit/Time*) apresentou medianas da correlação moderada e forte, pois as demais métricas baseadas em *commit* apresentaram medianas consideradas fracas.

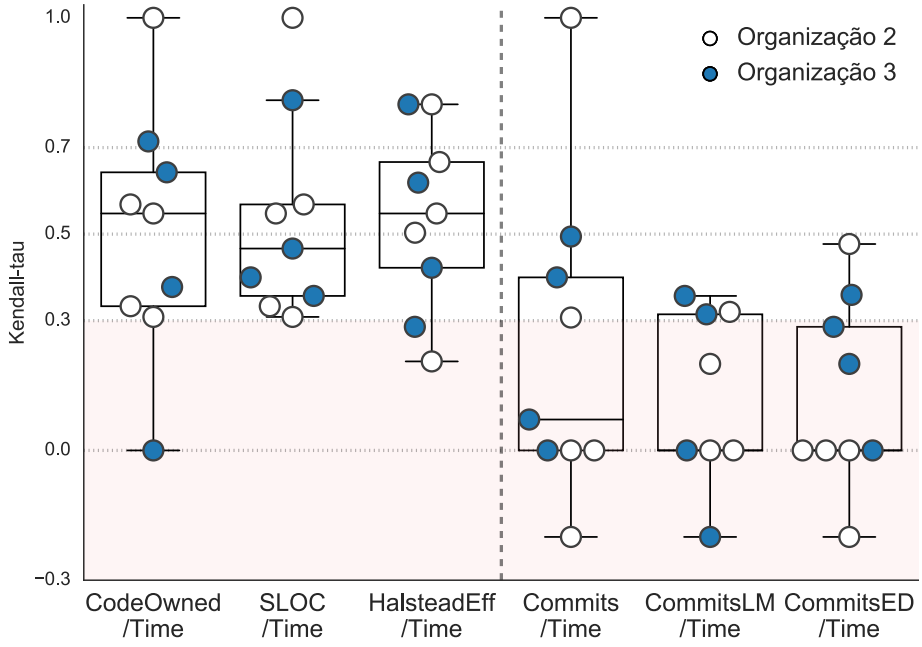

Figura 23. Resultados da correlação Kendall tau para as duas organizações.

A Tabela 44 apresenta um *ranking* das métricas de produtividade do desenvolvedor baseada nos resultados das correlações. A primeira coluna apresenta o *rank* de classificação para a métrica listada na segunda coluna. Da terceira à quinta coluna são apresentadas a mediana das correlações dos projetos para cada métrica, considerando os valores de alfa em 0,05, 0,10 e todas as correlações. O número de projetos abrangidos por cada métrica aparece entre parênteses. O ranking da métrica é ordenado pela mediana do nível alfa mais rígido ($\alpha = 0{,}05$).

Os resultados sugerem que as métricas baseadas em código melhor capturam as percepções dos líderes das equipes sobre a produtividade do desenvolvedor quando comparadas com as métricas baseadas em *commit*. Em seis de nove projetos, havia pelo menos uma métrica baseada em código que foi maior do que todas as métricas baseadas em *commits*. Além disso, mesmo nos três projetos onde as métricas baseadas em *commit* tiveram uma correlação maior, as métricas baseadas em código apresentaram uma correlação forte. Ademais, com base na classificação das métricas da Tabela 44, é possível confirmar a observação de que as métricas baseadas em código tiveram correlações maiores e em mais projetos quando consideramos as correlações estatisticamente significativas.

Tabela 44. Correlação Kendall tau obtidas para as duas organizações.

Rank	Métrica	Mediana (α = .05)	Mediana (α = .10)	Mediana (todos)
1	*SLOC/Time*	**.81** (3/9)	**.69** (4/9)	**.55**
2	*HalsteadEff/Time*	**.71** (4/9)	**.57** (6/9)	**.62**
3	*CodeOwned/Time*	**.68** (4/9)	**.64** (4/9)	**.64**
4	*Commits/Time*	**.51** (3/9)	**.51** (5/9)	**.40**
5	*CommitsLM/Time*	**.48** (2/9)	**.32** (3/9)	.20
6	*CommitsED/Time*	.00 (1/9)	**.48** (3/9)	.20

Embora não exista um padrão consistente nos resultados de todos os projetos, as métricas baseadas em código, que avaliam a produtividade combinando o montante de código fonte produzido com a propriedade do código, melhor se aproximaram das percepções dos líderes de projetos na maioria dos projetos.

7.3.3 Qual o impacto do contexto da organização na percepção dos líderes de projeto sobre a produtividade de seus desenvolvedores?

Para responder a essa subquestão, foram avaliadas se as métricas de produtividade do desenvolvedor apresentaram variação da correlação com as percepções dos líderes de projeto, dependendo da organização em análise.

7.3.3.1 *Organização 2*

Entre os quatro projetos dessa organização, os projetos P2 e P4 eram novos desenvolvimentos de *software*, enquanto os projetos P1 e P3 eram projetos em manutenção evolutiva. Além disso, os projetos P1, P2 e P4 adotam a linguagem *Java*, enquanto o projeto P3 adota *PHP*. Apesar disso, nenhum padrão observado nas correlações obtidas para a Organização 2 pode ser relacionado com essas diferenças. Um total de três líderes de projeto e 20 desenvolvedores estavam envolvidos nesses projetos de *software*. A Figura 24 mostra todas as correlações obtidas entre os *rankings* fornecidos pelos líderes do projeto e os rankings obtidos a partir das métricas de produtividade do desenvolvedor calculadas.

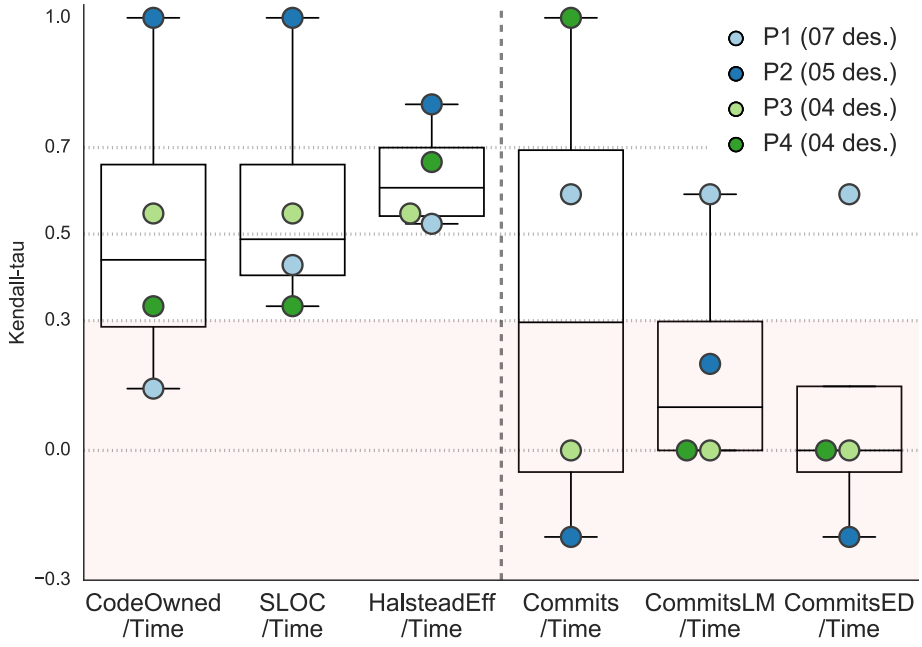

Figura 24. Resultados da correlação Kendall tau para a Organização 2.

Em geral, as métricas baseadas em código se aproximaram mais das percepções dos líderes do que as métricas baseadas em *commits*, de acordo com as correlações obtidas. Isso pode ser observado pelas medianas dos *box-plots* apresentados na Figura 24. Apesar da métrica *Commits/Time*, diferente das outras métricas baseadas em *commit*, ter obtido correlações fortes ($\tau \geq .5$) para dois projetos (P1 e P4), o resultado para os outros projetos (P2 e P3) apresentaram correlações zero e negativas ($\tau \leq 0$), respectivamente. Todas as métricas baseadas em arquivos apresentaram correlações fortes ($\tau \geq .5$) com os *ranking* fornecidos pelos líderes, exceto pela correlação do projeto P1 com a métrica *CodeOwned/Time*, contrastando com a maioria das correlações obtidas com as métricas baseadas em *commits* ($\tau \leq .3$).

A Tabela 45 mostra todos os valores das correlações obtidas juntamente com sua significância estatística correspondente. A significância estatística foi avaliada considerando dois níveis diferentes de alfa (.05 e .10). Durante a análise, foi observado que várias correlações estavam próximas a obter a significância estatística, mas sem atingir o valor de alfa comumente adotado de .05. Isto aconteceu porque os valores críticos necessários para obter uma significância estatística, em correlações Kendall tau, com esse valor alfa dependem muito do tamanho da amostra, especialmente em amostras pequenas. No caso dos projetos da Organização 2, isso é notadamente difícil, pois são projetos que possuem poucos

desenvolvedores (tamanho da amostra). Para exemplificar essa situação, com um tamanho de amostra de apenas quatro desenvolvedores, como nos projetos P3 e P4, é necessário alcançar uma correlação perfeita ($\tau = 1.00$) ou uma correlação inversamente perfeita ($\tau = -1.00$) para obter-se um resultado estatisticamente significativo. Assim sendo, foram utilizados dois valores de alfa comumente adotados em pesquisas científicas (.05 e .10) para dar suporte nas conclusões a respeito das correlações calculadas.

Tabela 45. Correlação Kendall tau obtidas para a Organização 2.

Métrica	P1	P2	P3	P4
FilesOwned/Time	.14	**1.00****	.55	.33
SLOC/Time	.43	**1.00****	.55	.33
HalsteadEff/Time	**.52***	**1.00****	.55	.67
Commits/Time	**.59***	-.20	**.00****	**1.00****
CommitsLM/Time	**.59***	.20	**.00****	.00
CommitsED/Time	**.59***	-.20	**.00****	.00

* $\alpha = .10$, ** $\alpha = .05$

Observando a Tabela 45, nota-se que os projetos P3 e P4 não apresentaram nenhum correlação significativa estatisticamente para a métrica *HalsteadEff/Time*, apesar de possuírem correlações fortes ($\tau \geq .5$) e maiores (P4, $\tau = .67$) que outros projetos (P1, $\tau = .52$). Esse é mais um exemplo da dependência do tamanho da amostra, acontecendo justamente pela menor quantidade de desenvolvedores nos projetos P3 e P4 em relação ao projeto P1. Do mesmo modo, o projeto P1 teve quatro correlações fortes ($\tau \geq .5$) com diversas métricas, mas que não foram consideradas estatisticamente significativas para um valor de alfa de .05.

Considerando somente as correlações com significância estatística, as métricas baseadas em arquivos foram melhores do que as métricas baseadas em *commits*, incluindo um projeto com uma perfeita correlação ($\tau = 1.00$) com as percepções do líder (*l6*). Porém, as métricas baseadas em arquivo somente cobriram metade dos projetos, quando consideramos um alfa de .10; enquanto que as métricas baseadas em *commits* cobriram 3/4 dos projetos com correlações estatisticamente significativas para o mesmo alfa. No entanto, desses 3/4, uma das correlações estatísticas significativas (projeto P3) foi zero, significando, na

realidade, a ausência de qualquer correlação do *ranking* das métricas com percepção do líder (*l6*).

Por fim, a Tabela 46 apresenta uma classificação sumarizada das métricas de produtividade para a Organização 2. Esta tabela apresenta a mediana das correlações dos todos projetos para cada métrica, considerando valores de alfa em .05 e .10 e considerando todas as correlações. O número de projetos abrangidos com significância estatística aparece entre parênteses. As classificações das métricas foram ordenadas pela maior mediana do valor alfa mais rígido ($\alpha = .05$).

Tabela 46. Classificação das métricas de produtividade para a Organização 2.

Rank	Métrica	Mediana ($\alpha = .05$)	Mediana ($\alpha = .10$)	Mediana (todos)
1	*SLOC/Time*	1.00 (1/4)	1.00 (1/4)	.49
=	*FilesOwned/Time*	1.00 (1/4)	1.00 (1/4)	.44
3	*HalsteadEff/Time*	.80 (1/4)	.66 (2/4)	.53
4	*Commits/Time*	.50 (2/4)	.59 (3/4)	.29
5	*CommitsLM/Time*	0.00 (1/4)	.30 (2/4)	.00
=	*CommitsED/Time*	0.00 (1/4)	.30 (2/4)	.10

Conforme é possível notar (Tabela 46), as medianas das métricas baseadas em arquivos foram maiores em todos os casos e com todos os projetos. Apesar de somente ter um resultado com correlação estatisticamente significativa, as métricas *SLOC/Time* e *FilesOwned/Time* foram as que melhor corresponderam às percepções dos líderes de projeto da Organização 2, com leve superioridade para *SLOC/Time* pela mediana considerando todas as correlações obtidas (.49). Além disso, os valores das correlações das métricas foram todos moderados ($\tau \geq .3$) ou fortes ($\tau \geq .5$), enquanto que as métricas baseadas em *commits* apresentaram em sua maioria correlações fracas ($\tau \leq .3$) e até negativas em alguns casos (P2). Esses resultados apontam, que para a Organização 2, as correlações entre as métricas baseadas em arquivos e as percepções dos líderes são superiores e mais fortes do que com as métricas baseadas em *commits*.

7.3.3.2 *Organização 3*

Cinco projetos da Organização 3 foram avaliados (P1, P2, P3, P4 e P5). Diferente dos projetos da Organização 2, todos esses projetos apresentaram características bem similares: novos desenvolvimentos de *software* utilizando *Java*

como principal linguagem de programação com tempo de desenvolvimento menor ou igual a 12 meses. A Figura 25 mostra todas as correlações obtidas entre os *rankings* fornecidos pelos líderes do projeto e os rankings obtidos a partir das métricas de produtividade do desenvolvedor calculadas.

