UNIVERSIDADE ESTADUAL DE MARINGÁ
CENTRO DE TECNOLOGIA
DEPARTAMENTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

BRUNO HENRIQUE CAVALCANTE

Diretrizes para o Desenvolvimento de Software em *Startups*

Maringá
2018

BRUNO HENRIQUE CAVALCANTE

Diretrizes para o Desenvolvimento de Software em *Startups*

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação do Departamento de Informática, Centro de Tecnologia da Universidade Estadual de Maringá, como requisito parcial para obtenção do título de Mestre em Ciência da Computação.

Orientadora: Profª. Drª. Gislaine Camila Lapasini Leal
Coorientador: Prof. Dr. Renato Balancieri

Maringá
2018

Dados Internacionais de Catalogação-na-Publicação (CIP)
(Biblioteca Central - UEM, Maringá – PR., Brasil)

Cavalcante, Bruno Henrique

C376d Diretrizes para o desenvolvimento de software em startups/ Bruno Henrique Cavalcante. -- Maringá, 2018.
218 f., il., color., figs., tabs.

Orientadora: Prof.a. Dr.a. Gislaine Camila Lapasini Leal.
Coorientador: Prof. Dr. Renato Balancieri.

Dissertação (mestrado) – Universidade Estadual de Maringá, Centro de Tecnologia, Programa de Pós-graduação em Ciência da Computação, 2018.

1. Desenvolvimento de Software. 2. Startups. 3. Diretrizes. I. Leal, Gislaine Camila Lapasini, orient. II. Balancieri, Renato, coorient. III. Universidade Estadual de Maringá. Centro de Tecnologia. Programa de Pós-Graduação em Ciência da Computação. IV. Título.

CDD 22. ED.005.12
Jane Lessa Monção CRB 1173/97

FOLHA DE APROVAÇÃO

BRUNO HENRIQUE CAVALCANTE

Diretrizes para o desenvolvimento de software em startups

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação do Departamento de Informática, Centro de Tecnologia da Universidade Estadual de Maringá, como requisito parcial para obtenção do título de Mestre em Ciência da Computação pela Banca Examinadora composta pelos membros:

BANCA EXAMINADORA

Profa. Dra. Gislaine Camila Lapasini Leal
Universidade Estadual de Maringá – DEP/UEM

Prof. Dr. Renato Balancieri
Universidade Estadual de Maringá – DIN/UEM

Prof. Dr. Edson Alves de Oliveira Junior
Universidade Estadual de Maringá – DIN/UEM

Prof. Dr. Ivaldir Honorio de Farias Junior
Universidade de Pernambuco – UPE

Aprovada em: 04 de dezembro de 2018.
Local da defesa: Sala 101, Bloco C56, *campus* da Universidade Estadual de Maringá.

AGRADECIMENTOS

Aos meus pais, Rute e Valter, pelo amor e carinho, por estarem sempre ao meu lado.

À Juliana, pela paciência, compreensão e, acima de tudo, carinho e incentivo.

À professora Dr.ª Camila Lapasini Leal e ao professor Dr. Renato Balancieri, por todos os ensinamentos, pela confiança e disposição em sempre ajudar.

Ao professor Dr. Ivaldir de Farias Junior pelas inúmeras contribuições.

À minha família e aos meus amigos, pelos vários bons momentos ao longo dessa jornada e por todo o apoio.

Aos professores do PCC por acreditarem em mim e por todos os ensinamentos.

À Inês pela disponibilidade e auxílio.

Ao CNPq pelo apoio financeiro.

Diretrizes para o Desenvolvimento de Software em *Startups*

RESUMO

Startups são organizações que estão buscando por um modelo de negócio por meio da oferta de produtos ou serviços inovadores e que necessitam de investimentos externos para operar até que possam estabilizar-se no mercado e, a partir disto, começarem a crescer até se tornarem uma organização madura. Entretanto, muitas dessas organizações deixam de existir antes de atingirem a fase de crescimento, surgindo a necessidade de buscar por ferramentas e métodos que apoiem estas *startups* a alcançarem seus objetivos. Neste trabalho é proposto um conjunto de diretrizes voltadas ao desenvolvimento de software em *startups* com o objetivo de proporcionar a essas organizações um conjunto de ferramentas relevantes para que as mesmas possam prosperar em meio as incertezas da indústria. O método de pesquisa deste trabalho é formado por: uma fase exploratória, em que foi realizada uma fundamentação teórica, um mapeamento sistemático e um *survey*, que subsidiaram as fases seguintes; uma fase de desenvolvimento, onde foi realizada a especificação das diretrizes; e uma fase de refinamento, onde estas diretrizes foram refinadas por meio de *survey* confirmatório, grupo focal e painel com especialistas. Este trabalho permitiu fornecer evidências de que a pesquisa em *startups* vem crescendo nos últimos anos e, neste sentido, por meio de seus processos empíricos, organizou e descobriu informações relevantes para estas organizações. Destacam-se os aspectos técnicos levantados no mapeamento sistemático, que permitiram ter uma visão da área de pesquisa, e as descobertas do *survey*, que proporcionaram uma perspectiva do desenvolvimento de software em *startups* brasileiras. No mapeamento sistemático, foi possível identificar um total de 24 técnicas, 31 práticas e 37 ferramentas, entre os 19 artigos encontrados. No *survey*, foram obtidas respostas de 100 *startups* brasileiras, permitindo evidenciar o perfil dessas organizações e contribuindo para a confecção das diretrizes. A principal contribuição é que as diretrizes auxiliam as atividades de desenvolvimento de software e aumentam as chances de sucesso dessas organizações.

Palavras-chave: Desenvolvimento de Software, Startups, Diretrizes

Guidelines for Software Development in Startups

ABSTRACT

Startups are companies seeking for a business model based on innovative products, they need external investments until they can stabilize this product in the market and then start to grow to become a mature company. However, many of these companies fail before the growth phase, emerging the need to seek for methods that help these companies to achieve their goals. In this work, a set of guidelines will be proposed for the software development in startups to provide these organizations with relevant tools to thrive amidst the uncertainties of the industry. The research method of this work consists of: an exploratory phase, consisting of a theoretical basis, a systematic mapping review and a survey which subsidized the following phases; a development phase, consisting of the guidelines specification; and a refinement phase, where the guidelines were refined through a confirmatory survey, focus group and expert panel. This work allowed to provide evidence that the research in startups has been growing in recent years and, in this manner, through its empirical processes, organized and discovered relevant information for these organizations. We can highlight the technical aspects found in the systematic mapping, that allowed to have a view of the research area, and the survey findings, which provided a perspective of software development in brazilian startups. In the systematic mapping, it was possible to identify a total of 24 techniques, 31 practices and 37 tools, among the 19 articles found. In the survey, responses were obtained from 100 Brazilian startups, allowing to highlight the profile of these organizations and contributing to the writing of the guidelines. The contribution of this work is that the guidelines can help software development activities in startups and raise the odds of success for these organizations.

Keywords: Software Development, Startups, Guidelines

SUMÁRIO

1

Introdução

1.1 Contextualização

Startups são organizações que buscam por um modelo de negócio, por meio da oferta de um produto ou serviço, que possa expandir exponencialmente, sendo que essas organizações, geralmente, necessitam de investimentos externos para operar (Paternoster *et al.*, 2014; Sutton, 2000). Também é pode-se dizer que, uma *startup* é um grupo de pessoas à procura de um modelo de negócios que seja repetível e escalável, trabalhando em condições de extrema incerteza[1]. É comum que essas organizações acabem por criar mercados até então desconhecidos, entregando ao público produtos ou serviços que os mesmos até então sequer tinham necessidade. Os casos de Snapchat, Instagram e Facebook são alguns dos exemplos de *startups* que criaram novos segmentos de serviços e se tornaram organizações bilionárias, inspirando novos empreendedores e sendo influência da bolha de *startups* (Paternoster *et al.*, 2014).

Sutton (2000) caracteriza *startup* por meio de quatro desafios que este tipo de organização enfrenta, são eles: imaturidade, recursos limitados, múltiplas influências e tecnologias e mercados dinâmicos. Crowne (2002) descreve a evolução das *startups*, da ideia até a maturidade, por meio de uma divisão em três fases, do pontapé inicial, passando pela estabilização e alcançando o crescimento. Ao fim das fases definidas por Crowne (2002), a organização deixa de ser uma *startup* e se transforma em uma organização madura, que possui um produto estável no mercado e que atende as necessidades de seus

[1] `http://www.sebrae.com.br/sites/PortalSebrae/sebraeaz/o-que-e-uma-startup,616913074c0a3410VgnVCM1000003b74010aRCRD` Acesso em 16 de jan. de 2019.

consumidores e passa a ter a habilidade de desenvolver novos produtos e lançá-los no mercado. Chegar a esta fase de maturidade é, entretanto, um grande desafio para as *startups*, sendo que a maioria dessas organizações não consegue atingir dois anos de vida (Crowne, 2002).

Startups de software proporcionam um ambiente ágil para a concepção e desenvolvimento de ideias inovadoras a um baixo custo (Kon e Monteiro, 2014), porém, além dos desafios listados por Sutton (2000), inerentes a qualquer tipo de *startup*, um dos principais desafios que uma *startup* de software enfrenta é o ato de implementar as metodologias necessárias para estruturar e controlar as atividades de desenvolvimento (Coleman e O'Connor, 2008). Por causa da natureza ágil e flexível dessas organizações, introduzir processos e medidas burocráticas é algo que foge de suas características naturais (Sutton, 2000), sendo assim, adaptar métodos é uma prática comum (Tegegne, 2018). Além disto, por causa da limitação de recursos que enfrentam, essas organizações normalmente preferem investir no desenvolvimento do produto, e não na implementação de processos (Coleman e O'Connor, 2008).

Desta forma, entender como essas *startups* podem se beneficiar de suas práticas de trabalho é importante para apoiar empreendedores (Paternoster *et al.*, 2014). Ao investigar quais práticas de trabalho estão alinhadas com o dinamismo e flexibilidade das *startups* seria possível criar uma base de conhecimento com as práticas mais adequadas, sendo que este conhecimento forneceria o suporte necessário para aquelas *startups* que estão adentrando o mercado, ou até mesmo para aquelas que já estão em fase de crescimento, permitindo que estas organizações tomem decisões melhores embasadas e que foram validadas empiricamente, contribuindo para a diminuição do número de *startups* que encerram suas atividades precocemente. Identificar estas práticas é importante tanto para os profissionais da área, ao fornecer por meio da Engenharia de Software Baseada em Evidência (ESBE) indícios suficientes para a transferência deste conhecimento para a indústria (Dyba *et al.*, 2005; Kitchenham *et al.*, 2004), quanto para os pesquisadores, que poderão identificar, por meio do conhecimento gerado pelo estudo, novas oportunidades de pesquisa.

1.2 Motivação e Justificativa

Um relatório de 2014 do Crunchbase, uma das maiores bases de dados sobre *startups*, aponta que, nos últimos 10 anos, mais de 200 mil *startups* foram fundadas (Crunchbase, 2014). De acordo com a base de dados do site Startup Ranking, a grande maioria das *startups* é concentrada nos Estados Unidos, que aparece com mais de 46 mil *startups*. O

ranking continua com Índia em segundo, com mais de 6 mil *startups* e Reino Unido em terceiro mais de 4 mil. O Brasil aparece na décima posição, com cerca de mil *startups*[2]. Apesar da base de dados do site não ser completamente fiel aos números encontrados na realidade, a partir dela é possível ter uma noção da distribuição de *startups* por país e, devido ao número muito superior de *startups* situadas nos Estados Unidos em relação aos países restantes, generalizar os dados referentes ao cenário estadunidense para os demais.

Um levantamento, realizado pelo Mashable também em 2014[3], demonstra alguns dados quanto ao percentual de falha das *startups*. Segundo a pesquisa, 25% das *startups* falham no primeiro ano, e das que restaram 36% falham no segundo ano, em seguida, novamente das que restaram, 44% falham no terceiro ano e por fim, das remanescentes, 50% falham no quarto ano. Com base nos dados é possível verificar que no segundo ano o percentual de sucesso é de 48%, índice que fica próximo ao afirmado por Crowne (2002) há mais de dez anos. No quarto ano o percentual de sucesso cai para apenas 28%. Não é explícito no artigo, porém o mesmo leva a crer que os dados são referentes ao cenário estadunidense de *startups*. No Brasil, um estudo publicado em 2014 pela Fundação Dom Cabral (FDC)[4], mostrou que uma em cada quatro *startups* deixam de existir em menos de um ano.

Conforme levantado por Paternoster *et al.* (2014), um estudo de 1994 realizado por Carmel (1994) em uma *startup* já apontava para a necessidade de mais pesquisas quanto as práticas de desenvolvimento de software neste tipo de ambiente com o intuito de replicar tais práticas e então transferi-las para outros setores da indústria de tecnologia. Porém, apesar do desenvolvimento de software ser o centro das atividades de uma *startup*, este não possui uma base de conhecimento cientificamente suportada (Paternoster *et al.*, 2014). Sutton (2000) aponta que implementar medidas e processos engessados em *startups* não é algo natural devido ao dinamismo destas organizações, desta forma, implementar métodos que auxiliem as atividades de desenvolvimento de software é um dos principais desafios que elas enfrentam (Coleman e O'Connor, 2008).

Em uma análise da literatura sobre o desenvolvimento de software em *startups*, buscando identificar as práticas de trabalho realizadas nestas organizações, Paternoster *et al.* (2014) concluíram que os estudos encontrados não eram suficientes para entender o processo de desenvolvimento. Em outra análise, por meio de um mapeamento sistemático, Klotins *et al.* (2015) buscaram aprofundar o estudo das práticas de trabalho relacionadas as *startups* e, mais uma vez, os estudos encontrados não foram suficientes. Desta forma,

[2] `http://www.startupranking.com/countries`. Acesso em 11 de dez. de 2018.

[3] `http://mashable.com/2014/01/30/startup-success-infographic/#o_x_guspP0qj`. Acesso em: 07 de ago. de 2017.

[4] `http://www.fdc.org.br/blogespacodialogo/Lists/Postagens/Post.aspx?ID=384`. Acesso em: 25 de jan. de 2017.

ambos os estudos concluiram que a lacuna apontada por Sutton (2000) há mais de dez anos continua em aberto. Além disto, na agenda de pesquisa apresentada por Unterkalmsteiner *et al.* (2016) é apontada a necessidade de estudos que verifiquem o uso de ferramentas de suporte no contexto de *startups*, visto que até o momento a investigação é insuficiente.

Ainda em 2000, Fayad *et al.* (2000) apontavam que a maioria das organizações que desenvolviam software eram pequenas e que o número de *startups* continuava a crescer rapidamente. Apesar destas pequenas organizações estarem interessadas em utilizar as melhores práticas, o ato de identificar e separar quais destas práticas seriam úteis para elas dentre aquelas utilizadas por grandes organizações é algo difícil. *Startups* buscam adotar métodos de desenvolvimento e práticas que sejam flexíveis, leves e que permitam mudanças rápidas (Tegegne, 2018). A partir disto, surge a necessidade de criar um conjunto de diretrizes especificamente voltadas a essas organizações (*startups*). Estas diretrizes são compostas de um conjunto de sugestões e melhores práticas que visam melhorar e guiar a execução de uma determinada rotina (Sjoberg *et al.*, 2007).

1.3 Objetivos

Este trabalho busca responder à seguinte questão de pesquisa: De que forma a Engenharia de Software (ES) pode apoiar as atividades de desenvolvimento de software em *startups*? Desta forma, o objetivo deste trabalho é elaborar diretrizes para o desenvolvimento de software em *startups*.

Como objetivos específicos, derivam-se do objetivo geral os seguintes:

- Proporcionar, por meio das diretrizes, um conjunto de ferramentas de apoio que possam auxiliar as atividades de desenvolvimento de software em *startups*, de forma que estas organizações tenham maiores chances de sobrevivência;
- Fornecer um conjunto de evidências que serão obtidas por meio de um mapeamento sistemático e de um *survey* para os pesquisadores e profissionais da indústria que buscam entender o contexto de desenvolvimento de software em *startups*;
- Incentivar uma aproximação entre a academia e a indústria, de forma que as *startups* possam fazer uso do conhecimento gerado pela academia de forma ágil e que a academia possa aproveitar o ambiente dinâmico e desafiador das *startups* para a aplicação de novas pesquisas.

1.4 Método de Pesquisa

O método de pesquisa adotado para a realização do trabalho apresenta as estratégias de pesquisa selecionadas, visando alcançar o objetivo geral do trabalho.

1.4.1 Caracterização da Pesquisa

Esta pesquisa pode ser caracterizada por meio de quatro perspectivas: natureza, abordagem do problema, objetivos almejados e procedimentos adotados (Gil, 2002):

- Natureza: a pesquisa pode ser considerada de natureza básica, visto que está direcionada à aquisição de novos conhecimentos com o objetivo de resolver problemas práticos.
- Abordagem do problema: a pesquisa será de caráter qualitativo, visto que envolve a análise de dados de forma indutiva.
- Objetivos almejados: a pesquisa pode ser considerada como exploratória e descritiva. Exploratória, pois busca uma maior familiaridade com o problema, visando torná-lo explícito e possibilitar a formulação de hipóteses. Descritiva, pois visa caracterizar o problema, buscando identificar as prováveis relações entre as variáveis.
- Procedimentos adotados: a pesquisa é apoiada, inicialmente, em uma fundamentação teórica, mapeamento sistemático e um *survey*. Estes procedimentos irão subsidiar a elaboração da proposta. Em seguida, a pesquisa irá avançar para a pesquisa de campo, conduzindo um *survey* confirmatório, grupo focal e painel com especialistas.

1.4.2 Estruturação da Pesquisa

A pesquisa é estruturada em três fases: Estudo Exploratório, Desenvolvimento e Refinamento. A Figura - 1.1 apresenta as fases com suas respectivas etapas, além da sequência de execução das etapas.

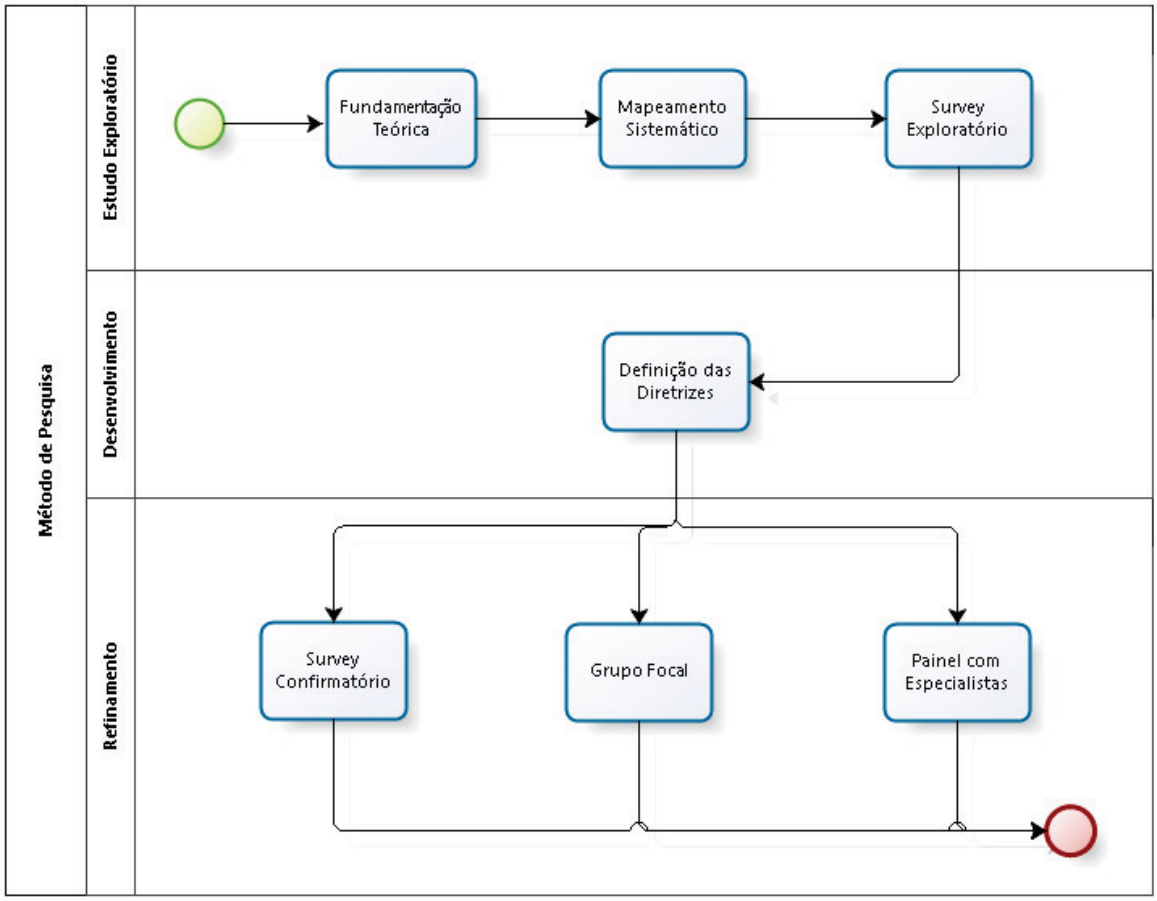

Figura 1.1: Fases da Pesquisa

1.4.2.1 Estudo Exploratório

O estudo exploratório tem como objetivo direcionar a pesquisa e proporcionar a familiarização com o tema de estudo. Esta fase é composta de três etapas: fundamentação teórica, mapeamento sistemático e *survey*, descritas a seguir.

1.4.2.1.1 Fundamentação Teórica

Sumarizar o atual estado de evidência de um problema ou assunto e identificar lacunas em seu estado de pesquisa atual, que podem ser tomadas como ponto de partida para novas pesquisas, são algumas das razões para se realizar uma fundamentação teórica.

Neste trabalho, a fundamentação teórica teve como objetivo formar um referencial teórico consistente, permitindo a visualização do estado da arte e então subsidiar o

desenvolvimento das demais etapas. Nesta etapa, foram abordadas as principais áreas envolvidas na pesquisa, sendo elas: *Startups* e Desenvolvimento de Software em *Startups*.

1.4.2.1.2 Mapeamento Sistemático

O mapeamento sistemático proporciona uma visão geral dos estudos e seus resultados, pois este tipo de estudo possibilita a identificação de lacunas, onde novos ou melhores estudos primários são necessários (Budgen *et al.*, 2008). Um mapeamento sistemático é recomendado para áreas de pesquisa que possuem uma falta de estudos primários relevantes e de boa qualidade (Kitchenham e Charters, 2007), permitindo uma análise geral do estado de publicação na área, ajudando desta forma na tomada de decisão quanto a validade de se fazer um estudo mais profundo na área, ou se ainda é necessário que mais estudos primários sejam publicados para que isto se torne viável (Budgen *et al.*, 2008).

Neste trabalho, o mapeamento sistemático foi realizado visando ampliar a cobertura da fundamentação teórica, com o objetivo de identificar estudos que abordassem as técnicas, práticas e ferramentas utilizadas no desenvolvimento de software em *startups*. O mapeamento sistemático conduzido foi formado pelas seguintes etapas: definição da questão de pesquisa, definição das estratégias de pesquisa, definição dos critérios e procedimentos de seleção e extração dos dados.

1.4.2.1.3 *Survey* Exploratório

O *survey* é um tipo específico de estudo que envolve a coleta de dados de uma amostra de elementos obtidos a partir de uma população bem definida por meio da aplicação de um questionário (Visser *et al.*, 2000). De acordo com Travassos *et al.* (2002), o objetivo do *survey* é descrever, explanar e explorar informações preliminares, quantitativas ou qualitativas, por meio do questionário, além de levantar as variáveis do estudo a serem avaliadas.

Neste trabalho, o *survey* será conduzido com o objetivo de avaliar perspectivas, baseado nas informações coletadas por meio do mapeamento sistemático. Sendo assim, o *survey* pode ser classificado como exploratório, pois visa analisar uma área de pesquisa pouco explorada. Em relação à forma de coleta de dados, caracteriza-se como corte transversal, pois a coleta ocorrerá em apenas um dado momento (Sampieri *et al.*, 2010). A população buscada para a aplicação do *survey* serão quaisquer *startups* que possuam o desenvolvimento de software como atividade principal ou como uma de suas atividades chave.

Este trabalho foi desenvolvido em parceria com o Prof. Dr. Ivaldir Farias Junior da Universidade de Pernambuco, este que possui papel importante na divulgação do *survey*. O questionário utilizado para o *survey* pode ser verificado no Apêndice C.

1.4.2.2 Desenvolvimento

A fase de desenvolvimento será realizada com base no conhecimento e nas lacunas identificadas no estudo exploratório. Esta fase envolve a definição das diretrizes.

1.4.2.2.1 Definição das Diretrizes

A definição das diretrizes será realizada com base no conhecimento levantado por meio da fundamentação teórica, do mapeamento sistemático e do *survey* realizados durante a fase exploratória, visto que estas etapas permitirão verificar e analisar quais práticas estão relacionadas com o aumento das chances de sucesso de uma *startup*.

A definição visa sistematizar a confecção das diretrizes e será realizada por meio das seguintes etapas:

- Definição do formato de especificação: busca estabelecer um formato padrão para a estruturação e especificação das diretrizes, visando caracterizar os elementos que as compõem e assim facilitar o seu entendimento.
- Definição do objetivo das diretrizes: busca estabelecer o propósito das diretrizes, ou seja, identificar o que se pretende alcançar por meio das mesmas.
- Detalhamento das diretrizes: visa detalhar, de acordo com o formato de especificação definido, as diretrizes que serão propostas.

1.4.2.3 Refinamento

A fase de refinamento terá como objetivo analisar as perspectivas propostas para a confecção das diretrizes, visando avaliar a sua aplicabilidade por meio do *feedback* de atores da indústria e, com isto, identificar oportunidades de melhoria e os benefícios que podem ser alcançados. Esta fase será dividida em três etapas que serão realizadas em paralelo: aplicação um *survey* confirmatório, condução de um grupo focal e aplicação de um painel com especialistas, descritos a seguir.

1.4.2.3.1 *Survey* Confirmatório

O *survey* confirmatório envolve os mesmos processos do *survey* exploratório, sendo que neste o objetivo é de avaliar as perspectivas coletadas por meio do primeiro. Sendo assim, este questionário pode ser classificado como de avaliação (Wohlin e Aurum, 2015), visto que busca avaliar as diretrizes propostas sob a perspectiva das *startups*.

1.4.2.3.2 Grupo Focal

O grupo focal é um tipo de entrevista em profundidade realizada em grupo onde as reuniões apresentam características definidas em relação a proposta, tamanho, composição e procedimentos de condução. Entre as características gerais de um grupo focal estão o envolvimento de pessoas, a realização de reuniões, homogeneidade dos participantes de acordo com os aspectos de interesse da pesquisa, geração de dados, natureza qualitativa e, principalmente, discussão foca em um tópico, de acordo com o propósito da pesquisa. O objeto de análise é a interação dentro do grupo e seu uso é apropriado quando o objetivo é entender como as pessoas consideram uma experiência, ideia ou evento (Oliveira e Freitas, 1998).

1.4.2.3.3 Painel com Especialistas

A aplicação do painel envolve uma série de entrevistas com especialistas da área de pesquisa, trazendo, desta forma, benefícios para a pesquisa dado a sua natureza aplicada. A aplicação de um painel com especialistas tem como objetivo obter uma interpretação das evidências, sendo assim, a aplicação do painel não é uma evidência *per se* (Dybå e Dingsøyr, 2008).

Combinar o conhecimento e o julgamento de um determinado número de especialistas aumentam as chances de que a pesquisa se aproxime da verdade. Além disto, ao obter a visão dos atores que estão diretamente ligados à área, esta técnica permite um melhor entendimento dos fenômenos envolvidos (Conboy e Fitzgerald, 2010).

1.5 Organização do Trabalho

O restante deste trabalho está organizado da seguinte forma: o Capítulo 2 apresenta o referencial teórico sobre métodos ágeis, *Lean Software Development*, *startups* e desenvolvimento de software em *startups* e também os trabalhos relacionados. O Capítulo 3 destaca o Mapeamento Sistemático a respeito das técnicas, práticas e ferramentas utilizadas no desenvolvimento de software em *startups*, que busca subsidiar os demais estágios da

pesquisa. O Capítulo 4 apresenta o estudo empírico realizado por meio de um *survey*, trazendo o seu planejamento, execução e resultados. O Capítulo 5 destaca a proposta das diretrizes, apresentando sua especificação e representação. O Capítulo 6 apresenta as etapas de validação e refinamento das diretrizes propostas. Por fim, o Capítulo 7 traz a conclusão do trabalho e apresenta algumas sugestões de trabalhos futuros.

2

Referencial Teórico

2.1 Considerações Iniciais

Neste capítulo é apresentada a fundamentação teórica sobre os temas que subsidiaram o desenvolvimento deste trabalho: *Startups* e Desenvolvimento de Software em *Startups*. Também são apresentados neste capítulo os trabalhos similares.

2.2 Startups

Startups são organizações recém fundadas com pouco ou nenhum histórico de operação e que estão enfrentando mercados e tecnologias instáveis (Giardino *et al.*, 2014). Estas organizações buscam por um modelo de negócio, por meio da oferta de um produto ou serviço, que possa ser escalável, o que faz com que as mesmas geralmente necessitem de investimentos externos para financiar suas operações (Sutton, 2000). Esta busca por um produto que possua um crescimento exponencial e a necessidade de investimentos são as principais diferenças entre *startups* e outras pequenas organizações. Ao contrário de organizações bem estabelecidas que, independente do seu tamanho, estão focadas em otimizar um modelo de negócio existente, *startups* estão focadas em encontrar um (Ries, 2011).

O tempo de vida de uma *startup* se inicia a partir da concepção da ideia até que a mesma atinja a maturidade (Paternoster *et al.*, 2014). Crowne (2002) descreve o ciclo de vida de uma *startup* por meio de uma evolução em três fases: inicialização, estabilização e crescimento. Crowne (2002) define a fase de inicialização como a fase entre a concepção

do produto até a realização da primeira venda. Nesta fase, o empreendedor precisa identificar uma oportunidade de mercado e então, por meio da tecnologia disponível, determinar uma maneira de aproveitá-la (Giardino *et al.*, 2014; Nguyen-Duc *et al.*, 2016). A fase de estabilização começa no momento em que o primeiro consumidor recebe o produto, durando até o momento em que o produto está estável o suficiente para ser vendido a um novo consumidor sem causar sobrecarga no desenvolvimento do produto (Nguyen-Duc *et al.*, 2016). É nesta fase que o empreendedor, geralmente, precisa buscar por investidores externos para financiar suas operações. Então, a fase de crescimento se inicia exatamente ao fim da fase de estabilização e acaba quando as taxas de crescimento e fatia de mercado estão estáveis e todos os processos necessários para o desenvolvimento e venda do produto estão estabilizados (Klepper, 1996). Ao fim destas fases a *startup* se torna uma organização madura.

Em um estudo a respeito dos desafios que *startups* em fase inicial enfrentam, Giardino *et al.* (2015) sumarizaram as dez dificuldades mais citadas entre as *startups* investigadas, sendo elas: prosperar em meio a incerteza tecnológica, adquirir os primeiros consumidores, adquirir fundo inicial, montar o time empreendedor, entregar valor ao cliente, gerenciar múltiplas tarefas, definir um MVP, almejar um nicho de mercado, se manter focado e disciplinado e atingir o ponto de equilíbrio, que seria balancear os ganhos e perdas de forma que seja possível se manter trabalhando no projeto. Giardino *et al.* (2015) apontam que a possível causa para os desafios encontrados na pesquisa podem estar relacionados a falta de um processo de aprendizagem, visto que, ao adentrar o mercado sem o conhecimento necessário pode resultar em um desalinhamento entre o modelo de negócio e as atividades de desenvolvimento.

Em relação as organizações estabelecidas, *startups* apresentam algumas desvantagens como, mais problemas de comunicação e coordenação, além da falta de produtos, parceiros e consumidores estabelecidos (Sutton, 2000). Além disto, há ainda os desafios descritos por Sutton (2000):

- Imaturidade: uma das características mais básicas de uma *startup* é ser uma organização relativamente nova, ou no mínimo sem experiência em comparação com organizações mais estabelecidas. Assim, essas organizações possuem uma história curta ou pouca experiência.

- Recursos limitados: outra característica marcante de *startups* é a falta de recursos. Os primeiros recursos, geralmente, são focados no lançamento e promoção do produto, e em construir alianças estratégicas. Cumprir estas atividades são importantes para a sobrevivência da *startup*.

- Múltiplas influências: no início *startups* podem ser influenciadas por diversos fatores, como, por exemplo, investidores, clientes, parceiros e competidores. Influências divergentes podem inclusive partir de dentro da própria organização. Isto obriga a organização a estar constantemente ajustando e reajustando o que ela faz e como ela faz.

- Tecnologias e mercados dinâmicos: essas novas organizações podem se aproveitar do surgimento de novas tecnologias, como, por exemplo, um aumento na variedade de dispositivos de comunicação, novos canais de comunicação, novas arquiteturas de sistemas, entre outros. *Startups* muitas vezes são estabelecidas no desenvolvimento de produtos inovadores[1] e, para desenvolvê-los necessitam de ferramentas e técnicas de desenvolvimento de ponta.

Estes desafios ajudam a compreender as dificuldades que uma *startup* enfrenta em comparação a uma organização bem estabelecida. É claro que, em algum nível, organizações estabelecidas também se deparam com estes mesmos desafios, porém, no caso das *startups* isto é levado a um nível extremo (Sutton, 2000).

Devido ao surgimento de novos mercados, tecnologias acessíveis e ao capital de risco o surgimento de novas *startups* tem aumentado dia após dia (Smagalla, 2004), aumentando também o papel das *startups* na criação de novos empregos, sendo estas um fator importante para a economia (Giardino *et al.*, 2014; Unterkalmsteiner *et al.*, 2016). Com a Internet se tornando ubíqua e a onipresença dos *smartphones*, somado ao fácil acesso a potenciais mercados e a redução dos custos dos serviços de distribuição, resulta em condições chamativas ao empreendedor (Marmer *et al.*, 2012). Do ponto de vista da inovação, *startups* abrem as portas para a introdução de novas e disruptivas inovações. A exemplo disto, pode-se mencionar o Facebook, que mudou a forma de se consumir conteúdo na Internet e, junto ao Google, domina o mercado de anúncios digitais[2], o Spotify, que oferece um novo modo de ouvir músicas e, o Instagram, que apresentou um novo modo de compartilhar fotos.

Facebook, Instagram, LinkedIn e Spotify são alguns exemplos de *startups* que conseguiram se tornar um negócio de sucesso, porém, muitas destas organizações deixam de existir antes de atingir o sucesso (Crowne, 2002). Consta que 60% destas organizações não sobrevivem durante os cinco primeiros anos e 75% das que possuem investimentos externos falham (Nobel, 2011). Ainda, mais de 90% falham principalmente por autodestruição em

[1] Produtos que buscam resolver problemas simples ou complexos do dia a dia por meio de abordagens pioneiras.

[2] `http://br.reuters.com/article/internetNews/idBRKBN1AD2HF-OBRIN`. Acesso em: 30 de jul. de 2017.

vez de competição (Marmer *et al.*, 2012). Crowne (2002) aponta que a maioria destas organizações não conseguem atingir dois anos de vida. Segundo Giardino *et al.* (2014), isto se deve ao alto risco que *startups* enfrentam, oportunidades perdidas e outras razões relacionadas a negócios. Fatores de mercado e problemas financeiros, também, estão entre os motivos (Klotins *et al.*, 2015).

Entretanto, a investigação dos motivos que levam as *startups* a falharem ainda é escassa (Paternoster *et al.*, 2014) e, principalmente, o impacto que a falta de práticas e metodologias de engenharia representam neste índice ainda é desconhecido devido ao prematuro estado de pesquisa na área (Giardino *et al.*, 2014; Klotins *et al.*, 2015). Conforme levantado por Paternoster *et al.* (2014), um estudo de 1994 realizado por Carmel (1994) em *startups*, apontava para a necessidade de mais pesquisas quanto as práticas de desenvolvimento de software, com o intuito de replicar tais práticas e então transferi-las para outros setores da indústria de tecnologia. Porém, apesar do desenvolvimento de software ser o centro das atividades de uma *startup*, este não possui uma base de conhecimento cientificamente suportada.

Paternoster *et al.* (2014) apresentam uma análise da literatura de desenvolvimento de software em *startups* e identificam as práticas de trabalho realizadas nestas organizações. Os autores concluiram que os estudos encontrados não eram suficientes para entender o desenvolvimento de software nas *startups*, sendo que até 2014, ano do estudo, a lacuna apresentada por Sutton (2000) há mais de uma década, onde este relatou a falta de estudos voltados para as *startups*, foi apenas parcialmente preenchida. Em 2015, Klotins *et al.* (2015) buscaram aprofundar o estudo das práticas de trabalho relacionadas as *startups* por meio de um mapeamento sistemático com o objetivo de identificar e classificar as áreas de conhecimento de Engenharia de Software dentro destas organizações, porém, mais uma vez os estudos encontrados foram insuficientes, mantendo desta forma em aberto a lacuna apontada por Sutton (2000).

2.2.1 Desenvolvimento de Software em Startups

A implementação de metodologias para estruturar e controlar as atividades de desenvolvimento de software em *startups* é um dos principais desafios da Engenharia de Software (Coleman e O'Connor, 2008). *Startups* são flexíveis por natureza e, devido a isto, relutantes em adotar processos e medidas burocráticas (Sutton, 2000). Com recursos limitados, *startups* normalmente preferem investir no desenvolvimento do produto em vez da definição de um processo (Coleman e O'Connor, 2008). As metodologias ágeis estão entre as práticas mais encontradas em *startups*, com destaque para as metodologias

Lean/Lean Startup, Scrum, *Extreme Programming* (XP), *Experiment Driven Development* (EDD), *Feature Driven Development* (FDD) e Scrumban. No caso de algumas *startups*, é possível identificar que várias das práticas de trabalho são derivadas da metodologia XP, entretanto não se pode dizer que essas organizações de fato seguem a metodologia já que a programação em pares não é adotada (Martin *et al.*, 2007; Tingling e Saeed, 2007). Coleman e O'Connor (2008) buscam identificar como é formado o processo de desenvolvimento de software dentro de *startups*, tendo-se como principal influência as experiências anteriores daqueles que são os encarregados do trabalho de desenvolvimento.

O período de tempo entre a concepção do produto até a sua disponibilização para venda (*time-to-market*) é, de acordo com os pesquisadores, um dos pontos chave para a operação de uma *startup* (Och Dag, 2002; Sawyer *et al.*, 1999), visto que, pela falta de recursos e devido as incertezas do mercado, deve-se tentar entregar o produto de forma mais rápida possível. Estas características apontam para o que é conhecido na literatura como *market-driven development*, em tradução literal, desenvolvimento orientado ao mercado (Alves *et al.*, 2006). A partir disto surgem alguns problemas relacionados aos requisitos do produto, sendo eles: requisitos sendo inventados pela própria *startup* (Potts, 1995), mal documentados (Karlsson *et al.*, 2002) e sendo validados apenas após o lançamento do produto no mercado (Keil e Carmel, 1995). Desta forma, a falha de um produto está normalmente relacionada ao mesmo não atender as necessidades do clientes (Alves *et al.*, 2006).

Devido ao dinamismo do ambiente em que as *startups* estão inseridas, é natural que as metodologias ágeis sejam o tipo de processo mais viável. Entregar resultados rápidos por meio de um processo iterativo e incremental permite uma diminuição do *time-to-market* (Giardino *et al.*, 2014). Alinhada as metodologias ágeis se encontra a metodologia *lean*, que prega uma aprendizagem rápida ao identificar as principais partes do negócio e desenvolver um MVP a partir disto, permitindo que seja possível testar e modificar rapidamente o produto de acordo com o *feedback* recebido (Giardino *et al.*, 2014). Duc e Abrahamsson (2016) conduzem uma investigação a respeito do papel do MVP em *startups*, do qual são apresentados alguns dos principais tipos de MVP utilizados neste contexto, sendo eles:

- Wizard of Oz MVP: interface de usuário que é igual a um produto funcional, porém o processo é feito de maneira manual, onde o objetivo é demonstrar todo o trabalho feito pelo produto.

- Concierge MVP: serviço manual que consiste exatamente dos mesmos passos que o usuário executaria com o produto.

- Piecemeal MVP: similar ao Wizard of OZ, porém aqui a execução das tarefas é feita por ferramentas já existentes.
- Mockup MVP: representação da interface do produto, sem conter qualquer funcionalidade.
- Single feature MVP: protótipo que implementa apenas a funcionalidade mais importante do produto.

O estudo observou que as técnicas de MVP são úteis, pois os artefatos gerados servem como limites do que pode ser realizado e também podem ser reutilizados em outros estágios do projeto.

Giardino *et al.* (2014) apontam que para ter sucesso, *startups* precisam de métodos transferíveis e confiáveis, ou seja, que são fáceis de colocar em prática e que tragam resultados rápidos. A natureza de incerteza que compõe uma *startup* exige que os processos sejam desenvolvidos com velocidade, assim, as atividades devem ser desenhadas para permitir flexibilidade e reatividade. Inclusive, diante deste contexto, é preferível que se falhe rápido e consequentemente se aprenda rápido. A partir disto, Giardino *et al.* (2014) citam alguns pontos que devem ser atendidos para permitir esta velocidade, e consequentemente o sucesso, às *startups*:

- Uso de *frameworks* conhecidos, permitindo mudanças rápidas no produto para atender ao mercado.
- Uso de prototipagem evolucionária e experimentação por meio de componentes existentes.
- Validação contínua com grupos de usuários chave.
- Entrega de valor contínua focando em funcionalidades chaves que engajem o usuário.
- Empoderamento do time como forma de aumentar o desempenho e o sucesso.
- Utilização de métricas para aprender por meio dos usuários.
- Uso de ferramentas fáceis de implementar para tornar o desenvolvimento do produto mais dinâmico.

O número de estudos no âmbito do desenvolvimento de software em *startups* que apresentam resultados transferíveis para a indústria ainda é baixo. Dentre os itens levantados por Giardino *et al.* (2014), todos apresentam lacunas e se beneficiariam de

mais estudos empíricos. São necessários estudos que busquem soluções para atender a estes itens, a fim de criar cenários onde técnicas, práticas e ferramentas possam ser validadas e, possivelmente a partir dos dados levantados, criarem guias de métodos confiáveis que podem ser adotados. Há, também, a necessidade de estudos comparativos, que busquem por exemplo avaliar os *frameworks* de desenvolvimento mais populares do mercado, e verificar sua validade para a aplicação no contexto de uma *startup*.

O mesmo pode ser dito para os demais pontos levantados por Giardino *et al.* (2014), mas principalmente quanto a questão da capacitação da equipe, já que este é um dos pontos menos explorados entre os citados. Apenas o estudo desenvolvido por Coleman e O'Connor (2008) aborda esta questão, onde a abordagem de gerenciamento "incluir e capacitar" se mostra interessante pois proporciona liberdade ao time de desenvolvimento, que necessita de menos supervisão direta, indo ao encontro dos benefícios apontados por Giardino *et al.* (2014).

Outro ponto importante para a manutenção e o desenvolvimento das *startups* de software é a existência de ecossistemas, tema este que vem ganhando atenção e sendo estudado recentemente (Cukier, 2017; Cukier *et al.*, 2015a,b; Kon *et al.*, 2014, 2015; Kon e Lyons, 2016; Santos, 2016). Estes estudos apontam que os ecossistemas criam um ambiente favorável à prosperidade destas organizações, uma vez que permite uma maior interação entre estas *startups*, permitindo que se organizem de forma conjunta. Além disto, o agrupamento destas organizações atrai a atenção de potenciais investidores e de mão de obra qualificada.

2.3 Trabalhos Similares

2.3.1 Diretrizes

As diretrizes são um conjunto de sugestões e melhores práticas que tem por ojetivo melhorar e orientar a execução de uma determinada rotina (Sjoberg *et al.*, 2007). Para atingir seus objetivos, essas diretrizes devem ser relatadas de forma a atender as necessidades da indústria e da academia (Kitchenham *et al.*, 2002).

As diretrizes apresentadas por Kitchenham *et al.* (2002) são amplamente citadas e são um importante guia sobre como realizar estudos empíricos em Engenharia de Software. Kitchenham *et al.* (2002) estrutura suas diretrizes por meio de seis tópicos, apresentando em cada um deles o que deve e o que não deve ser feito. A partir disto, cada um destes tópicos é apresentado por meio de uma breve introdução onde são apresentados os elementos e os objetivos do tópico em questão. Após esta introdução as diretrizes são

descritas, sendo indicadas por meio de um marcador (e.g., A1, A2, etc.) e de um texto curto em modo imperativo (e.g., se certifique de especificar...). Em seguida os autores detalham e fundamentam as diretrizes em questão por meio das suas crenças e por meio da literatura disponível. Apesar de possuírem uma estrutura simples e pouca formalização, as diretrizes propostas por Kitchenham *et al.* (2002) são claras e objetivas, permitindo que as sugestões sejam facilmente seguidas por seus leitores.

Steinmacher *et al.* (2018) apresentam um conjunto de diretrizes para auxiliar no processo de entrada de recém-chegados em projetos *open source*. Os autores distinguem as diretrizes em três categorias e, dentro dessas categorias, as diretrizes são descritas sem o uso de qualquer formalismo, sendo que a ideia geral da diretriz é destacada por meio de texto em negrito, seguida pela sua fundamentação com base nas evidências adquiridas pelos autores. As separações entre as diretrizes podem ser identificadas por meio do formato do texto.

Os trabalhos de Jorgensen (2005) e de Fischer *et al.* (2009) apresetam suas diretrizes de forma similar. O primeiro apresenta diretrizes sobre a estimativa de esforço em software baseado no julgamento de especialistas, enquanto os segundos apresentam diretrizes para o apoio de especialistas de domínio no desenvolvimento de software. Em ambos os trabalhos as diretrizes são apresentadas por meio de um título que sintetiza a ideia da diretriz em questão. Em seguida, sem qualquer formalismo, são apresentadas a fundamentação e o debate em relação aquela diretriz.

Tanveer (2017) apresenta um conjunto de diretrizes para a análise de impactos de mudanças quando estimando o esforço no desenvolvimento de software ágil. As diretrizes são divididas em dois tipos, seleção e, integração e aplicação. As diretrizes são nomeadas em ordem numérica (e.g., Diretriz 1, Diretriz 2, etc.) e são apresentadas por meio de um título resumido em negrito que especifica a ideia geral da diretriz. Em seguida, em poucas linhas e em modo imperativo, é apresentado o detalhamento de cada uma das diretrizes. Além das diretrizes, o autor apresenta o processo realizado ao seguir um conjunto de diretrizes (e.g., processo seguindo as diretrizes 1 e 2, processo seguindo as diretrizes 3 e 4), fornecendo algo próximo de um passo a passo de como seguir as diretrizes apresentadas.

O trabalho desenvolvido por van Gompel *et al.* (2016) apresenta um conjunto de diretrizes para qualidade de software. Este é o que possui uma maior formalização na apresentação das diretrizes entre os trabalhos relacionados. As diretrizes são apresentadas de acordo com quatro cenários diferentes, sendo cada uma nomeada por um número de configuração seguido do cenário na qual a mesma está localizada (e.g., Configuração 4: Software Experimental Não Suportado). Portanto, a diretriz é composta por itens

numerados que trazem as sugestões a serem seguidas. Alguns destes itens contêm outros itens que os compõem, sendo que estes itens são marcados alfabeticamente.

As diretrizes se repetem em todos os cenários, porém aquelas que não se aplicam ao cenário em questão possuem cor de fonte em cinza. As diretrizes são detalhadas com o que deve ser feito, sendo que a parte principal da diretriz é destacada em negrito. Além disto, dentro dos itens existem marcações (e.g., ID2, AC3, etc.) que apontam para critérios de avaliação de qualidade que complementam as diretrizes em questão. Estes critérios estão escritos em formato de pergunta e resposta, onde cada um deles apresenta uma questão relacionada a qualidade, que é então respondida com o que deve e o que não deve ser feito. Alguns destes critérios também possuem marcações que apontam para outros critérios relacionados. Os critérios estão divididos em dois grupos: i) usabilidade; ii) sustentabilidade e manutenção, que por sua vez também possuem divisões, como por exemplo, compreensibilidade, instalabilidade, etc. dentro de usabilidade e testabilidade, portabilidade, etc. dentro de sustentabilidade e manutenção.

As diretrizes são um recurso comumente utilizado em Engenharia de Software, porém, é possível verificar que não há um método definido ou formato específico para a sua apresentação, visto que em geral as diretrizes são apresentadas por meio de um objetivo seguido de uma descrição/fundamentação onde a mesma é detalhada.

2.3.2 Lições Aprendidas

Pompermaier *et al.* (2017) apresentam uma série de entrevistas e observações a partir de *startups* localizadas em um parque tecnológico buscando entender os fatores que influenciam positivamente e negativamente no desenvolvimento de software dessas *startups*, além de apresentar algumas soluções. De acordo com os autores, a escolha de *startups* que fazem parte de um parque tecnológico se deve ao fato de que essas organizações recebem as mesmas oportunidades, enriquecendo a análise do estudo.

Pompermaier *et al.* (2017) conduziram a coleta de dados por meio de entrevistas semiestruturadas e observações de campo. O estudo foi conduzido com 8 *startups* de software, onde as entrevistas foram direcionadas aos fundadores das organizações de modo a entender a forma como a organização começou e quais as decisões técnicas adotadas em relação ao desenvolvimento do produto de software. De acordo com os achados do estudo, o tempo médio de interação com o parque tecnológico foi de dois anos e meio, sendo que durante este tempo 62,5% das *startups* não possuiam um MVP, demonstrando que a maioria das organizações iniciam suas atividades dentro do parque com o objetivo de desenvolver um produto mínimo viável.

O estudo segue ao comparar as respostas das entrevistas com as áreas de conhecimento do SWEBOK (Bourque e Fairley, 2014) e identifica os pontos críticos do desenvolvimento de software, sendo eles: requisitos de software, estrutura e arquitetura de software e testes de software.

Em relação aos requisitos de software, o estudo aponta que 62,5% das *startups* entrevistadas fazem uso de um método pseudo ágil para gerenciar os requisitos do MVP. Uma ferramenta de gerenciamento visual, como Kanban, e algumas práticas de Scrum são utilizadas, sendo que, basicamente, o gerenciamento é realizado por meio de *post-its* em uma parede, que indica os requisitos a serem feitos, aqueles que estão em andamento e aqueles que já foram concluídos. Além disto, todos os entrevistados disseram que muitos dos requisitos não são gerenciados e/ou documentados de forma alguma, sendo simplesmente passados verbalmente pelo usuário chave ou pelo líder da *startup*. Quando questionados sobre o motivo de utilizar algumas práticas porém não outras, os entrevistados disseram que o tempo deve ser utilizado para o desenvolvimento do produto (codificação) e assim evitar que o processo de desenvolvimento se torne burocrático. Com base nestas informações, os autores listam os seguintes achados: i) a adoção de técnicas de gerenciamento de requisitos leva a um aumento significativo no tempo de desenvolvimento; ii) a não documentação e/ou gerenciamento de requisitos não interfere na qualidade do MVP desenvolvido.

A respeito da estrutura e arquitetura de software, os entrevistados apontam que melhorias nessa área são necessárias, visto que o desenvolvimento é realizado sem o planejamento adequado, sem a utilização de estruturas e padrões de código que permitem o crescimento do produto, seja em número de funcionalidades ou de usuários. Linguagem de programação, base de dados, servidor de hospedagem, ferramenta de implantação de código e abordagem de segurança são os itens relacionados a arquitetura e estrutura que apresentam problemas quando negligenciados de acordo com os entrevistados. Em relação a esta área de conhecimento os autores concluem que: a não definição ou uma definição mal feita da arquitetura e estrutura do software na fase inicial de desenvolvimento aumenta exponencialmente a dívida técnica da organização.

Por fim, em relação aos testes de software, as entrevistas demonstraram que não são usadas técnicas de teste de software durante a construção da primeira versão do produto, porém este cenário se altera durante as fases seguintes, onde 75% dos entrevistados passam a utilizar alguma técnica de teste. Entre as técnicas de testes utililizadas estão o uso de um cliente piloto, testes unitários, testes *ad hoc* funcionais e o uso de um especialistas em testes. A partir disto, os autores concluem que: estruturar ou formalizar uma etapa de teste de software irá aumentar a aceitação de mercado dos requisitos desenvolvidos.

Em outro trabalho, Shah (2006) aponta para o desafio de se contratar os melhores desenvolvedores possíveis e para o risco de perder um destes desenvolvedores após expô-lo ao produto por um longo período de tempo até o mesmo se tornar proficiente neste produto, visto que o custo de substituição deste desenvolvedor será muito alto. Em uma experiência com uma empresa fundada por si, o autor relata a respeito do sucesso da organização que pode ser atribuído ao fato de que, nos primeiros sete anos de operação, não perdeu nenhum dos seus desenvolvedores chave. Isto não só resultou em economia com contratação de pessoal como também para a manutenção de um produto mais elegante e manutenível.

Shah (2006) descreve a respeito do processo de terceirização do processo de desenvolvimento, onde o mesmo relata que apesar dos benefícios dos pontos de vista de redução de custos e de disponibilidade de talento, a terceirização das atividades chave do desenvolvimento não faz sentido para *startups* de software que estão iniciando suas atividades. Segundo o autor, o projeto de desenvolvimento é de difícil gerenciamento mesmo quando existe uma proximidade entre os envolvidos, ao terceirizar este processo a coordenação se torna ainda mais difícil, diminuindo as chances de sucesso do produto. Além disto, manter um excelente desenvolvedor por meio de um processo de terceirização é altamente improvável devido a natureza de contratos em que estas terceirizações ocorrem. O autor acrescente que esta visão é compartilhada com todos os empreendedores que foram entrevistados para a execução do trabalho.

A respeito do uso de software *open source*, Shah (2006) relata que, 65% dos empreendedores que responderam um questionário disponibilizado pelo mesmo, indicaram fazer uso de algum tipo de código *open source* como parte do desenvolvimento dos seus produtos. O nível de aceitação pela indústria quanto ao uso de software *open source* tem aumentado, sendo que o maior impactor pode ser visto na camada de infraestrutura, consistindo em sistemas operacionais, bases de dados e ferramentas de desenvolvimento. Um dos principais benefícios do uso de software *open source* da perspectiva de uma *startup* é a redução de custos. Ao utilizar estes produtos uma *startup* pode evitar os custos de aquisição de ferramentas e tecnologias necessárias para o desenvolvimento dos seus produtos. O autor exemplifica isto com a *stack* LAMP, que consiste do Linux como sistema operacional, do Apache como servidor *web*, do MySQL como base de dados e do PHP como linguagem de programação. Em experiência com sua *startup*, o autor relata que o total de economia calculada com o uso de software *open source* é de aproximadamente $250.000,00 em custos de desenvolvimento, além de cerca de uma redução de 6 a 9 meses no *time-to-market*.

Shah (2006) por fim sumariza alguns dos pontos que considera importantes para o sucesso de uma *startup*, sendo: i) forte experiência em desenvolvimento de software de pelo menos um dos membros fundadores; ii) colocar o produto no mercado, mesmo que imperfeito, o mais rápido possível; iii) entendimento e uso das ferramentas básicas para um desenvolvimento eficiente por parte do time de desenvolvimento, como *builds* automáticas, *bug* e *feature tracking*, testes automáticos, etc.; iv) habilidade de encontrar e recrutar excelentes desenvolvedores por parte do líder de desenvolvimento; v) conhecimento das ferramentas *open source* disponíveis que podem se encaixar no desenvolvimento do produto por parte do líder de desenvolvimento.

2.3.3 Modelos

No trabalho de Kaysen *et al.* (2016), os autores desenvolvem um modelo a partir das experiências em um estudo de caso desenvolvendo uma solução de software. Além disto, o modelo possui influência de experiências anteriores dos autores, trabalhando principalmente com a metodologia *Lean Startup* (Ries, 2011) e com a sua extensão, o modelo *Early Stage Software Startup Development Model* (ESSSDM) (Bosch *et al.*, 2013). Em uma destas experiências, os autores descrevem que o uso de práticas de XP em conjunto com a metodologia *Lean Startup* obteve benefícios durante o desenvolvimento de software. Os autores incluíram as práticas que consideram se encaixar dentro das necessidades de desenvolvimento dos seus MVPs, sendo elas: i) programação em pares, melhorando a qualidade do código; ii) propriedade coletiva do código, possibilitando que todos da equipe colaborassem, evitando sobrecarga em apenas um membro do time; iii) padrões de código, aumentando a legibilidade e a compreensibilidade do código; iv) testes unitários, assegurando que novas *features* não quebrassem as existentes.

Em relação aos testes unitários, a metodologia XP recomenda o uso de *Test Driven Development* (TDD) para a sua realização, porém, os autores relatam que o uso de TDD aumentou o tempo de desenvolvimento em cerca de 15%-16%, indo na contramão da noção de velocidade e mínimo esforço pregada pelo MVP na metodologia *Lean Startup*. Em adição aos conceitos de XP, os autores também utilizaram os princípios de projeto de quase decomposição propostos por Sarasvathy (2003). Estes princípios permitiram a aceitação de mudanças rápidas, uma vez que os componentes são decompostos de forma que se torne possível a troca de um componente por um novo, proporcionando uma funcionalidade extentida ou totalmente diferente.

O modelo proposto pelos autores é denominado *Entrepreneurial Software Innovation* (ESI), e seus conceitos são formados com base nas ideias da metodologia *Essence*, proposta

por Aaen (2015), e nos conceitos da metodologia *Lean Startup* e do modelo ESSSDM. O modelo define um conceito chamado *Define-Build-Evaluate* (DBE), que por sua vez é inspirado no conceito BML do *Lean Startup* e baseado no modelo *Entry Criteria, Task, Validation and eXit Criteria* (ETVX) contido na metodologia *Essence*. Por meio do DBE é possível desenvolver ideias, protótipos e MVPs. Nas duas iterações realizadas no estudo de caso utilizando o modelo ESI os resultados foram positivos, sendo que na avaliação da empresa do estudo de caso, o software conseguiu atender o problema de acordo com os requisitos especificados, além de ser estável e suportar uma grande quantidade de usuários.

O modelo ESSSDM foi proposto como resposta para os desafios encontrados por Bosch *et al.* (2013) por meio de entrevistas com profissionais de *startups*. De acordo com os achados, apesar de familiaziados com práticas ágeis em seu desenvolvimento de software, as *startups* investigadas encontraram dificuldades para a implementação da metodologia *Lean Startup*. O modelo é formado a partir de uma extensão de princípios da metodologia *Lean Startup*, dos resultados das entrevistas conduzidas pelos autores e das experiências anteriores dos autores trabalhando com *startups* de software. Diferentemente da metodologia *Lean Startup*, o ESSSDM permite a investigação de múltiplas ideias de produto em paralelo.

O processo de avaliação do ESSSDM ocorreu por meio de um projeto de uma *startup*, onde dois dos autores eram cofundadores, além de entrevistas com profissionais da indústria. O projeto foi realizado durante um período de oito meses em uma incubadora, sendo providenciados à equipe o financiamento inicial, escritório, profissionais com experiência, consultores de negócio e peritos legais. O objeto da *startup* era encontrar um produto promissor dentro do segmento de pequenas empresas. Ao fim das análises, os autores concluiram que o modelo proporciona suporte operacional para a implementação dos princípios do *Lean Startup*, tanto nas etapas de planejamento quanto nas etapas de execução.

2.3.4 Qualidade de Software

Por fim, Pohja (2016) apresenta em seu trabalho uma investigação a respeito do controle de qualidade no desenvolvimento de software. O autor destaca que em um ambiente de *startup* é particularmente importante assegurar a qualidade do software, de modo a possibilitar a continuidade do desenvolvimento do produto. Em comparação com organizações estabelecidas, *startups* não podem arcar com os custos de muitos erros durante o desenvolvimento do produto. A partir disto o autor lista o que seria as fundações da alta qualidade no desenvolvimento de software: código limpo e integridade.

O código é, naturalmente, fundamental em qualquer produto de software, desta forma, manter um código limpo é essencial para o desenvolvimento do projeto, uma vez que um código confuso irá resultar em uma desaceleração do processo de desenvolvimento e consequentemente em uma diminuição do *time-to-market*. Quanto a integridade, esta pode ser percebida por meio de dois tipos diferentes: integridade percebida e integridade conceitual. A integridade percebida é um equilíbrio entre função, usabilidade, confiabilidade e economia observado pelo cliente e é afetada pela experiência completa do sistema, desde a entrega até a habilidade do mesmo resolver o problema. A integridade conceitual está ligada a suavidade e uniformidade dos conceitos centrais de todo o sistema. A arquitetura do sistema deve possuir um equilíbrio entre flexibilidade, manutebilidade, eficiência e responsividade. A integridade conceitual é um pré-requisito para a integridade percebida.

Pohja (2016) apresenta um estudo de caso em uma *startup* de baixo orçamento onde apresenta as medidas de qualidade adotas para o projeto. Por ter sido realizado anteriormente a escrita do trabalho, sem a intenção de fazer parte do mesmo, o estudo de caso não segue os requisitos de um estudo científico adequado. O objetivo do projeto era de criar um novo tipo de sistema de gravação para jardins de infância, onde a principal funcionalidade do produto seria que as criadas poderiam marcar as presenças das crianças por meio de um dispositivo móvel. A partir disto, os seguintes métodos e processos fizeram parte da garantia de qualidade no projeto: i) desenvolvimento ágil, permitindo que a implementação de funcionalidades fossem avaliadas em relação ao entendimento do cliente; ii) análises estatísticas, visto que boa parte do desenvolvimento foi realizado por meio de ambiente de desenvolvimento integrados (IDEs), que por sua vez proporcionam uma análise automática e contínua das estatísticas do código fonte, permitindo que erros fossem identificados antes de serem inseridos na aplicação; iii) sessões de revisão em pares, sendo realizados em situações onde o desenvolvedor responsável pela implementação de um componente possuía dúvidas a respeito dos detalhes de implementação; iv) testes, incluindo testes de integração e testes de interfaces de usuário.