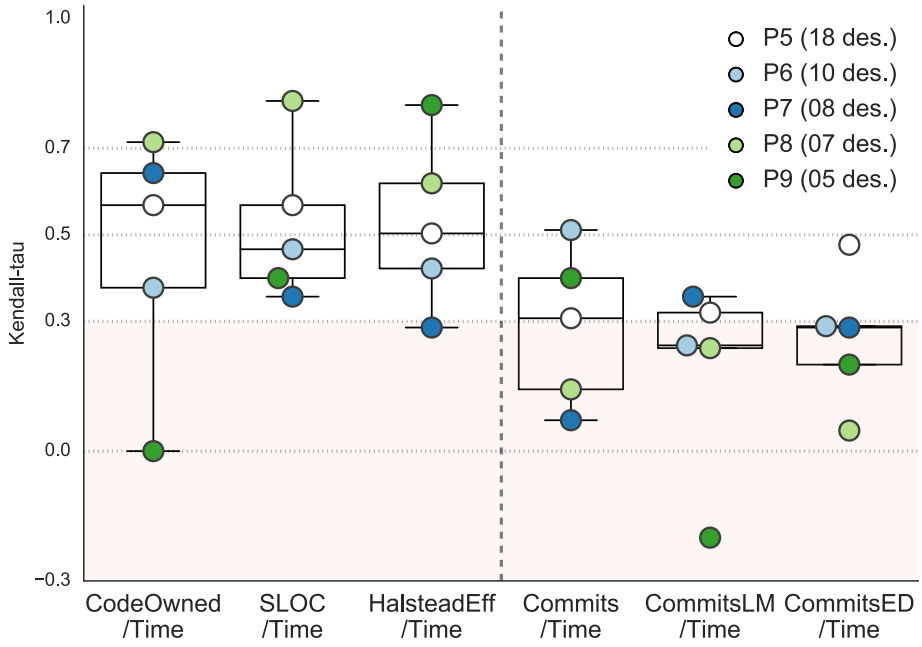

Figura 25. Resultados da correlação Kendall tau para a Organização 3.

Assim como na Organização 2, também na Organização 1 é possível observar (Figura 25) que quase todas as correlações para métricas baseadas em arquivo foram maiores que as correlações obtidas com as métricas baseadas em *commits*. A principal exceção foi com o projeto P6 para a métrica *Commits/Time*, que foi maior do que todas as métricas baseadas em arquivos. Além disso, sem considerar o projeto P9 para a métrica *FilesOwned/Time*, todas as correlações das métricas baseadas em arquivos foram moderadas ($\tau \geq .3$) ou fortes ($\tau \geq .5$). As métricas baseadas em *commits*, em contraste, apresentaram a maioria das correlações obtidas como fracas ($\tau \leq .3$), incluindo uma correlação negativa para o projeto P9.

A Tabela 47 mostra todos os valores das correlações Kendall tau obtidas juntamente com sua significância estatística correspondente para a Organização 3. Considerando $\alpha = .10$, todas as correlações para o projeto P1 foram estatisticamente significativas, enquanto que para os outros projetos, as métricas baseadas em arquivo cobriram todos os projetos com pelo menos uma correlação estatisticamente significativa, com resultados em sua maioria fortes ($\tau \geq .5$). Por outro lado, as

métricas baseadas em *commits* não alcançaram correlações estatisticamente significativas para os projetos P7, P8 e P9. Além deste resultado, a métrica *CommitsLM/Time* obteve uma correlação negativa com o projeto P9, indicando um *ranking* invertido em comparação com o *ranking* informado pelo líder desse projeto (*l12*). Considerando α = .05, a principal diferença é que as métricas baseadas em arquivo não obtiveram uma única correlação significativa com o projeto P6.

Tabela 47. Correlação Kendall tau obtidas para a Organização 3.

Métrica	P5	P6	P7	P8	P9
FilesOwned/Time	**.57****	.38	**.64****	**.71****	.00
SLOC/Time	**.57****	**.47***	.36	**.81****	.40
HalsteadEff/Time	**.50****	**.42***	.29	**.62****	**.80****
Commits/Time	**.31***	**.51****	.07	.14	.40
CommitsLM/Time	**.32***	.24	.36	.24	-.20
CommitsED/Time	**.48****	.29	.29	.05	.20

* α = .10, ** α = .05

A Tabela 48 apresenta uma classificação sumarizada das métricas de produtividade para a Organização 3. Como na Organização 2, as métricas baseadas em arquivos tiveram um resultado superior na correlação com a percepção dos líderes de projeto. Ademais, todas as métricas baseadas em arquivos cobriram mais projetos e com todas correlações consideradas fortes (τ ≥ .5), enquanto que as métricas baseadas em *commits* apresentaram em sua maioria correlações fracas (τ ≤ .3), exceção à métrica *Commits/Time* (τ ≥ .5, com α = .05).

Tabela 48. Classificação das métricas de produtividade para a Organização 3.

Rank	Métrica	Mediana (α = .05)	Mediana (α = .10)	Mediana (todos)
1	*SLOC/Time*	.69 (2/5)	.57 (3/5)	.47
2	*FilesOwned/Time*	.64 (3/5)	.64 (3/5)	.57
3	*HalsteadEff/Time*	.62 (3/5)	.56 (4/5)	.50
4	*Commits/Time*	.51 (1/5)	.41 (2/5)	.31
5	*CommitsLM/Time*	.48 (1/5)	.48 (1/5)	.29
6	*CommitsED/Time*	– (0/5)	.32 (1/5)	.24

Novamente a métrica *SLOC/Time* obteve a melhor correlação comparada à percepção do líder do projeto, contudo cobrindo um projeto a menos e com

resultados menores quando considerado $\alpha = .10$ e todas as correlações. As métricas baseadas em *commits* novamente apresentaram um resultado com significância estatística em um número menor de projetos e com correlações mais fracas ($\tau \leq .3$). Os resultados dessa organização reforçam os resultados da organização anterior, mostrando que as métricas baseadas em arquivos melhor correspondem às percepções dos líderes de projeto quanto à produtividade de seus desenvolvedores do que as métricas baseadas na frequência de *commits*.

É possível concluir que, nas duas organizações analisadas, as métricas baseadas em código obtiveram um resultado geral melhor do que as métricas baseadas em *commits*. No entanto, os resultados também mostraram que a correlação com as percepções dos líderes de projetos com as métricas apresenta variação de organização para organização e até de projeto para projeto.

7.4 DISCUSSÃO DOS RESULTADOS

7.4.1 As métricas de produtividade do desenvolvedor de software correspondem às percepções das organizações?

Este estudo teve como objetivo analisar como as métricas de produtividade dos desenvolvedores do *software* se relacionam com as percepções dos líderes do projeto. Foram selecionadas métricas de produtividade do desenvolvedor frequentemente utilizadas pelos pesquisadores da produtividade na Engenharia de *Software*. Essas métricas foram ainda agrupadas em métricas baseadas em arquivos de código fonte e métricas baseada na frequência de *commits*. Um total de nove projetos, com oito líderes de projetos correspondentes de duas organizações, foram analisados e tiveram seus repositórios processados para calcular a produtividade dos desenvolvedores membros. A partir dos *rankings* gerados pelas métricas de produtividade e dos *rankings* informados pelos líderes de projeto correspondentes, um estudo correlacional foi realizado.

Foi mostrado que as métricas baseadas em arquivos de código fonte estão mais fortemente ($\tau \geq .5$) correlacionadas, segundo o guia de interpretação da intensidade da correlação proposto por Cohen (1988), em mais projetos com os *rankings* fornecidos pelos líderes do projeto do que as métricas baseadas em frequência de *commits*. A métrica *SLOC/Time*, apesar de ter recebido muitas críticas na literatura [Barb *et al.* 2014; Rahman e Devanbu 2013; Sheetz *et al.* 2009], obteve

as maiores correlações estatisticamente significativas ($\alpha = .05$). As outras métricas baseadas em arquivos (*FilesOwned/Time* e *HalsteadEff/Time*) a seguiram com resultados muito próximos. As métricas baseadas em *commits* apresentaram correlações consideravelmente fracas ($\tau \leq .3$), em menos projetos e, em alguns casos, apresentando *rankings* invertidos em comparação com os *rankings* das percepções dos líderes do projeto.

Um fator que pode ter influenciado nesses resultados pode ter sido a escolha de se utilizar o conceito de propriedade (*ownership*) conforme proposto por Bird *et al.* (2011). Para avaliar o impacto desta escolha, introduzimos entre as métricas baseadas em arquivos a métrica *FilesOwned/Time* neste estudo. Os resultados dessa métrica em particular indicam que a propriedade do código tem forte relação com a percepção dos líderes de projeto, pois que essa estratégia foi utilizada em todas as métricas baseadas em arquivos. Uma explicação possível pode estar na alocação dos desenvolvedores para as tarefas do desenvolvimento, onde o líder escolhe seus desenvolvedores mais produtivos para as tarefas mais complexas que, por sua vez, comumente envolvem mais arquivos, com mais linhas de código e com código mais complexo. Dessa forma, os desenvolvedores vão se apropriando de códigos do projeto segundo a alocação definida pelo líder e, possivelmente, essa alocação é refletida nas métricas baseadas em arquivos deste estudo. Nesta perspectiva, a quantidade de linhas de código (*SLOC*) e o esforço de *Halstead* atuam como ponderações em cima dos códigos apropriados pelos líderes de projeto. Essas ponderações podem ser interpretadas como a complexidade do código [Booch 2008]. Um hipótese a ser investigada no futuro é que as fortes correlações associadas às métricas baseadas em arquivos de código fonte são em decorrência da alocação realizada pelos líderes de projeto. Isto porque os líderes alocam seus desenvolvedores mais produtivos para trabalhar em mais arquivos de códigos fonte e/ou com mais complexidade do que os demais desenvolvedores.

As métricas baseadas em código apresentaram principalmente correlações fortes ($\tau \geq .5$) ou moderadas ($\tau \geq .3$) e com baixa variação. Em uma configuração alternativa de nosso estudo, foram calculadas as mesmas métricas de produtividade para determinados períodos de tempo menores que o tempo total do projeto. Não foram observadas diferenças significativas com relação às métricas baseadas em arquivos, a não ser em casos específicos em que o *ranking* aparece com menos

desenvolvedores do que os informados pelo líder, dado que alguns desenvolvedores entraram tardiamente no projeto. Embora seja necessária mais pesquisa, esses resultados específicos sugerem que as características da implementação das métricas baseadas em arquivos são estáveis durante o desenvolvimento. Já com relação às métricas baseadas em frequência de *commits,* estas tiveram mais variação no *ranking*, especialmente nos casos em que os desenvolvedores eram temporariamente alocados a outros projetos ou tiravam férias.

As métricas baseadas em *commits* foram fracamente correlacionadas ($\tau \leq .3$) e tiveram mais variação com relação às percepções dos líderes de projeto do que as métricas baseadas em arquivos de código fonte. As métricas baseadas em *commits* são muito dependentes da frequência dos *commits* realizados pelos desenvolvedores. Como resultado, essas métricas acabam penalizando aqueles desenvolvedores que realizam *commits* em um ritmo mais lento; que realizem tarefas mais complexas fora do repositório principal; que auxiliam outros desenvolvedores a realizarem suas tarefas (*mentoring* ou *pair programming)*, enfim, que realizem trabalhos cujo resultado não será registrado no repositório ou que será registrado por outro desenvolvedor, já que somente um autor poderá registrar o *commit*. Dessa forma, as métricas baseadas na frequência de *commits* não consideram diversos casos de trabalho dos desenvolvedores, ainda ignorando por completo o histórico das tarefas realizadas pelo desenvolvedor durante o projeto. Assim, desenvolvedores que realizam *commits* com frequência são considerados mais produtivos por essas métricas, mesmo que estejam trabalhando em tarefas de codificação bem menos complexas. Esses cenários podem explicar as correlações fracas ($\tau \leq .3$) obtidas com essas métricas, incluindo os casos das correlações negativas, com as percepções dos líderes, já que os mesmos levaram em conta todo o histórico do projeto para dedução dos seus *rankings* de produtividade.

Finalmente, considerando todos os resultados aqui apresentados, é possível responder à questão de pesquisa: *as métricas de produtividade do desenvolvedor correspondem às percepções das organizações de software, sendo mais preciso afirmar que as métricas de produtividade baseadas em arquivos de código correspondem mais às percepções dos líderes do projeto sobre a produtividade de seus desenvolvedores*. Esses resultados sugerem que as métricas baseadas em código, usando a propriedade de código (*Code Ownership*), podem complementar a percepção de

produtividade das organizações de *software*, indicando possíveis desenvolvedores produtivos não percebidos pelos líderes de projeto e confirmando os desenvolvedores mais produtivos do projeto.

7.5 AMEAÇAS À VALIDADE

A principal ameaça à validade das conclusões do estudo é a generalização de dos resultados obtidos. A quantidade de projetos investigados não pode ser considerada como representativa, porém buscou-se mitigar essa ameaça pela obtenção e coleta do maior número de projetos das organizações pesquisadas, porém a decisão final estava com as organizações. Infelizmente, nem todos os projetos disponibilizados pelas organizações puderam ser utilizados para contribuir para uma conclusão estatística mais confiável. Além disso, utilizamos diversas métricas de produtividade do desenvolvedor visando à triangulação dos resultados, fortalecendo os achados do estudo.

Outra ameaça para nossas descobertas é quanto à observação em apenas duas organizações, pois que elas representam uma limitada diversidade de contextos. No entanto, essas duas organizações pesquisadas representam empresas típicas de desenvolvimento de software no Brasil e em outros lugares. O perfil e contexto dessas organizações está descrito em detalhes no Capítulo 1, Subseção 1.5.2.

Ainda outra ameaça à validade de construção é que as percepções dos entrevistados podem ser tendenciosas em relação a suas próprias crenças sobre seus desenvolvedores. Dessa forma, essas crenças podem causar algumas distorções ao interpretar a realidade. Para reduzir essa ameaça, os líderes de projeto foram novamente contatados para os casos de maior discrepâncias entre os dados existentes nos repositórios do projeto e as informações preenchidas no questionário aplicado.