2.4 Considerações Finais

Este capítulo apresentou temas que irão subsidiar o restante do trabalho. O capítulo apresenta o que são *startups* e discorre a respeito do desenvolvimento de software nestas organizações. Sendo o principal objeto de estudo deste trabalho, é importante apresentar um referencial teórico consistente sobre as mesmas

É possível verificar que apesar de pesquisas na área de *startup* existirem há algum tempo, só agora existe um interesse maior na área, fazendo crescer o número de estudos publicados nos últimos dez anos. Outro ponto a ser notado, é que apesar das metodologias ágeis ser uma metodologia recorrentemente relacionada com *startups*, dificilmente essas organizações adotam o seu uso de forma total. É comum adotarem apenas algumas das práticas que a metodologia prega, adequando os métodos de forma a encaixar em suas rotinas de trabalho (Tegegne, 2018).

Em relação aos trabalhos similares, não foi possível encontrar nenhum estudo que tivesse uma proposta próxima a deste estudo, sendo assim, a similidade está mais voltada aos métodos e processos adotados em suas execuções do que nos resultados.

3

Mapeamento Sistemático da Literatura

3.1 Considerações Iniciais

Neste capítulo são apresentados os conceitos presentes em um mapeamento sistemático da literatura, os processos envolvidos em sua condução, como questão de pesquisa, estratégia de busca, estratégia de seleção, estratégia de extração, avaliação do protocolo de pesquisa, os resultados e a discussão dos dados, as limitações e ameaças à validade. O objetivo do mapeamento é subsidiar as próximas fases do trabalho, principalmente na definição das questões presentes no *survey* conduzido no Capítulo 4 e no desenvolvimento do Capítulo 5.

3.2 Planejamento

O mapeamento sistemático realizado neste trabalho busca identificar quais são as técnicas, práticas e ferramentas usadas no âmbito do desenvolvimento de software em *startups*. Para o autor, técnica se refere aquilo que possui um conjunto de instruções formais, ou seja, possuem métodos e processos próprios. Enquanto prática refere-se a uma ação costumeira, cotidiana de algo, isto é, está ligada à experiência adquirida, um modo particular de agir, e não a um conjunto de regras preestabelecido.

Identificar tais questões é importante tanto para os profissionais da área, ao fornecer por meio da Engenharia de Software Baseada em Evidência (ESBE) evidências suficientes para a transferência deste conhecimento para a indústria (Dyba *et al.*, 2005; Kitchenham *et al.*, 2004), quanto para os pesquisadores, que poderão identificar por meio do mapeamento

áreas de pesquisa que ainda são pouco exploradas ou áreas que carecem de uma análise mais profunda e que se beneficiariam de uma Revisão Sistemática de Literatura (RSL).

3.2.1 Avaliação do Protocolo

Devido a importância do protocolo para a realização do MSL, é necessário garantir que o mesmo seja adequado, utilizando para isto um procedimento de avaliação. Conforme sugerem as diretrizes propostas por Kitchenham e Charters (2007), esta avaliação deve ser conduzida por um grupo independente de pesquisadores. Especificamente, no estudo conduzido, foi desenvolvido um questionário de avaliação do protocolo (verificar Apêndice A) e o mesmo foi disponibilizado por meio da ferramenta de Formulários do Google, onde especialistas foram convidados a responderem o questionário.

Obteve-se respostas de quatro especialistas e os principais pontos levantados em relação ao protocolo foram: i) verificar a validade de realizar a busca manual visto que a busca automática deve cobrir os estudos a serem encontrados; ii) verificar se a Restrição 13 deveria mesmo conter uma exceção para a Restrição 3, ou se isto não deveria já estar especificado na própria Restrição 3; iii) analisar se o *snowballing* e a avaliação da qualidade não deveriam estar contidos dentro do processo de seleção e assim demonstrados na Figura - 3.1; iv) dúvida quanto ao processo de análise do protocolo, se no caso seria uma compilação sintética baseada na lista de informações a serem extraídas; v) verificar se não seria interessante incluir nas fontes manuais a conferência *International Conference on Software Engineering and KnowledgeEngineering* (SEKE).

Em relação aos pontos levantados pelos especialistas o seguinte foi considerado: i) decidiu-se prosseguir com a execução das buscas manuais, até como forma de evitar que estudos potenciais não fossem encontrados devido a alguma limitação da *string* de busca eletrônica; ii) a observação é válida, porém decidiu-se manter desta forma para manter o padrão direto e instrutivo dos critérios de exclusão, deixando as questões subjetivas para serem tratadas pelos critérios de inclusão; iii) O processo de seleção apresentado na Figura - 3.1 visa tratar exclusivamente da reunião de todos os potenciais estudos, sendo que o produto do último passo do mesmo será a lista final de artigos selecionados. Desta forma, a avaliação da qualidade é entendida como atividade a ser executada após o processo de seleção, visto que o objetivo é avaliar a qualidade apenas dos artigos selecionados. Quanto ao *snowballing* aplica-se o mesmo princípio, visto que o objetivo é aplicar a técnica apenas no conjunto final de estudos, pois aplicá-la ainda durante a fase de seleção pode gerar desperdício de esforço em casos onde o estudo fosse eventualmente excluído antes da seleção final; iv) a princípio o protocolo busca apenas realizar esta compilação sintética

das informações extraídas, porém, a partir da obtenção destes dados será conduzida uma análise das informações; v) a conferência sugerida (SEKE) foi adicionada como fonte para a busca manual.

3.2.2 Questão de Pesquisa

Visto que o objetivo do mapeamento é identificar as técnicas, práticas e ferramentas utilizadas no desenvolvimento de software em *startups*, a questão de pesquisa (QP1) definida foi:

- **QP1: Quais são as técnicas, práticas e ferramentas utilizadas para o desenvolvimento de software em *startups*?**

3.2.3 Estratégia de Busca

A parte mais importante da estratégia de busca é a definição da *string* de busca a ser utilizada durante as pesquisas. Seguindo as diretrizes sugeridas por Kitchenham e Charters (2007), os seguintes pontos foram levados em consideração para a sua montagem:

- A partir da pergunta de pesquisa são identificados os termos principais que irão compor a *string*;
- São identificados os sinônimos dos termos;
- Todos os termos são traduzidos para o inglês, já que este é o idioma mais utilizado na literatura de Ciência da Computação, assim como em outras áreas;
- A *string* é composta da combinação dos termos definidos, principais e sinônimos, utilizando-se dos operadores lógicos OR (ou) e AND (e), sendo estes, respectivamente, utilizados entre os sinônimos e os termos.

A Tabela - 3.1 mostra o conjunto das palavras-chave que compuseram a *string* de busca.

Tabela 3.1: Termos e sinônimos (adaptada de Paternoster *et al.* (2014))

Conceitos chave	**Termos**
Startups	software startup*; software start-up*; start-up compan*; startup compan*; lean startup*; lean start-up* IT start-up*; IT startup*; internet start-up*; internet startup*; web startup*; web start-up*; mobile startup*; mobile start-up*;
Desenvolvimento	develop*; engineer*; implement*; cod*; creat*; build*;

De acordo com as palavras-chave que formam a Tabela - 3.1, a *string* geral de busca definida foi:

software startup OR software start-up* OR start-up compan* OR startup compan* OR lean startup* OR lean start-up* OR IT start-up* OR IT startup* OR internet start-up* OR internet startup* OR web startup* OR web start-up* OR mobile startup* OR mobile start-up* AND (develop* OR engineer* OR implement* OR cod* OR creat* OR build*)*

3.2.3.1 Fontes de Busca

As fontes de busca foram divididas em: fontes de busca automáticas (bases eletrônicas) e fontes de busca manuais (periódicos e conferências). Para as fontes de busca automáticas foram observadas as fontes já utilizadas por especialistas em ESBE (Kitchenham e Charters, 2007) e observadas as fontes utilizadas em revisões anteriores sobre o desenvolvimento de software em *startups*, tendo como principal o trabalho realizado por Paternoster *et al.* (2014). As seguintes fontes foram selecionadas para as buscas eletrônicas: *Inspec/Compendex*, *IEEE Xplore*, *Scopus*, *ScienceDirect*, *ACM Digital Library* e *Google Scholar*. Para as fontes de busca manuais foram observadas as fontes com maior probabilidade de retornar artigos de qualidade (Petersen *et al.*, 2015), onde muitas delas são meios de publicação das fontes eletrônicas. Além disto, foram selecionadas algumas das fontes manuais verificadas por Klotins *et al.* (2015).

3.2.4 Estratégia de Seleção

De acordo com Kitchenham e Charters (2007), a estratégia de seleção deve identificar os estudos que proporcionem uma evidência clara sobre a questão de pesquisa, desta forma, para evitar vieses, os critérios de exclusão e inclusão devem ser definidos durante

a definição do protocolo.

3.2.4.1 Critérios de Exclusão

Foram excluídos os artigos:

- (Restrição 1) Obsoletos (mais de 10 anos, visto que um dos focos da pesquisa é permitir a transferência do conhecimento para a indústria);
- (Restrição 2) Não revisados em pares (literatura cinza);
- (Restrição 3) Que não estejam escritos em inglês;
- (Restrição 4) Relacionado a organizações que não se caracterizam como *startups*;
- (Restrição 5) Não relacionados ao processo de desenvolvimento de software.
- (Restrição 6) Que contemplam somente a execução de estudos teóricos sobre *startups*.
- (Restrição 7) Que não estejam disponíveis na internet;
- (Restrição 8) Quando forem encontrados artigos duplicados, apenas o mais completo será mantido.

3.2.4.2 Critérios de Inclusão

Foram incluídos os artigos:

- (Restrição 9) Que contemplem a execução de estudos empíricos a respeito das técnicas, práticas e ferramentas utilizadas no desenvolvimento de software dentro do contexto de *startups*.
- (Restrição 10) Quando mais de um artigo conter resultados distintos de um mesmo estudo, todos serão incluídos.
- (Restrição 11) Durante as buscas manuais artigos publicados em português também serão aceitos, abrindo assim uma exceção para a Restrição 3.

3.2.4.3 Processo de Seleção

O processo de seleção dos estudos foi conduzido a partir de quatro etapas, conforme demonstrado na Figura - 3.1.

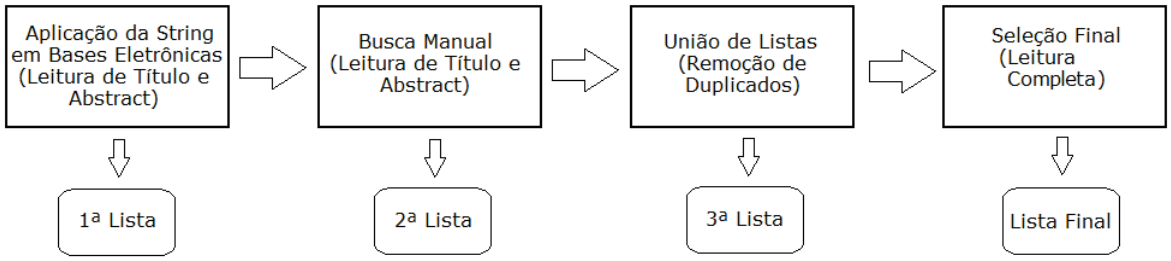

Figura 3.1: Processo de Seleção

3.2.4.3.1 Aplicação da String em Bases Eletrônicas

Neste passo foram realizadas as buscas nas bases eletrônicas a fim de encontrar os estudos primários. Foram lidos o título e *abstract* de cada estudo observando-os em relação aos critérios de exclusão definidos anteriormente. Aqueles artigos que não geraram confiança suficiente para serem excluídos foram mantidos para análise em um passo futuro.

A Tabela - 3.2 apresenta as fontes de busca eletrônica juntamente com a quantidade total de artigos retornados e de artigos aceitos durante a aplicação da *string*.

Tabela 3.2: Total de artigos retornados por fonte de busca eletrônica

ID	Fonte	Total de artigos retornados	Total de artigos aceitos
1	Inspec/Compendex	657	37
2	IEEE Xplore	282	3
3	Scopus	19	0
4	ScienceDirect	265	8
5	ACM Digital Library	429	4
6	Google Scholar	182	7
	Total	1.834	59

Ao fim da busca, foram iniciadas as leituras de título, *abstract* e palavras-chave dos artigos retornados, observando-os em relação aos critérios de inclusão e exclusão definidos no protocolo.

3.2.4.3.2 Busca Manual

Neste passo foram realizadas as buscas nas fontes manuais onde periódicos foram analisados ao percorrer a lista de artigos publicados nos volumes e edições da publicação. Já para as conferências a busca foi feita ao analisar os artigos aceitos para publicação. A Tabela - 3.3 apresenta as fontes utilizadas (periódicos e conferências) já com a quantidade de artigos retornados por cada fonte.

Tabela 3.3: Total de artigos retornados por fonte de busca manual

ID	**Fonte**	**Qtde.**
1	Americas Conference on Information Systems	0
2	Annals of Software Engineering	0
3	Communications of the ACM (CACM)	2
4	Empirical Software Engineering	2
5	IEEE Computer	4
6	IEEE Software	11
7	IEEE Transactions on Software Engineering (TSE)	1
8	IET Software (EE Proceedings Software)	1
9	Information and Software Technology (IST)	6
10	Information Systems Journal	2
11	International Conference on Cooperation and Promotion of Information Resources in Science and Technology (COINFO)	0
12	International Conference on eXtreme Programming and Agile Processes in Software Engineering (XP)	10
13	International Conference on Global Software Engineering (ICGSE)	4
14	International Conference on Software Engineering (ICSE)	1
15	International Conference on Software Engineering and Knowledge Engineering (SEKE)	2
16	International Journal of e-Collaboration	0
17	International Journal of Project Management	2
18	International Symposium on Empirical Software Engineering and Measurement (ESEM)	1
19	Journal of Computer-Mediated Communication	0
20	Journal of Global Information Management	1
21	Journal of Global Information Technology Management	0
22	Journal of Software: Evolution and Process	10
23	Journal of Systems and Software (JSS)	7
24	Lean Enterprise Software and Systems (LESS)	3
25	Software Practice and Experience (SPE)	0
26	Software Process: Improvement and Practice	0
27	Transactions On Software Engineering and Methodology (TOSEM)	1
	Total	71

Foram lidos o título, *abstract* e as palavras-chaves de cada estudo observando-os em relação aos critérios de exclusão e inclusão definidos anteriormente. Aqueles artigos que não geraram confiança suficiente para serem excluídos foram mantidos para análise em um passo futuro. Durante este passo, é comum que uma grande quantidade de artigos totalmente irrelevantes para a pergunta de pesquisa sejam encontrados (Kitchenham e Charters, 2007), ou seja, que não estão relacionados ao desenvolvimento de software ou a *startups*, desta forma, estes artigos foram permanentemente descartados, não sendo registrados nas listas de artigos excluídos. Como forma de complementar a busca manual, foi utilizada a técnica de *snowballing*, que consiste em análisar os artigos referenciados nos estudos encontrados e avaliar se os mesmos devem ser inclusos no conjunto final ou não. A técnica foi conduzida durante a etapa de extração de dados dos artigos selecionados.

3.2.4.3.3 União de Listas

Neste passo foi realizada a união das listas de artigos encontrados durante as buscas. A união das listas dos artigos encontrados durante as buscas eletrônicas (59) e manuais (71) resultou em um total de 130 artigos encontrados, porém destes, 18 artigos encontrados na busca manual foram considerados duplicados pois já haviam sido retornados durante a busca eletrônica, sendo então removidos das futuras listas.

3.2.4.3.4 Seleção Final

Por fim, neste passo realizou-se a leitura por completo dos potenciais estudos obtidos do passo anterior e, também, é nesta etapa que os artigos que geraram incertezas nos passos anteriores foram novamente analisados. Durante a análise dos estudos foram observados os critérios de exclusão e os critérios de inclusão, onde ao fim desta análise obteve-se a seleção final de artigos. Após a união das listas um total 112 artigos chegaram até esta fase para serem analisados. Ao término da leitura, um total de 17 artigos foram aceitos por satisfazerem a pergunta de pesquisa proposta no protocolo e então obteve-se a seleção final de artigos e considerou-se finalizada a etapa de seleção dos estudos. A Figura - 3.2 sintetiza as etapas e resultados do processo de seleção.

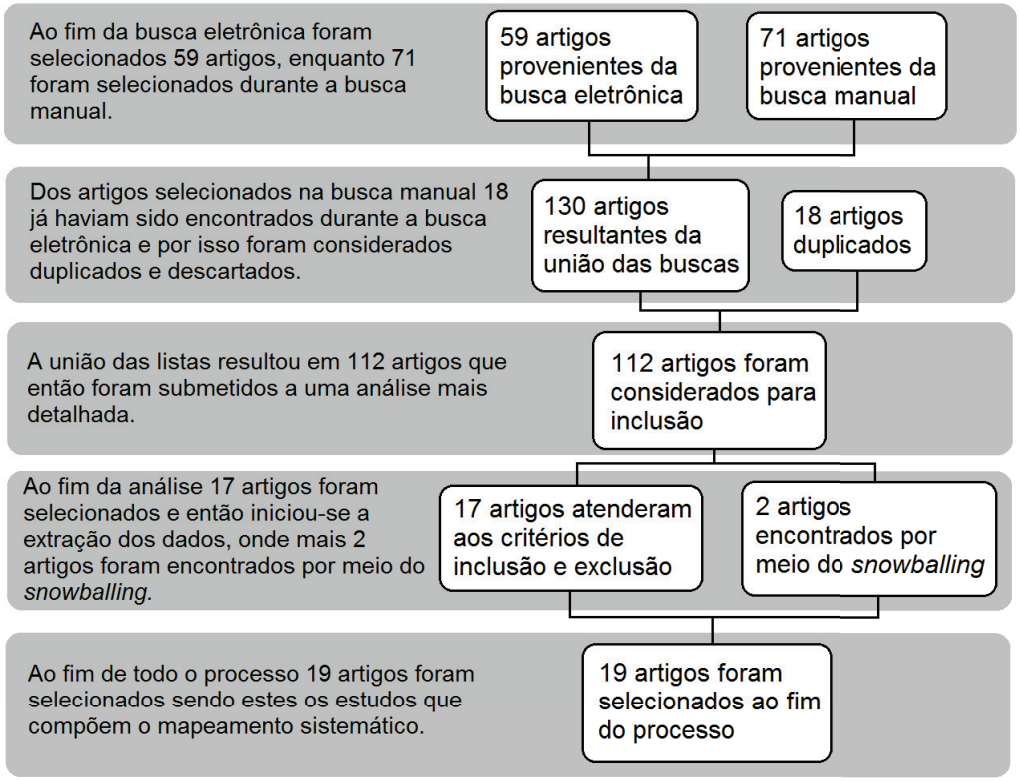

Figura 3.2: Resultados do processo de seleção

A Figura - 3.3 demonstra a distribuição dos artigos aceitos por país. Ao cruzar os dados da figura com a origem dos pesquisadores que constituem a agenda de pesquisa apresentada por Unterkalmsteiner *et al.* (2016) podemos verificar que grande parte dos estudos relacionados a *startups* surgem de países como Finlândia, Itália e Suécia.

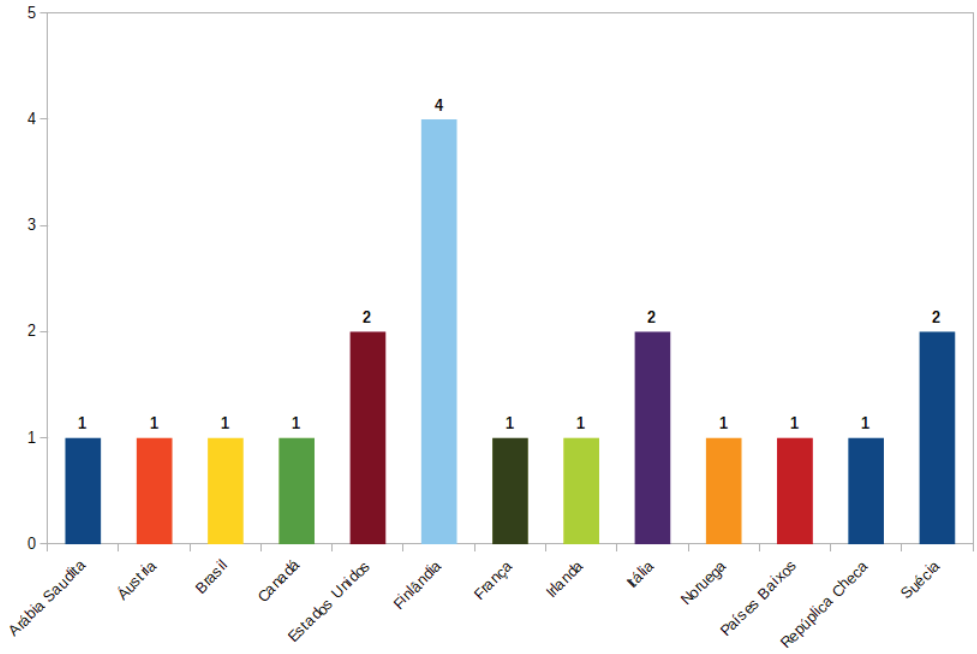

Figura 3.3: Distribuição dos artigos aceitos por país durante a seleção final.

A Figura - 3.4 demonstra a distribuição por ano onde, por meio da linha de tendência, é possível notar a evolução do tema ao longos dos anos.

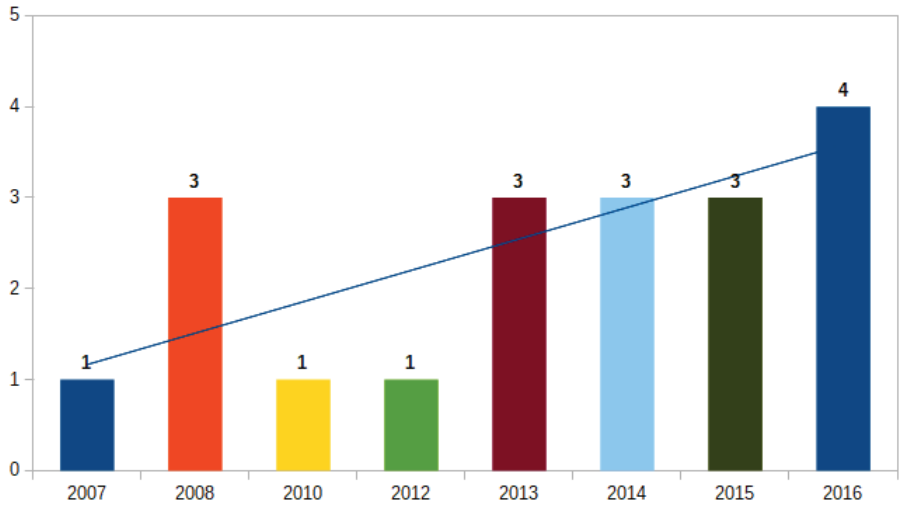

Figura 3.4: Distribuição dos artigos aceitos por ano durante a seleção final.

3.2.4.4 Avaliação da Qualidade

Segundo Kitchenham e Charters (2007), a qualidade de um estudo está relacionado com a sua habilidade de minimizar o viés e maximizar a validação interna e externa. Um estudo

enviesado é aquele onde os resultados se afastam dos resultados reais, uma vez que um artigo imparcial, ao contrário de enviesado, será internamente válido. Desta forma, um estudo internamente válido é aquele onde os resultados são os mais próximos possíveis da realidade, sendo este um requisito para a validação externa. A validação externa diz respeito a replicação do estudo, ou seja, se os efeitos observados no estudo podem ser alcançados fora dele.

Avaliar a qualidade dos estudos permite verificar a importância de estudos individuais durante a sintetização dos resultados, auxilia a guiar as recomendações para pesquisas futuras e também a guiar a interpretação dos resultados e determinar a força das inferências (Kitchenham, 2004).

Então, para avaliação da qualidade dos estudos selecionados, conduzida pelo próprio autor, os mesmos foram verificados em relação as questões abaixo por meio de respostas de "Sim" ou "Não" (Dyba *et al.*, 2007):

1. O artigo é baseado em pesquisas ou é apenas um relatório de lições aprendidas com base na opinião de especialistas?
2. Há uma declaração clara dos objetivos da pesquisa?
3. Existe uma descrição adequada do contexto no qual a pesquisa foi realizada?
4. O projeto da pesquisa foi adequado para resolver os objetivos da pesquisa?
5. A estratégia de recrutamento foi adequada aos objetivos da pesquisa?
6. Havia um grupo de controle com o qual se comparar os tratamentos?
7. Os dados foram coletados de uma forma que abordou a questão de pesquisa?
8. A análise de dados foi suficientemente rigorosa?
9. A relação entre o pesquisador e os participantes foi devidamente considerada?
10. Há uma declaração clara dos resultados?
11. O estudo é de valor para a pesquisa ou a prática?

Como forma de medir a qualidade dos estudos as questões foram particionadas sob os critérios de rigor e relevância. O rigor representa a precisão utilizada pelo estudo em seu método de pesquisa e a forma como o estudo é apresentado. A relevância representa o

valor do estudo para a comunidade de pesquisa e para a indústria (Ivarsson e Gorschek, 2011). As questões 1 a 8 estão relacionadas ao rigor e as questões 9 a 11 são referentes a relevância. Cada resposta "Sim" possui valor 1 e cada resposta "Não" possui valor 0, sendo assim, o valor máximo para o critério de rigor é 8 e o valor máximo para o critério de relevância é 3.

A Figura - 3.5 demonstra a visão geral dos artigos encontrados analisados por meio dos critérios de rigor e relevância. Nota-se que a maior parte dos artigos (16) estão localizados no canto superior direito do gráfico, o que denota alto rigor e alta relevância. Apenas 2 artigos possuem um rigor mediano, porém ainda apresentam máxima relevância. Por fim, apenas 1 dos artigos apresenta baixo rigor e baixa relevância.

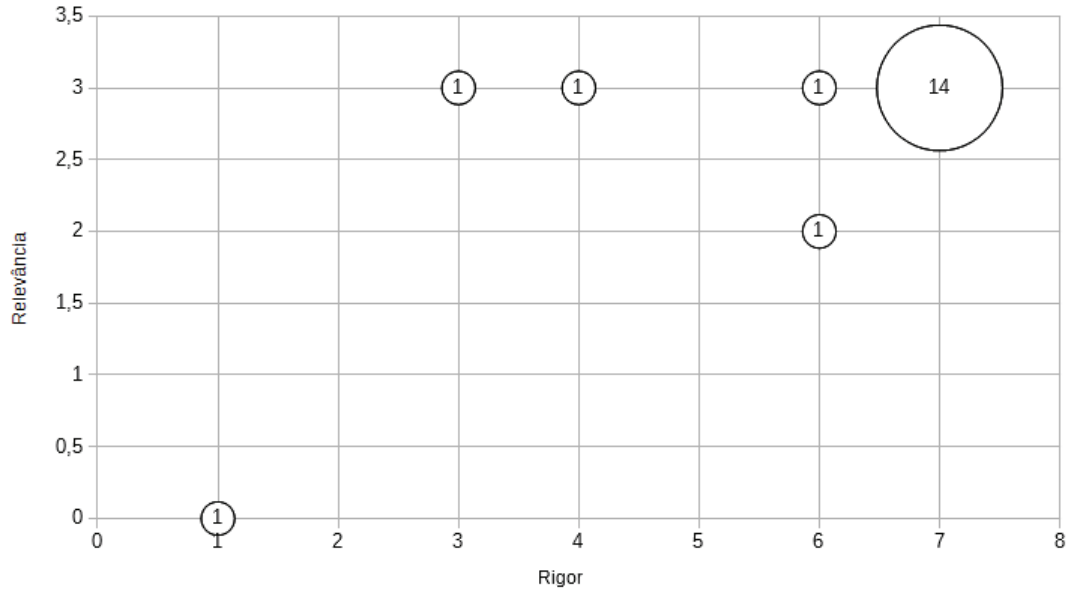

Figura 3.5: Avaliação dos critérios de rigor e relevância.

Além das questões, também foi preenchido um esquema de classificação para listagem final de artigos. O esquema de classificação é adaptado de um MSL sobre desenvolvimento de software em *startups* (Paternoster *et al.*, 2014), e com ele busca-se categorizar os estudos encontrados de uma forma que seja possível determinar sua qualidade. O esquema é particionado sob 4 aspectos: tipo de pesquisa, tipo de contribuição, foco e significância.

O tipo de pesquisa visa identificar os diferentes tipos de estudo (pesquisa de avaliação, proposta de solução, artigo filosófico, artigo de opinião ou artigo de experiência), sem considerar as metodologias empregadas em cada estudo.

O tipo de contribuição permite identificar o tipo de contribuição que o estudo proporciona, dos quais estes tipos podem ser divididos entre fracos (diretrizes, lições aprendidas, conselho/implicações e ferramenta) e fortes (modelo, teoria e *framework*/métodos).

Em relação ao foco, suas categorias (desenvolvimento de software, gestão de processo, ferramentas e tecnologias, gestão/organizacional) permitem identificar os estudos diretamente relacionados ao desenvolvimento de software, as ferramentas e tecnologias utilizadas para suportar o desenvolvimento ou então aqueles que possuem maior foco quanto a gestão do processo de desenvolvimento.

Quanto ao aspecto de significância (total, parcial e marginal), o mesmo permite identificar o grau de relacionamento com as atividades de engenharia em *startups*, onde aqueles estudos com maior relação terão maior significância para a pergunta de pesquisa.

Classificar os estudos sob estes aspectos permite verificar a relevância dos estudos em relação a questão de pesquisa. Ao analisar a significância dos estudos, por exemplo, pode-se verificar quais estudos estão diretamente ligados ao desenvolvimento de software em *startups* e por consequência quais serão aqueles que provavelmente terão maiores contribuições para responder a questão de pesquisa. De maneira geral, os aspectos permitem identificar de uma forma ágil os tipos de resultados que são esperados de cada estudo e assim guiar o leitor para aqueles estudos que provavelmente serão mais relevantes para o mesmo.

3.2.5 Estratégia de Extração

Para Kitchenham e Charters (2007) a extração dos dados deve ser projetada para coletar toda a informação necessária para responder a questão de pesquisa, assim como para analisar os estudos por meio dos critérios de seleção.

Desta forma, seguindo as diretrizes (Kitchenham e Charters, 2007), as seguintes informações devem ser extraídas a fim de se formar um padrão: i) Título do artigo; ii) Autor(es); iii) Ano de publicação; iv) Tipo de fonte (revista, conferência ou jornal); v) Título da fonte.