7.6 CONSIDERAÇÕES FINAIS

Este capítulo descreveu um estudo correlacional entre as métricas de produtividade do desenvolvedor e as percepções de organizações de *software*, segundo a percepção dos seus líderes de projeto. O estudo foi conduzido envolvendo oito líderes de projeto em nove projetos de *software*, com um total de 68

desenvolvedores das duas organizações pesquisadas. Os resultados mostram que existe uma relação entre as métricas de produtividade do desenvolvedor e as percepções dos líderes de projeto de *software*. Entre os dois grupos de métricas investigadas, baseadas em arquivos de código fonte e baseadas em frequência de *commits*, as métricas baseadas em arquivos de código fonte apresentaram correlações mais fortes com a maioria dos projetos investigados e com menor variação. Além disso, esses resultados sugerem que as métricas baseadas em código podem complementar a percepção de produtividade dos desenvolvedores por parte das organizações de *software*.

Capítulo 8 – FATORES DE INFLUÊNCIA DOS DESENVOLVEDORES PRODUTIVOS DE *SOFTWARE*

Este capítulo apresenta um estudo qualitativo para identificar os fatores de influência da produtividade presentes, segundo a percepção dos líderes de projetos, nos desenvolvedores considerados produtivos, e de acordo os rankings fornecidos pelos líderes e gerados pelas métricas de produtividade aplicadas.

8.1 INTRODUÇÃO

Este capítulo apresenta um estudo para identificar os fatores de influência da produtividade presente, segundo a percepção dos líderes de projetos de *software*, nos desenvolvedores considerados mais produtivos, de acordo com os *rankings* fornecidos pelos líderes e com os *rankings* gerados pelas métricas de produtividade (Estudo *E5*, Capítulo 7). Após a identificação dos fatores de influência da produtividade dos desenvolvedores junto aos líderes de projeto, realizadas no Estudo *E4* (Capítulo 6), objetivou-se neste estudo investigar a existência desses fatores nos desenvolvedores mais produtivos das organizações.

Além disso, também teve como objetivo avaliar se os fatores identificados nos desenvolvedores mais produtivos têm suporte na literatura (Capítulo 4) e/ou na teoria fundamentada (Capítulo 6, Seção **Error! Reference source not found.**). Por fim, os resultados identificados neste estudo levaram a uma evolução da teoria fundamentada preliminar apresentada no Capítulo 6.

8.2 PLANO DE ESTUDO

O objetivo principal desse estudo foi buscar nos fatores de influência da produtividade dos desenvolvedores produtivos da organização, evidências concretas dos fatores de influência identificados na teoria fundamentada apresentada no Capítulo 6. Para isso, os líderes dos projetos avaliados no capítulo anterior foram entrevistados novamente para identificar os fatores de influência na produtividade dos desenvolvedores mais produtivos de seus respectivos projetos. A seleção dos desenvolvedores mais produtivos continha os desenvolvedores no topo do *ranking*

de cada líder, assim como também os desenvolvedores no topo do *ranking* das métricas de produtividade que não estavam presentes no topo do *ranking* dos líderes de projeto.

As duas questões principais (*QP1* e *QP2*) dessa investigação e as subquestões derivadas estão apresentadas na Tabela 49. A primeira questão principal (*QP1*) foi desdobrada em duas subquestões, que visaram identificar os fatores de influência nos desenvolvedores produtivos da organização (*SQ1*), conforme a percepção de seus líderes de projeto e conforme as métricas de produtividade, além de também identificar quais fatores de influência os diferenciavam (*SQ2*). A segunda questão principal (*QP2)* também foi desdobrada em duas subquestões, que visaram verificar quais dos fatores de influência identificados tinham suporte na literatura científica (*SQ3*) e/ou na teoria fundamentada (*SQ4*) identificada nesta pesquisa.

Tabela 49. Questões de pesquisa para os líderes de projeto.

QP1	Quais são os fatores de influência na produtividade dos desenvolvedores produtivos da organização?
SQ1	Quais são os fatores de influência na produtividade nos desenvolvedores produtivos segundo os *rankings* de produtividade?
SQ2	Quais são os fatores de influência na produtividade que diferenciam os desenvolvedores mais produtivos?
QP2	Quais fatores de influência na produtividade identificados nos desenvolvedores produtivos têm suporte em outros estudos?
SQ4	Quais dos fatores de influência identificados nos desenvolvedores produtivos têm suporte na teoria fundamentada preliminar desta pesquisa?
SQ5	Quais dos fatores de influência identificados nos desenvolvedores produtivos têm suporte na revisão terciária desta pesquisa?

8.2.1 Procedimentos de coleta de dados

A coleta de dados foi realizada utilizando entrevistas semiestruturadas. Para este estudo, somente foram entrevistados os líderes das organizações cujos projetos de *software* foram investigados no estudo sobre as métricas de produtividade do desenvolvedor de *software* (Capítulo 7), permitindo a triangulação de dados para seleção dos desenvolvedores produtivos pelas organizações. Dessa forma, a Organização 1 ficou de fora deste estudo.

Um total de oito líderes de projeto de *software* foram re-entrevistados neste estudo, sendo três na Organização 2 (Org. 2) e cinco na Organização 3 (Org. 3). A

Tabela 50 apresenta os líderes de projeto de *software* participantes deste estudo, seus respectivos projetos e a quantidade de desenvolvedores produtivos sobre os quais indicaram seus fatores de influência na produtividade.

Tabela 50. Líderes de projeto de *software* entrevistados.

Id	Org.	Educação	Projetos
l5	2	Especialização em Gestão de Projetos	P1
l6	2	Especialização em Engenharia de SW	P2, P3
l7	2	Especialização em Engenharia de SW	P4
l8	3	Bacharel em Ciência da Computação	P5
l9	3	Especialização em Gestão de Projetos	P6
l10	3	Bacharel em Análise de Sistemas	P7
l11	3	Especialização em Engenharia de SW	P8
l12	3	Bacharel em Análise de Sistemas	P9

Os líderes de projeto selecionados indicaram os fatores de influência na produtividade de 37 desenvolvedores selecionados como mais produtivos nas organizações. A partir do *ranking* dos líderes de projeto, foram considerados como desenvolvedores mais produtivos aqueles que estavam posicionados nas três primeiras posições do *ranking*. Para os casos de projetos com somente quatro desenvolvedores, foram considerados somente os dois primeiros indicados pelo líder de projeto. Também foram considerados como desenvolvedores produtivos, aqueles desenvolvedores que mesmo não estando nas primeiras posições de acordo com os líderes, estavam nas primeiras posições de acordo com os *rankings* das métricas de produtividade baseadas em código.

A Tabela 51 e Tabela 52 apresentam os 35 desenvolvedores considerados mais produtivos, de um total de 64 desenvolvedores a partir da triangulação dos *rankings* dos líderes e das métricas de produtividade. Nestas tabelas, além do projeto e do líder respectivo, também é apresentado o *ranking* de cada desenvolvedor conforme o líder e conforme as métricas, considerando as três métricas baseadas em código e também as três métricas baseadas em *commits*. Ademais, é indicado (utilizando um asterisco) qual o *ranking* foi utilizado para selecionar o desenvolvedor como produtivo, esclarecendo se o desenvolvedor foi

selecionado por causa do *ranking* do líder ou a partir do *ranking* das métricas baseadas em código.

Tabela 51. Desenvolvedores produtivos da Organização 2.

Id	Projeto	Desenvolvedor	*Ranking* pelo Líder	*Rankings* Código
l5	P1 (4)	d01	1*	2, 2, 1
		d02	2*	3, 3, 3
		d03	3	1*, 1*, 2
l6	P2 (5)	d04	1*	1, 1, 1
		d05	2*	2, 2, 3
		d06	3*	3, 3, 2
l6	P3 (4)	d07	1*	1, 1, 1
		d08	2*	4, 4, 4
		d09	3	2*, 2*, 2*
l7	P4 (7)	d10	1*	1, 1, 1
		d11	2*	5, 5, 5
		d12	3*	3, 3, 3
		d13	4	2*, 4, 2*
		d14	7	4, 2*, 4

A Tabela 51 apresenta os 14 desenvolvedores selecionados na Organização 2. O projeto P2 não acrescentou mais nenhum desenvolvedor além dos três primeiros indicados pelo líder de projeto. Os projetos P1 e P3 incluíram o terceiro desenvolvedor mais produtivo segundo o líder de projeto. O projeto P4 incluiu mais dois desenvolvedores na seleção, já que estavam na segunda posição do *ranking* segundo as métricas de produtividade baseadas em arquivo. Como os projetos dessas organizações possuíam poucos desenvolvedores, foram investigados junto aos líderes de projeto a grande maioria dos desenvolvedores (14/20).

A Tabela 52 apresenta os 21 desenvolvedores selecionados como mais produtivos na Organização 3. Os projetos P6 e P7 incluíram mais dois desenvolvedores pelas métricas de produtividade, enquanto os projetos P8 e P9 somente incluíram mais um. O projeto P7 não incluiu o primeiro desenvolvedor mais produtivo segundo o líder, pois era o próprio líder do projeto. Nessa organização, foram investigados junto aos líderes de projeto somente 43% dos

desenvolvedores, uma vez que a quantidade de desenvolvedores envolvidos nos projetos dessa organização era muito maior (48 desenvolvedores).

Tabela 52. Desenvolvedores produtivos da Organização 3.

Id	Projeto	Desenvolvedor	*Ranking* pelo Líder	*Rankings* Código
l8	P5 (18)	d15	1*	2, 2, 2
		d16	2*	4, 4, 4
		d17	3*	3, 3, 1
		d18	8	1*, 1*, 3
l9	P6 (10)	d19	1*	4, 4, 6
		d20	2*	2, 2, 3
		d21	3*	7, 6, 4
		d22	4	3, 3, 2*
		d23	8	1*, 1*, 1*
l10	P7 (8)	d24	2*	2, 4, 4
		d25	3*	1, 2, 2
		d26	4	5, 1*, 1*
		d27	6	4, 3*, 3*
l11	P8 (7)	d28	1*	2, 1, 2
		d29	2*	1, 2, 1
		d30	3*	4, 4, 4
		d31	4	3*, 3*, 3*
l12	P9 (5)	d32	1*	3, 2, 1
		d33	2*	2, 1, 2
		d34	3*	4, 4, 3
		d35	5	1*, 3, 4

Para responder à questão de pesquisa *QP1*, foram adotados procedimentos semelhantes aos estudos qualitativos anteriores, utilizando um roteiro semiestruturado de acordo com as diretrizes propostas por Runeson *et al.* (2012). Para cada líder de projeto, considerando cada desenvolvedor produtivo segundo seu *ranking*, foram questionados (*SQ1*) os fatores de influência da produtividade (Tabela 53). Quando o desenvolvedor em foco não era o desenvolvedor mais produtivo segundo o líder entrevistado, também foi questionado (*SQ2*) quais fatores diferenciavam esse desenvolvedor do desenvolvedor mais produtivo (Top 1). Assim

como nos procedimentos aplicados nos estudos qualitativos anteriores, as entrevistas foram realizadas simultaneamente com o processo de análise, realizando ciclos compostos de entrevistas, transcrições e análise.

Tabela 53. Roteiro semiestruturado utilizado nas entrevistas com os líderes de projeto.

Subquestão	Descrição
SQ1	Quais as características e fatores de influência na produtividade do desenvolvedor produtivo X, segundo seu *ranking*?
	Quais as características e fatores de influência na produtividade do desenvolvedor produtivo X, segundo o *ranking* das métricas?
SQ2	Quais as características e fatores de influência na produtividade do desenvolvedor X o diferenciam do desenvolvedor Top 1 do seu *ranking*?

Para a segunda questão de pesquisa (*SQ2*), utilizamos como referência os fatores relevantes identificados na teoria fundamentada preliminar desta pesquisa (Capítulo 6, Subseção **Error! Reference source not found.**), assim como também os fatores extraídos da literatura a partir do resultado da revisão terciária (Capítulo 4).

8.2.2 Procedimentos de análise de dados

O *software* Atlas.ti[17] foi novamente utilizado para realizar as análise das entrevistas realizadas. Também foram utilizados alguns procedimentos da Teoria Fundamentada (*Grounded Theory – GT*) [Glaser e Strauss 1968]. Dos procedimentos de GT, foram utilizadas a codificação aberta, a codificação axial, mas não a codificação seletiva. Novamente, decidiu-se não eleger uma categoria central, pois a mesma não emergiu naturalmente dos dados. Dessa forma, não se pode afirmar que o método GT foi aplicado por completo, mas que foram utilizados somente alguns procedimentos específicos de GT.

A codificação foi novamente realizada pelo autor desta pesquisa, também responsável pela codificação, pelas redes identificadas nas categorias e pelos *memos* descrevendo as relações identificadas. Outra pesquisadora realizou a revisão da codificação e das redes identificadas. Foram utilizados turnos de entrevistas, transcrições e análise, repetidos até não haver mais participantes disponíveis.

[17] http://atlasti.com

Após a extração dos fatores utilizando os procedimentos de GT, foi realizada uma análise comparativa entre os fatores identificados com os existentes em outros estudos. Essa análise comparativa foi executada utilizando a técnica de análise de características (*feature analysis*), categorizando e tabulando os fatores, facilitando a análise dos resultados.

8.3 RESULTADOS OBTIDOS

8.3.1 SQ1. Quais são os fatores de influência sobre a produtividade dos desenvolvedores produtivos da organização?

> *"Ele conseguia atender com uma capacidade muito rápida" – l9*

Os líderes de projeto indicaram diversos fatores que influenciaram na produtividade dos desenvolvedores considerados mais produtivos dos seus respectivos projetos dentro da organização. Além dos fatores, foram citadas diversas características dos desenvolvedores produtivos que contribuíram para sua produtividade. Os fatores citados foram agrupados em três categorias principais: (i) **comportamento**, (ii) **conhecimento e experiência** e (iii) **contexto organizacional**.