Em adição, seguindo o esquema de classificação descrito na seção anterior, os seguintes dados foram extraídos de acordo com os aspectos contidos no esquema: i) Tipo de pesquisa; ii) Tipo de contribuição; iii) Foco; iv) Significância.

Durante esta etapa de extração dos dados foi aplicada a técnica de *snowballing* para os estudos selecionados, dos quais aqueles estudos que se mostraram promissores foram analisados por meio dos critérios de inclusão e exclusão. A técnica consiste em verificar a lista dos artigos referenciados nos estudos aceitos com intuito de identificar novos

artigos para inclusão (Wohlin, 2014). Os artigos que se mostraram irrelevantes por meio do título ou foram considerados obsoletos não entraram na lista de artigos excluídos, sendo permanentemente descartados. Durante a aplicação da técnica um total de 21 potenciais artigos foram encontrados, porém, 17 deles foram considerados duplicados, restando apenas 4 artigos que foram analisados, e após a análise dos mesmos apenas 2 foram aceitos observando todos os critérios de inclusão e exclusão. Ao fim do *snowballing* um total de 19 artigos foram aceitos durante toda a busca, sendo estes os artigos que tiveram os dados extraídos.

A Figura - 3.6, Figura - 3.7 e Figura - 3.8 agregam a quantidade de artigos, por meio das bolhas, encontrados entre os cruzamentos dos aspectos definidos no esquema de classificação.

A Figura - 3.6 demonstra a distribuição e relacionamento dos artigos encontrados por meio dos aspectos tipo de pesquisa, tipo de contribuição e foco. Nesta figura pode-se verificar que o cruzamento entre o tipo de contribuição lições aprendidas e o tipo de foco desenvolvimento de software representam o maior número de artigos (6). Ao cruzar o tipo de pesquisa com o foco, o maior número de artigos (4) está entre as categorias de pesquisa de avaliação e gestão de processo. Separadamente, o tipo de contribuição lições aprendidas agrega o maior número de artigos, com 12. Por sua vez, o tipo de pesquisa pesquisa de avaliação representa 7 artigos, enquanto para o foco a categoria desenvolvimento de software, com 8 artigos, representa o maior número.

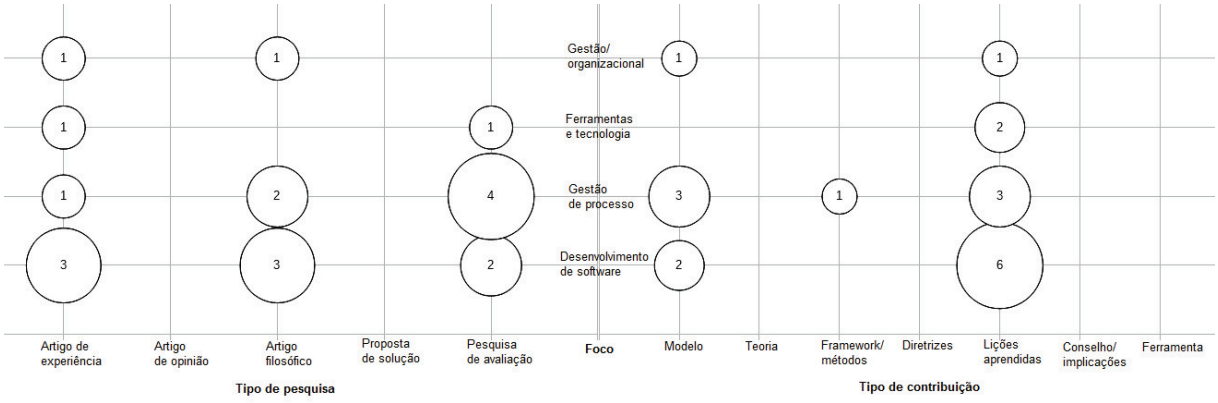

Figura 3.6: Tipo de pesquisa, tipo de contribuição e foco.

A Figura - 3.7 apresenta o relacionamento entre os aspectos tipo de pesquisa, significância e tipo de contribuição. Nesta figura nota-se que os cruzamentos entre o foco lições aprendidas e significância total representam um total de 12 artigos, enquanto ao cruzar o foco lições aprendidas com o tipo de pesquisa artigo de experiência temos um total de 6

artigos. Esta figura também demonstra que apenas 2 artigos não possuem total relação com as atividades de engenharia de software em *startups*.

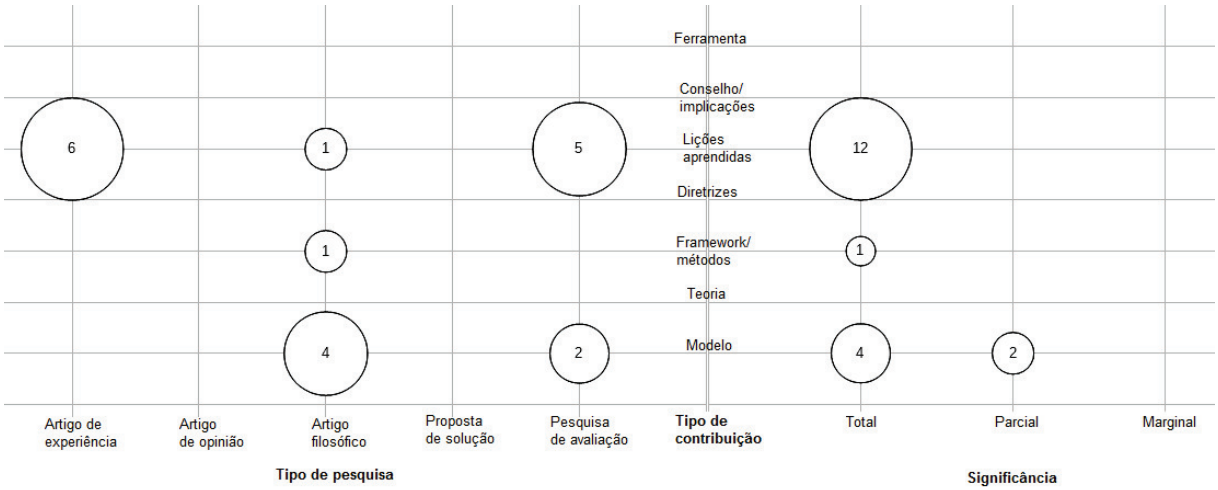

Figura 3.7: Tipo de pesquisa, significância e tipo de contribuição.

Por fim, a Figura - 3.8 demonstra a relação entre os aspectos tipo de pesquisa, significância e foco. A figura demonstra que o cruzamento entre a significância total e tipo de pesquisa pesquisa de avaliação (7) e significância total de foco desenvolvimento de software (8) são onde estão concentrados a maioria dos artigos.

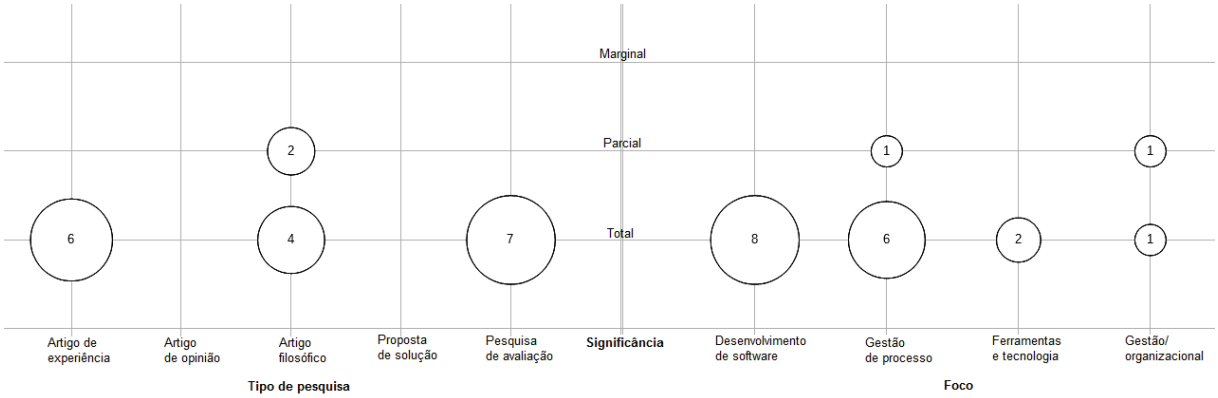

Figura 3.8: Tipo de pesquisa, significância e foco.

A Tabela - 3.4 apresenta a extração dos dados dos 19 artigos encontrados no mapeamento, onde são apresentados os títulos dos artigos e seus respectivos autores, ano de publicação, tipo e título da fonte na qual os mesmos foram publicados.

Tabela 3.4: Extração dos dados

ID	Título do artigo	Autores	Ano	Tipo de fonte	Título da fonte
1	A Feature-Based Tool-Selection Classification for Agile Software Development	Mohsen Taheri; S. Masoud Sadjadi	2015	Conferência	SEKE
2	Agile Methods In Tech-Startup	Tomas Langer; Pavel Vanecek	2012	Conferência	IMEA
3	An investigation into software development process formation in software start-ups	Gerry Coleman; Rory V. O'Connor	2008	Periódico	JEIM
4	Development of a mobile application using the lean startup methodology	Adnan Miski	2014	Periódico	IJSER
5	Experience report - A pure shirt fits reflections on Haskell at Bluespec	Ravi Nanavati	2008	Conferência	ICFP
6	Extreme programming in action - A longitudinal case study	Akbar Saeed; Peter Tingling	2013	Periódico	IJBMI
7	Good Organisational Reasons for Bad Software Testing - An Ethnographic Study of Testing in a Small Software Company	David Martin; John Rooksby; Mark Rouncefield; Ian Sommerville	2007	Conferência	ICSE
8	Huitale - A story of a Finnish lean startup	Marko Taipale	2010	Conferência	LESS
9	Lean product development in early stage startups	Jens Bjork; Jens Ljungblad; Jan Bosch	2013	Conferência	ICSOB
10	Lean startup meets software product lines - Survival of the fittest or letting products bloom	Henri Terho; Sampo Suonsyrja; Ari Jaaksi; Tommi Mikkonen; Rick Kazman; Hong-Mei Chen	2015	Conferência	SPLST
11	Minimum viable product or multiple facet product - The role of MVP in software startups	Anh Nguyen Duc; Pekka Abrahamsson	2016	Conferência	XP Conference
12	Pragmatic and opportunistic reuse in innovative start-up companies	Slinger Jansen; Sjaak Brinkkemper; Ivo Hunink; Cetin Demir	2008	Periódico	IEEE Software
13	Raising the odds of success - the current state of experimentation in product development	Eveliina Lindgren; Jurgen Munch	2016	Periódico	IST
14	Software development in startup companies - A systematic mapping study	Nicolò Paternoster; Carmine Giardino; Michael Unterkalmsteiner; Tony Gorschek; Pekka Abrahamsson	2014	Periódico	IST
15	Software Development in Startup Companies - The Greenfield Startup Model	Carmine Giardino; Nicolò Paternoster; Michael Unterkalmsteiner; Tony Gorschek; Pekka Abrahamsson	2016	Periódico	TSE
16	UX work in startups - Current practices and future needs	Laura Hokkanen; Kaisa Vaananen-Vainio-Mattila	2015	Conferência	XP Conference
17	What do we know about software development in startups	Carmine Giardino; Michael Unterkalmsteiner; Nicoló Paternoster; Tony Gorschek; Pekka Abrahamsson	2014	Periódico	IEEE Software
18	When Software Impacts the Economy and Environment	Edleno Silva de Moura; Mauro Rojas Herrera; Leonardo Santos; Tayana Conte	2016	Periódico	IEEE Software
19	Why modern mobile and web-based development need a Lean Agile Web Approach (LAWA)	Jaroslav Skrabálek; Christina Bohm	2013	Conferência	IDIMT

3.3 Resultados e Discussão dos Dados

Nesta seção são apresentadas as técnicas, práticas e ferramentas extraídas, visando proporcionar ao profissional da indústria uma visão sobre a utilização das mesmas e assim apoiar o mesmo nos processos de decisão da *startup*. Também são discutidos os achados de acordo com o esquema de classificação e as áreas de pesquisa que são pouco exploradas e que se beneficiariam de uma análise mais detalhada. A seção está particionada entre as contribuições para a indústria e as contribuições para a academia.

3.3.1 Contribuição para a Indústria

Como contribuição para a indústria são citadas as técnicas, práticas e ferramentas encontradas durante a extração. Nesta subseção é apresentada uma breve descrição e que tipo de contribuição cada um dos estudos encontrados demonstra, de forma que o leitor possa identificar aqueles artigos que mais lhe interessam e então se aprofundar nos mesmos.

Dos 19 artigos encontrados foram extraídas 24 técnicas, 31 práticas e 37 ferramentas. Dentre as técnicas extraídas, a que apresenta maior frequência entre os estudos é a metodologia *Lean/Lean Startup* com 7 ocorrências, seguida pela metodologia Scrum com 4. Metodologias ágeis em geral, Kanban e cartões de estórias aparecem 3 vezes nos artigos encontrados, seguidas por *Extreme Programming*, *Planning Game*, *Backlog* e programação em pares com duas ocorrências. As demais técnicas encontradas (*Join Application Design*, *Experiment Driven Development* (EDD), Teste A/B, *Feature Driven Development* (FDD), *Rational Unified Process* (RUP), Scrumban, *logs* e estatísticas e *Code Review*) aparecem apenas uma vez cada. Técnicas de *Minimum Viable Product* (MVP) aparecem 7 vezes distribuídas em tipos diferentes. A Tabela - 3.5 demonstra em qual artigo cada técnica foi encontrada.

Em relação as práticas, integração contínua possui a maior frequência, aparecendo 5 vezes entre os artigos. Utilização de soluções *commercial off-the-shelf* (COTS), conhecido como software de "prateleira", ou *open source*, padrões de código/*design*/*frameworks*, refatoração de código e entregas frequentes/contínua são as próximas com 4 ocorrências cada.

Testes de aceitação e reuso de software contam com 3 ocorrências, seguidas por cliente no local de desenvolvimento e testes de integração com duas ocorrências. O restante das práticas (semana de trabalho de 40 horas, código pertence a todos, teste contínuo, *design* simples, *releases* curtos, teste unitário, teste de usabilidade, entrega manual,

Tabela 3.5: Rastreamento das técnicas

Técnica	**ID Artigo**
Lean/Lean Startup	4, 8, 10, 13, 14, 15, 17
Scrum	9, 13, 18, 19
Metodologias ágeis	13, 14, 17
Kanban	9, 13, 18
Cartões de estórias	7, 14, 15
Extreme Programming	3, 14
Planning Game	6, 14
Backlog	14, 16
Programação em pares	14, 18
Técnicas de MVP	11
Join Application Design	6
Experiment Driven Development	13
Teste A/B	15
Feature Driven Development	2
Rational Unified Process	3
Scrumban	14
Logs e estatísticas	16
Code Review	18

gerenciamento de requisitos, apresentação de materiais, backups diários em produção, *rollback* automático em produção em caso de falha, *bug-tracking*, capacitação dos membros da equipe, indicadores chave de performance, prototipagem evolucionária, controle de versão, entrevistas, usuários de testes, pesquisa de mercado, comando e controle e incluir e capacitar) possuem uma ocorrência cada. A Tabela - 3.6 demonstra em qual artigo cada prática foi encontrada.

No conjunto das ferramentas extraídas a que possui maior ocorrência é o Google Analytics, com apenas duas, onde o restante das ferramentas (Atlassian Jira, Axosoft On-Time, Target Proccess, Microsoft TFS, Rally Platform, Mingle, Version One, Blossom.io, Scrumwise, Base Camp, LeanKit, AgileZen, PlanBox, Kanbanize, ScrumWorksPro, BananaScrum, AgileFant, IceScrum, Xplanner, Trello, Asana, XPStoryStudio, JustInMind, Betalist.com, Mixpanel, Hadoop, Storm, Kafka, Flume, Hbase e AIMLBot) e linguagens (Haskell, C++, PHP, Python e .NET) aparecem apenas uma vez cada. A Tabela - 3.7 demonstra em qual artigo cada ferramenta foi encontrada.

Dos artigos considerados para a seleção final destaca-se o mapeamento sistemático de literatura realizado por Paternoster *et al.* (2014). Apesar de abrangente, uma das perguntas de pesquisa do estudo foca nas práticas de trabalho relacionadas a engenharia de software em *startups*, sendo que os achados desta pergunta proporcionaram um volume

Tabela 3.6: Rastreamento das práticas

Prática	**ID Artigo**
Integração contínua	6, 8, 13, 14, 18
Soluções COTS ou *open source*	12, 14, 15, 17
Padrões de código	6, 14, 15, 17
Refatoração de código	6, 14, 17, 19
Entregas frequentes/contínua	7, 8, 14, 18
Testes de aceitação	7, 14, 17
Reuso de software	12, 14, 17
Cliente no local de desenvolvimento	6, 7
Testes de integração	7, 19
Semana de trabalho de 40 horas, código pertence a todos, teste contínuo, *design* simples, *releases* curtos	6
Teste unitário, gerenciamento de requisitos, apresentação de materiais	7
Teste de usabilidade, entrega manual, *bug-tracking*, capacitação dos membros da equipe, indicadores chave de performance	14
Backups diários em produção, *Rollback* automático em caso de falha	8
Prototipagem evolucionária, controle de versão	15
Entrevistas, usuários de testes, pesquisa de mercado	16
Comando e controle, incluir e capacitar	3

Tabela 3.7: Rastreamento das ferramentas

Ferramenta	**ID Artigo**
Google Analytics	4, 16
Atlassian Jira, Axosoft On-Time, Target Proccess, Microsoft TFS, Rally Platform, Mingle, Version One, Blossom.io, Scrumwise, Base Camp, LeanKit, AgileZen, PlanBox, Kanbanize, ScrumWorksPro, BananaScrum, AgileFant, IceScrum, Xplanner, Trello, Asana, XPStoryStudio	1
Hadoop, Storm, Kafka, Flume, Hbase, C++, PHP, Python	18
Betalist.com, Mixpanel	16
AIMLBot, .NET	12
JustInMind	11
Haskell	5

considerável de técnicas e práticas extraídas com 20 contribuições. O trabalho de Tingling e Saeed (2007) apresentaram uma contribuição considerável, com 11 ocorrências entre técnicas e práticas. No estudo, os autores identificam e analisam a adoção dos princípios

de XP em uma *startup*, onde, curiosamente, a programação em pares não é adotada, mesmo sendo uma das principais técnicas da metodologia.

O estudo desenvolvido por Martin *et al.* (2007) é focado em testes e faz uma análise de como os testes são conduzidos em uma *startup*. Várias das práticas de trabalho identificadas na *startup* são derivadas da metodologia XP, entretanto não se pode dizer que a organização de fato segue a metodologia já que, assim como no trabalho de Tingling e Saeed (2007), a programação em pares não é adotada. No caso de estudo analisado por Martin *et al.* (2007) foi identificado que os processos de testes ainda precisam ser pesquisados mais a fundo, pois ainda há uma distância entre o que se tem em pesquisas e a realidade. Outro estudo focado em uma das partes que compõem o desenvolvimento, desta vez voltado a interface, mais especificamente *User Experience* (UX), é apresentado por Hokkanen e Väänänen-Vainio-Mattila (2015). O estudo investiga, por meio de entrevistas, as práticas atuais relacionadas a UX dentro de *startups*. Ao fim do estudo são apontados alguns passos a serem seguidos por *startups*, visto que de acordo com os autores os processos de UX são de grande importância para estas organizações, principalmente para aquelas que estão planejando entrar em novos mercados por meio de produtos inovadores. Em outro estudo, Duc e Abrahamsson (2016) conduzem uma investigação a respeito do papel do MVP em *startups*, do qual são apresentados alguns dos principais tipos de MVP utilizados neste contexto. O estudo observou que as técnicas de MVP são úteis pois os artefatos gerados servem como limites do que pode ser realizado e também podem ser reutilizados em outros estágios do projeto. Além disto, por meio dos estudos de caso, o estudo identificou que metodologias de desenvolvimento ágil são o processo mais viável para *startups*, pois permitem entregas rápidas e encurtam o tempo entre a concepção da ideia e a produção da mesma.

Na pesquisa realizada por Langer e Vaněček (2012) é conduzida uma investigação breve a respeito de qual metodologia será utilizada para um determinado projeto de uma *startup*, sendo apontada ao final a metodologia considerada mais adequada, no caso FDD, pois a mesma proporcionou vantagens como a possibilidade de mudanças no escopo e uma lista das funções que podem ser facilmente implementadas no futuro. Lindgren e Münch (2016) por sua vez apresentam um estudo investigativo a respeito do estado da experimentação no desenvolvimento de produto. Apesar de não ser focado especificamente em *startups* alguns dos casos de estudo analisados podem ser consideradas como *startups*, onde é possível identificar as metodologias usadas nas mesmas, além de que, em pelo menos uma, utiliza-se da experimentação no desenvolvimento do produto.

Giardino *et al.* (2016) conduziram um estudo visando criar-se um modelo de auxílio às *startups* para que estas possam lançar seus produtos no mercado com maior agilidade.

Durante as investigações é possível identificar algumas práticas utilizadas em *startups*, sendo estas informações principalmente de fontes externas. Este estudo é realizado por meio de uma RSL e, assim como os estudos realizados por Paternoster *et al.* (2014) e Klotins *et al.* (2015), aponta que existe uma falta de pesquisas que suportem as atividades de desenvolvimento de software em *startups*. Outro estudo de Giardino *et al.* (2014) questiona o que se sabe sobre o desenvolvimento de software em *startups*, apresentando algumas das técnicas e práticas que são utilizadas no contexto, além de trazer pontos interessantes sobre quais seriam os fatores responsavéis pelo sucesso de uma *startup*. Coleman e O'Connor (2008) buscam identificar como é formado o processo de desenvolvimento de software dentro de *startups*, tendo-se como principal influência as experiências anteriores daqueles que são os encarregados do trabalho de desenvolvimento. No estudo é possível identificar algumas metodologias e formas de gerenciamento que são utilizadas. Entre os achados do estudo, a abordagem de gerenciamento "incluir e capacitar"se mostra interessante pois proporciona liberdade ao time de desenvolvimento, que necessita de menos supervisão direta, seguindo na mesma direção que um dos pontos apontados por Giardino *et al.* (2014) como sendo fator de sucesso para *startups*.

Alguns trabalhos a respeito da metodologia *Lean/Lean Startup* foram encontrados. Miski (2014) apresenta um estudo acerca do desenvolvimento de uma aplicação *mobile* utilizando da metodologia *Lean Startup*. Taipale (2010) apresenta um estudo parecido, porém de forma mais abrangente, não focando no desenvolvimento de apenas uma aplicação. Björk *et al.* (2013) propõem um modelo de auxílio para que *startups* possam aplicar os princípios do *Lean Startup*, visto que o estudo aponta que a metodologia é pouco aplicada pois seus métodos são muito vagos e imprecisos para serem implementados na prática. Das organizações investigadas durante as entrevistas todas utilizam algum tipo de metodologia ágil, principalmente Scrum e Kanban. Škrabálek e Böhm (2013) buscam criar uma abordagem *Lean* por meio de um caso de estudo em uma *startup*, do qual são identificadas algumas das práticas utilizadas pela mesma, além da metodologia Scrum. No estudo de Terho *et al.* (2015) duas *startups* são investigadas em relação a utilização do *Lean Startup*, sendo que estas fazem uso de MVP para iterar por meio de um processo de construção, medição e aprendizagem, o que de acordo com os achados aumenta a produtividade e reduz o *time-to-market* (o tempo entre a concepção do produto até a sua disponibilização para venda).

Jansen *et al.* (2008) conduziram um estudo sobre reuso em *startups*, onde nos casos de estudo apresentados também é possível identificar algumas das ferramentas utilizadas. O reuso se provou rentável para as *startups* analisadas no estudo visto que possibilitou a elas desenvolverem seus produtos por meio da reutilização de funcionalidades que as mesmas

não conseguiriam desenvolver por si próprias. Nanavati (2008) apresenta as motivações e os benefícios da utilização de Haskell como principal linguagem de programação em uma *startup*. O uso da linguagem se provou como um forte diferencial para a *startup* em foco no estudo e, segundo o autor, linguagens de programação puras como o Haskell podem ser fonte de uma vantagem competitiva para *startups* visto que as vantagens trazidas pela pureza são muito maiores que as desvantagens. Em outro artigo, apesar de não ser o foco do estudo de de Moura *et al.* (2016), o trabalho apresenta as linguagens de programação e ferramentas utilizadas pela *startup* investigada no estudo. Por fim, Taheri e Sadjadi (2015) montam um esquema de seleção de ferramentas baseado em característica para o desenvolvimento de software ágil. Apesar de não ser focado apenas em *startups*, o esquema gerado apresenta uma distribuição de ferramentas de acordo com uma classificação para o uso das mesmas nesta fase de vida de uma organização.

3.3.2 Contribuição para a Academia

Nesta subseção são discutidos os principais pontos investigados pelos estudos e é destacada uma visão geral das contribuições dos mesmos. Além disto, é apresentado um panorama dos estudos encontrados de acordo com o esquema de classificação e, também, são apontados quais tipos de estudos poderiam ser conduzidos visando preencher as lacunas da área.

Conforme demonstram os dados extraídos a partir do esquema de classificação, dos 19 artigos encontrados apenas 2 não possuem total ligação com as atividades de desenvolvimento de software em *startups*, porém ainda estão parcialmente relacionados, o que indica que os achados possuem um grande valor para este tipo de organização. Entre os artigos, 6 deles são identificados como do tipo de pesquisa artigo filosófico, ou seja, estes artigos buscam estruturar o campo de conhecimento em forma de uma taxonomia ou de um *framework* conceitual.

Outros 7 artigos são categorizados como de pesquisa de avaliação, onde uma metodologia é aplicada na prática e uma avaliação da mesma é conduzida, sendo que junto com os 6 artigos de experiência, que demonstram como algo foi feito na prática, são os que provavelmente irão fornecer um conhecimento que seja de fácil aproveitamento pela indústria. Somado a isto estão 12 artigos cujo tipo de contribuição são de lições aprendidas onde é possível analisar os resultados diretamente obtidos em pesquisa. Ainda sobre o tipo de contribuição, apenas 1 artigo consta na categoria de *framework*/métodos, que apresenta modelos relacionados à construção de software ou gestão do processo de

desenvolvimento. Os 6 artigos restantes apresentam modelos, ou seja, representam uma realidade observada por meio de conceitos.

Em relação ao aspecto foco, 8 dos artigos encontrados podem ser categorizados como de desenvolvimento de software, o que ajuda a suportar a ideia de que os estudos identificados possuem grande relevância para a indústria, visto que os artigos desta categoria estão diretamente ligados as atividades de engenharia usadas para escrever e manter o código fonte. Além disto, 7 artigos são classificados como de gestão de processo, sendo que estes estão relacionados aos métodos e técnicas utilizadas para o gerenciamento das atividades de desenvolvimento. Entretanto, apenas 2 artigos tratam especificamente dos instrumentos usados para criar, depurar, manter e suportar as atividades de desenvolvimento. Os 2 artigos restantes são categorizados como gestão/organizacional e tratam dos aspectos voltados ao gerenciamento da estrutura.

Pode-se verificar que boa parte das técnicas extraídas estão alinhadas com a agilidade necessária às *startups*. Lean e Scrum como metodologias ágeis, Lean e MVP como forma de aprender por meio de interações com o usuário são alguns dos exemplos. Em relação as práticas o mesmo pode ser percebido. Testes e ciclos curtos são algumas das práticas que mais aparecem, porém de forma muito distribuída. O mesmo pode ser dito para questões voltadas a capacitação do time. Para Giardino *et al.* (2014), o empoderamento da equipe possui um papel importante para o sucesso de uma *startup*, visto que é o conhecimento e a capacidade da equipe que irão compensar a falta de recursos da organização. Em relação as ferramentas, são poucos os artigos que focam especificamente em estudá-las, sendo citadas esporadicamente durante os trabalhos. Apenas o estudo desenvolvido por Taheri e Sadjadi (2015) é focado em analisar ferramentas, porém seu objetivo é montar um esquema de classificação para seleção de ferramentas para o desenvolvimento ágil, sem focar especificamente em *startups*.

Conforme demonstrado pelos estudos, existe uma certa diversidade entre os trabalhos a respeito do desenvolvimento de software em *startups*. O esquema de classificação gerado na extração dos dados demonstra que boa parte dos artigos encontrados são de lições aprendidas, enquanto o restante, com algumas exceções, são de modelos. São poucos os estudos que focam em determinada área, como testes ou interface, sendo que mais estudos deste tipo seriam importantes, pois ajudam na construção de uma base de conhecimento para a solução de problemas específicos. Alguns dos estudos encontrados demonstram resultados interessantes para a sobrevivência das *startups*, como por exemplo a utilização de técnicas de MVP, que permitem que estas organizações aprendam durante o desenvolvimento do produto e então consigam lançar no mercado um produto com mais chances de sucesso. A utilização de métodologias ágeis é o ponto mais comum entre os

estudos, sendo que mesmo aqueles que não citam diretamente alguma metodologia ágil pelo menos faz uso de algum tipo de princípio ágil. Estas metodologias são de grande valor para *startups* pois tempo é essencial para estas organizações, visto que a maioria delas falham dentro de um período curto de tempo, e as mesmas permitem um desenvolvimento mais rápido do produto por meio de entregas frequentes, maior produtividade e redução to *time-to-market*.

Estudos mais abrangentes auxiliam na visão geral do desenvolvimento dentro do contexto de *startups*, permitindo que se veja as áreas carentes de informações e que se beneficiariam de mais estudos focados. São poucos os estudos, por exemplo, que focam em avaliar ferramentas, sendo que apenas um dos estudos buscou criar um modelo para seleção de ferramentas, porém, o estudo em questão não é focado apenas em *startups*. Desta forma, estudos de avaliação ou de comparação de ferramentas são necessários. O uso de *frameworks* está entre as práticas encontradas, entretando nenhum estudo comparativo entre *frameworks* de desenvolvimento disponíveis foi identificado. Na realidade, muitas das práticas identificadas podem ser aplicadas por meio de ferramentas, porém estudos que avaliem ou comparem estas ferramentas são escassos. Portanto, estas seriam as áreas mais carentes no momento, e estudos em torno da mesma seriam de grande importância em virtude da facilidade de transferir este conhecimento para a indústria. De maneira resumida, o número de estudos ainda é pequeno, e mais trabalhos neste contexto são necessários para a formação de uma base de conhecimento sólida que possa ser transferida para indústria com maior frequência e aproveitamento.

3.4 Limitações e Ameaças à Validade

Assim como revisões sistemáticas, a tendência a publicação de apenas bons resultados, é um possível viés ao mapeamento sistemático (Brereton *et al.*, 2007; Kitchenham e Charters, 2007). Entretanto, esta não é uma grande ameaça à validade do estudo conduzido visto que não é o objetivo do mesmo avaliar o desempenho, eficácia, produtividade ou qualquer outro medidor de qualidade das técnicas, práticas e ferramentas extraídas. Ainda assim, isto pode ser uma limitação já que possivelmente diminui o número de estudos publicados a respeito de *startups*, já que há uma grande tendência de que as mesmas falhem (Crowne, 2002).