Os **comportamentos** dos desenvolvedores considerados produtivos foram os fatores mais citadas pelos líderes de projeto. O foco, o comprometimento, a proatividade e a motivação foram fatores citados que estão presentes nos desenvolvedores produtivos. Segundo os líderes de projeto, o foco não é perdido quando o desenvolvedor pede ajuda para completar sua tarefa, mas sim quando ele deixa de buscar completar sua tarefa para dedicar-se a uma outra atividade, possivelmente fora do escopo de suas responsabilidades com a organização. Também foi citado que certos desenvolvedores precisam de desafios para se manterem motivado e, por isso, tomam iniciativas para buscar novos conhecimentos técnicos em novas tecnologias de desenvolvimento de *software*.

> *"Ele, do momento que chega, do momento que sai, é focado. Focado no trabalho que ele faz. Se tiver um sistema para fazer em um dia, ele faz em um dia. É alguém que eu recomendaria para outras equipes" – l6*

> *"No caso do foco, é uma questão do cara que está ali com um problema para resolver, e não quando está com o celular na mão no Facebook" – l12*
>
> *"Ele é um cara que se compromete a fazer e entrega naquele prazo" – l7*
>
> *"Ele é bem motivado" – l12*
>
> *"É mais uma característica dele mesmo de buscar e de ser proativo. Talvez a produtividade dele não fosse tão grande se eu desse uma coisa simples, pois ele iria estar fazendo aquilo ali, mas sem a vontade de estar fazendo. Para ele tem que ser alguma tarefa difícil e desafiadora" – l5*

Além desses fatores do comportamento já conhecidos, também foram citadas características individuais que se destacam, segundo as percepções dos líderes, nos desenvolvedores produtivos, tais como a paciência, a disciplina, a resiliência e a competência. Essas características serão consideradas dentro do escopo desse estudo como novos fatores de influência na produtividade do desenvolvedor, pois foram fatores não citados anteriormente nos Estudos *E3* (Capítulo 5) e *E4* (Capítulo 6). A paciência para colaborar com o restante da equipe foi citada como uma característica que contribui, mesmo indiretamente, para a produtividade da equipe. A disciplina é uma característica que também contribui para a produtividade, pois permite que o desenvolvedor direcione seu comportamento para cumprir suas obrigações dentro projeto. A resiliência às pressões existentes no projeto foi outra característica citada que contribui para a produtividade do desenvolvedor, pois, mesmo com a pressão, o desenvolvedor mantém seu nível de produtividade. Por fim, a competência em realizar suas tarefas da melhor maneira, com a melhor solução, é também uma característica marcante nos desenvolvedores produtivos.

> *"Colaborava com a equipe, sempre ajudando, sempre disponível e sempre muito paciente" – l10*
>
> *"Você passa o serviço e ela cumpre. É uma pessoa disciplinada e sempre séria e em algum canto produzindo" – l5*
>
> *"Ele aguenta a pressão e coisas do tipo, mas sem cair a produtividade" – l7*

> *"Você percebe que ele abraça o problema e tenta encontrar a solução, e não qualquer solução, mas com qualidade de código e tudo mais" – l8*

Os líderes de projeto também citaram fatores de **conhecimento e experiência**. Os desenvolvedores produtivos possuíam conhecimento técnico, experiência e conhecimento do domínio do problema. O conhecimento técnico abre alternativas de implementação para do desenvolvedor, permitindo escolher a forma mais simples, facilitando as futuras manutenções do código por ele desenvolvido. A experiência contribui para que o desenvolvedor possa atender as suas tarefas em um menor tempo, dado que o mesmo já tenha realizado essas mesmas tarefas anteriormente. Por fim, o conhecimento do negócio também permite ao desenvolvedor melhor atender as necessidades do cliente, pois pode considerar diversas soluções alternativas, escolhendo, ao final, aquela que seja a solução mais adequada para o cliente.

> *"Ele é um bom técnico e a característica técnica dele é muito boa. O conhecimento técnico é muito bom, desenvolvendo da maneira mais simples possível" – l7*

> *"O destaque é porque ele tem uma experiência maior do que os demais. Nesse sentido, ele tinha uma habilidade muito grande de aprender novas tecnologias e, em relação à produtividade, ele realmente conseguia atender as* 'user stories' *no menor tempo, quando comparado com os pares" – l9*

> *"Ele tinha o conhecimento do domínio do problema. Isso ajudava ele entender um pouco melhor e aí pensava numa solução que ajudava" – l11*

Por último, também citaram fatores relacionados ao **contexto organizacional**. São fatores que estão presentes nas organizações de *software* e que impactaram negativamente na produtividade dos desenvolvedores produtivos: a inconsistência dos requisitos e a troca de contexto. A inconsistência dos requisitos acontece quando ocorrem muitas mudanças de requisitos ou quando os mesmos não estão claramente definidos, causando a desmotivação dos trabalhadores e queda na produtividade. A troca de contexto ocorre quando o desenvolvedor tem que

direcionar sua atenção para outra atividade, deixando de realizar as tarefas previamente acordadas com o líder. Nem sempre essas mudanças dependem do líder, pois podem ser tarefas urgentes de outros projetos ou novas responsabilidades para o desenvolvedor, como por exemplo, assumir um novo cargo na organização.

> *"A gente percebeu que o cliente não sabia o que estava fazendo, por isso começaram a pedir alterações no código. Isso desmotivou todo mundo e a produtividade caiu também" – l8*

> *"Agora o que é ruim pode ser a falta de definição de alguns itens [requisitos] do produto, pois pode gerar uma desmotivação e com isso baixar a produtividade. Tem um caso que eu conheço, demorou para definir algumas coisas e a gente executou baseado na nossa percepção. Nesse caso, a falta de definição com o Top1 teve um impacto significativo, ficou desmotivado" – l6*

> *"Você percebe aqui que o [meu] Top1 foi para terceiro, justamente porque ele assumiu um cargo de supervisão, então fica mais difícil de ele manter [a produtividade]" – l7*

8.3.2 SQ2. Quais são os fatores de influência que diferem o desenvolvedor mais produtivo do projeto em relação aos demais?

> *"Porque ele está em terceiro? O comportamental acaba se destacando" – l11*

Diferente da questão anterior, nesta questão o intuito foi de identificar os fatores que discriminam o desenvolvedor mais produtivo dos demais, segundo a percepção dos líderes de projeto. Os fatores identificados foram agrupados em duas categorias: (i) **comportamento** e (ii) **conhecimento e experiência**.

O **comportamento** do desenvolvedor foi o conjunto de fatores mais citados pelos líderes de projeto. A capacidade de focar na sua atividade e a motivação do desenvolvedor foram os fatores que mais diferenciaram o desenvolvedor considerado mais produtivo em relação aos demais desenvolvedores do projeto. Os desenvolvedores mais produtivos apresentavam, segundo os líderes de projeto, maior foco em suas tarefas do que os demais. A desmotivação também foi citada como um fator de influência que diferenciava os desenvolvedores mais produtivos, pois os demais desenvolvedores quando têm divergências técnicas com o projeto

acabam desmotivados, causando prejuízo na sua produtividade. Em geral, isso não ocorre com os desenvolvedores mais produtivos de cada projeto.

> ***Sobre os desenvolvedores mais produtivos:***
>
> *"O Top 1 tem mais objetividade, mais foco" – l6*
>
> *"Por incrível que pareça, o foco dele parece meio disperso, mas quando ele entra no foco, é para entregar" – l7*
>
> *"Mas em termo de foco, não tem nem como comparar ele com os outros três" – l8*
>
> ***Sobre os demais desenvolvedores:***
>
> *"Ele é um cara assim, como posso dizer... quando ele não concorda com o que está sendo feito, ele se desmotiva muito rápido" – l8*
>
> *"Assim, negativamente ele tem algumas opiniões técnicas que diferiam um pouco da equipe. Então no momento em que ele foi convencido sobre isso, eu observei que houve uma queda da participação dele" – l6*

Além desses fatores do comportamento, dois novos outros fatores dos desenvolvedores mais produtivos que os diferem dos demais desenvolvedores foram citados: a facilidade de comunicação e a autossuficiência técnica. Os desenvolvedores mais produtivos têm uma melhor colaboração com a equipe, facilitada pela sua capacidade de comunicação. Como resultado, colaboram mais efetivamente com a equipe. Os desenvolvedores que não possuem a mesma facilidade de comunicação (por serem, por exemplo, naturalmente mais introspectivos) apresentam dificuldades adicionais ao se isolarem, não solicitar ajuda, etc. Os líderes ainda citaram que mesmo quando têm um comportamento mais introspectivo, os desenvolvedores mais produtivos normalmente contrabalanceiam sua desvantagem com grande autossuficiência técnica. Em outras palavras, eles não têm uma dependência técnica recorrente de outros desenvolvedores, uma vez que usualmente, segundo os líderes de projeto, possuem todo o conhecimento técnico necessário para realizar as suas tarefas.

Sobre os desenvolvedores mais produtivos:

"Ele (Top 1) é um cara que estuda bastante. Ele ensina mais [os outros]" – l8

"Provavelmente a falta de comunicação é o que coloca esse desenvolvedor aqui embaixo, porque tecnicamente é melhor do que outros" – l10

"Se esse Top1 for introspectivo talvez para ele não impacte, porque ele tem bastante conhecimento [técnico] e ele vai conseguir desenrolar tudo. Mas talvez para um outro que não tem um conhecimento técnico tão bom, talvez ele não consiga produzir muito, porque ele vai acabar ficando ali na dele" – l12

Sobre os demais desenvolvedores:

"Isso porque ele segurava algumas tarefas com ele ou tinha algumas dificuldades porque não tinha tirava dúvidas com outros. Então ele bloqueava. Mas ele é muito bom tecnicamente, é muito bom mesmo, mas nesse caso a introspecção é o que coloca ele lá embaixo" – l10

"Em termos de produtividade, esse desenvolvedor é bem produtivo. Ele só não consegue, às vezes, tocar as coisas sozinho, e aí ele pede ajuda dos outros" – l8

O **conhecimento e experiência** também foram citados pelos líderes de projeto como os diferenciais dos desenvolvedores mais produtivos. Os desenvolvedores mais produtivos utilizam todo seu conhecimento técnico para transpor os obstáculos enfrentados na realização de suas tarefas. A experiência e familiaridade com o projeto em que o desenvolvedor está trabalhando também foi citada pelos líderes, pois apesar de possuir o conhecimento técnico necessário, a falta dessa experiência e familiaridade com o projeto pode fazer com que o desenvolvedor não consiga realizar suas tarefas eficientemente.

Sobre os desenvolvedores mais produtivos:

"Ele (Top 1) tem bastante conhecimento [técnico] e vai conseguir desenrolar tudo" – l12

"Não é o conhecimento técnico, é mais tempo de projeto, [pois] tem menos tempo, acho que dois anos de projeto" – l7

Sobre os demais desenvolvedores:

"[Conhecimento técnico] menor. Por isso que ela pegava as tarefas menos complexas" – l10

8.3.3 SQ3. Quais dos fatores de influência identificados nos desenvolvedores produtivos têm suporte na teoria preliminar desta pesquisa?

As características e os fatores de influência na produtividade identificados nos desenvolvedores produtivos da organização foram comparados com os fatores extraídos relacionados na teoria fundamentada preliminar descrita nesta pesquisa. Para facilitar a resposta para esta subquestão de pesquisa, a desdobramos em termos de fatores técnicos, humanos e organizacionais. A relação de suporte ainda foi dividida para o caso do desenvolvedor mais produtivo e para o caso dos desenvolvedores produtivos de cada projeto.

Praticamente todos os fatores técnicos da teoria fundamentada preliminar foram citados pelos líderes de projeto como presentes nos desenvolvedores produtivos das organizações, com exceção do nível de escolaridade (Tabela 54), sem que nenhum novo fator fosse identificado. O conhecimento na tecnologia esteve presente em todos os desenvolvedores produtivos, porém, de acordo com os líderes de projeto, o conhecimento é maior nos desenvolvedores mais produtivos. Eles também foram citados com o diferencial de possuírem domínio, experiência e familiaridade nas tecnologias, no negócio e no projeto, respectivamente.

Tabela 54. Fatores Técnicos identificados com suporte na teoria fundamentada preliminar.

Fatores Técnicos teoria fundamentada	Desenvolvedor mais Produtivo	Desenvolvedores Produtivos
Conhecimento técnico:		
Conhecimento na tecnologia	√	√
Nível de escolaridade		
Domínio e experiência:		
Domínio do negócio	√	
Experiência na tecnologia	√	
Familiaridade com o projeto	√	

Com relação aos fatores humanos da teoria fundamentada preliminar (Tabela 55), todos os fatores da teoria foram citados como presentes nos desenvolvedores produtivos das organizações. Contudo, é preciso considerar que são os mesmos líderes de projeto que foram entrevistados tanto para obtenção da teoria, quanto para este estudo. Já os novos fatores, descritos a partir das características individuais citadas pelos líderes, como a resiliência, a paciência, a disciplina e a competência, não foram citados nos fatores de influência da teoria fundamentada preliminar. Além disso, a facilidade de comunicação dos desenvolvedores mais produtivos apareceu associada com o relacionamento interpessoal presente na teoria, mas somente no contexto do relacionamento com a equipe.

Tabela 55. Fatores Humanos com suporte na Teoria Fundamentada.

Fatores Humanos Teoria Fundamentada	Desenvolvedor mais Produtivo	Desenvolvedores Produtivos
Comportamento e Atitudes:		
Foco	√	√
Comprometimento	√	√
Motivação	√	
Proatividade	√	√
Relacionamento Interpessoal:		
Relacionamento com o líder		
Relacionamento com a equipe	√	√
Relacionamento com a gerência		
Relacionamento com o cliente		

Poucos foram os fatores organizacionais (Tabela 56) citados pelos líderes de projeto como influenciando os desenvolvedores produtivos das organizações. Os líderes citaram o impacto da indefinição e das mudanças não planejadas dos requisitos na motivação e, consequentemente, na produtividade de todos desenvolvedores produtivos. Além disso, os líderes citaram a troca de contexto (que aparece na teoria fundamentada como a alocação dos desenvolvedores) como fator de influência, tenha sido ela causada pela necessidade de participação do desenvolvedor em outras atividades urgentes fora do projeto ou pelo fato dele ter assumindo novas responsabilidades por determinação da organização. Essas alocações afetam a produtividade de todos desenvolvedores envolvidos no projeto.