A construção da *string* de busca eletrônica é um dos principais desafios no planejamento do mapeamento sistemático. No caso das *startups* existem vários sinônimos que podem ser utilizados e com isto é preciso cuidado para não tornar a *string* muita genérica, trazendo uma quantidade quase imensurável de artigos. Como aponta Paternoster *et al.*

(2014), os termos *startup* e *start-up* se assemelham com o verbo especial *to start up* da lingua inglesa, que para outras áreas da literatura está ligada com o processo de início de movimento de um motor. Durante a busca eletrônica, vários dos artigos retornados que foram excluídos já durante a leitura de seus títulos tratavam a respeito de sistemas de controle ou segurança de plantas nucleares, onde o termo *to start up* é bastante utilizado para determinar o mesmo onde estes sistemas entram em ação. Para atenuar esta ameaça a formulação da *string* foi baseada na construída por Paternoster *et al.* (2014), que se provou confiável durante o seu trabalho de mapeamento.

Ameaças quanto a seleção dos estudos são mitigadas por meio dos critérios de inclusão e de exclusão (Kitchenham e Charters, 2007). Porém, visto que o processo de seleção dos estudos foi realizado por apenas um pesquisador, existe a possibilidade de enviesamento na seleção em virtude de opiniões pessoais. Para tratar esta ameaça o processo de seleção foi tratado com o máximo de critério, neutralidade e imparcialidade, dedicando-se um tempo de leitura e reflexão acima da média para aqueles estudos que geraram mais incertezas. Além disto, as discussões com outros pesquisadores eram frequentes, sendo que em mais de um caso outro pesquisador foi consultado como forma de obter uma opinião a respeito da inclusão ou exclusão de algum dos estudos encontrados.

A avaliação da qualidade também pode ser considerada uma ameaça à validade, uma vez que, assim como o processo de seleção, foi realizada por apenas um pesquisador, o que pode ter resultado em uma avaliação enviesada dos estudos selecionados. Visando atenuar essa ameaça, o processo de avaliação utilizado não é de autoria do pesquisador em questão, pois de outra forma o viés poderia existir já na confecção do método, onde o pesquisador elaborasse as perguntas de forma a buscar um certo tipo de resposta.

3.5 Considerações Finais

Giardino *et al.* (2014) apontam que para ter sucesso, *startups* precisam de métodos transferíveis e confiáveis, ou seja, que são fáceis de colocar em prática e que tragram resultados rápidos. A natureza de incerteza que compõe uma *startup* exige que os processos sejam desenvolvidos com velocidade, assim, as atividades devem ser desenhadas para permitir flexibilidade e reatividade. Inclusive, diante deste contexto, é preferível que se falhe rápido e consequentemente se aprenda rápido. A partir disto, Giardino *et al.* (2014) citam alguns pontos que devem ser atendidos para permitir esta velocidade, e consequentemente o sucesso, às *startups*:

- Uso de *frameworks* conhecidos, permitindo mudanças rápidas no produto para atender ao mercado.
- Uso de prototipagem evolucionária e experimentação por meio de componentes existentes.
- Validação contínua com grupos de usuários chave.
- Entrega de valor contínua focando em funcionalidades chaves que engajem o usuário.
- Empoderamento do time como forma de aumentar o desempenho e o sucesso.
- Utilização de métricas para aprender por meio dos usuários.
- Uso de ferramentas fáceis de implementar para tornar o desenvolvimento do produto mais dinâmico.

O número de estudos no âmbito do desenvolvimento de software em *startups* que apresentam resultados transferíveis para a indústria ainda é baixo. Dentre os itens levantados por Giardino *et al.* (2014), todos apresentam lacunas e se beneficiariam de mais estudos empíricos. São necessários estudos que busquem soluções para atender a estes itens, a fim de criar cenários onde técnicas, práticas e ferramentas possam ser validadas e, possivelmente a partir dos dados levantados, criarem-se guias de métodos confiáveis que podem ser adotados. Há também a necessidade de estudos comparativos, que busquem por exemplo avaliar os *frameworks* de desenvolvimento mais populares do mercado, e verificar sua validade para a aplicação no contexto de uma *startup*. O mesmo pode ser dito para os demais pontos levantados por Giardino *et al.* (2014), mas principalmente quanto a questão da capacitação da equipe, já que este é um dos pontos menos explorados entre os citados.

Neste sentido, este trabalho buscou mapear as técnicas, práticas e ferramentas utilizadas no desenvolvimento de software em *startups* e assim criar uma base de conhecimento que pode servir como guia para a tomada de decisão de *startups*. Ao identificar um total de 24 técnicas, 31 práticas e 37 ferramentas, pode-se dizer que o objetivo foi alcançado. Apesar da necessidade de um número maior de estudos empíricos para que a área de pesquisa esteja suficientemente madura, as informações coletadas neste mapeamento constituem um ponto de partida para os profissionais de *startups*.

Em comparação aos mapeamentos apresentados por Klotins *et al.* (2015) e Paternoster *et al.* (2014), o primeiro analisa as áreas de conhecimento de Engenharia de Software que são usadas em *startups* enquanto o segundo é mais focado nos aspectos organizacionais

que compõem o desenvolvimento de software em *startups*, sendo que quando descreve a respeito das práticas encontradas o faz por meio de um panorama geral, já este mapeamento busca preencher a lacuna apontada por Sutton (2000) ao focar no aspecto técnico, identificando as técnicas, práticas e ferramentas que dão suporte ao desenvolvimento de software nas *startups*, além disto, por considerar apenas estudos realizados nos últimos dez anos, espera-se que os achados do mapeamento tenham maior relevância para o cenário atual das *startups*.

Como trabalhos futuros, há a possibilidade explorar os pontos apontados por Giardino *et al.* (2014), dos quais estes, de acordo com o autor, são os principais fatores para o sucesso de uma *startup*. Entre estes pontos, explorar os *frameworks* e ferramentas utilizados no desenvolvimento, seja por meio de avaliações ou de estudos comparativos, são áreas com grande potencial para pesquisa pois, conforme verificado por meio do mapeamento, são áreas ainda pouco exploradas e com alta possibilidade de transferência para a indústria. O mesmo pode ser dito a respeito do uso de métricas para aprender por meio dos usuários. Um forte indicador para isto é que atividades relacionadas ao uso de métricas e aprendizagem por meio do usuário aparecem em 15 dos 19 estudos em relação as técnicas extraídas, além de 4 ocorrências entre as práticas e 2 ocorrências entre as ferramentas, demonstrando que esta atividade é ponto chave no desenvolvimento de software em *startups*.

4

Estudo Empírico: Survey Exploratório

4.1 Considerações Iniciais

Neste capítulo são apresentados os passos e os resultados do estudo empírico conduzido no trabalho, sendo este um *survey* exploratório. A pesquisa foi realizada com *startups* brasileiras de desenvolvimento de software, sendo observadas as suas características, as características do indivíduo que respondeu o questionário e aspectos técnicos relacionados ao desenvolvimento de software dentro dessas organizações.

4.2 Condução do Estudo Empírico

Os estudos empíricos são formas de adquirir conhecimento, buscando descrever, prever, explorar e avaliar fenômenos a partir das evidências obtidas por meio de observação sistemática ou experimento, ao invés de lógica dedutiva (Sjoberg *et al.*, 2007). De acordo com Wohlin *et al.* (2012), esses estudos permitem qualificar as tarefas desempenhadas por pessoas de forma sistemática, quantificável, controlada e disciplinada.

Esses estudos são importantes, pois possibilitam avaliar tecnologias da Engenharia de Software e determinar em quais cenários e em quais tarefas as mesmas são úteis, e assim permitem criar o conhecimento necessário para apoiar os pesquisadores e profissionais da indústria na decisão de quais processos, técnicas e ferramentas adotar. Como, por exemplo, durante o processo de desenvolvimento de software de uma *startup* (Sjøberg *et al.*, 2005).

O estudo empírico neste trabalho foi conduzido por meio de um *survey*. O *survey* é um método de pesquisa quantitativo frequentemente utilizado em pesquisas empíricas em

Engenharia de Software como forma de obter dados de uma variedade de fontes (Wohlin e Aurum, 2015). O *survey* foi conduzido buscando compreender o desenvolvimento de software em *startups* a partir das práticas, técnicas, ferramentas e metodologias adotadas. Este *survey* pode ser classificado como exploratório (Wohlin e Aurum, 2015), pois busca obter dados da indústria visando explorar uma área ainda pouco estudada.

Em relação a técnica utilizada para a coleta dos dados, a mesma é caracterizada como de corte transversal, uma vez que a compilação dos dados ocorreu em um período específico de tempo (Sampieri *et al.*, 2010). O *survey* foi conduzido de acordo com o método proposto por Forza (2002), que é subdivido em quatro fases: Planejamento, Experimento Piloto, Coleta dos Dados e Análise dos Resultados.

4.2.1 Planejamento

A fase de Planejamento visa a prevenção de problemas durante as fases seguintes do *survey* e busca garantir a qualidade do processo de pesquisa. Esta fase é composta das seguintes atividades: seleção de contexto, seleção dos participantes e desenvolvimento dos instrumentos de pesquisa.

4.2.1.1 Seleção de Contexto

O *survey* considerou qualquer *startup* do Brasil que tenham entre as suas atividades principais o desenvolvimento de software. Não foi definida nenhuma restrição em relação ao domínio de atuação da *startup*, visto que durante a elaboração do referencial teórico e da condução do mapeamento sistemático não foram encontrados indícios de que o domínio pudesse ter alguma influência.

4.2.1.2 Seleção dos Participantes

A lista de *startups* foi obtida por meio de sites de associações e agrupamentos, como por exemplo a Associação Brasileira de *Startups*[1]. Todas as *startups* encontradas, considerando o contexto definido, foram convidadas via e-mail (Apêndice B) a participarem da pesquisa.

4.2.1.3 Desenvolvimento dos Instrumentos de Pesquisa

O instrumento de pesquisa consiste de dois documentos, sendo eles a Carta de Apresentação e o Questionário. A Carta de Apresentação (verificar Apêndice B) buscou

[1] `https://abstartups.com.br/`

apresentar formalmente os pesquisadores e o propósito da pesquisa para as *startups*, visando obter sua colaboração para responder o questionário. O Questionário (verificar Apêndice C) apresenta as questões que compõem o *survey*, desenvolvidas de acordo o propósito da pesquisa, ele está estruturado em 3 seções: i) caracterização do respondente, buscando identificar o perfil do respondente do questionário; ii) caracterização da *startup*, que visa identificar o perfil da organização e em qual estágio de vida se encontra; iii) desenvolvimento de software, que tem por finalidade identificar os métodos, práticas, técnicas e ferramentas utilizadas durante as atividades de desenvolvimento de software dentro da *startup*.

4.2.1.3.1 Confiabilidade do Questionário

A confiabilidade das questões do questionário foram avaliadas por meio do coeficiente α de Cronbach. A confiabilidade reflete o quanto os valores observados estão correlacionados aos valores verdadeiros e permite observar o quanto uma medida está livre de variância e de erros aleatórios (Carmines e Zeller, 1979; Crocker e Algina, 1986). O α mede a correlação entre respostas em um questionário por meio da análise do perfil das respostas dadas pelos respondentes. Seu cálculo é realizado a partir da variância dos itens individuais e das covariâncias entre os itens, conforme apresenta a Figura - 4.1.

Figura 4.1: Equação para cálculo do α de Cronbach

$$\alpha = \left(\frac{k}{k-1}\right)\left(1 - \frac{\sum_{i=1}^{k} S_i^2}{S_t^2}\right)$$

Onde:

- *k:* número de itens do questionário
- S_i^2: variância do item i
- S_t^2: variância total do questionário

Não existe um consenso em relação à interpretação da confiabilidade obtida a partir do valor do coeficiente α de Cronbach. Para alguns autores, um valor aceitável para o α de Cronbach é de 0,70 (Gliem e Gliem, 2003; Streiner, 2003), sendo que abaixo desse valor a consistência interna é considerada baixa. O coeficiente α de Cronbach obtido para o questionário foi de 0,4717, que pode ser considerado como um valor moderado

(Landis e Koch, 1977). Este valor pode ter sido afetado pela heterogeneidade dos *scores* das questões, uma vez que algumas questões possuem tipos de respostas diferentes.

4.2.2 Experimento Piloto

O Experimento Piloto visa analisar os processos de medição do *survey* e verificar a viabilidade da sua aplicação, ou seja, seu objetivo é testar o que foi planejado. Sendo assim, o Experimento Piloto foi conduzido com o objetivo de analisar o comportamento do instrumento de pesquisa em uma situação real de coleta de dados, possibilitando verificar se as questões estavam compreensíveis, se as opções de respostas estavam completas e se o mesmo estava suficientemente sucinto, de forma a facilitar que fosse respondido e, assim, aumentar a adesão. O experimento piloto foi conduzido em ambiente acadêmico com três pesquisadores da área de Engenharia de Software, que possuem, respectivamente, doze, doze e quinze anos de experiência na academia.

O primeiro dos pesquisadores, da Universidade Estadual de Maringá, atua na área de apoio à tomada de decisão, englobando as áreas de pesquisa operacional e gestão de tecnologia da informação, e possui interesse nas áreas de otimização, desenvolvimento distribuído de software, mapeamento e melhoria de processos, capitalização de conhecimento e qualidade de software. O segundo pesquisador, da Universidade do Pernambuco, atua nas áreas de desenvolvimento e inovação em Engenharia de Software, melhoria de processos de software e gerência de projetos e métodos ágeis, e também atua como consultor, implementador e avaliador dos modelos MPT.BR e implementador dos modelos MR-MPS-SV. O terceiro dos pesquisadores, também da Universidade Estadual de Maringá, atua nas áreas de Engenharia de Software, computação aplicada à saúde e desenvolvimento distribuído de software, e atualmente é bolsista CNPq de Desenvolvimento Tecnológico e Científico, categoria DTC-A.

Após a condução do experimento piloto, alguns ajustes foram realizados para melhorar a organização do questionário, adequando a ordenação das questões a fim de melhorar o fluxo do questionário e realizando ajustes na escrita das questões para deixá-las mais objetivas, e, também, foram incluídas duas novas questões que não haviam sido consideradas inicialmente, sendo uma em relação aos tipos de testes utilizados pela *startup* e outra a respeito das ferramentas de controle de versão e configuração que são utilizadas pela *startup*. O experimento piloto também permitiu estimar o tempo médio para responder o questionário em 15 minutos.

4.2.3 Coleta de Dados

A Coleta de Dados tem por objetivo reunir as informações relevantes ao *survey*, proporcionando as informações necessárias para o cumprimento do objetivo da pesquisa. A Coleta de Dados foi realizada utilizando a ferramenta de formulários do Google (`forms.google.com`). A escolha de uma ferramenta on-line para a coleta dos dados permite uma maior abrangência, principalmente geográfica, pois qualquer *startup* com conexão a Internet tem acesso ao formulário, e facilita a sua divulgação, visto que *startups* que responderam o questionário podem encaminhar o link de acesso do mesmo para outras *startups* que ainda não participaram da pesquisa.

O procedimento de coleta consistiu do envio de e-mail para as *startups* encontradas contendo a carta de apresentação e o link de acesso ao formulário (`https://goo.gl/forms/QqksndXyuNCSRPFw1`). O formulário ficou disponível por um período de 120 dias e as solicitações para participar da pesquisa foram enviadas duas vezes para cada *startup*.

4.2.4 Análise dos Resultados

A Análise dos Resultados é fundamental para os objetivos da pesquisa, pois permite investigar os dados por diversos ângulos e também compará-los, de forma a produzir novos conhecimentos. Neste *survey*, a Análise dos Resultados foi realizada por meio da ferramenta de planilhas LibreOffice Calc. A partir da ferramenta os dados foram estruturados e entrelaçados conforme as visões desejadas, permitindo destacar os fatos mais importantes e possibilitando o compreendimento das informações coletadas.

4.2.5 Demografia da Pesquisa

A pesquisa foi conduzida com *startups* brasileiras de desenvolvimento de software, em que o total de participantes foi de 104 *startups* diferentes. Após a exclusão dos *outliers* (dados inconsistentes), sendo 3 deles por respostas incompletas e 1 por duplicidade, o total da amostra foi reduzido para 100 participantes. O questionário foi enviado para 627 *startups* diferentes, dessa forma a taxa de resposta da pesquisa foi de 16,58%.

Conforme demonstra a Figura - 4.2, a distribuição dos respondentes por Estado é de 29 em São Paulo (29%), 26 no Paraná (26%), 24 em Santa Catarina (24%), 6 no Rio de Janeiro (6%), 4 em Pernambuco (4%), 3 em Minas Gerais (3%) e 2 no Amazonas (2%). Amapá, Bahia, Distrito Federal, Mato Grosso do Sul, Rio Grande do Sul e Roraima aparecem com 1 respondente cada, representando juntos 6% do total.

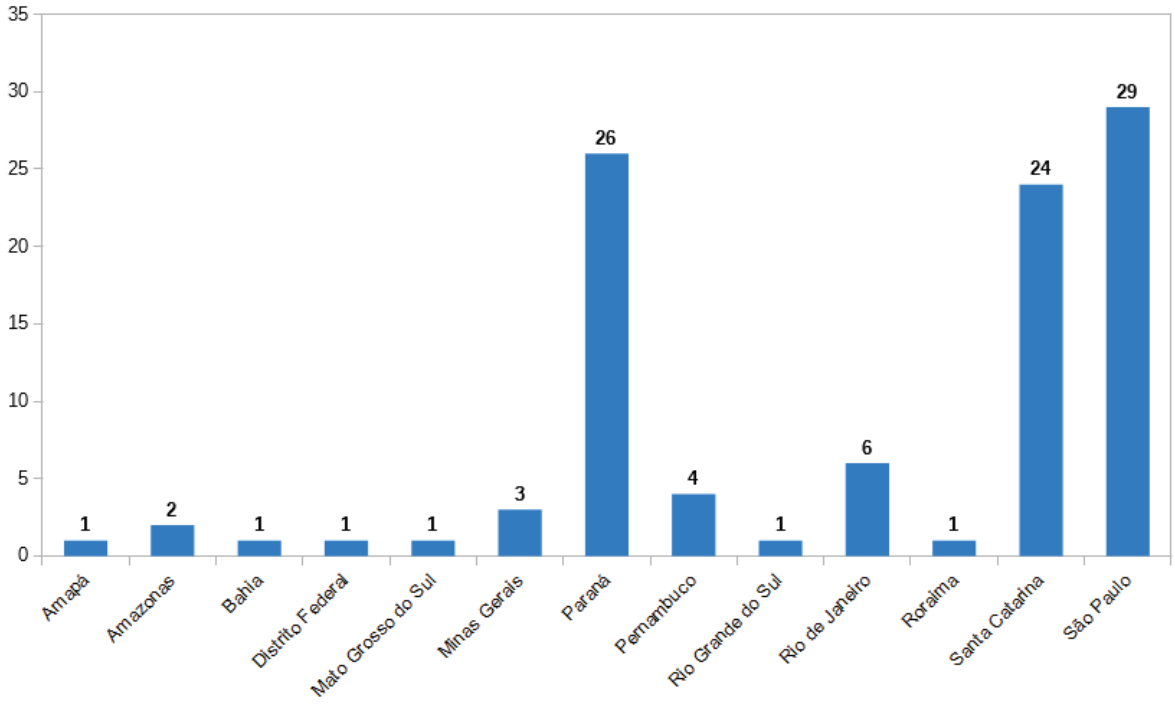

Figura 4.2: Distribuição dos respondentes por Estado

4.2.6 Caracterização de Startup

Essa seção apresenta o ano de fundação da *startup*, qual o seu nível de maturidade, se a *startup* faz parte de algum grupo ou rede de colaboração (ecossistema, incubadora, arranjos produtivos, etc), se a *startup* possui algum tipo de investidor externo e quantos colaboradores a *startup* possui. Esses elementos visam caracterizar o perfil das *startups* que responderam o questionário.

A Figura - 4.3 apresenta a distribuição das *startups* de acordo com o seu respectivo ano de fundação. O ano de 2016 é o que apresenta o maior número de *startups* fundadas, com 23, seguido de 2015, com 21, e 2013, com 13. Os anos de 2014 e 2017 aparecem ambos com 11 *startups* fundadas. Em seguida tem-se 2011 com 6, 2010 com 5, 2012 com 4 e 2008 com 3. Os anos de 2000, 2006 e 2009 aparecem com 1 *startup* fundada cada.

Por meio da linha de tendência é possível verificar o crescimento exponencial do número de *startups* fundadas nos últimos 10 anos até agora. É importante salientar que o questionário foi disponibilizado em meados de julho de 2017, ficando disponível até meados de outubro de 2017, sendo assim, é possível que a quantidade de *startups* fundadas neste ano esteja incompleta.

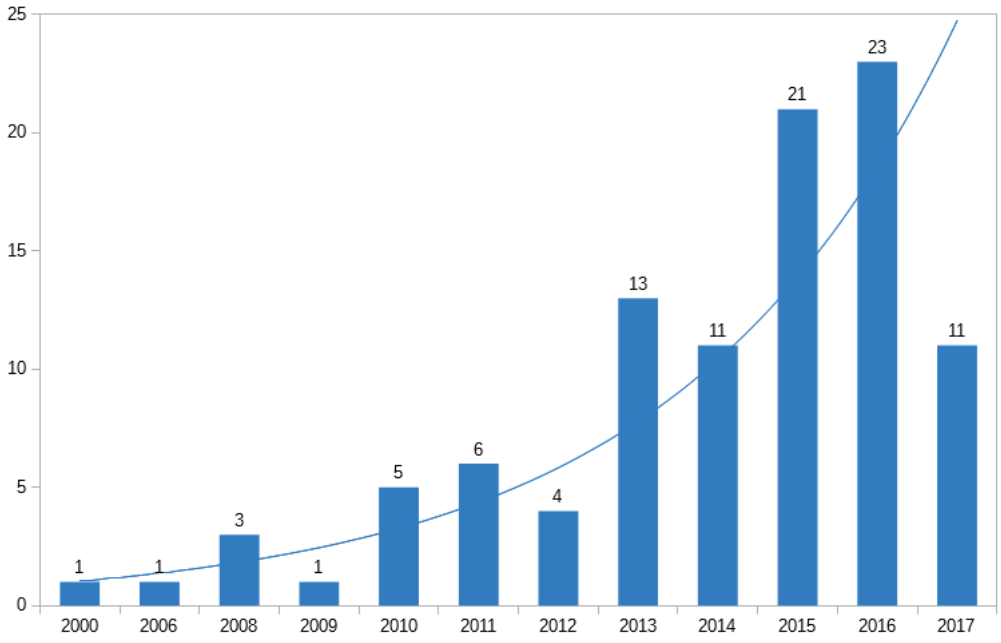

Figura 4.3: Ano de fundação das *startups*

Em relação ao nível de maturidade das *startups* participantes da pesquisa, conforme apresenta a Figura - 4.4, cerca de metade das *startups* participantes se encontram em fase de crescimento (49%), enquanto que as restantes estão dividas entre em fase de início (26%) e em estabilização (25%). De acordo com Crowne (2002), a fase de início é definida como a fase entre a concepção do produto até a realização da primeira venda. A fase de estabilização começa no momento em que o primeiro consumidor recebe o produto, durando até o momento em que o produto está estável o suficiente para ser vendido a um novo consumidor sem causar sobrecarga no desenvolvimento do produto (Nguyen-Duc *et al.*, 2016). A fase de crescimento se inicia exatamente ao fim da fase de estabilização e acaba quando as taxas de crescimento e fatia de mercado estão estáveis e todos os processos necessários para o desenvolvimento e venda do produto estão estabilizados (Klepper, 1996).

A Figura - 4.4 permite concluir que quase metade das *startups* participantes se encontram em fase de crescimento, ou seja, estas *startups* são capazes de entregar seus produtos para mais de um cliente e em breve irão se estabilizar, tornando-se organizações maduras. Por outro lado, a outra metade das *startups* participantes estão bem divididas entre as fases de início e de estabilização. Estas em fase de início ainda não entregaram um produto, ou seja, estão na fase de concepção da ideia. Enquanto as que estão em fase de estabilização já conseguiram entregar um produto e agora buscam expandir as suas vendas para alcançarem a fase de crescimento.

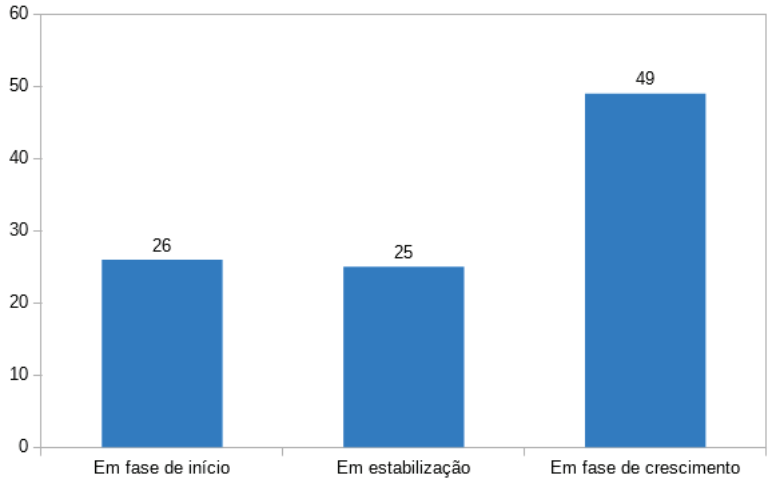

Figura 4.4: Nível de maturidade das *startups*

A Tabela - 4.1 apresenta uma relação entre o ano de fundação das *startups* e o seu nível de maturidade, demonstrando por ano de fundação o percentual de *startups* que se encontra em cada um dos níveis de maturidade. A tabela permite verificar que todas as *startups* fundadas entre 2000 e 2010 já se encontram em fase de crescimento, ou seja, são as *startups* mais próximas de se tornarem organizações maduras. A partir de 2011 o número de *startups* que estão em fase de crescimento diminui, representando 66,67% do total de *startups* fundadas no ano, as restantes estão em fase de estabilização, que representa 33,33% do total. Nenhuma das *startups* fundadas em 2011 estão em fase de início, o que também ocorre com aquelas fundadas em 2012. Neste ano existe um equilíbrio, onde metade das *startups* fundadas em 2012 estão em fase de estabilização e a outra metade se encontra em fase de crescimento.

A partir de 2013 surgem *startups* que estão em fase de início, com 23,07% do total, o mesmo percentual de organizações que se encontram em fase de estabilização. Enquanto aquelas que estão em fase de crescimento representam 53,86% do total que foram fundadas no ano. Das fundadas em 2014, a quantidade que se encontra em fase de início é menor, com 9,09%, enquanto aquelas em fase de estabilização demonstra um percentual parecido com o do ano anterior, com 27,27%, sendo que a maioria, 63,64% estão em fase de crescimento. Em 2015, o percentual de *startups* em fase de crescimento ainda é maioria, com 54,55%, e as que estão em fase de estabilização se mantém, representando 27,27%. As que estão em fase de início representam 18,18%. A partir de 2016 o percentual de *startups* em fase de crescimento cai, representando 27,28%, e ocorre um equilíbrio entre aquelas que estão em fase de estabilização e fase de início, ambas representando 36,36%. Por fim, das fundadas em 2017, apenas 9,09% estão em fase de estabilização, sendo que a grande

maioria se encontra em fase de início, com 90,91%. Nenhuma das *startups* fundadas neste ano se encontra em fase de crescimento.

Tabela 4.1: Relação entre o ano de fundação e o nível de maturidade

Ano	Em fase de início	Em fase de estabilização	Em fase de crescimento
2000	-	-	100%
2006	-	-	100%
2008	-	-	100%
2009	-	-	100%
2010	-	-	100%
2011	-	33,33%	66,67%
2012	-	50,00%	50,00%
2013	23,07%	23,07%	53,86%
2014	9,09%	27,27%	63,64%
2015	18,18%	27,27%	54,55%
2016	36,36%	36,36%	27,28%
2017	90,91%	9,09%	-

Este dados permitem verificar que na maioria dos casos, as *startups* levam cerca de um ano até que comecem a sair da fase de início para a fase de estabilização. No segundo ano o número já apontam igualdade entre as duas primeiras fases, com um percentual menor já aparecendo em fase de crescimento. Com o decorrer dos anos é possível verificar que o número de *startups* em fase de início vai diminuindo, com exceção a 2013, enquanto o número de *startups* em fase de estabilização se mantém parecido, com exceção do primeiro ano de fundação. Também é possível dizer que após seis anos de fundação a maioria das *startups*, entre aquelas que sobreviveram, se encontram em fase de crescimento e todas a partir do sétimo ano de fundação.

Ao serem questionadas se fazem parte de algum grupo ou rede de colaboração (ecossistema, incubadora, arranjos produtivos, etc), 54% das *startups* responderam que sim, enquanto 46% responderam que não fazem parte. Em outra questão, em relação a *startup* possuir ou não algum tipo de investidor externo, 34% indicaram que sim, enquanto as que não possuem constam com 66%.

A Tabela - 4.2 demonstra a relação entre o nível de maturidade em que as *startups* se encontram e o percentual daquelas que fazem parte ou não de um grupo ou rede de colaboração. Entre aquelas que estão em fase de início, 53,85% fazem parte de um grupo ou rede de colaboração, enquanto 46,15% não fazem. Entre aquelas que estão em fase de estabilização é de 56% o total que fazem parte e de 44% as que não fazem. Por fim, entre

as que estão em fase de crescimento, as que fazem parte representam 59,19% e as que não fazem 40,81%.

Tabela 4.2: *Startups* que fazem parte de grupo ou rede de colaboração em relação ao nível de maturidade

	Faz parte de algum grupo ou rede de colaboração?	%
Em fase de início	Sim	53,85%
	Não	46,15%
Em fase de estabilização	Sim	56,00%
	Não	44,00%
Em fase de crescimento	Sim	59,19%
	Não	40,81%

Estes dados permitem verificar que em qualquer nível de maturidade em que as *startups* se encontram o número daquelas que fazem parte de um grupo ou rede de colaboração é maior do que daquelas que não fazem parte, porém, em todos os casos a diferença é pequena. Os dados não permitem traçar uma conclusão a respeito do papel que esta participação tem para o sucesso das *startups* participantes da pesquisa, porém, é possível afirmar que a participação traz benefícios como, maior facilidade de comunicação entre as organizações que fazem parte do grupo e assim facilitando a troca de conhecimento, facilitação da entrada de matérias-primas e concentração de mão-de-obra especializada devido a proximidade geográfica (Santos, 2016).

A Tabela - 4.3 apresenta a relação entre o nível de maturidade das *startups* e os números daquelas que possuem ou não algum tipo de investidor externo. Entre as que estão em fase de início é de 30,77% as que possuem algum tipo de investidor externo e de 69,23% as que não possuem. Entre aquelas que estão em fase de estabilização, as que possuem representam apenas 24% e as que não possuem 76%. Entre aquelas que estão em fase de crescimento, o percentual das que possuem é de 40,81% e das que não possuem é de 59,19%.