Portanto, todos os fatores observados pelos líderes como presente nos desenvolvedores produtivos fazem parte da teoria fundamentada descrita nesta pesquisa. Porém, nem todos os fatores descritos na teoria foram observados pelos líderes nos desenvolvedores produtivos, em especial os fatores organizacionais. Isso pode ser explicado pelo próprio método de pesquisa adotado, que direcionou o foco do líder para cada desenvolvedor produtivo de sua equipe.

Tabela 56. Fatores Organizacionais com suporte na Teoria Fundamentada.

Fatores Organizacionais Teoria Fundamentada	Desenvolvedor mais Produtivo	Desenvolvedores Produtivos
Políticas Organizacionais:		
Processo de desenvolvimento		
Definição dos requisitos	√	√
Gestão do relacionamento com o cliente	√	√
Gestão dos Desenvolvedores:		
Falta de reconhecimento		
Tamanho da equipe		
Alocação dos desenvolvedores	√	√
Ambiente de Trabalho:		
Ambiente salubre		
Ambiente tranquilos		
Equipamentos adequados		

8.3.4 SQ4. Quais dos fatores de influência identificados nos desenvolvedores produtivos tem suporte na revisão terciária desta pesquisa?

Nesta seção, as características e os fatores de influência na produtividade identificado nos desenvolvedores produtivos foram comparados com os fatores extraídos da literatura de acordo com a revisão terciária realizada nesta pesquisa. Assim como na subquestão anterior, a resposta foi desdobrada em fatores técnicos, humanos e organizacionais, focando tanto no caso dos desenvolvedores mais produtivo quanto apenas nos produtivos. Porém, os fatores identificados foram relacionados diretamente com os fatores encontrados nos estudos secundários selecionados e não com os fatores unificados na revisão terciária. Isto se deu em razão da diversidade e granularidade dos fatores presentes na literatura, o que

facilitou o mapeamento daqueles fatores para os identificados pelos líderes. Por causa da grande quantidade de fatores extraídos dos estudos secundários, os resultados só listaram aqueles com suporte na literatura.

Os fatores técnicos identificados nos desenvolvedores produtivos (Tabela 57) foram todos relacionados com fatores existentes na literatura. O conhecimento técnico do desenvolvedor foi relacionado como compatível com a sua capacidade de programação (*programmer capability*) necessária para realização das suas tarefas. Os demais fatores identificados pelos líderes como fortemente presentes nos desenvolvedores mais produtivos de cada projeto também encontraram suporte na literatura. O domínio do negócio (*domain of the application*), a experiência (*experience*) e a familiaridade (*application experience & familiarity*) com o projeto foram fatores que diferenciaram esses desenvolvedores mais produtivos dos demais membros da equipe de desenvolvimento.

Tabela 57. Fatores Técnicos identificados com suporte na Revisão Terciária.

Fatores Técnicos Revisão Terciária	Desenvolvedor mais Produtivo	Desenvolvedores Produtivos
Conhecimento:		
Domain of the application	√	
Capacidade e Experiência:		
Experience	√	
Application experience & familiarity	√	
Programmer capability	√	√

Nem todos os fatores humanos (Tabela 58) identificados pelos líderes de projeto nos desenvolvedores produtivos foram encontrados nos fatores extraídos pela revisão terciária. Em especial, o foco do desenvolvedor não constou em nenhum fator extraído dos estudos secundários. Também não foram encontradas a paciência, a disciplina, a resiliência, a competência, nem a autossuficiência. Porém, a facilidade de comunicação, não incluída na teoria, apareceu como um fator na literatura, associada com a coesão da equipe de desenvolvimento. A motivação (*motivation*), a facilidade de comunicação (*communication*) e o correspondente relacionamento interpessoal (*interpersonal relationship*) foram fatores identificado pelos líderes nos desenvolvedores mais produtivos. Além desses, outros fatores

humanos identificados, presente em todos os desenvolvedores produtivos, incluíram o comportamento proativo (*attitudes*) e o comprometimento assumido (*commitment*) na realização das tarefas com eficiência.

Tabela 58. Fatores Humanos identificados com suporte na Revisão Terciária.

Fatores Humanos Revisão Terciária	Desenvolvedor mais Produtivo	Desenvolvedores Produtivos
Motivação		
Motivation	√	
Coesão e Comunicação na Equipe:		
Communication	√	
Interpersonal relationship	√	
Outras Características Individuais:		
Attitudes	√	√
Commitment	√	√

Por fim, todos os fatores organizacionais (Tabela 59) identificados pelos líderes de projeto como de impacto na produtividade tiveram suporte em estudos na literatura. Esses fatores não foram, segundo os líderes, discriminantes entre os desenvolvedores produtivos e os desenvolvedores mais produtivos de cada projeto. A inconsistência (*consistent requirements*) e instabilidade (*requirements stability*) dos requisitos influenciam, segundo os líderes de projeto, na produtividade de qualquer desenvolvedor da equipe, incluindo os desenvolvedores produtivos.

Tabela 59. Fatores Organizacionais identificados com suporte na Revisão Terciária.

Fatores Organizacionais Revisão Terciária	Desenvolvedor mais Produtivo	Desenvolvedores Produtivos
Requisitos Consistentes:		
Consistent requirements	√	√
Requirementes stability	√	√
Fragmentação do Tempo:		
Time fragmentation	√	√

Além disso, os líderes citaram a troca de contexto como um fator limitante da produtividade dos desenvolvedores produtivos do projeto, causada por alocações para tarefas fora do projeto determinadas pela organização. Na literatura, é possível

encontrar um fator de fragmentação do tempo (*time fragmentation*) compatível com esse fator apontado pelos líderes de projeto. A fragmentação do tempo é uma consequência natural de qualquer realocação, assim como uma consequência da troca de contexto, afetando igualmente a produtividade dos desenvolvedores.

Portanto, todos os fatores técnicos descritos pelos líderes como presente nos desenvolvedores produtivos da organização foram identificados na revisão terciária realizada nesta pesquisa. Porém, nem todos fatores humanos e organizacionais identificados na revisão terciária foram descritos pelos líderes de projeto. Os fatores organizacionais foram os menos citados. Assim como na subquestão anterior, a explicação pode ser o próprio método de pesquisa empregado, uma vez que direcionou o foco do líder para cada desenvolvedor produtivo de sua equipe e não, por exemplo, para a organização.

8.4 TEORIA FUNDAMENTADA EVOLUÍDA SOBRE FATORES DE INFLUÊNCIA NA PRODUTIVIDADE DO DESENVOLVEDOR DE ORGANIZAÇÕES DE *SOFTWARE*

Com os resultados apresentados quanto ao suporte pela teoria fundamentada preliminar dos fatores identificados nos desenvolvedores produtivos, verificou-se a que os líderes de projeto citaram características de desenvolvedores produtivos que podem ser consideradas como fatores de influência na produtividade. A identificação de novos fatores é um fato que não deve causar surpresa, pois era até mesmo esperado. Os líderes de projeto foram questionados no estudo realizado (Capítulo 6) sobre os fatores de influência que eles acreditavam de maior influência, enquanto no estudo deste capítulo esse tema é questionado sob a perspectiva dos seus desenvolvedores mais produtivos. A utilização de diferentes perspectivas, permite a triangulação dos dados, fortalecendo os resultados obtidos.

As características individuais citadas pelos líderes não podem ser classificadas como fatores técnicos ou organizacionais, mas fatores humanos, uma vez que as mesmas são inatas ao desenvolvedor, e não controladas ou definidas pela organização. Esses novos fatores humanos citados pelos líderes de projeto foram a resiliência, a paciência, a disciplina e a competência.

Dessa forma, torna-se necessário revisitar a teoria fundamentada preliminar e evoluí-la a fim de contemplar esses novos fatores identificados. A Figura 26

apresenta a teoria fundamentada evoluída, incluindo os novos fatores dentro dos fatores humanos que influenciam a produtividade dos desenvolvedores de organizações de *software*.

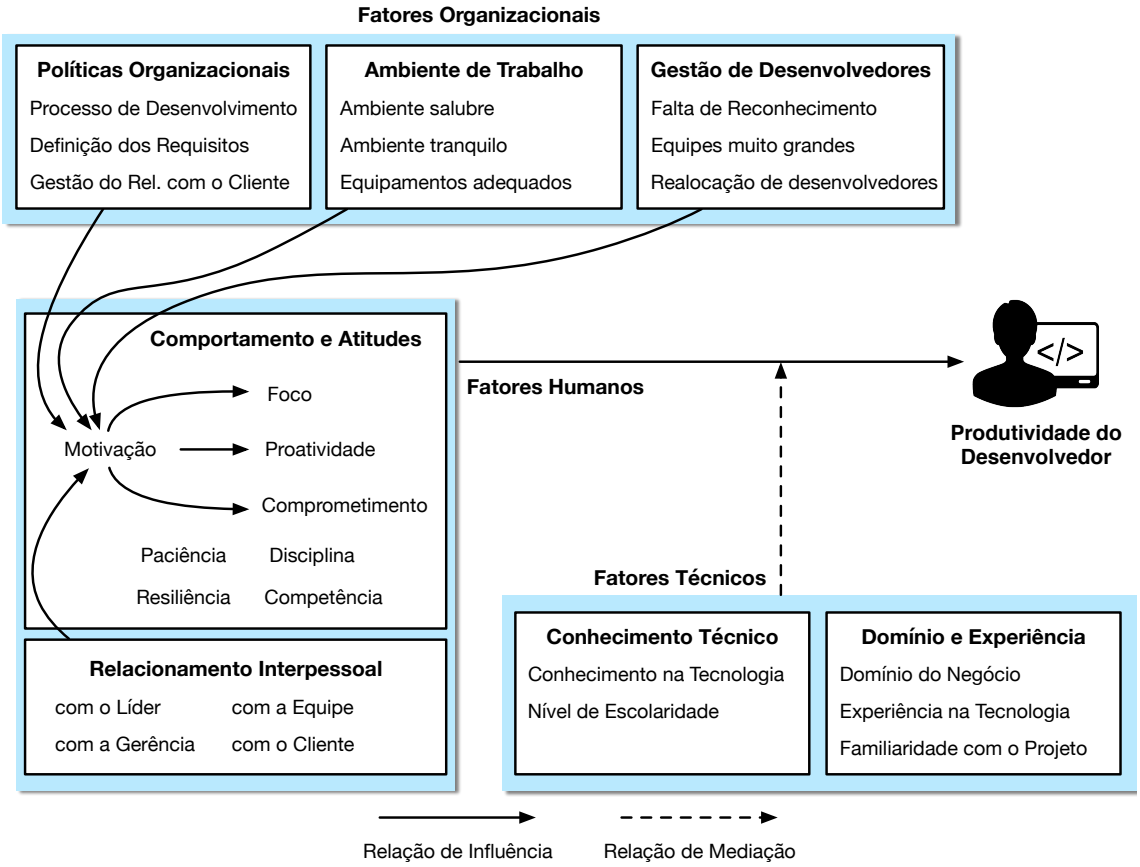

Figura 26. Teoria Fundamentada Evoluída.

8.5 DISCUSSÃO DOS RESULTADOS

8.5.1 QP1. Quais são os fatores de influência na produtividade dos desenvolvedores produtivos da organização?

Este estudo teve como um dos objetivos identificar os fatores de influência da produtividade presente, segundo a percepção dos líderes de projetos de *software*, nos desenvolvedores considerados produtivos pelas organizações. Além disso, também foram investigados os fatores e as características percebidas pelos líderes que distinguem o desenvolvedor mais produtivo dos demais. Um total de oito líderes de projeto de *software* foram entrevistados neste estudo, respondendo ao roteiro de perguntas pré-estabelecido sobre um conjunto de 31 desenvolvedores mais

produtivos em seus projetos, selecionados a partir da percepção dos líderes e das métricas de produtividade baseadas em arquivos.

Foi mostrado que, segundo os líderes de projeto, os fatores que predominam nos desenvolvedores produtivos são relacionados com o seu comportamento, como o foco, o comprometimento, a proatividade e a motivação, assim como o seu conhecimento, como o conhecimento técnico, conhecimento do negócio e a experiência, além de alguns fatores relacionados ao contexto organizacional, como a inconsistência dos requisitos. Ademais, os líderes de projeto também citaram algumas características presentes nos desenvolvedores mais produtivos, tais como a paciência, a disciplina, a resiliência, a competência e a eficiência. Comparando o desenvolvedor mais produtivo de cada projeto com os demais desenvolvedores produtivos, os líderes de projeto encurtaram ainda mais esta lista de fatores. O desenvolvedor mais produtivo de cada projeto tem como característica predominante o foco e a motivação, além de possuir o conhecimento técnico e a experiência necessária. Alguns outros fatores também foram atribuídos aos desenvolvedores que não são os mais produtivos, tais como a falta de comunicação e a dependência técnica. Disto, é possível deduzir que o oposto dessas características está presente nos desenvolvedores mais produtivos, ou seja, eles devem ser (a) autossuficientes, desenvolvendo e concluindo suas tarefas de forma autônoma; (b) capazes de buscar apoio na equipe quando necessário, (c) avessos ao isolamento e (d) solícitos à colaboração aos demais desenvolvedores.

Paiva *et al.* (2010) investigaram os fatores de influência da produtividade pela visão dos desenvolvedores de *software*. Segundo os desenvolvedores, a motivação é o fator de maior influência. Além dele, entre os dez fatores mais influentes na produtividade, também foram citados o comprometimento, a experiência e o relacionamento interpessoal. Os resultados obtidos por Paiva *et al.* (2010) são similares aos obtidos nesta pesquisa. Pela percepção dos líderes de projeto, os desenvolvedores mais produtivos também possuem motivação, comprometimento, relacionamento interpessoal e experiência. Porém os líderes, pela sua percepção, além desses fatores consideram também o foco, a proatividade e conhecimento técnico do desenvolvedor como fatores importantes na influência da produtividade. Por fim, para os líderes de projeto, o desenvolvedor mais produtivo

de seus projetos possui esses fatores num grau superior aos demais desenvolvedores da equipe.

Esses resultados corroboram a preponderância dos fatores considerados humanos na produtividade do desenvolvedor de organizações de *software*, pois durante o projeto são seus comportamentos e atitudes que fazem com que suas tarefas fluam para a conclusão dentro do tempo esperado. Aqueles desenvolvedores que conseguem ultrapassar os obstáculos encontrados durante o ciclo do projeto, seja por iniciativas de estudo ou busca de inovações, ainda mesmo que não seja uma tarefa sua, são considerados como os desenvolvedores mais produtivos, pois os líderes estão indicando a sua contribuição para a produtividade do projeto. Esses desenvolvedores, além de seu amplo conhecimento técnico, tem facilidade de comunicação com o restante da equipe, facilitando o conhecimento técnico aos demais membros, contribuindo majoritariamente para a produtividade da equipe e, com isso, para a conclusão do projeto com o sucesso esperado.

8.5.2 QP2. Quais fatores de influência na produtividade identificados nos desenvolvedores produtivos têm suporte em outros estudos?

Este estudo teve como outro objetivo identificar se os fatores de influência da produtividade presente nos desenvolvedores considerados produtivos pelas organizações possuíam suporte em outros estudos. Para isso foram considerados o estudo realizado nesta pesquisa que resultou em uma teoria fundamentada sobre os fatores de influência dos desenvolvedores, assim como também foram considerados os fatores existentes na literatura científica, a partir da revisão terciária também realizada nesta pesquisa. Os mesmos oito líderes de projeto de *software* foram entrevistados, identificando fatores de produtividade existentes em 31 desenvolvedores mais produtivos de seus projetos.

Ficou demonstrado que, segundo os líderes de projeto, todos os fatores técnicos e organizacionais de influência na produtividade identificados nos desenvolvedores produtivos encontraram suporte na teoria fundamentada e na literatura. Contudo, nem todos os fatores técnicos e organizacionais constantes na teoria fundamentada desta pesquisa e na literatura foram identificados nos desenvolvedores produtivos investigados, sendo a explicação mais provável o escopo deste estudo bem menor, somente em duas organizações de *software*, do que o

escopo da grande quantidade de estudos existentes na literatura. O próprio foco deste estudo em desenvolvedores específicos da organização também pode ter influenciado esse resultado. Ainda assim, o conhecimento, tanto técnico, quando do domínio da aplicação, e a experiência são fatores predominante nos desenvolvedores produtivos, permitindo uma maior flexibilidade junto aos desafios encarados pelos desenvolvedores em suas tarefas. Esse resultado reforça a ideia de que o conhecimento é uma habilitação que permite ao desenvolvedor ser um colaborador mais útil para organização, atuando, portanto, como um mediador da produtividade. Aqueles desenvolvedores que não o possuem, são limitados em suas tarefas e, comumente, alocados para as tarefas mais simples do projeto. Além disso, a ideia de que os fatores organizacionais também influenciam na produtividade também foi reforçada. Esses fatores estabelecem os caminhos do projeto, ora simplificando-os, ora dificultando-os, como no caso de realocação de desenvolvedores para tarefas pontuais e urgentes, impactando diretamente na produtividade dos desenvolvedores envolvidos.

Os fatores humanos, considerados mais relevantes pelos líderes, identificados nos desenvolvedores produtivos da organização, tais como a motivação, o comprometimento e proatividade, tiveram suporte tanto na literatura, quanto na teoria. Exceção ao foco, fator muito citado pelo líder, que, surpreendentemente, não possui nenhuma referência específica na literatura. Uma possível explicação para isso é que o foco possa estar relacionado a outros fatores mais estudados, como a motivação. Essa explicação também pode ser adotada para as ausências de outras características como a paciência, a resiliência, a disciplina e até mesmo a competência, pois esses fatores são difíceis de medir e, por isso, talvez sejam colocados em segundo plano em relação a outros fatores mais tangíveis e comensuráveis.

A grande diferença observada entre os desenvolvedores participantes dos projetos reside, para os líderes de projeto, em fatores como motivação, comprometimento e proatividade. Considerando que tais fatores descrevem a própria natureza do ser humano como agente propulsor, é possível sintetizar o papel dos fatores, de acordo com as crenças dos atores envolvidos, afirmando que *os fatores técnicos habilitam, os organizacionais estabelecem os caminhos e os humanos fornecem a propulsão que determina o grau de produtividade de cada um*. Nesta

pesquisa, esta propulsão foi principalmente vinculada à motivação do desenvolvedor, o que nos leva a concluir que, de acordo com os líderes, são os fatores humanos os preponderantes na produtividade dos desenvolvedores de organizações de *software*.

Esta perspectiva é de muita importância para as organizações de *software*, pois as permite direcionar seus esforços de melhoria da produtividade para os seus desenvolvedores. Esse esforço pode estabelecer, nas organizações, novas práticas para motivar os seus desenvolveres a aprimorar seus conhecimentos, ampliando as suas capacidades, simplificando os processos, facilitando os caminhos a serem percorridos e, acima de tudo, transformando seus dias de trabalho em uma experiência prazerosa, impulsionando-os a desenvolver suas tarefas de forma mais efetiva.

8.6 AMEAÇAS À VALIDADE

Uma das principais ameaças à validade deste estudo é quanto à generalização das conclusões para todas as outras existentes organizações de *software*. Estudos qualitativos não utilizam a amostragem estatística para reivindicar a generalização de seus resultados. Contudo, a generalização do trabalho se constata pela harmonia entre os sujeitos da pesquisa. Eles caracterizam uma base natural de generalização que se fundamenta na relação entre profundidade e tipo de experiência vivida, a expressão da experiência e a compreensão da mesma. Ainda assim, para mitigar essa ameaça, foram entrevistados 12 líderes de projetos de duas organizações de desenvolvimento de *software*.

Além da ameaça acima mencionada, as outras principais ameaças à validade deste estudo são as mesmas já descritas em estudo anterior (Capítulo 6, Seção 6.6), tendo sido mitigadas usando as mesmas estratégias descritas anteriormente.

8.7 CONSIDERAÇÕES FINAIS

Este capítulo teve como objetivos identificar os fatores de influência da produtividade presente, segundo a percepção dos líderes de projetos, nos desenvolvedores considerados produtivos pelas organizações, assim como também identificar se esses fatores de influência possuíam suporte em outros estudos, considerando a teoria fundamentada e a revisão terciária realizadas nesta pesquisa.

Um total de oito líderes de projeto de *software* foram entrevistados, respondendo ao roteiro de perguntas pré-estabelecido sobre um conjunto de 31 desenvolvedores mais produtivos em seus respectivos projetos, selecionados a partir da percepção dos líderes e das métricas de produtividade baseadas em arquivos. Os resultados obtidos a partir da percepção dos líderes de projeto indicaram que os fatores humanos, percebidos pelo seu comportamento durante a realização das atividades do projeto, são os preponderantes para a produtividade dos desenvolvedores das organizações. Os fatores técnicos têm a sua importância especial na preparação e capacitação do desenvolvedor para realização de suas tarefas, habilitando-o, de acordo com o seu conhecimento e experiência, a realizar funções mais diversas com maior eficiência. Por fim, os fatores organizacionais foram considerados, pelos líderes de projeto, como fatores de influência na produtividade dos desenvolvedores, simplificando ou dificultando a realização do trabalho, estabelecendo assim, os ritos e processos a serem percorridos pela equipe de desenvolvimento para atingir o sucesso do projeto.

Capítulo 9 – CONCLUSÕES E PERSPECTIVAS FUTURAS

Este capítulo apresenta conclusões desta pesquisa, apresentando as suas principais contribuições. Além disso, são apresentadas as perspectivas futuras, fornecendo possíveis direções de continuidade desta pesquisa.

9.1 CONCLUSÕES FINAIS

Esta pesquisa teve como questão principal a busca pelos fatores de influência da produtividade preponderantes nos desenvolvedores de organizações de *software*, de acordo com a perspectiva dos líderes dos projetos e da literatura especializada. Com este objetivo e uma metodologia apoiada em uma visão pragmática de pesquisa, foram investigadas respostas com resultados práticos e aplicáveis às organizações de *software*. Primeiro buscou-se na literatura os conceitos aplicáveis e as formas de medição da produtividade dentro da Engenharia de *Software* e, mais ao final da pesquisa, para evitar a criação de viés nos estudos qualitativos desenvolvidos, buscou-se, ainda na literatura, os fatores de influência na produtividade dos desenvolvedores em estudos secundários recentes. Estudos qualitativos foram desenvolvidos em organizações reais de *software*, para identificar os conceitos e fatores *in vivo*, junto aos gerentes e líderes de projetos de *software*. Além disso, no estudo qualitativo com os líderes de projeto, a investigação dos fatores de produtividade gerou uma teoria fundamentada. Um estudo quantitativo foi realizado para avaliar, em projetos reais, as métricas de produtividade comumente utilizadas pelos pesquisadores para avaliação da produtividade dos desenvolvedores de *software*. Como resultado, identificou-se que métricas baseadas em arquivos de código fonte são ainda úteis, pois tem uma forte relação com as percepções dos líderes de projeto. Esse estudo com as métricas proporcionou ainda uma identificação dos desenvolvedores produtivos da organização, contando com aqueles indicados pelo líder e pelas métricas de produtividade. Essa seleção foi o insumo necessário para a realização de mais um estudo qualitativo, visando verificar a teoria fundamentada identificada nesta pesquisa.

Após a realização de toda essa metodologia, aplicada através de estudos diretamente aplicados na indústria, foram obtidos uma série de resultados importantes. Esses resultados indicaram a preponderância dos fatores humanos em relação aos fatores técnicos e organizacionais, quando ponderadas as crenças dos líderes com insumos da literatura. É importante frisar que fatores técnicos e organizacionais ainda são fatores importantes para a produtividade, mas ficam em segundo plano quando o foco é a produtividade do desenvolvedor dentro de organizações de *software*.

9.2 PRINCIPAIS RESULTADOS

As principais contribuições desta tese de doutorado são descritas abaixo:

- Identificação, classificação e análise das métricas de produtividade de *software* existentes na literatura, utilizadas pelos pesquisadores na Engenharia de *Software*, conforme mapeamento sistemático realizado;
- Identificação e classificação em técnicos, humanos ou organizacionais de fatores de influência na produtividade dos desenvolvedores de *software*, conforme revisão terciária realizada;
- Identificação dos conceitos, percepções e utilidade da produtividade dos desenvolvedores de organizações de *software*, conforme estudo qualitativo realizado com gerentes de organizações de *software*;
- Identificação de uma teoria fundamentada sobre os fatores de influência na produtividade dos desenvolvedores de organizações de *software*, conforme estudo qualitativo realizado com líderes de projeto de *software*;
- Identificação de métricas de produtividade do desenvolvedor representativa das percepções da organização, conforme estudo quantitativo com métricas de produtividade nos repositórios de software, disponibilizados pelas organizações, com os líderes de projeto;
- Identificação de fatores característicos em desenvolvedores produtivos das organizações de *software*, conforme um segundo estudo qualitativo realizado com líderes de projeto;

- Identificação de fatores preponderantes na produtividade dos desenvolvedores, conforme análise de todos os estudos realizados, respondendo à questão de pesquisa desta tese.

9.3 ARTIGOS PUBLICADOS

Durante esta pesquisa de doutorado, alguns artigos foram publicados relacionados ao tópico de pesquisa desta tese. As referências resumidas dos artigos originados desta pesquisa são apresentados a seguir em ordem cronológica de publicação:

- Oliveira, E., Conte, T., Cristo, M., Mendes, E. (2016). Software Project Managers' Perceptions of Productivity Factors. In *Proceedings of the 10th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement* – ESEM 2016;
- Oliveira, E., Viana, D., Cristo, M., Conte, T. (2017). How Have Software Engineering Researchers Been Measuring Software Productivity? A Systematic Mapping Study. In *Proceedings of the 19th International Conference on Enterprise Information Systems* – ICEIS 2017;
- Oliveira, E., Fernandes, E., Steinmacher, I., Garcia, A., Cristo, M., Conte, T., Branco, D., Braga, L. (2017). Do Developers' Productivity Metrics Correspond to Software Organizations' Perceptions? Submetido para: 2018 IEEE/ACM 40th *International Conference on Software Engineering: Software Engineering in Practice Track* – ICSE-SEIP 2018;

Além disso, outros artigos publicados em colaboração com outros pesquisadores estão listados a seguir:

- Rabelo, J., Oliveira, E., Viana, D., Braga, L., Santos, G., Steinmacher, I., Conte, T. (2015). Knowledge Management and Organizational Culture in a Software Organization – a Case Study. In 8th *International Workshop on Cooperative and Human Aspects of Software Engineering* – CHASE 2015;
- Martínez C. Branco, D. T., Cunha de Oliveira, E. C., Galvão, L., Prikladnicki, R., Conte, T. (2015). An Empirical Study about the Influence of Project Manager Personality in Software Project Effort. In *Proceedings of*

the 17th International Conference on Enterprise Information Systems – ICEIS 2015;

- Valentim, N. M. C., Lopes, A., César, E., Conte, T., Vincenzi, A. M. R., Maldonado, J. C. (2017). An Acceptance Empirical Assessment of Open Source Test Tools. In *Proceedings of the 19th International Conference on Enterprise Information Systems* – ICEIS 2017.
- Oliveira, R., Sousa, L., De Mello, R., Valentim, N., Lopes, A., Conte, T., Garcia, A., Oliveira, E., Lucena, C. (2017). Collaborative Identification of Code Smells: A Multi-Case Study. In 2017 IEEE/ACM 39th International Conference on Software Engineering: Software Engineering in Practice Track (ICSE-SEIP);
- Sousa, L., Oliveira, R., De Mello, R., Oizumi, W., Lee, J., Oliveira, E., R., Valentim, N., Lopes, A., Conte, T., Garcia, A., Lucena, C. (2017). How Do Software Developers Identify Design Problems? A Qualitative Analysis. In *Proceedings of the 31st Brazilian Symposium on Software Engineering* – SBES 2017.