Independentemente do nível de maturidade da *startup*, o percentual das que não possuem algum tipo de investidor externo é sempre maior do que daquelas que possuem, principalmente entre aquelas que se encontram em fase de estabilização, onde a diferença é maior, sendo que é justamente nesta fase que geralmente o empreendedor precisa buscar por investidores externos para financiar suas operações (Crowne, 2002). Sendo assim, os dados demonstram que o investimento externo não é um dos principais fatores para o sucesso da *startup*, porém, talvez seja na fase de crescimento que essas organizações precisem buscá-lo, servindo como forma de alavancar seu crescimento para se tornarem

organizações maduras, visto que nesta fase o percentual daquelas que possuem algum tipo de investidor externo é maior do que nas fases anteriores.

Tabela 4.3: *Startups* que possuem investidor externo em relação ao nível de maturidade

	Possui algum tipo de investidor externo?	%
Em fase de início	Sim	30,77%
	Não	69,23%
Em fase de estabilização	Sim	24,00%
	Não	76,00%
Em fase de crescimento	Sim	40,81%
	Não	59,19%

Por fim, a Figura - 4.5 apresenta o número de colaboradores das *startups*. Como forma de classificação, as faixas de número de colaboradores por *startup* foram definidas de acordo com os portes utilizados pelo Serviço Brasileiro de Apoio às Micro e Pequenas Empresas (SEBRAE), onde empresas até 9 colaboradores são consideradas como microempresas, de 10 até 49 colaboradores são consideradas de pequeno porte, de 50 a 99 colaboradores são consideradas de porte médio e de 100 ou mais colaboradores são consideradas como grande.

Dessa forma, a Figura - 4.5 demonstra que 73% das *startups* respondentes se enquadram como microempresas, 20% se enquadram como empresas de pequeno porte, 6% se enquadram como empresas de porte médio e apenas 1% se encaixa na faixa de empresas de grande porte. Assim podemos verificar que a grande maioria das *startups* participantes é formada por microempresas, ou seja, possuem no máximo 9 colaboradores. Empresas de pequeno porte representam apenas 20% das participantes, enquanto empresas de maior porte, como médias e grandes, possuem um número ainda menor. Visto que o objetivo do SEBRAE é justamente auxiliar as micro e pequenas empresas, o serviço pode desempenhar um papel fundamental no desenvolvimento da maioria das *startups* brasileiras.

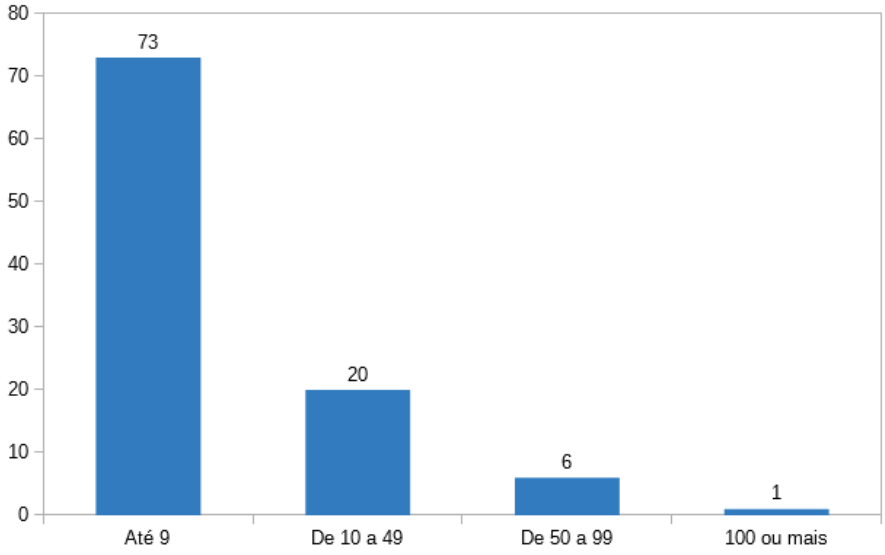

Figura 4.5: Número de colaboradores das *startups*

4.2.7 Caracterização do Respondente

Essa seção apresenta o tempo de experiência com desenvolvimento de software, o tempo de experiência com *startups* e o grau de formação do respondente. Esses elementos visam caracterizar o perfil do respondente do questionário.

A Figura - 4.6 apresenta o tempo de experiência dos respondentes com desenvolvimento de software. É possível verificar que 72% possuem mais do que 5 anos de experiência, indicando que a grande maioria dos envolvidos nas *startups* possuem uma quantidade considerável de experiência com desenvolvimento de software. Do restante, 12% possuem entre 3 e 5 anos de experiência, 10% possuem entre 1 e 3 anos e 6% possuem menos do que 1 ano de experiência.

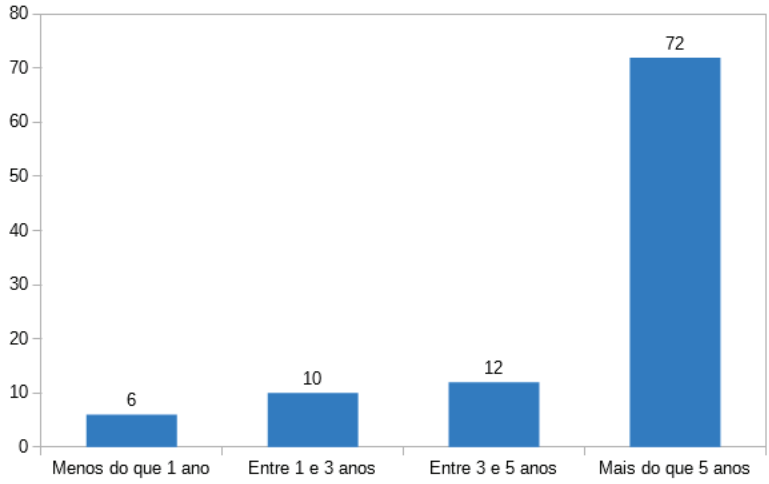

Figura 4.6: Tempo de experiência com desenvolvimento de software

Quanto ao tempo de experiência com *startups*, conforme demonstra a Figura - 4.7, existe um equilíbrio entre as faixas de mais do que 5 anos (29%), entre 3 e 5 anos (28%) e entre 1 e 3 anos (28%). O restante (15%) apresenta menos do que 1 ano de experiência com *startups*.

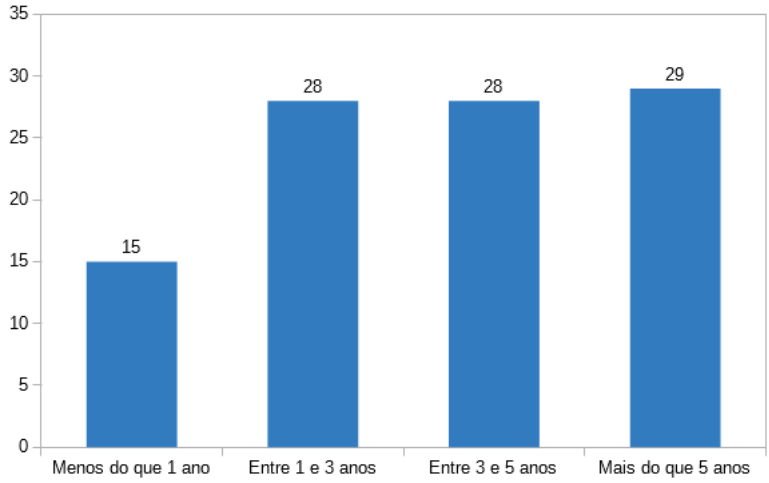

Figura 4.7: Tempo de experiência com *startups*

No que se refere ao grau de formação, nenhum dos respondentes indicou Curso Técnico, sendo que 9% é Graduando, 39% é Graduado, 29% é Especialista, 6% Mestrando, 8% Mestre, 2% Doutorando e 7% Doutor, conforme ilustrado pela Figura - 4.8. Isto demonstra que boa parte das pessoas que estão envolvidas com *startups* são recém-formadas ou acabaram de terminar uma pós-graduação, ou então são pessoas que decidiram não investir em uma carreira acadêmica além da graduação e da especialização.

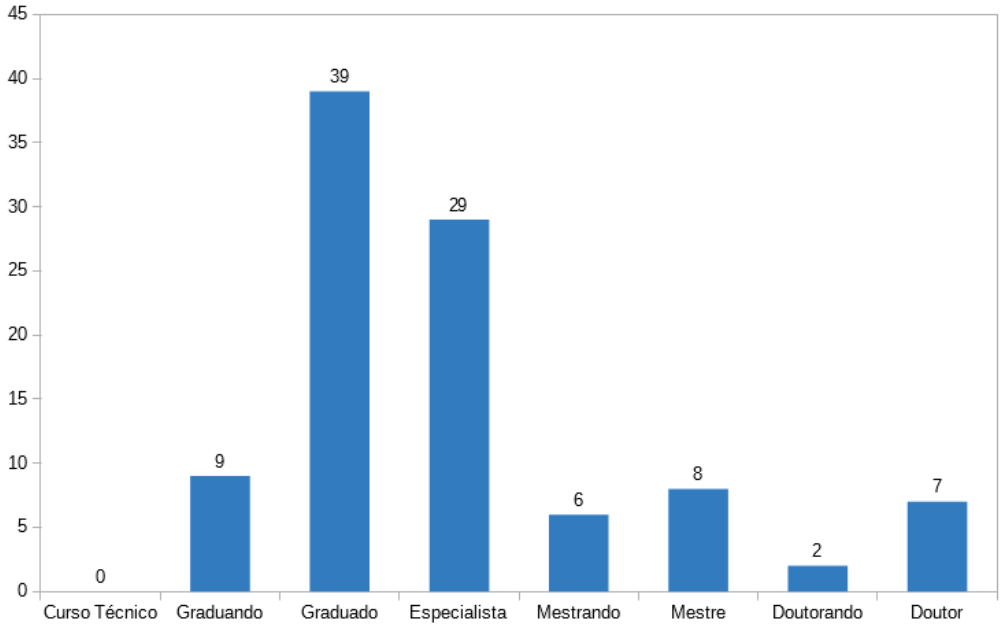

Figura 4.8: Grau de formação dos respondentes

4.2.8 Desenvolvimento de Software em Startups

Essa seção apresenta se a *startup* utiliza metodologias ágeis e/ou metodologias tradicionais e quais, apresenta também quais as práticas e técnicas são utilizadas, quais tipos de testes, quais ferramentas de controle de versão e configuração, quais ferramentas de gerenciamento de projeto, quais suítes/plataformas de desenvolvimento e quais ferramentas de gestão do conhecimento as *startups* utilizam. Esses elementos visam caracterizar as atividades de desenvolvimento de software das *startups* que responderam o questionário.

Em relação a utilização de metodologias ágeis, das *startups* participantes, a grande maioria (86%) indicou utilizar, enquanto o restante (14%) indicou que não utiliza. Os dados demonstram uma maior tendência de adoção dos métodos ágeis, conforme outros autores também verificaram (Cohen *et al.*, 2003; Qumer e Henderson-Sellers, 2008; Taromirad e Ramsin, 2008; West *et al.*, 2010), inclusive no Brasil (Melo *et al.*, 2013). A Figura - 4.9 apresenta as metodologias ágeis utilizadas pelas *startups*. A metodologia Scrum foi indicada por 65 *startups*, seguida pela metodologia *Lean/Lean Startup* que foi indicada por 53, a seguir aparece a metodologia *Test Driven Development* (TDD) com 18 indicações, *Extreme Programming* (XP) com 10 indicações, *Feature Driven Development* (FDD) com 3 indicações e a metodologia *Experiment Driven Development* (EDD) com 2 indicações. Além disso, 8 *startups* indicaram utilizar outras metodologias, sendo que 7 delas indicaram o Kanban e 1 indicou *Extreme Go Horse*.

Para efeitos da pesquisa o Kanban é considerado como um técnica, e não como uma metodologia, inclusive sendo uma das opções em outra questão do questionário. Entretanto, não é incomum que a técnica seja encarada como uma metodologia por diversas organizações. A "metodologia" *Extreme Go Horse* é na realidade uma metodologia fictícia cunhada pela indústria como forma de satirizar as organizações que não utilizam um processo de desenvolvimento, onde as atividades são realizadas sem qualquer tipo de planejamento, pulando de uma atividade para outra conforme as mesmas vão surgindo[2]

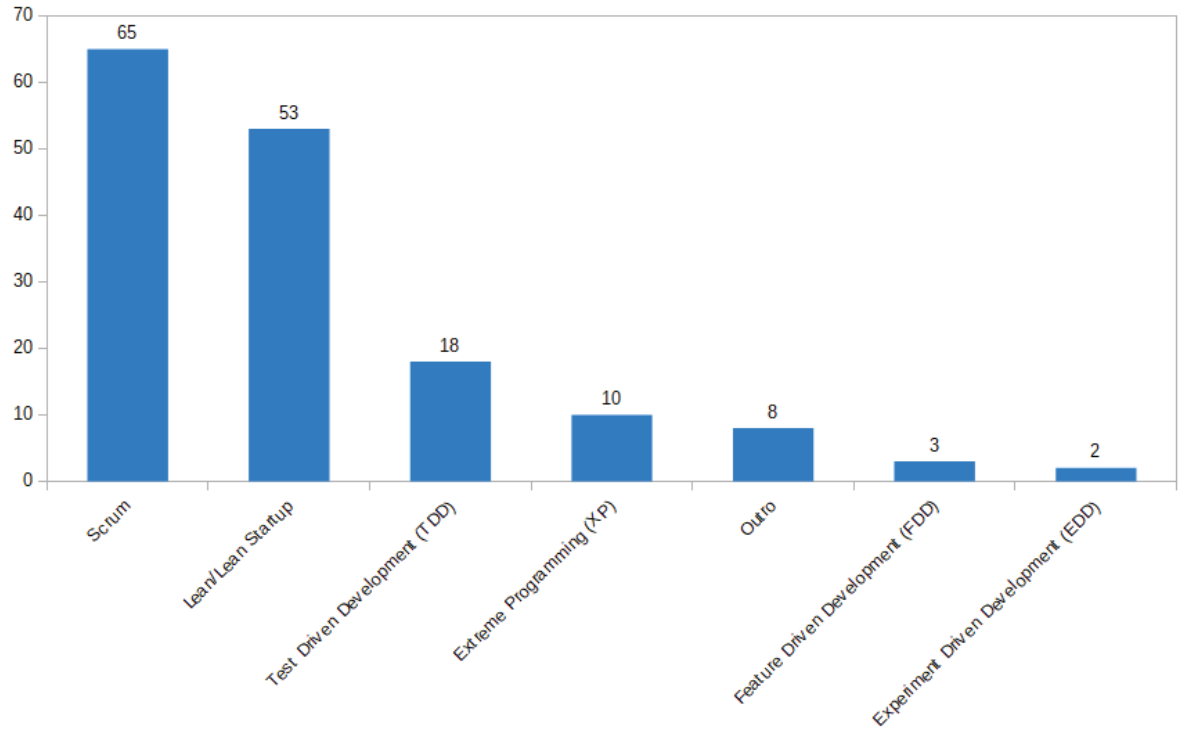

Figura 4.9: Metodologias ágeis utilizadas pelas *startups*

Quanto ao uso de métodos tradicionais, os resultados demonstram que, das *startups* participantes, apenas 14% utilizam métodos tradicionais, enquanto 86% não utilizam. Em conjunto com os dados a respeito do uso de métodos ágeis, os dados de uso de métodos tradicionais demonstram que aquelas *startups* que não fazem uso das metodologias ágeis fazem das metodologias tradicionais, e vice-versa. Na Figura - 4.10 são apresentadas quais metodologias tradicionais são utilizadas. O modelo em cascata aparece como mais utilizado, com 6 indicações, enquanto o *Rational Unified Process* (RUP) aparece com 4. Houve três *startups* que não informaram qual ou quais metodologias tradicionais utilizam

[2]`http://sou.gohorseprocess.com.br/extreme-go-horse-xgh/`. Acesso em: 16 de nov. de 2017.

e foram citadas 3 metodologias que não constavam entre as opções selecionáveis, sendo elas fluxogramas, canvas e sistema de gestão operacional.

Os fluxogramas são uma representação gráfica de um processo ou de um fluxo de trabalho, geralmente feitos por meio do uso de figuras geométricas e setas que conectam essas figuras, representando o fluxo de execução. Em relação ao canvas, não é possível definir se a *startup* utiliza algum modelo específico, como *Business Model Canvas* ou *Software Development Canvas*, porém, em linhas gerais, é possível definir que um canvas é um mapa visual de itens, onde estes itens estão particionados em blocos, e dentro destes blocos estão contidas informações referentes a estes itens. Quanto ao sistema de gestão operacional, não é possível definir do que se trata visto que é algo muito genérico, podendo ser um sistema próprio da organização, um sistema desenvolvido por terceiros e etc.

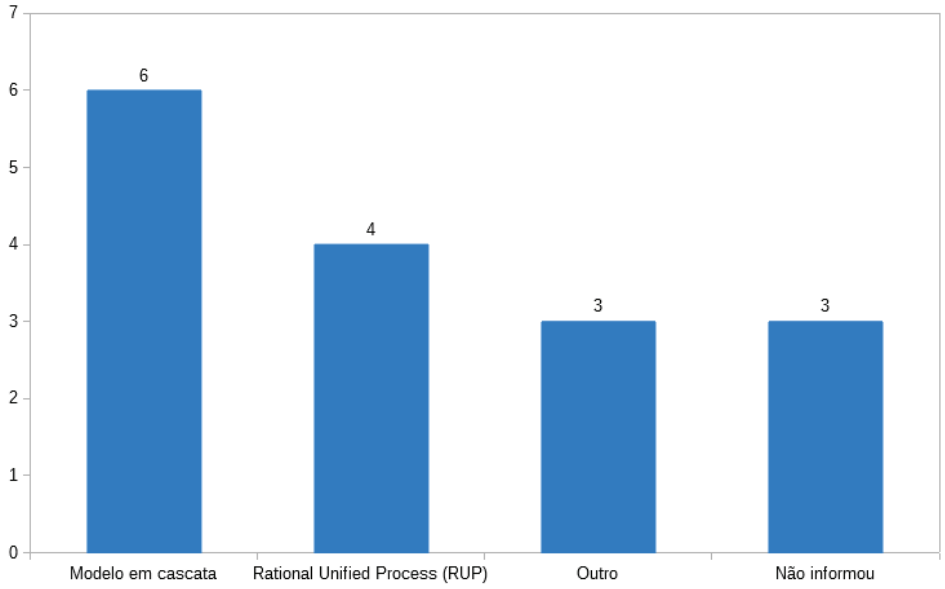

Figura 4.10: Metodologias tradicionais utilizadas pelas *startups*

A Figura - 4.11 apresenta quais são as práticas utilizadas pelas *startups* participantes da pesquisa. Por meio da figura podemos verificar que o uso de *frameworks* é a prática mais utilizada, contando com 79 indicações. Em seguida aparecem as práticas de entregas frequentes/contínuas e *backlog*, com 73 indicações cada. As demais práticas, com o seu respectivo número de indicações, são: padrões de código (64), *releases* curtos (61), empoderamento da equipe de desenvolvimento (50), capacitação dos membros da equipe (49), reuso de software (48), uso de *logs* e estatísticas (44), integração contínua (42), *bug-tracking* (40), uso de indicadores de desempenho (39), uso de soluções COTS ou *open*

source (38), prototipagem evolucionária (37), gerenciamento de requisitos (37) e cliente no local de desenvolvimento (12).

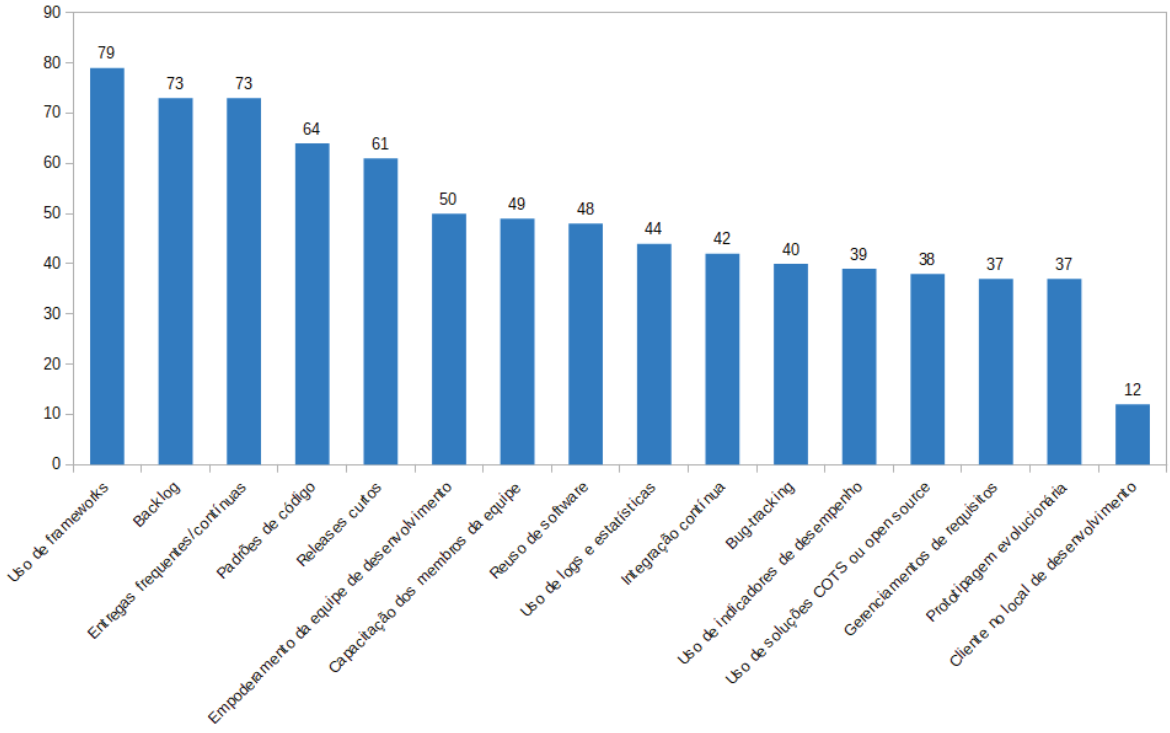

Figura 4.11: Práticas utilizadas pelas *startups*

O uso de *frameworks* está entre as práticas mais utilizadas, uma vez que agiliza o processo de desenvolvimento ao retirar a necessidade de implementar algo que já está pronto, sendo uma das práticas mais importantes para o sucesso de uma *startup* (Giardino *et al.*, 2014). Outra prática popular é a realização de entregas frequentes/contínuas, pois permitem um ciclo de desenvolvimento mais rápido, alinhado ao processo empregado pelas metodologias ágeis, além de acelerarem o ciclo de *feedback* com o cliente, permitindo que ajustes sejam feitos de forma muito mais rápida para atender as demandas dos clientes. O uso de *backlog* também está entre as práticas mais utilizadas, sendo que esta é uma das principais práticas da metodologia Scrum, a metodologia mais utilizada entre os respondentes.

Apesar de ser utilizada por apenas 10% das *startups* participantes, várias das práticas da metodologia XP são utilizadas frequentemente, inclusive algumas das que foram mais indicadas pelos respondentes como, integração contínua (42%), padrões de código (64%), *releases* curtos (61%). Outras práticas com uma boa representação são o uso de prototipagem evolucionária (37%) e o empoderamento da equipe de desenvolvimento,

sendo estas práticas essenciais para o sucesso da *startup* de acordo com Giardino *et al.* (2014).

A Figura - 4.12 traz os dados relacionados as técnicas utilizadas pelas *startups* respondentes. Conforme apresentado, a técnica com o maior número de indicações é o uso de MVP, com 81. Kanban é a segunda técnica com mais indicações, com um total de 68. Em seguida estão as técnicas: *code review* (35), programação em pares (23) e *Planning Game* (13). Foram indicadas 3 técnicas que não estavam entre as selecionáveis, por três *startups diferentes*, sendo elas: técnicas motivacionais, escopo e OKRs (*Objectives and Key Results*). Apenas uma *startup* indicou que não utiliza nenhuma das técnicas.

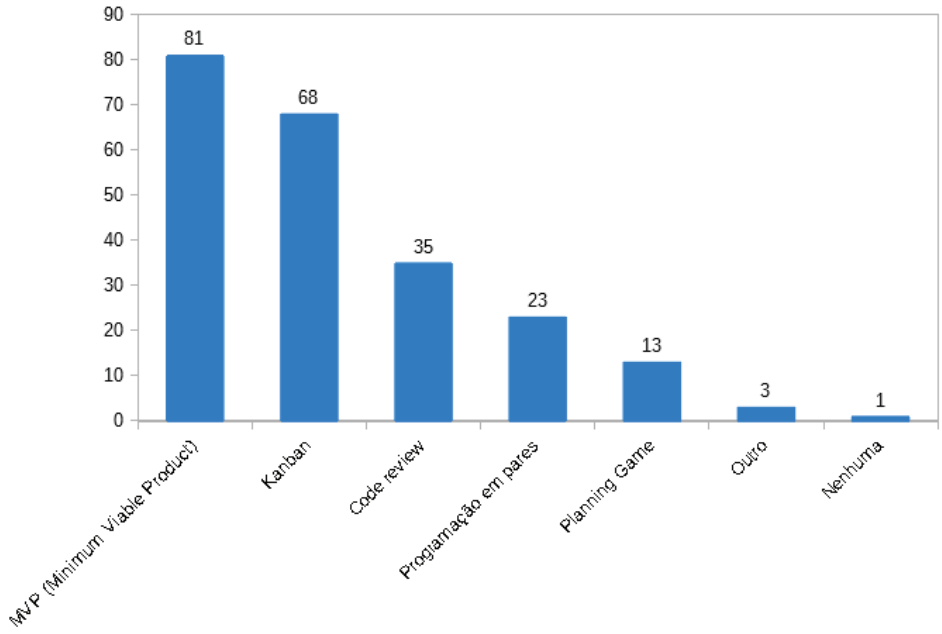

Figura 4.12: Técnicas utilizadas pelas *startups*

OKR é um *framework* de definição de metas criado pela Intel e guiado por objetivos e resultados-chave, onde os objetivos são as entregas desejadas que se quer alcançar e os resultados-chave são a forma de medir se o caminho para alcançar estes objetivos está correto.

Há também partes do XP entre as técnicas utilizadas, entre eles o *Planning Game*, *code review* e a programação em pares, demonstrando que mesmo aquelas *startups* que não utilizam o XP diretamente, fazem uso de suas técnicas alinhadas a outras metodologias. A programação em pares, que em alguns casos não é completamente adotada nem mesmo em ambientes que fazem uso do XP (Tingling e Saeed, 2007), aqui aparece com uma adoção até maior que a própria metodologia XP.

A técnica mais popular é o uso de MVP (81%), visto que cumpre um papel importante nos processos de uma *startup* (Duc e Abrahamsson, 2016), servindo como um artefato de design e como artefato reusável, além de ser ponto-chave da metodologia *Lean/Lean Startup* (Ries, 2011), onde tem parte importante no processo de aprendizagem e de melhoramento do produto. O mesmo pode ser dito em relação a metodologia EDD. Outra técnica popular é o Kanban, utilizado por 68% das *startups*. Isto se deve ao fato de que o Kanban é hoje adotado como complemento ao Scrum e outros métodos ágeis (Ahmad *et al.*, 2016), visto que a técnica proporciona velocidade na produção e um ciclo rápido e contínuo de *feedback* com o cliente (Al-Baik e Miller, 2015).

Na Figura - 4.13 são demonstrados os dados em relação aos tipos de testes utilizados pelas *startups*. A figura demonstra que 59% utilizam testes de usabilidade, 49% utilizam usuários de testes, 44% utilizam teste unitário, 38% fazem uso de testes de aceitação, 37% usam testes de integração e 25% utilizam teste contínuo. Duas *startups* indicaram outros testes, onde uma indicou o uso de teste de cobertura e a outra indicou testes, sem qualquer especificação. Apenas uma *startup* não informou quais tipos de testes utiliza.

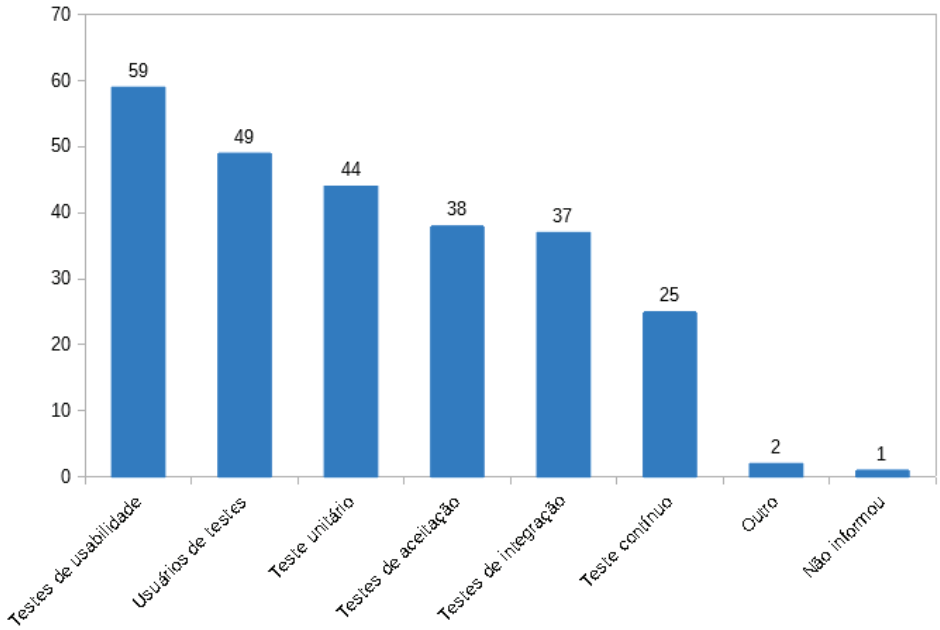

Figura 4.13: Tipos de testes utilizados pelas *startups*

Testes de usabilidade e usuários de testes estão entre as práticas de teste mais adotadas pois permitem ganhar *feedback* do cliente e assim realizar ajustes de acordo com o conhecimento adquirido. Para Giardino *et al.* (2014), a validação contínua com usuários chave está entre os principais fatores de sucesso para as *startups*, e estão em conformidade

com o ciclo de aprendizagem defendido por Ries (2011), justificando a maior popularidade destes tipos de testes.

Teste unitário também está entre os mais populares, uma vez que este tipo de teste permite verificar se o sistema atende aos requisitos que foram definidos durante a especificação (Runeson, 2006), assim como os testes de aceitação. Testes de integração geralmente ocorrem após os testes unitários, e permitem verificar se os módulos ou partes do sistema irão funcionar quando combinados, portanto, é esperado que estes tipos de testes possuam um percentual de adoção parecido. Quanto ao teste contínuo, visto que o mesmo é realizado a partir da execução automática de testes, é justificável que seu uso seja menor, já que implementar testes automáticos é mais difícil do que implementar testes manuais (Berner *et al.*, 2005; Ramler e Wolfmaier, 2006).