9.4 PERSPECTIVAS FUTURAS

A realização desta pesquisa resultou, além de outros resultados, na identificação dos fatores de influência preponderantes nos desenvolvedores de organizações de *software*. Esse resultado abre novas perspectivas de pesquisa, que podem ser exploradas em trabalhos futuros. Alguns desses trabalhos futuros são relatados a seguir:

- Replicação do estudo com as métricas de produtividade em repositórios de projetos *open source*, avaliando as mesmas de acordo com a percepção dos líderes do projeto;
- Um estudo para identificação e análise das causas dos casos discrepantes entre a percepção do líder de projeto e o resultado apresentado nas métricas, permitindo a evolução das formas de avaliação da produtividade;

- Aplicação de um *survey* com gerentes e líderes de projetos de organizações *software*, permitindo avaliar em larga escala se os resultados obtidos nesta pesquisa são representativos também em diversas outras organizações de desenvolvimento de *software*.
- Um estudo para a definição de uma ontologia para os diversos fatores de influência de produtividade existentes na literatura, classificando e coletando as formas de medição utilizadas, facilitando futuros estudos sobre fatores de produtividade, permitindo uma melhor agregação dos resultados em revisões sistemáticas.
- Proposição de métricas ou procedimentos que tentem avaliar aspectos de impacto sobre o processo de desenvolvimento, de natureza menos comensurável, tais como a motivação do desenvolvedor.
- Definição de diretrizes para líderes e organizações de software que promovam a formação e manutenção de ambientes mais prazerosos e produtivos.

REFERÊNCIAS

Abdel-Hamid, T. K. (1996). The slippery path to productivity improvement. *IEEE Software*, v. 13, n. 4, pp. 43–52. DOI= http://dx.doi.org/10.1109/52.526831.

Albrecht, A. J. (1979). Measuring application development productivity. In *Proceedings of the Joint SHARE, GUIDE, and IBM application development symposium*. v. 10, pp. 83–92.

Amrit, C., Daneva, M., Damian, D. (2014). Human factors in software development: On its underlying theories and the value of learning from related disciplines. A guest editorial introduction to the special issue. *Information and Software Technology*, v. 56, n. 12, pp. 1537–1542. DOI= http://dx.doi.org/10.1016/j.infsof.2014.07.006.

Anselmo, D., Ledgard, H., Kelvin, Lord, Constantine, L., Anselmo, D., Ledgard, H. (2003). Measuring Productivity in the Software Industry. *Communications of the ACM*, v. 46, n. 11, pp. 121–125. DOI= http://dx.doi.org/10.1145/948383.948391.

Aquino Junior, G. S., Meira, S. R. L. (2009). Towards effective productivity measurement in software projects. In *Proceedings of the 4th International Conference on Software Engineering Advances (ICSEA 2009)*. pp. 241–249. DOI= http://dx.doi.org/10.1109/ICSEA.2009.44.

Asmild, M., Paradi, J. C., Kulkarni, A. (2006). Using Data Envelopment Analysis in software development productivity measurement. *Software Process: Improvement and Practice*, v. 11, n. 6, pp. 561–572. DOI= http://dx.doi.org/10.1002/spip.298.

Avison, D. E., Lau, F., Myers, M. D., Nielsen, P. A. (1999). Action research. *Communications of the ACM*, v. 42, n. 1, pp. 94–97. DOI= http://dx.doi.org/10.1145/291469.291479.

Banker, R. D., Datar, S. M., Kemerer, C. F. (1991). A Model to Evaluate Variables Impacting the Productivity of Software Maintenance Projects. *Management Science*, v. 37, n. 1, pp. 1–18. DOI= http://dx.doi.org/10.1287/mnsc.37.1.1.

Barb, A. S., Neill, C. J., Sangwan, R. S., Piovoso, M. J. (2014). A statistical study of the relevance of lines of code measures in software projects. *Innovations in Systems and Software Engineering*, pp. 243–260. DOI= http://dx.doi.org/10.1007/s11334-014-0231-5.

Basili, V., Gianluigi, C., Rombach, H. (1994). Goal Question Metric Paradigm. *Encyclopedia of Software Engineering*, v. 2.

Basili, V. R., Freburger, K. (1981). Programming measurement and estimation in the software engineering laboratory. *Journal of Systems and Software*, v. 2, n. 1, pp. 47–57. DOI= http://dx.doi.org/10.1016/0164-1212(81)90046-7.

Basili, V. R., Rombach, H. D. (1988). Tame Project: Towards Improvement-

Oriented Software Environments. *IEEE Transactions on Software Engineering*, v. 14, n. 6, pp. 758–773. DOI= http://dx.doi.org/10.1109/32.6156.

Beecham, S., Baddoo, N., Hall, T. (2008). Motivation in Software Engineering : A Systematic Literature Review. *ACM*,

Belbin, M. (2010). *Management teams : why they succeed or fail*. Amsterdam. Elsevier.

Bessen, J., Hunt, R. M. (2007). An empirical look at software patents. *Journal of Economics \& Management Strategy*, v. 16, n. 1, pp. 157–189.

Bird, C., Nagappan, N., Murphy, B., Gall, H., Devanbu, P. (2011). Don't touch my code! In *Proceedings of the 19th ACM SIGSOFT symposium and the 13th European conference on Foundations of software engineering - SIGSOFT/FSE '11*. pp. 4. DOI= http://dx.doi.org/10.1145/2025113.2025119.

Boehm (1987). Improving Software Productivity. *Computer*, v. 20, n. 9, pp. 43–57. DOI= http://dx.doi.org/10.1109/MC.1987.1663694.

Boehm, B., Clark, B., Horowitz, E., Westland, C., Madachy, R., Selby, R. (1995). Cost models for future software life cycle processes: COCOMO 2.0. *Annals of Software Engineering*, v. 1, n. 1, pp. 57–94. DOI= http://dx.doi.org/10.1007/BF02249046.

Boehm, B. W. (1981). *Software Engineering Economics*. Englewood Cliffs, New Jersey. Prentice Hall.

Booch, G. (2008). Measuring Architectural Complexity. *IEEE Software*, v. 25, n. 4, pp. 14–15. DOI= http://dx.doi.org/10.1109/MS.2008.91.

Branco, D. T., Oliveira, E. C., Galvão, L., Prikladnicki, R., Conte, T. (2015). An Empirical Study about the Influence of Project Manager Personality in Software Project Effort. In *Proceedings of the 17th International Conference on Enterprise Information Systems*. pp. 102–113. DOI= http://dx.doi.org/10.5220/0005373001020113.

Briand, L. C., Wieczorek, I. (2002). Resource Estimation in Software Engineering. In: *Encyclopedia of Software Engineering*,. Hoboken, NJ, USA. John Wiley & Sons, Inc. pp. 1–67. DOI= http://dx.doi.org/10.1002/0471028959.sof282

Card, D. N. (2006). The Challenge of Productivity Measurements. *Pacific Northwest Software Quality Conference*, pp. 1–10.

Chand, D. R., Gowda, R. G. (1993). An exploration of the impact of individual and group factors on programmer productivity. In *Proceedings of the 1993 ACM conference on Computer science - CSC '93*. pp. 338–345. DOI= http://dx.doi.org/10.1145/170791.170867.

Chatman, V. V. (1995). CHANGE-POINTs: A proposal for software productivity measurement. *Journal of Systems and Software*, v. 31, n. 1, pp. 71–91. DOI= http://dx.doi.org/10.1016/0164-1212(94)00088-5.

Cheikhi, L., Al-Qutaish, R. E., Idri, A. (2012). Software Productivity: Harmonization in ISO/IEEE Software Engineering Standards. *Journal of*

Software, v. 7, n. 2, pp. 462–470. DOI= http://dx.doi.org/10.4304/jsw.7.2.462-470.

Chrysler, E. (1978). Some basic determinants of computer programming productivity. *Communications of the ACM*, v. 21, n. 6, pp. 472–483. DOI= http://dx.doi.org/10.1145/359511.359523.

Cohen, J. (1960). A coefficient of agreement of nominal scales. *Educational and Psychological Measurement*, v. 20, n. 1, pp. 37–46. DOI= http://dx.doi.org/10.1177/001316446002000104.

Cohen, J. (1988). *Statistical Power Analysis for the Behavioral Sciences*. 2nd. ed. Hillsdale, NJ. Lawrence Erlbaum Associates.

Colazo, J. a (2008). Following the sun: Exploring productivity in temporally dispersed teams. *14th Americas Conference on Information Systems, AMCIS 2008*, v. 3, pp. 1833–1839.

Creswell, J. W. (2013). *Research design: Qualitative, quantitative, and mixed methods approaches*. Sage publications.

Cruz, S., Da Silva, F. Q. B., Capretz, L. F. (2015). Forty years of research on personality in software engineering: A mapping study. *Computers in Human Behavior*, v. 46, pp. 94–113. DOI= http://dx.doi.org/10.1016/j.chb.2014.12.008.

Curtis, B., Hefley, B., Miller, S. (2009). People Capability Maturity Model (P-CMM) Version 2.0, Second Edition.

Dalcher, D. (2006). Supporting software development: enhancing productivity, management and control. *Software Process: Improvement and Practice*, v. 11, n. 6, pp. 557–559.

Dale, C., Van der Zee, H. (1992). Software productivity metrics: who needs them? *Information and Software Technology*, v. 34, n. 11, pp. 731–738. DOI= http://dx.doi.org/10.1016/0950-5849(92)90168-O.

DeMarco, T. (1986). *Controlling Software Projects: Management, Measurement, and Estimation*. Upper Saddle River, NJ. Prentice Hall PTR.

DeMarco, T., Lister, T. (1987). *Peopleware: productive projects and teams*. 1st. ed. New York. Dorset House Publishing.

Deng-Jyi, C., Lee, P. J. (1993). On the study of software reuse using reusable C++ components. *Journal of Systems and Software*, v. 20, n. 1, pp. 19–36. DOI= http://dx.doi.org/10.1016/0164-1212(93)90046-Z.

Dixon-Woods, M., Agarwal, S., Jones, D., Young, B., Sutton, A. (2005). Synthesising qualitative and quantitative evidence: a review of possible methods. *Journal of Health Services Research and Policy*, v. 10, n. 1, pp. 45–53. DOI= http://dx.doi.org/10.1258/1355819052801804.

Dutra, A. C. S., Prikladnicki, R., Franca, C. (2015). What Do We Know about High Performance Teams in Software Engineering? Results from a Systematic Literature Review. In *2015 41st Euromicro Conference on Software Engineering and Advanced Applications*. pp. 183–190. DOI=

http://dx.doi.org/10.1109/SEAA.2015.24.

Fenton, N. E., Pfleeger, S. L. (1998). *Software Metrics: A Rigorous and Practical Approach*. 2nd. ed. Boston, MA, USA. PWS Publishing Co.

França, A. C. C., Gouveia, T. B., Santos, P. C. F., Santana, C. A., Da Silva, F. Q. B. (2011). Motivation in software engineering: a systematic review update. In *Proceedings of the 15th Annual Conference on Evaluation & Assessment in Software Engineering (EASE 2011)*. pp. 154–163. DOI= http://dx.doi.org/10.1049/ic.2011.0019.

Gencel, C., Demirors, O. (2008). Functional size measurement revisited. *ACM Transactions on Software Engineering and Methodology*, v. 17, n. 3, pp. 1–36. DOI= http://dx.doi.org/10.1145/1363102.1363106.

Gill, G. K., Kemerer, C. F. (1991). Cyclomatic complexity density and software maintenance productivity. *IEEE Transactions on Software Engineering*, v. 17, n. 12, pp. 1284–1288. DOI= http://dx.doi.org/10.1109/32.106988.

Glaser, B. G., Strauss, A. L. (1968). *The Discovery of Grounded Theory: Strategies for Qualitative Research.* New York, NY. Aldine Transaction.

Gummesson, E. (1992). Quality dimensions: what to measure in service organizations. *Advances in Services Marketing and Management*, pp. 177–205.

Guthridge, M., Komm, A. B., Lawson, E. (2008). Making talent a strategic priority. *The McKinsey Quarterly*, n. 1, pp. 49–59.

Halstead, M. H. (1977). *Elements of software science*. Holland, New York. Elsevier North.

Hernández-López, A., Colomo-Palacios, R., Garcia-Crespo, A. (2012). Productivity in software engineering: a study of its meanings for practitioners Understanding the concept under their standpoint. In *Proceedings of the 7th Iberian Conference on Information Systems and Technologies*. pp. 1–6.

Hernández-López, A., Colomo-Palacios, R., García-Crespo, A. (2013). Software Engineering Job Productivity – A Systematic Review. *International Journal of Software Engineering and Knowledge Engineering*, v. 23, n. 3, pp. 387–406. DOI= http://dx.doi.org/10.1142/S0218194013500125.

Hernández-López, A., Colomo-Palacios, R., García-Crespo, Á., Cabezas-Isla, F. (2011). Software Engineering Productivity. *International Journal of Information Technology Project Management*, v. 2, n. 1, pp. 37–47. DOI= http://dx.doi.org/10.4018/jitpm.2011010103.

Huang, H., Yang, Q., Xiao, J., Zhai, J. (2011). Automatic mining of change set size information from repository for precise productivity estimation. In *Proceeding of the 2nd workshop on Software engineering for sensor network applications - SESENA '11*. pp. 72. DOI= http://dx.doi.org/10.1145/1987875.1987889.

Humphrey, W. S., Singpurwalla, N. D. (1991). Predicting (individual) software productivity. *IEEE Transactions on Software Engineering*, v. 17, n. 2, pp. 196–207. DOI= http://dx.doi.org/10.1109/32.67600.

Jalali, S., Wohlin, C. (2012). Systematic literature studies. In *Proceedings of the ACM-IEEE international symposium on Empirical software engineering and measurement - ESEM '12*. pp. 29. DOI= http://dx.doi.org/10.1145/2372251.2372257.

Jeffery, D. R., Lawrence, M. J. (1981). Some issues in the measurement and control of programming productivity. *Information & Management*, v. 4, n. 4, pp. 169–176. DOI= http://dx.doi.org/10.1016/0378-7206(81)90057-4.

Jeffery, D. R., Lawrence, M. J. (1985). Managing programming productivity. *Journal of Systems and Software*, v. 5, n. 1, pp. 49–58. DOI= http://dx.doi.org/10.1016/0164-1212(85)90006-8.

Jones, C. (2005). Software Cost Estimating Methods for Large Projects. *CrossTalk, The Journal of Defense Software Engineering*, v. 18, n. 4, pp. 8–12.

Kendall, M. G. (1955). *Rank correlation methods*. 2nd. ed. New York, NY, USA. Hafner Publishing Co.

Kitchenham, B. A., Budgen, D., Pearl Brereton, O. (2011). Using mapping studies as the basis for further research – A participant-observer case study. *Information and Software Technology*, v. 53, n. 6, pp. 638–651. DOI= http://dx.doi.org/10.1016/j.infsof.2010.12.011.

Kitchenham, B., Charters, S. (2007). Guidelines for performing Systematic Literature Reviews in Software Engineering.

Kitchenham, B., Mendes, E. (2004). Software productivity measurement using multiple size measures. *IEEE Transactions on Software Engineering*, v. 30, n. 12, pp. 1023–1035. DOI= http://dx.doi.org/10.1109/TSE.2004.104.

Kitchenham, B., Pretorius, R., Budgen, D., Pearl Brereton, O., Turner, M., Niazi, M., Linkman, S. (2010). Systematic literature reviews in software engineering – A tertiary study. *Information and Software Technology*, v. 52, n. 8, pp. 792–805. DOI= http://dx.doi.org/10.1016/j.infsof.2010.03.006.

Landis, J. R., Koch, G. G. (1977). The Measurement of Observer Agreement for Categorical Data. *Biometrics*, v. 33, n. 1, pp. 159. DOI= http://dx.doi.org/10.2307/2529310.

Laranjeira, L. A. (1990). Software size estimation of object-oriented systems. *IEEE Transactions on Software Engineering*, v. 16, n. 5, pp. 510–522. DOI= http://dx.doi.org/10.1109/32.52774.

Lawrence, M. J. (1981). Programming methodology, organizational environment, and programming productivity. *Journal of Systems and Software*, v. 2, n. 3, pp. 257–269. DOI= http://dx.doi.org/10.1016/0164-1212(81)90023-6.

Levenshtein, V. I. (1966). Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*,

Lim, W. C. (1994). Effects of reuse on quality, productivity, and economics. *IEEE Software*, v. 11, n. 5, pp. 23–30. DOI= http://dx.doi.org/10.1109/52.311048.

Lui, K. M., Chan, K. C. C. (2006). Pair programming productivity: Novice–novice vs. expert–expert. *International Journal of Human-Computer Studies*, v. 64, n. 9,

pp. 915–925. DOI= http://dx.doi.org/10.1016/j.ijhcs.2006.04.010.

Malheiros, V., Hohn, E., Pinho, R., Mendonca, M., Maldonado, J. C. (2007). A Visual Text Mining approach for Systematic Reviews. In *First International Symposium on Empirical Software Engineering and Measurement (ESEM 2007)*. n. 7465, pp. 245–254. DOI= http://dx.doi.org/10.1109/ESEM.2007.21.

Marks, M. A., Mathieu, J. E., Zaccaro, S. J. (2001). A temporally based framework and taxonomy of team processes. *Academy of management review*, v. 26, n. 3, pp. 356–376.

Maxwell, K. D. K. D., Van Wassenhove, L., Dutta, S. (1996). Software development productivity of European space, military, and industrial applications. *IEEE Transactions on Software Engineering*, v. 22, n. 10, pp. 706–718. DOI= http://dx.doi.org/10.1109/32.544349.

McGarry, J., Card, D., Jones, C., Layman, B., Clark, E., Dean, J., Hall, F. (2002). *Practical Software Measurement: Objective Information for Decision Makers*.

Melo, C., Cruzes, D. S., Kon, F., Conradi, R. (2011). Agile Team Perceptions of Productivity Factors. In *Proceedings of the 2011 Agile Conference*. pp. 57–66. DOI= http://dx.doi.org/10.1109/AGILE.2011.35.

Meyer, A. N., Fritz, T., Murphy, G. C., Zimmermann, T. (2014). Software developers' perceptions of productivity. In *Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering*. pp. 19–29. DOI= http://dx.doi.org/10.1145/2635868.2635892.

Mockus, A. (2009). Succession: Measuring transfer of code and developer productivity. In *2009 IEEE 31st International Conference on Software Engineering*. pp. 67–77. DOI= http://dx.doi.org/10.1109/ICSE.2009.5070509.

Mockus, A., Fielding, R. T., Herbsleb, J. (2000). A case study of open source software development: the Apache server. In *Proceedings of the 2000 International Conference on Software Engineering. ICSE 2000 the New Millennium*. v. 0, n. 11, pp. 263–272. DOI= http://dx.doi.org/10.1109/ICSE.2000.870417.

Mockus, A., Fielding, R. T., Herbsleb, J. D. (2002). Two case studies of open source software development: Apache and Mozilla. *ACM Transactions on Software Engineering and Methodology*, v. 11, n. 3, pp. 309–346. DOI= http://dx.doi.org/10.1145/567793.567795.

Morisio, M., Stamelos, I., Spahos, V., Romano, D. (1999). Measuring functionality and productivity in Web-based applications: a case study. In *Proceedings Sixth International Software Metrics Symposium (Cat. No.PR00403)*. pp. 111–118. DOI= http://dx.doi.org/10.1109/METRIC.1999.809732.

Moser, R., Abrahamsson, P., Pedrycz, W., Sillitti, A., Succi, G. (2008). A case study on the impact of refactoring on quality and productivity in an agile team. *Lecture Notes in Computer Science*, v. 5082, pp. 252–266.

Moser, S., Nierstrasz, O. (1996). The effect of object-oriented frameworks on developer productivity. *Computer*, v. 29, n. 9, pp. 45–51. DOI= http://dx.doi.org/10.1109/2.536783.

Munson, J. C., Elbaum, S. G. (1998). Code churn: a measure for estimating the impact of code change. In *Proceedings. International Conference on Software Maintenance (Cat. No. 98CB36272)*. pp. 24–31. DOI= http://dx.doi.org/10.1109/ICSM.1998.738486.

Myers, I. B. (1962). *The Myers-Briggs Type Indicator: Manual (1962)*.

Oliveira, E., Viana, D., Cristo, M., Conte, T. (2017). How Have Software Engineering Researchers Been Measuring Software Productivity? A Systematic Mapping Study. In *In Proceedings of the 19th International Conference on Enterprise Information Systems (ICEIS 2017)*. v. 2, pp. 76–87. DOI= http://dx.doi.org/10.5220/0006314400760087.

Paiva, E., Barbosa, D., Lima, R., Albuquerque, A. (2010). Factors that Influence the Productivity of Software Developers in a Developer View. In: *Innovations in Computing Sciences and Software Engineering*, T. Sobh, K. Elleithy. Dordrecht. Springer Netherlands. pp. 99–104. DOI= http://dx.doi.org/10.1007/978-90-481-9112-3_17

Patton, M. (1990). Qualitative Evaluation and Research Methods. *Qualitative Evaluation and Research Methods*, pp. 169–186. DOI= http://dx.doi.org/10.1002/nur.4770140111.

Petersen, K. (2011). Measuring and predicting software productivity: A systematic map and review. *Information and Software Technology*, v. 53, n. 4, pp. 317–343. DOI= http://dx.doi.org/10.1016/j.infsof.2010.12.001.

Petersen, K., Feldt, R., Mujtaba, S., Mattsson, M. (2008). Systematic mapping studies in software engineering. *EASE'08 Proceedings of the 12th international conference on Evaluation and Assessment in Software Engineering*, pp. 68–77. DOI= http://dx.doi.org/10.1142/S0218194007003112.

Petersen, K., Wohlin, C. (2009). Context in industrial software engineering research. In *2009 3rd International Symposium on Empirical Software Engineering and Measurement - ESEM '09*. pp. 401–404. DOI= http://dx.doi.org/10.1109/ESEM.2009.5316010.

Petticrew, M., Roberts, H. (2006). *Systematic Reviews in the Social Sciences: A Practical Guide*.

Quesnay, F. (1766). Analyse de la formule arithmétique du Tableau Economique de la distribution des dépenses annuelles d'une nation agricole. *Journal de l'agriculture, du commerce et des finances*, v. 2, n. juin 1766, pp. 11–41.

Rabelo, J., Oliveira, E., Viana, D., Braga, L., Santos, G., Steinmacher, I., Conte, T. (2015). Knowledge Management and Organizational Culture in a Software Organization – a Case Study. In *8th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE 2015)*. pp. 89–92.

Rahman, F., Devanbu, P. (2013). How, and why, process metrics are better. In *Proceedings - International Conference on Software Engineering*. DOI= http://dx.doi.org/10.1109/ICSE.2013.6606589.

Rezende, D. A. (2005). *Engenharia de Software e Sistemas de Informação*.

Brasport.

Rocha, A. R., Santos, G., Barcellos, M. (2012). *Medição de Software - Controle Estatístico de Processos*. Ministério da Ciência, Tecnologia e Inovação; Secretaria de Política de Informática.

Rossman, G. B., Wilson, B. L. (1985). Numbers and Words: Combining Quantitative and Qualitative Methods in a Single Large-Scale Evaluation Study. *Evaluation Review*, v. 9, n. 5, pp. 627–643. DOI= http://dx.doi.org/10.1177/0193841X8500900505.

Rothenberger, M. A., Dooley, K. J. (1999). A Performance Measure for Software Reuse Projects. *Decision Sciences*, v. 30, n. 4, pp. 1131–1153. DOI= http://dx.doi.org/10.1111/j.1540-5915.1999.tb00921.x.

Ruhe, M., Jeffery, R., Wieczorek, I. (2003). Cost estimation for web applications. *25th International Conference on Software Engineering, 2003. Proceedings.*, v. 6, pp. 285–294. DOI= http://dx.doi.org/10.1109/ICSE.2003.1201208.

Runeson, P., Host, M., Rainer, A., Regnell, B. (2012). *Case study research in software engineering: Guidelines and examples*. Hoboken, NJ. John Wiley & Sons.

Sampaio, S. C. de B., Barros, E. A., Aquino Junior, G. S. De, Silva, M. J. C. e, Meira, S. R. de L. (2010). A Review of Productivity Factors and Strategies on Software Development. In *Proceedings of the 5th International Conference on Software Engineering Advances (ICSEA 2010)*. pp. 196–204. DOI= http://dx.doi.org/10.1109/ICSEA.2010.37.

Scacchi, W. (1995). Understanding software productivity. In *Advances in Software Engineering and Knowledge Engineering*. v. 4, pp. 37–70.

Scholtes, I., Mavrodiev, P., Schweitzer, F. (2016). From Aristotle to Ringelmann: a large-scale analysis of team productivity and coordination in Open Source Software projects. *Empirical Software Engineering*, v. 21, n. 2, pp. 642–683. DOI= http://dx.doi.org/10.1007/s10664-015-9406-4.

SEI (2010). CMMI for Development, Version 1.3. *Carnegie Mellon University*, DOI= http://dx.doi.org/CMU/SEI-2010-TR-033 ESC-TR-2010-033.

Sheetz, S. D., Henderson, D., Wallace, L. (2009). Understanding developer and manager perceptions of function points and source lines of code. *Journal of Systems and Software*, v. 82, n. 9, pp. 1540–1549. DOI= http://dx.doi.org/10.1016/j.jss.2009.04.038.

Siegel, S., Castellan, N. J. (1988). *Nonparametric statistics for the behavioral sciences (2nd ed.)*.

Softex (2016a). MPS.BR - Melhoria de Processo do Software Brasileiro, Guia Geral MPS de Gestão de Pessoas.

Softex (2016b). MPS.BR - Melhoria de Processo do Software Brasileiro, Guia Geral MPS de Software.

Stensrud, E., Myrtveit, I. (2003). Identifying high performance ERP projects. *IEEE Transactions on Software Engineering*, v. 29, n. 5, pp. 398–416. DOI= http://dx.doi.org/10.1109/TSE.2003.1199070.

Strauss, A., Corbin, J. (2008). *Basics of qualitative research: Techniques and procedures for developing grounded theory, 2nd ed.* 3rd.ed. ed. Thousand Oaks, CA. SAGE publications.

Tomaszewski, P., Lundberg, L. (2005). Software development productivity on a new platform: An industrial case study. *Information and Software Technology*, v. 47, pp. 257–269. DOI= http://dx.doi.org/10.1016/j.infsof.2004.08.007.

Trendowicz, A., Münch, J. (2009). Factors Influencing Software Development Productivity – State-of-the-Art and Industrial Experiences. *Advances in Computers*, v. 77, pp. 185–241. DOI= http://dx.doi.org/10.1016/S0065-2458(09)01206-6.

Ulrich, D. (1998). A new mandate for nurses. *Harvard Business Review*, n. 1, pp. 124–134.

Wagner, S., Ruhe, M. (2008). A Systematic Review of Productivity Factors in Software Development. In *Proceedings of the 2nd International Software Productivity Analysis and Cost Estimation (SPACE 2008)*.

Wohlin, C., Runeson, P., Höst, M., Ohlsson, M. C., Regnell, B., Wesslén, A. (2012). *Experimentation in Software Engineering*. Berlin, Heidelberg. Springer Berlin Heidelberg. v. 9783642290

Wray, B., Mathieu, R. (2008). Evaluating the performance of open source software projects using data envelopment analysis. *Information Management & Computer Security*, v. 16, n. 5, pp. 449–462. DOI= http://dx.doi.org/10.1108/09685220810920530.

Yang, Z., Paradi, J. (2009). A DEA evaluation of software project efficiency. In *Industrial Engineering and Engineering Management, 2009. IEEM 2009. IEEE International Conference on*. pp. 1723–1727.

Yin, R. K. (2015). *Estudo de caso: planejamento e métodos*. Tradução Cristhian Matheus Herrera. 5.ed. ed. Porto Alegre. Bookman.