A Figura - 4.14 apresenta as respostas para quais ferramentas de controle de versão e configuração são utilizadas pelas *startups*. Github (48%) e Bitbucket (44%) são as mais utilizadas. Gitlab e Subversion aparecem respectivamente com 17% e 13%, enquanto Mercurial não foi indicada nenhuma vez. Uma das *startups* disse não utilizar nenhuma ferramenta de controle de versão e configuração e outra preferiu não informar.

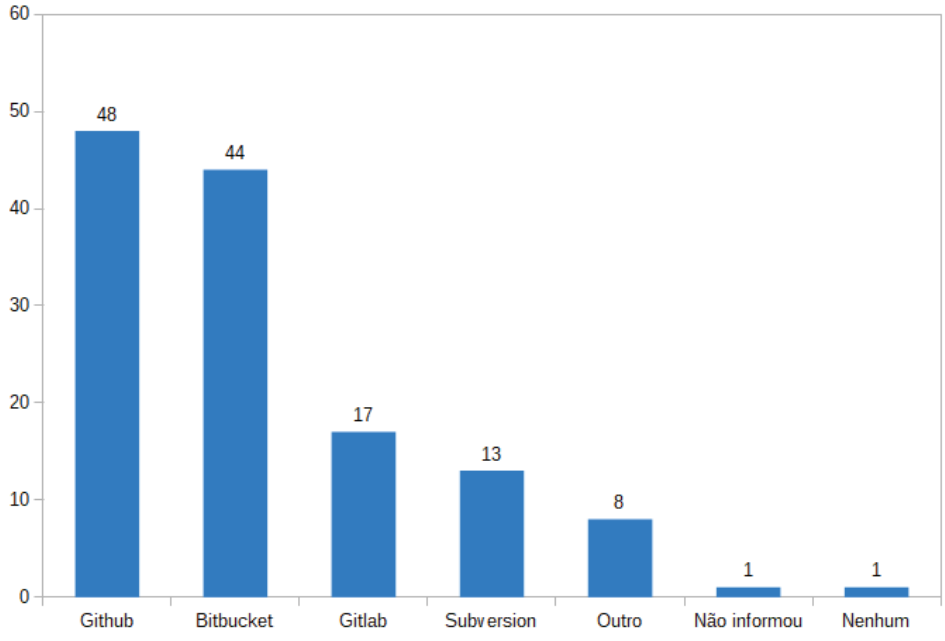

Figura 4.14: Ferramentas de controle de versão e configuração utilizadas pelas *startups*

Houve ainda 8 indicações de outras ferramentas que não constavam entre as selecionáveis. Dessas, 2 indicações são de uso de Git, possivelmente rodando em algum servidor próprio. Outras 2 são de uso de ferramenta própria, 1 que utiliza apenas

numeração, provavelmente por meio do sistema de arquivos do sistema operacional, 1 que utiliza um computador/servidor, porém não é possível identificar se utiliza algum sistema de controle de versão, como Git, ou apenas arquivos, 1 que utiliza Visual Studio Online e 1 que citou apenas Microsoft, sem especificar ferramenta.

É possível verificar a importância das ferramentas de controle de versão e configuração uma vez que todas as *startups*, com exceção de uma, utilizam alguma dessas ferramentas em seu processo de desenvolvimento. Estas ferramentas possibilitam voltar a versões específicas do sistema em caso de problemas, permitem que cada mudança seja gravada com uma mensagem de contexto, facilitando uma análise do histórico de mudanças (Blischak *et al.*, 2016) e portanto são poderosas ferramentas de *backup*. Além disto, por meio das versões é possível verificar quais trechos de código foram modificados, quem fez as mudanças (Ball *et al.*, 1997) e também métricas como, número de mudanças dentro de um período de tempo, número de erros, indicar inanição (Fischer *et al.*, 2003) e assim por diante, provendo detalhes importante sobre o curso de desenvolvimento.

Na Figura - 4.15 são apresentadas as ferramentas de gerenciamento de projeto que as *startups* utilizam. Trello aparece como a mais popular, com 66 indicações, seguida por Atlassian Jira com 19, Redmine com 8, Asana com 5, Basecamp com 3 e Pivotal Tracker com apenas 1. Mingle não foi indicada por nenhuma das *startups* e quatro delas disseram não utilizar nenhuma ferramenta de gerenciamento de projeto.

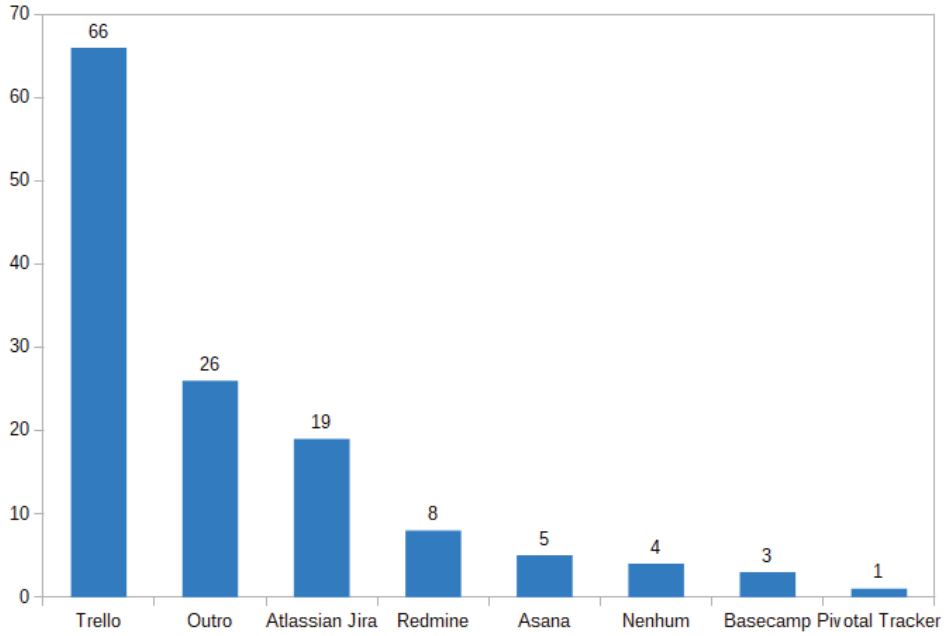

Figura 4.15: Ferramentas de gerenciamento de projeto utilizadas pelas *startups*

Outras ferramentas foram indicadas 26 vezes, sendo elas: Google Docs (2), Google Sheets (1), IBM Jazz (1), Taiga (1), Post-its (1), Zehnk (1), Wrike (1), Conflux (1), Visual Studio Team Services (1), Bitbucket (1), Clubhouse (1), Kanban (2), Artia (1), Toggl (1), Toodledo (1), Github (1), MeisterTask (1), Waffle (1), Work Box (1), Pipefy (1), ScrumHalf (1), Teamwork (1), Google (1) e ferramenta própria (1).

O motivo do Trello ser a ferramenta mais popular provavelmente se deve ao fato de o mesmo ser organizado da mesma forma em que o Kanban é executado, isto é, por meio de cartões de transitam entre painéis, além de ser uma ferramenta gratuita no plano mais básico. Essa similaridade facilita o seu uso, sendo a facilidade de uso o atributo que traz maior satisfação (Azizyan *et al.*, 2011), e visto que a técnica Kanban é amplamente utilizada e está fortemente conectada a metodologia Scrum, que por sua vez é a metodologia mais usada.

Azizyan *et al.* (2012) conduzem um estudo onde buscam selecionar uma ferramenta de gerenciamento que seja adequada a uma determinada organização. Os autores identificaram que, embora há várias opções disponíveis no mercado, selecionar uma ferramenta que seja adequada ao processo e as necessidades da empresa é uma tarefa difícil devido a uma série de motivos, entre eles: i) analisar as características gerais da ferramenta não proporciona a quantidade de informação necessária para tomar uma decisão quanto as necessidades da organização; ii) até mesmo pessoas que ocupam o mesmo papel dentro da organização possuem expectativas diferentes quanto as ferramentas; iii) mesmo após observar o processo da empresa é difícil encontrar uma ferramenta que seja adequada. A partir disto, ferramentas fáceis de implementar são preferíveis à *startups*, já que seus custos de manutenção e treinamento serão menores (Giardino *et al.*, 2014), logo, caso exista a necessidade de migrar de ferramenta, o impacto será menor e de baixo risco.

Na Figura - 4.16 são demonstradas quais são as suítes/plataformas de desenvolvimento utilizadas pelas *startups*. A suíte/plataforma Visual Studio foi citada 37 vezes, Eclipse 18 vezes, Xcode 17 vezes, JetBrains 15 vezes, NetBeans 14 vezes e PyCharm 4 vezes. Apenas uma *startup* respondeu que não utiliza qualquer suíte/plataforma de desenvolvimento e outras três não informaram.

Houve 56 citações entre outras opções de suíte/plataforma de desenvolvimento, sendo elas: Atom (12), Delphi (1), Ferramentas PHP (1), RadStudio (1), Android Studio (4), Sublime Text (18), WebStorm (1), Unity 3D (2), Vim (6), Rstudio (1), Geany (1), Cordova (1), Emacs (2), Ruby on Rails (2), Terminal Ubuntu (1), Arduino (1) e ferramenta própria (1).

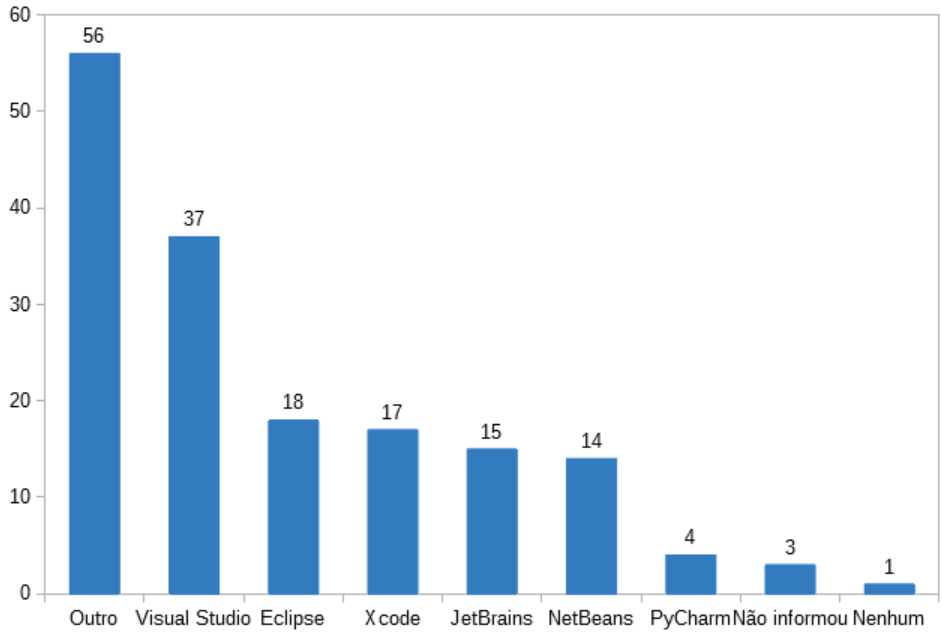

Figura 4.16: Suítes/plataformas de desenvolvimento utilizadas pelas *startups*

Por fim, a Figura - 4.17 apresenta os dados referentes as ferramentas de gestão do conhecimento que as *startups* utilizam. Conforme demonstra a figura, a ferramenta Atlassian Confluence possui 18 citações, seguida por Freshdesk e Collective Knowledge, ambas com 7 citações, OpenKM com 4 citações e eXo Platform com nenhuma citação. Uma das *startups* indicou que está estudando a respeito de qual ferramenta utilizar. O número de *startups* que não utilizam é relativamente alto, sendo 29 nesta situação. Outras oito *startups* não informaram a respeito.

Ainda, outras ferramentas foram indicadas em 33 ocasiões, entre elas: Atlassian Jira (1), World (1), Slack (2), Zehnk (1), Google Drive (4), Google Hangouts (1), Google Docs (3), Intercom (3), Zendesk (3), Google Sites (2), Vtiger (1), Wiki do VSTS (1), ferramenta própria (2), Code Academy (1), Driver (1), Wiki própria (1), Github (1), Notion (1), Bitbucket (1), TikiWiki (1) e DokuWiki (1).

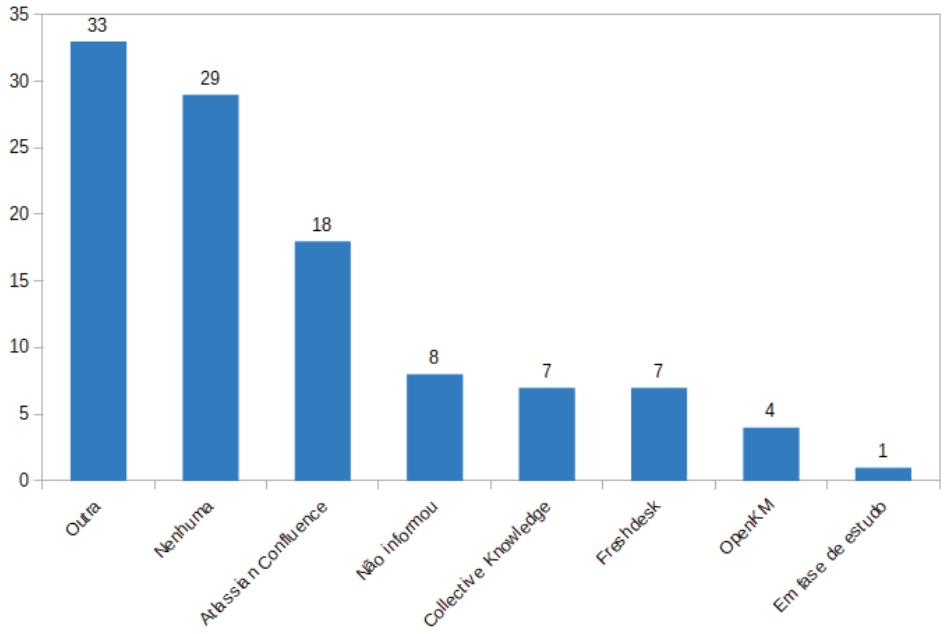

Figura 4.17: Ferramentas de gestão do conhecimento utilizadas pelas *startups*

4.3 Ameaças à Validade

As principais ameaças à validade identificadas neste estudo foram:

Os dados foram coletados e analisados de *startups* de todos os tamanhos, localizadas em várias regiões do Brasil e que se encontram em todos os níveis de maturidade e ocorreu um viés de auto-seleção (viés do voluntariado). De acordo com El-Attar e Miller (2009) a participação de voluntários invalida as ameaças, pois os mesmos estão realmente interessados em participar da pesquisa. Outra ameaça à validade está relacionada aos erros de especificação do instrumento de pesquisa (Land *et al.*, 2005), que é mitigada pela avaliação do instrumento de pesquisa por meio do experimento piloto (Sjoberg *et al.*, 2007).

O estudo foi conduzido no ambiente industrial/empresarial considerando *startups* de todo o Brasil, minimizando assim as ameaças relacionadas a representação de participantes. Esta interação com a indústria possibilita uma melhoria na qualidade e relevância dos estudos, visto que os participantes são o contexto real e possuem experiência (Rodríguez *et al.*, 2014; Wohlin *et al.*, 2012).

Também pode ser considerado uma ameaça à validade o grau de formação dos respondentes do questionário, causando um enviesamento nas respostas, visto que, conforme demonstra a Figura - 4.8, a maioria (68%) deles são graduados ou especialistas. Isto é,

pode ser maioria, as experiências destes respondentes podem ter influenciado a direção das respostas do questionário como um todo.

4.4 Considerações Finais

Como contribuição, o estudo empírico conduzido fornece indícios de caracterização do estado do desenvolvimento de software nas *startups* brasileiras, a partir das metodologias, ferramentas, técnicas e práticas utilizadas. O estudo também permite verificar o perfil das pessoas envolvidas no cenário por meio do tempo de experiência que possuem com desenvolvimento de software, o tempo de experiência que possuem com *startups* e qual o nível de escolaridade que possuem.

O estudo ajuda a evidenciar o perfil das *startups* brasileiras, sendo que por meio do mesmo é possível indicar qual a média de colaboradores que estas *startups* possuem, em qual nível de maturidade se encontram, quais delas possuem algum tipo de investimento externo e quais fazem parte de algum grupo ou rede de colaboração.

Estes resultados são importantes, pois, permitem monitorar o estado de desenvolvimento das *startups* brasileiras e possibilitam que novos estudos sejam conduzidos com base nos dados adquiridos, permitindo que novas hipóteses sejam formuladas a partir do que foi encontrado. Além disto, estes resultados servem como ponto de partida para identificar os fatores que levam estas organizações ao sucesso e a partir disto diminuir a incerteza sobre suas existências no futuro.

Do ponto de vista da indústria, estes resultados servem como um ponto inicial para aquelas *startups* que estão apenas iniciando suas operações e ainda não possuem bem definidas quais metodologias, ferramentas, técnicas ou práticas irão utilizar em seu processo de desenvolvimento. Para academia este estudo é importante, pois fornece dados até então desconhecidos e que podem gerar novas oportunidades de pesquisa e, assim, fomentar o interesse pela área.

5

Diretrizes para o Desenvolvimento de Software em Startups

5.1 Considerações Iniciais

Este capítulo apresenta o escopo das diretrizes, demarcando as áreas da Engenharia de Software que as mesmas irão abranger, o formato de especificação, que define a forma de apresentação das diretrizes, e, por fim, as diretrizes propostas.

5.2 Objetivos

O objetivo geral deste trabalho é desenvolver um conjunto de diretrizes para apoiar o desenvolvimento de software em *startups*. Para Kitchenham *et al.* (2002), o relato destas diretrizes deve atender tanto as necessidades de pesquisadores quanto as necessidades de empreendedores e demais pessoas da indústria. Singer (1999) aponta que se o relato destes resultados não forem apropriados, os problemas que eles buscam resolver nunca serão resolvidos. As diretrizes são propostas com base nas evidências coletadas por meio do *survey* e do mapeamento sistemático.

Em virtude dos desafios que as *startups* enfrentam, principalmente aqueles relacionados às atividades de desenvolvimento de software (Coleman e O'Connor, 2008; Sutton, 2000), estas diretrizes visam auxiliar as etapas de planejamento e execução do processo de desenvolvimento de software, e assim proporcionar uma maior produtividade e um aumento na qualidade dos produtos oferecidos pela *startup*. Além disto, as diretrizes

propostas visam permitir uma maior flexibilidade em sua adoção, pois podem ser adotadas de forma gradual de acordo com as necessidades da *startup* (Stagars, 2015).

5.3 Formato de Especificação das Diretrizes

As diretrizes propostas estão estruturadas a partir de quatro definições: i) título; ii) descrição; iii) resultados esperados; iv) fonte; v) ferramentas recomendadas. Este formato de especificação é inspirado e adaptado do modelo proposto por Leal (2015) em seu trabalho, uma vez que, conforme verificado no Capítulo 2, não há um formato padrão para a especificação de diretrizes.

Cada diretriz é especificada a partir dos seguintes elementos:

- **Título:** Apresenta um resumo da diretriz e propõe o que deve ser feito.
- **Descrição:** Aborda as características da diretriz, apresentando seus métodos e processos de forma que o leitor possa replicá-los na prática.
- **Resultados esperados:** Apresenta quais resultados são esperados ao adotar a diretriz em questão, permitindo que o leitor identifique qual é a aplicação da mesma e em quais cenários ela é aplicável.
- **Ferramentas recomendadas:** Apresenta ferramentas que podem ser utilizadas no apoio da implementação ou execução da diretriz em questão.
- **Fonte:** Indica qual ou quais fontes (literatura, *survey*, etc) dão origem à diretriz.

Como forma de organização e de identificação as diretrizes são ordenadas numericamente.

5.4 Diretrizes Propostas

As diretrizes foram definidas a partir de duas das áreas de conhecimento da Engenharia de Software especificadas no SWEBOK (Bourque e Fairley, 2014), sendo elas a Construção de Software e a Qualidade de Software. A Construção de Software está ligada à criação detalhada de software funcional por meio de uma combinação de codificação, verificação, teste unitário, teste de integração e depuração (Bourque e Fairley, 2014). A Qualidade de Software se refere às características desejáveis de produtos de software, a extensão que um produto particular de software possui essas características e aos processos, ferramentas e técnicas utilizadas para alcançar essas características (Bourque e Fairley, 2014).

A definição deste escopo evita que as diretrizes se tornem muito abrangentes, o que potencialmente tornaria difícil a leitura e a compreensão das diretrizes, indo na direção oposta da objetividade que se espera atingir com as mesmas. A escolha destas áreas por parte do autor são baseadas no que o mesmo acredita ser o mais apropriado de acordo com os dados coletados por meio do *survey* e do mapeamento sistemático, e também por serem áreas que têm potencial para gerar benefícios mais rápidos e valiosos para as *startups* que escolherem seguir as diretrizes.

5.4.1 Construção de Software

Diretriz 1: Utilizar uma metodologia ágil para guiar o desenvolvimento de software.

Descrição: As metodologias ágeis permitem um ciclo de desenvolvimento mais rápido, alinhado ao ambiente ágil no qual *startups* estão inseridas. Estes métodos permitem que o produto seja desenvolvido em pequenas iterações, e isto, se alinhado a uma prática de entregas frequentes, possibilita que sejam obtidos *feedbacks* constantes dos usuários. Estes *feedbacks* são importantes pois ajudam a visualizar se o desenvolvimento do produto está indo na direção certa, permitindo que sejam realizados ajustes quando este não é o caso.

A escolha da metodologia ágil deve ir de acordo com o perfil e necessidades da *startup*. Entretanto, adotar a metodologia possui alguns desafios, como mudança da cultura organizacional, aceleramento do ciclo de desenvolvimento e etc. Ou seja, uma equipe acostumada a trabalhar com Scrum provavelmente terá dificuldades em se adaptar ao EDD, o que pode tomar um tempo valioso que, a princípio, poderia ter sido investido no desenvolvimento do produto.

Por sua flexibilidade, há a possibilidade de que práticas ou técnicas de uma metodologia sejam utilizadas em conjunto com a metodologia escolhida. É possível, por exemplo, utilizar a técnica de programação em pares em um ambiente que não seja guiado pela metodologia XP, uma vez que seu conceito é bastante simples. A adaptação da metodologia é uma estratégia viável para o desempenho dos processos de uma *startup* (Souza *et al.*, 2017). Inclusive, no estudo apresentado por Rose *et al.* (2016), duas das organizações que foram investigadas apontaram utilizar formas customizadas da metodologia Scrum, descrevendo-as como ponto importante para a inovação durante os seus processos. Todavia é preciso ter cautela para que o ciclo de desenvolvimento não seja sobrecarregado por inúmeras práticas diferentes, aumentando a burocracia do mesmo, o que vai na direção contrária da principal característica dos métodos ágeis, a agilidade.

Resultados esperados: Os resultados esperados com a utilização de uma metodologia ágil para guiar o desenvolvimento de software são:

- ciclos de desenvolvimento mais rápidos;
- desenvolvimento do produto em pequenas iterações;
- constante obtenção de *feedback* do cliente;
- facilidade na implementação de ajustes no produto.

Ferramentas recomendadas: Visto que a escolha de uma metodologia depende do perfil da equipe de desenvolvimento, indicar uma que se encaixe independentemente deste perfil é uma tarefa difícil. Entretanto, a EDD, metodologia baseada em experimentação presente no trabalho de Lindgren e Münch (2016), pode ser uma boa escolha para aquelas *startups* mais incertas em relação a direção do seu produto, visto que possibilita que o mesmo seja desenvolvido incrementalmente por meio de um processo de aprendizagem validada ao realizar experimentos com os *stakeholders* e clientes.

Independente da metodologia adotada, Taheri e Sadjadi (2015) apresentam uma classificação com algumas ferramentas de gerenciamento de projetos ágeis que são mais adequadas para *startups*, sendo elas: Axosoft OnTime, Microsoft TFS, Rally Platform, Version One, Blossom.io, Base Camp, AgileZen, Kanbanize e IceScrum.

Fonte: Highsmith e Cockburn (2001), Williams e Cockburn (2003), Dyba e Dingsoyr (2009), Taheri e Sadjadi (2015) e resultados do *survey* (86% dos respondentes, questão 13).

Diretriz 2: Utilizar *frameworks open source* e/ou com grande adoção no mercado.

Descrição: O uso de *frameworks* é de extrema importância para uma *startup*, principalmente para aquelas que estão em fase inicial, pois permite que mais tempo seja investido no desenvolvimento das *features* do produto, já que facilitam o trabalho dos desenvolvedores ao providenciar uma estrutura já pronta. Isto permite que o produto seja lançado no mercado mais cedo, permitindo que o *feedback* dos usuários seja recebido imediatamente e assim as mudanças necessárias podem ser implementadas antes que o produto falhe. Para Giardino *et al.* (2014) o uso de *frameworks* está entre os itens mais importantes

para o sucesso de uma *startup*, pois permitem que mudanças rápidas sejam realizadas no produto visando atender o mercado.

Optar por *frameworks* populares, principalmente os que são *open source*, é menos arriscado, já que as chances de que o desenvolvimento e suporte do mesmo sejam abandonados são menores, evitando que a *startup* seja forçada a trocar de solução porque a mesma ficou defasada. Ainda, um *framework* com uma boa comunidade o mantendo estará sempre atualizado com o mercado, sendo que a *startup* poderá se aproveitar das melhores tecnologias disponíveis sem a necessidade de grandes investimentos.

Resultados esperados: Ao utilizar *frameworks open source* e/ou com grande adoção no mercado os resultados esperados são:

- maior tempo investido no desenvolvimento das *features* do produto;
- redução do *time-to-market*;
- agilidade no recebimento de *feedback* dos usuários;
- agilidade na aplicação de mudanças para atender às demandas do mercado;
- maior capacidade de reuso e customização.

Ferramentas recomendadas: O uso de *frameworks* é totalmente dependente da linguagem de programação que será utilizada, onde cada uma possui uma grande variedade de *frameworks* disponíveis, com funcionalidades diferentes a depender das características da linguagem de programação. Entre algumas das linguagens mais populares, alguns *frameworks* que podem ser indicados são: Spring, Hibernate e JUnit para a linguagem Java; Laravel, Zend, CakePHP e CodeIgniter para a linguagem PHP; AngularJS, ReactJS, e Vue.js para a linguagem JavaScript; .NET Framework e Entity Framework para a linguagem C#; Django para a linguagem Python; Ruby on Rails para a linguagem Ruby. O GitHub possui uma página[1] onde é possível verificar os *frameworks* mais ativos na plataforma.

Fonte: Giardino *et al.* (2014) e resultados do *survey* (questão 17, 79% dos respondentes utilizam *frameworks* e 39% utilizam soluções *open source*).

[1] `https://github.com/topics/framework`. Acesso em 22 de fev. de 2018.

Diretriz 3: Usar *backlog* em conjunto com o Kanban independente do método ágil utilizado.

Descrição: O Kanban é uma técnica versátil que pode ser utilizada em qualquer ambiente, independentemente da metodologia ágil que for adotada. Seu uso é recomendado independente do estágio de maturidade do processo de desenvolvimento da *startup*, porém, no início do processo de adoção de uma nova metodologia pode ser ainda mais importante, pois permite que a equipe identifique de maneira fácil o status das tarefas e auxilia na manutenção de um fluxo contínuo do processo de desenvolvimento, acelerando o ciclo de produção (Al-Baik e Miller, 2015). A técnica está fortemente ligada ao conceito *"just in time"* originado no Japão, onde procura-se eliminar os estoques e agilizar a produção.

Uma aplicação simples do Kanban envolve dividir o quadro de tarefas entre três estágios de desenvolvimento, sendo eles: para ser feito, em andamento e finalizado. Então, a partir das tarefas armazenadas no *backlog*, dividi-las de acordo com o seu status dentro do quadro do Kanban e controlar seu fluxo conforme as tarefas vão avançando durante o processo de desenvolvimento. Se for necessário, a *startup* pode adequar o Kanban para funcionar em um fluxo que se encaixe melhor para as suas demandas.

O *backlog* é uma prática tradicional da metodologia Scrum, porém não há impeditivos para o uso da prática em outros contextos, visto que seu conceito é simples. O *backlog* permite que se tenha uma lista com todas as *features* desejáveis para o produto, permitindo que a equipe de desenvolvimento tenha um registro sempre atualizado das tarefas que ainda devem ser realizadas.

Resultados esperados: Os resultados esperados da utilização de *backlog* em conjunto com o Kanban são:

- gestão visual do estado de desenvolvimento do produto;
- agilidade no ciclo de desenvolvimento;
- acesso rápido às *features* prontas, em desenvolvimento e não iniciadas.

Ferramentas recomendadas: Para o controle do Kanban podem ser utilizadas as ferramentas Kanbanize, Kanban Tool, GreenHopper e Trello. Para o *backlog*, pode-se rever as ferramentas recomendadas na Diretriz 1, além de outras como, Scrumwise, TargetProcess, Backlog by Nulab e alldone.io.

Fonte: Al-Baik e Miller (2015) e resultados do *survey* (68% dos respondentes, questão 18).

Diretriz 4: Estabelecer uma rotina de testes.

Descrição: Os testes são de suma importância, pois garantem a qualidade do produto, permitindo que o melhor produto possível seja entregue ao usuário, aumentando as suas chances de aceitação (Pompermaier *et al.*, 2017). A rotina de testes deve acontecer de acordo com as necessidades do produto, mas também deve respeitar as capacidades da *startup*. Se possível, definir uma rotina de testes automatizados pode ser uma boa opção, pois evita que muitos recursos sejam alocados na execução dos testes, principalmente se tratando dos testes que não dependam da interação dos usuários. Os testes automáticos garantem a qualidade do software por meio de uma extensiva cobertura de testes, a integração contínua e o *release* de software de qualidade proporciona um *feedback* rápido à equipe de desenvolvimento, fazendo com que problemas sejam eliminados na raiz (Rodríguez *et al.*, 2017).

A validação contínua com os usuários é um dos fatores de sucesso apontados por Giardino *et al.* (2014), pois proporciona receber *feedback* imediato quanto a qualidade percebida pelos usuários, permitindo que os problemas sejam corrigidos rapidamente pela equipe de desenvolvimento (Rodríguez *et al.*, 2017). Dessa forma, os testes de usabilidade e os usuários de testes estão entre os principais a serem implementados pela *startup*. Conforme os dados coletados por meio do *survey*, várias *startups* implementam testes de aceitação, testes de integração e teste unitário, sendo estes os principais tipos de testes a serem considerados no estabelecimento da rotina de testes. Organizações como *startups* enfrentam problemas em relação a execução de testes de software devido a falta de tempo para tal, todavia, as organizações entrevistadas no estudo de Gordón e O'Connor (2016) relataram executar dois tipos de testes: interno, com os próprios membros da equipe, e externo, com os clientes.

Em seu trabalho, Ren e Dong (2017) exploraram quais técnicas e processos de testes *startups* seguem para garantir a qualidade de seus produtos. Os autores identificaram que as principais técnicas utilizadas são os testes de caixa preta, testes automáticos e testes de função. Os testes de caixa preta permitem que o software seja testado sem a necessidade de que o testador conheça os detalhes de implementação do código, sua escolha se justifica por ser um método simples que não demanda muito tempo e nem uma habilidade alta para a sua execução. Os testes automáticos evitam a necessidade de execução de tarefas de testes enfadonhas e economizam muito tempo durante sua execução. Até mesmo pela limitação de recursos disponíveis, o testes automáticos são preferíveis para as *startups*

pois proporcionam uma economia de tempo e dinheiro. Quanto aos testes de função, sua escolha se limita ao pouco tempo necessário para a sua execução, sendo suficientes para testar as funções básicas do software.

Uma rotina de testes bem definida não acrescenta em muito o tempo de desenvolvimento, principalmente se estes testes forem automatizados. É possível, por exemplo, em um cenário de entregas contínuas, automatizar rotinas de testes de forma que a implantação do produto só seja realizado caso o mesmo tenha sido aceito em todos os testes (Diretriz 14). Em uma rotina de entregas contínuas os testes são essenciais, pois ajudam a garantir que os *releases* não irão apresentar problemas para os usuários e facilitam o trabalho da equipe de desenvolvimento, visto que o produto estará sendo testado em pequenas iterações, facilitando a correção de eventuais problemas.

Resultados esperados: Os resultados esperados do estabelecimento de uma rotina de testes são:

- agilidade na descoberta de problemas no software;
- gerência da rastreabilidade dos artefatos;
- maior facilidade na correção dos problemas.

Ferramentas recomendadas: Assim como no caso dos *frameworks*, as ferramentas, ou *frameworks*, de testes variam de acordo com a linguagem de programação. Entre algumas das linguagens mais populares, algumas ferramentas que podem ser indicados são: JTest, JMeter e JUnit para a linguagem Java; PHPUnit, CodeSniffer e PHP Mess Detector para a linguagem PHP; JSLint, QUnit, JSpec, Jest, Mocha e Enzyme para a linguagem JavaScript; NUnit e MbUnit para a linguagem C#; pytest, Ludibrio, behave para a linguagem Python; RSpec e Cucumber para a linguagem Ruby. Selenium, TestComplete e Jenkins são algumas das opções de ferramentas de testes que não dependem da linguagem de programação. O GitHub possui uma página[2] onde é possível verificar as ferramentas de testes mais ativas na plataforma.

Fonte: Giardino *et al.* (2014), Gordón e O'Connor (2016), Rodríguez *et al.* (2017), Pompermaier *et al.* (2017) e resultados do *survey* (questão 19, apenas 1 respondente não informou se utiliza testes).

[2] `https://github.com/topics/testing`. Acesso em 22 de fev. de 2018.

Diretriz 5: Validar a ideia antes de começar a desenvolver.

Descrição: É natural que a partir do momento que a ideia surge exista um desejo de começar a desenvolvê-la imediatamente. Dessa forma, é comum que a ideia não seja validada a princípio e que o produto comece a ser desenvolvido sem que se saiba se haverá demanda no mercado para o mesmo (Dande *et al.*, 2014). Sendo assim, é essencial para as *startups* que a ideia seja validada antes, evitando que tempo e dinheiro sejam desperdiçados no desenvolvimento de um produto que não irá vender. A validação da ideia deve ocorrer principalmente com os potenciais clientes, mas também pode ser realizada com investidores e especialistas da área (Dande *et al.*, 2014).

É comum que *startups* assumam que suas ideias irão funcionar (Gonçalves, 2017). Validar a ideia antes de iniciar o seu desenvolvimento permitirá verificar se haverá um número suficiente de clientes interessados na aquisição do produto, ou seja, se este produto irá atender as demandas e as necessidades do mercado, sendo que só então os recursos necessários serão alocados na construção deste produto. Visto a escassez de recursos que *startups* enfrentam, obter essa validação é de extrema importância independente do nível de maturidade da organização, pois o desenvolvimento de um produto comercialmente inviável significa um grande desperdício de tempo e de recursos por parte da *startup*, podendo até mesmo decretar o seu fim.

Resultados esperados: Os resultados esperados da validação da ideia antes de iniciar o desenvolvimento são:

- indicativo do número de clientes interessados na aquisição do produto;
- recursos não desperdiçados na construção de um produto inviável.

Ferramentas recomendadas: Mesmo ferramentas simples como papel e caneta podem ser suficientes para a validação da ideia, tudo depende do público alvo. Entrevistas informais, anúncios em jornais ou alguma outra estratégia de marketing também podem ser suficientes. Um dos métodos mais interessantes é construir uma *landing page*, que irá apresentar o produto e pode coletar informações dos potenciais consumidores, como por exemplo o e-mail, que pode ser utilizado posteriormente para a entrega de uma *newsletter* e servir de gatilho para o engajamento do usuário.

A prototipagem rápida também é uma boa estratégia para a validação, permitindo apresentar uma ideia mais palpável do produto para os potenciais clientes. A prototipagem

permite uma comunicação rápida e eficiente das ideias do projeto. Uma forma simples e eficiente de apresentar os conceitos do produto é por meio de *wireframes*, que podem ser construídos com ferramentas como Balsamiq Mockups, Pencil Project, OmniGraffle e MockFlow.

Fonte: Ries (2011), Dande *et al.* (2014) e Gonçalves (2017).

Diretriz 6: Construir um MVP.

Descrição: A construção de um MVP permite validar o produto com o menor investimento possível. Por meio desta prática é possível verificar se o produto cumpre a função para qual foi projetado ao desenvolver apenas as funcionalidades essenciais e minimamente necessárias para a sua operação. O MVP foca em minimizar o tempo para a aprendizagem do conjunto de características que compõem as requisitos do cliente (Dunkin, 2017). No contexto de *Lean startup*, o MVP é a versão mínima do produto para avançar no ciclo *Build-Measure-Lean* (Ries, 2011). O MVP permite, portanto, coletar informações do cliente e testar se determinadas suposições estão corretas.

Para *startups* é de extrema importância ser a primeira a lançar um determinado produto no mercado (Salerno *et al.*, 2015), sendo que o MVP auxilia no cumprimento deste objetivo visto que permite lançar uma versão mínima porém funcional do produto. Mesmo um curto período de tempo pode ser crucial para se chegar ao mercado a frente de um possível concorrente. Após o seu lançamento, o MVP pode continuar a ser desenvolvido por meio da disponibilização de novas versões para os clientes. Este processo pode ainda, por meio de atividades paralelas, ser utilizado para criar uma demanda, testar soluções, obter sugestões e para melhorar a versão final do produto (Salerno *et al.*, 2015).

A construção de um MVP é listada como um dos principais desafios que uma *startup* enfrenta (Wang *et al.*, 2016), demonstrando a sua importância para o sucesso da organização. Nesta etapa a *startup* pode ser menos rigorosa quanto aos requisitos do software (Diretriz 21), visto que a falta de documentação ou gerência de requisitos não afeta a qualidade do MVP desenvolvido (Pompermaier *et al.*, 2017). O principal objetivo do MVP é validar rapidamente o produto (Diretriz 5) por meio de um software funcional, a gerência de requisitos nesta etapa pode aumentar significativamente o tempo de desenvolvimento do MVP (Pompermaier *et al.*, 2017), desperdiçando tempo valioso para a validação da ideia

Um MVP bem sucedido serve de base para o desenvolvimento do produto e, ao permitir testar as hipóteses diretamente com o cliente, acelera o tempo de desenvolvimento. Além

disto, os artefatos gerados durante a implementação do MVP podem indicar os limites do que pode ser realizado em termos do produto, além de poderem ser reutilizados em outras etapas do processo de desenvolvimento (Duc e Abrahamsson, 2016).

Resultados esperados: Construir um MVP tem como resultados esperados:

- validação das ideias do produto investindo o mínimo possível;
- recursos não desperdiçados em um produto que não satisfaz o cliente;
- aprendizado com o cliente no decorrer das iterações;
- base para o desenvolvimento do produto;
- aceleração do tempo de desenvolvimento.

Ferramentas recomendadas: Duc e Abrahamsson (2016) apresentam uma investigação sobre o papel do MVP nas *startups*, onde são descritas algumas das principais ferramentas utilizadas neste contexto, entre elas: *Wizard of Oz MVP*, *Concierge MVP*, *Piecemeal MVP*, *Mockup MVP* e *Single feature MVP*.

Fonte: Ries (2011), Duc e Abrahamsson (2016), Dunkin (2017) e resultados do *survey* (81% dos respondentes, questão 18).

Diretriz 7: Utilizar uma ferramenta de gerenciamento de versão descentralizada.

Descrição: O controle de versão descentralizado permite acesso de primeira classe a todos os desenvolvedores, isto é, não é necessário ser um contribuidor direto para ter acesso as versões. Atomicidade nas alterações e uma maneira automática de fundi-lás também são vantagens que ajudam a manter a consistência do projeto, além de prevenir a ocorrência de erros e de código desatualizado em alguma das estações de trabalho. Diferente das ferramentas centralizadas, o controle de versão descentralizado permite que os desenvolvedores trabalhem mesmo quando não há uma conexão com o servidor, sendo assim, é possível que o desenvolver faça mudanças na sua cópia de trabalho mesmo fora do ambiente de trabalho, e posteriormente aplique essas mudanças assim que voltar a ter uma conexão com o servidor. Além disto, a descentralização do versionamento oferece um melhor suporte para fluxos de trabalho não centralizados, facilitando o trabalho em equipes distribuídas.

Ferramentas de controle de versão permitem que várias pessoas contribuam simultaneamente para o projeto sem que ocorram conflitos entre as modificações, permite uma melhor coordenação para o desenvolvimento de uma nova *feature*, auxilia na prática de *code review* e de como o suporte para código já lançado é organizado (De Alwis e Sillito, 2009). Em ferramentas de controle de versão descentralizadas não há a necessidade de que exista um repositório central, sendo que cada *checkout* é por sua vez um repositório de primeira classe, ou seja, é uma cópia contendo todo o histórico de *commits* (De Alwis e Sillito, 2009).

O Git é uma das ferramentas de controle de versão descentralizada mais populares da atualidade. O uso de uma ferramenta dessas pode resultar até mesmo em uma aceleração no processo e um aumento na qualidade, contribuindo para a inovação durante o processo de desenvolvimento e permitindo que a *startup* lance o produto em um menor tempo (Rose *et al.*, 2016).

Resultados esperados: Os resultados esperados ao utilizar uma ferramenta de gerenciamento de versão descentralizada são:

- consistência no projeto;
- diminuição da ocorrência de erros e de código desatualizado;
- histórico completo do desenvolvimento do produto;
- atomicidade nas alterações, ou seja, mudanças são aplicadas em sua totalidade ou não são aplicadas;
- possibilidade de trabalhar mesmo quando não há uma conexão com o servidor.

Ferramentas recomendadas: Git é o controle de versão mais popular entre as *startups* que compõem o questionário. A ferramenta pode ser utilizada em um servidor local ou então por meio de serviços que guardam o código na nuvem, como Github, Bitbucket e Gitlab, tornando a colaboração entre pessoas fisicamente distantes ainda mais fácil.

Fonte: De Alwis e Sillito (2009), Rose *et al.* (2016) e resultados do *survey* (questão 20, apenas 19% dos respondentes utilizam versionamento centralizado).

Diretriz 8: Priorizar a terceirização da infraestrutura.

Descrição: Grande parte das *startups* de software precisam de algum tipo de infraestrutura para construir e, principalmente, para distribuir seu produto. Isto inclui a necessidade de servidores altamente disponíveis, capacidade de armazenamento de informação e velocidade de acesso. Porém, até mesmo pela falta de recursos, proporcionar este tipo de infraestrutura é muito difícil para *startups*, devido a complexidade e custos envolvidos. Além disso, há a necessidade de pessoas capacitadas para gerenciar esta complexidade, aumentando ainda mais os custos.

A partir disto, a terceirização da infraestrutura passa a ser uma opção viável e atrativa, sendo possível colocar servidores em funcionamento com pouco esforço e armazenar dados conforme o necessário com os custos somente daquilo que está realmente sendo utilizado (Melegati e Goldman, 2015). Ao terceirar a infraestrutura é possível acelerar o crescimento da *startup* e economizar recursos, eliminando a necessidade de gerenciar essa parte. Isso permite desenvolver o produto com maior rapidez e agilidade, uma vez que a organização não precisará se preocupar em construir a infraestrutura e sim em apenas escolher qual irá utilizar.

Resultados esperados: Ao escolher a terceirização da infraestrutura os resultados esperados são:

- aceleração do desenvolvimento do produto;
- economia de recursos (financeiros, tempo e humanos);
- facilidade ao escalar o produto.

Ferramentas recomendadas: A terceirização da infraestrutura pode ocorrer em várias frentes, a depender das necessidades da *startup*, podendo ser terceirizado o servidor da aplicação/produto, o armazenamento dos dados, etc. Dessa forma, a *startup* pode contratar um servidor dedicado, um espaço de armazenagem ou qualquer outro provedor de serviço de acordo com o necessário.

Se tratando de *cloud computing*, as ferramentas relacionadas com a terceirização da infraestrutura podem ser dividas em três categorias: *Infrastructure as a Services* (IaaS), *Platform as a Service* (PaaS) e *Software as a Service* (SaaS). Entre elas as opções mais recomendadas de fornecedores são: Amazon S3, Amazon EC2, Amazon AWS, Microsoft Azure, DigitalOcean, Salesforce Platform, Red Hat OpenShift, Google App Engine e SAP Cloud Platform.

Fonte: Melegati e Goldman (2015) e Cukier (2017).

Diretriz 9: Modularizar a estrutura do produto.

Descrição: Conforme o produto evolui é necessário remover ou adicionar *features*. Em alguns casos, algumas *features* podem acabar se tornando até mesmo novos produtos (Dande *et al.*, 2014). Essas mudanças devem ocorrer rapidamente, de forma a não atrasar o desenvolvimento do produto, porém, se a estrutura do produto não foi construída com essas mudanças em mente, as alterações podem se tornar complexas. A não definição da estrutura do produto pode contribuir significativamente para um aumento da dívida técnica (Diretriz 19) (Pompermaier *et al.*, 2017). Em *startups*, a adição e a remoção de *features* ocorrem frequentemente, visto que o produto está em constante mudança, modularizar a estrutura do produto facilita que estas ações ocorram, pois as *features* se tornam independentes.

A modularização da estrutura do produto também permite o reuso de software, pois facilita a inserção de código externo no produto ou até mesmo o reaproveitamento de módulos de outros produtos da própria *startup*. No estudo realizado por Jansen *et al.* (2008) o reuso se mostrou rentável para as *startups* analisadas, visto que possibilitou a elas desenvolverem seus produtos por meio da reutilização de funcionalidades de terceiros que as mesmas não conseguiriam desenvolver por meios próprios.

Resultados esperados: Os resultados esperados da modularização da estrutura do produto são:

- facilidade na inserção e remoção de *features*;
- encurtamento do tempo de desenvolvimento;
- facilidade na manutenção;
- reaproveitamento de código (reuso).

Ferramentas recomendadas: Existem algumas formas de estruturar o software em módulos, entre as mais comuns estão as técnicas de programação orientadas a objetos, a estrutura *model-view-controller* (MVC) e a construção por meio de *application programming interface* (API).

Fonte: Jansen *et al.* (2008) e Dande *et al.* (2014).

Diretriz 10: Utilizar estilos de codificação.

Descrição: Os estilos de codificação, também conhecidos como estilos de programação, são um assunto pouco debatido na literatura, porém, um trabalho de 1974 já destacava a sua importância (Kernighan e Plauger, 1978). Para os autores, o software deve ser codificado não apenas para satisfazer as necessidades do compilador, mas também deve ser legível para humanos. O livro apresenta 56 lições que devem ser seguidas visando a escrita do melhor código possível. Apesar de antigo, sendo que muitas das linguagens de programação atuais ainda não existiam na época de sua publicação, os pontos levantados no livro geralmente dizem respeito a questões estilísticas e estruturais que vão além de detalhes particulares de cada linguagem.

Em *startups*, conforme o produto escala para atender novos clientes e compreender novas funcionalidades, muitas vezes há a necessidade de que novos desenvolvedores sejam inseridos no projeto. Isto traz problemas em relação a leitura e a compreensão do código, uma vez que estes novos desenvolvedores não estão familiarizados com o projeto, implicando em uma constante necessidade de reescrita de código, que por sua vez aumenta o *time-to-market* (Ribeiro e Travassos, 2015). Dessa forma, o uso de estilos de codificação é uma estratégia viável para evitar a reescrita de código, principalmente se for adotado desde o início do projeto, podendo inclusive influenciar para um aumento na qualidade do software (Nascimento e Ribeiro, 2017).

Seguir um padrão na escrita de código auxilia para que os desenvolvedores compreendam código escrito por terceiros com maior facilidade e consequentemente cometam um menor número de erros. Apesar de transcender as linguagens de programação, os padrões ou estilos de programação são definidos de acordo com a linguagem de programação utilizada. Atualmente algumas das maiores empresas de tecnologia (Airbnb, Google, WordPress, etc) disponibilizam seus estilos de código para que possam ser seguidos, porém nada impede que a *startup* crie o seu próprio estilo de código, de acordo com as preferências dos seus desenvolvedores.

Resultados esperados: Os resultados esperados ao utilizar estilos de codificação são:

- maior legibilidade do código;
- facilidade na manutenção;

- facilidade na inserção de novos desenvolvedores.

Ferramentas recomendadas: Os estilos de codificação são uma escolha pessoal e por isso é difícil indicar estilos a serem seguidos. Além disto, os estilos variam de acordo com a linguagem de programação utilizada pela organização. Alguns guias de estilo específicos para a linguagem JavaScript que podem ser analisados são do Airbnb[3], Node.js[4] e WordPress[5].

Fonte: Nascimento e Ribeiro (2017) e resultados do *survey* (64% dos respondentes, questão 17).

Diretriz 11: Usar prototipagem evolucionária durante o desenvolvimento do produto.

Descrição: A prototipagem é uma forma de experimentação durante o desenvolvimento do produto e um ponto importante para a inovação (Martin, 2011). A inovação cria oportunidades para o crescimento de mercado, diferenciação e vantagem competitiva (Song e Montoya-Weiss, 1998). A prototipagem é uma estratégia viável para *startups*, visto que os custos para sua realização são baixos e não demanda tecnologia de ponta.

Giardino *et al.* (2014) apontam a prototipagem evolucionária como um dos principais pontos para o sucesso das *startups*, mais especificamente a prototipagem evolucionária por meio de componentes existentes. A prototipagem evolucionária pode ser executada por meio do ciclo de vida *Build-Measure-Learn* apresentado por Ries (2011). Neste ciclo, primeiro a *startup* desenvolve um protótipo do seu produto, com o menor número possível de funcionalidades, porém suficientes para testar as suas premissas, ou seja, um MVP. A partir deste protótipo a *startup* deve observar como o produto se comporta, isto é, colher o *feedback* dos usuários. Este processo pode ser auxiliado por técnicas de testes, de monitoramento, entre outros. Por fim, o ciclo de aprendizagem ocorre, onde os dados coletados na segunda fase serão analisados. Com base nos resultados a *startup* deve decidir qual o próximo passo a ser executado no produto, se irá continuar com as mesmas ideias que foram introduzidas no início ou se irá mudar radicalmente a direção do produto.

[3] `https://github.com/airbnb/javascript`. Acesso em 23 de abr. de 2018.

[4] `https://github.com/felixge/node-style-guide`. Acesso em 23 de abr. de 2018.

[5] `https://make.wordpress.org/core/handbook/best-practices/coding-standards/javascript/`. Acesso em 23 de abr. de 2018.

Resultados esperados: Ao utilizar a prototipagem evolucionária durante o desenvolvimento do produto os resultados esperados são:

- refinamento do produto, sendo amadurecido e melhorado no decorrer das iterações;
- ajustes no produto de acordo com os *feedbacks* coletados dos usuários;
- maiores chances de que o produto final atenda às expectativas.

Ferramentas recomendadas: Ferramentas simples como planilhas e documentos de texto podem ser suficientes em alguns casos para a organização dos dados obtidos, porém, ferramentas de gerenciamento, como as apresentadas na Diretriz 1 também podem auxiliar. Balsamiq Mockups, Verify e Kiss Metrics são algumas das ferramentas que podem auxiliam, respectivamente, nos ciclos de *build*, *measure* e *learn*.

Fonte: Giardino *et al.* (2014), Ries (2011) e resultados do *survey* (37% dos respondentes, questão 17).

5.4.2 Qualidade de Software

Diretriz 12: Empoderar a equipe de desenvolvimento.

Descrição: A equipe de desenvolvimento deve estar apta para absorver e aprender por meio de tentativa e erro de forma rápida o suficiente para se adaptar às novas práticas emergentes no mercado (Giardino *et al.*, 2014). Empoderar a equipe é importante, pois diminui a interdependência e também a dependência dos membros com cargos superiores, deixando a estrutura do time mais horizontal, onde decisões importantes podem ser tomadas por qualquer um dos integrantes da equipe. Isso auxilia na criação de uma mente coletiva, onde o conhecimento não está concentrado em apenas em alguns membros mais capacitados, mas sim em todos os membros do grupo, permitindo que os mesmos estejam aptos a realizar as tarefas e a coordenar as suas próprias ações, adaptando seu comportamento de acordo com a demanda das tarefas e de outros membros da equipe (Cannon-Bowers e Salas, 2001). O conhecimento e o processo de equipe estão entre os principais impulsores para a inovação (Rose *et al.*, 2016).

Estabelecer uma cultura que encoraje os membros da equipe a surgirem com ideias inovadoras e a fazer parte das decisões de gestão e de estratégias de inovação são

um dos pontos chave para a inovação dentro da organização (Shahzad *et al.*, 2017). De acordo com Hartmann (2006), uma cultura organizacional forte pode estimular significativamente a criatividade e o comportamento inovativo entre os membros da equipe, proporcionando um ambiente propício para o surgimento de ideias criativas e para o estabelecimento da inovação como um valor essencial da organização. Deve haver uma ênfase no desenvolvimento dos recursos humanos e deve ser proporcionado um ambiente aberto para estratégias de pesquisa e desenvolvimento que levem a um aumento no desempenho de inovação da organização e à obtenção de uma vantagem competitiva (Shahzad *et al.*, 2017).

Para Coleman e O'Connor (2008), o conhecimento e a capacidade da equipe irão compensar a falta de recursos da *startup*. Empoderar a equipe permite maior confiança na execução de tarefas com uma menor supervisão, possibilitando que maiores responsabilidades sejam delegadas aos membros. Dessa forma, os desenvolvedores podem exercer uma maior influência durante o processo de desenvolvimento, ficando muito mais confortáveis do que em ambientes restritos e controlados. Com uma maior capacidade para a tomada de decisão, a equipe irá depender menos dos gestores do projeto, isto ajuda a diminuir o tempo de desenvolvimento já que encurta o número de passos necessários para a implementação de uma mudança. O empoderamento dos membros da equipe representa a estratégia mais viável para melhorar o desempenho e o sucesso da *startup* (Giardino *et al.*, 2014).

Resultados esperados: Os resultados esperados ao empoderar a equipe de desenvolvimento são:

- melhora o desempenho da equipe;
- autonomia na tomada de decisão;
- menor necessidade de supervisão;
- encurtamento do tempo de desenvolvimento.

Ferramentas recomendadas: Não é possível indicar ferramentas que auxiliem no empoderamento da equipe, porém é possível indicar algumas práticas que podem auxiliar no processo, entre elas: encorajar a comunicação entre os membros, cultivar uma cultura de transparência, apresentar novos desafios e oportunidades, incentivar a formação, respeitar as limitações de cada membro, proporcionar um ambiente flexível e proporcionar

liberdade para que erros sejam cometidos (Giardino *et al.*, 2014; Kirkman e Rosen, 1999).

Fonte: Coleman e O'Connor (2008), Giardino *et al.* (2014), Cannon-Bowers e Salas (2001), Shahzad *et al.* (2017) e resultados do *survey* (50% dos respondentes, questão 17).

Diretriz 13: Realizar ciclos de *feedback* com *stakeholders* e clientes.

Descrição: Ries (2011) apresenta a aprendizagem validada como o principal fator de sucesso para uma *startup*, pois esta permite que, com o mínimo de esforço e, logo no início do desenvolvimento do produto, seja verificado se o mesmo irá atender as necessidades dos clientes. Desta forma, há a possibilidade da *startup* "pivotar", ou seja, mudar de direção, antes de realizar grandes investimentos em produto sem apelo no mercado. Além disto, esses ciclos de *feedback* com os clientes auxiliam na compreensão do domínio do usuário, sendo este um conhecimento considerado como impulsor para a inovação (Lee e Cole, 2003).

O contato direto com os usuários permite que os desenvolvedores obtenham um melhor compreendimento em relação as ideias dos mesmos, fazendo com que os desenvolvedores foquem nos aspectos certos imediatamente. Além disto, ao apresentar novas *features* aos usuários para que possam testar e experimentar permite receber um *feedback* imediato quanto a satisfação dos mesmos em relação as soluções desenvolvidas. Para os desenvolvedores isto é importante pois valida o seu trabalho e os dá motivação (Bosch-Sijtsema e Bosch, 2015).

Visando obter esta aprendizagem validada, recomenda-se que se siga com um ciclo de validação contínua com usuários chave do produto, procurando sempre entregar funcionalidades que possuem valor a estes usuários. Isto implica em deixar de lado funcionalidades que parecem interessantes do ponto de vista técnico, mas que entregam pouco valor ao cliente final. Giardino *et al.* (2014) também defende a validação contínua e a entrega de valor contínua como pontos essenciais para o sucesso de uma *startup*.

Resultados esperados: Os resultados esperados da realização de ciclos de *feedback* com os clientes e *stakeholders* são:

- aprendizagem validada;
- verifica se o produto irá atender as necessidades dos clientes;
- evita que grandes investimentos sejam feitos em um produto sem apelo no mercado;

- indica se é necessário mudar a direção do produto.

Ferramentas recomendadas: Para que estes ciclos de *feedback* ocorram regularmente é preciso ter um canal de comunicação fácil e sempre disponível com o cliente, para isto, ferramentas como Slack, Mattermost e Rocket.Chat são alternativas. Estas ferramentas permitem a criação de canais que podem ser disponibilizados aos clientes e permitem que arquivos sejam enviados diretamente por meio da ferramenta, mantendo um histórico pesquisável de todas as mensagens trocadas. Com o auxílio de outras ferramentas, existe a possibilidade de criar canais de mensagens automáticas com os clientes, sendo possível, por exemplo, informar o cliente sobre *releases* automaticamente assim que os mesmos são liberados.

Outras abordagens interessantes na obtenção de *feedback* dos usuários são a realização de grupos de foco e a execução de entrevistas e questionários.

Fonte: Giardino *et al.* (2014) e Ries (2011).

Diretriz 14: Realizar entregas frequentes/*releases* curtos.

Descrição: Para manter a sua competitividade, organizações intensivas de software, como *startups*, precisam entregar *features* valiosas de forma muito rápida para o seus clientes, muitas vezes próximo de ser em tempo real (Rodríguez *et al.*, 2017). As metodologias ágeis permitem que o produto seja desenvolvido em pequenas iterações, logo, liberar versões do produto em curtos períodos de tempo se torna uma tarefa fácil, porém de grande importância. Ao liberar novas versões constantemente é possível obter *feedbacks* mais precisos e tomar medidas mais rápidas em relação aos mesmos, gerando um ciclo de aprendizagem contínua e de melhoramento do produto. Além disso, ao optar por entregas frequentes, os riscos de que uma grande atualização faça o produto funcionar de maneira inesperada são menores, pois permite que os desenvolvedores tenham um maior controle sobre as *features* que estão sendo implementadas.

A entrega contínua, ou *Continuous Deployment* (CD), é a prática de liberar código imediatamente em ambiente de produção para os clientes (Claps *et al.*, 2015; Fitzgerald e Stol, 2014; Järvinen *et al.*, 2014), podendo ser uma opção interessante para as *startups* dispostas a implementá-la. Entre os benefícios da implementação de CD estão: diminuição do *time-to-market*, aumento da satisfação do cliente, *feedback* contínuo, inovação rápida, melhora na confiabilidade de *release* e melhora na produtividade (Rodríguez *et al.*, 2017).

As *startups* investigadas por Souza *et al.* (2017) apresentam um grande foco em adotar ferramentas bem integradas que as auxiliam a automatizar os processos e a comunicação, sendo que a entrega contínua, em conjunto com a integração contínua, ou *Continuous Integration* (CI) (Ståhl e Bosch, 2014; Vasilescu *et al.*, 2015), possuem um papel essencial para que isso ocorra. Entretanto, para garantir um produto de qualidade durante as entregas automáticas, a *startup* deve ter bem definido um processo de testes automáticos (Rodríguez *et al.*, 2017) (Diretriz 4).

As entregas devem ser realizadas por meio de um processo bem definido de *deploy*, preferencialmente de forma que seja possível reverter qualquer atualização que comprometa a funcionalidade do produto. Atualmente isso pode ser feito por meio de ferramentas automatizadas, sendo que em alguns casos, até mesmo nos casos de falha a ferramenta é capaz de reverter as mudanças sem intervenções. Entretanto é preciso monitorar constantemente o processo para que surpresas não ocorram, visto que a entrega de um produto funcional ao cliente é de suma importância para o seu sucesso.

Resultados esperados: Como resultados esperados da prática de entregas frequentes/*releases* curtos tem-se:

- diminuição do tempo de desenvolvimento;
- desenvolvimento do produto em pequenas iterações;
- coleta mais rápida de *feedback* dos usuários;
- ciclo de aprendizagem e melhoramento do produto;
- evita que uma grande atualização faça o produto funcionar de maneira inesperada;
- maior controle sobre as *features* que estão sendo lançadas.

Ferramentas recomendadas: Algumas ferramentas que são capazes de gerenciar o processo de *deployment* são: Codeship, AWS CodeDeploy, Chef, Travis CI, Jenkins e Microsoft TFS.

Caso opte por criar um ambiente de CD e CI, ferramentas como Github, Gitlab e Slack estão entre as mais recomendadas.

Fonte: Giardino *et al.* (2014), Rodríguez *et al.* (2017), Souza *et al.* (2017) e resultados do *survey* (67% dos respondentes, questão 17).

Diretriz 15: Gerenciar o conhecimento do projeto de software.

Descrição: Armazenar o conhecimento de software permite manter um histórico do desenvolvimento do produto, possibilitando visualizar o estado das suas *features* através do tempo. A possibilidade de consultar as decisões que foram tomadas no passado é importante até mesmo para decisões futuras, pois podem indicar o que se esperava do produto e em qual direção o mesmo se encontra atualmente. A partir disto ajustes podem ser realizados para colocar o produto de volta ao seu caminho original ou para a inserção de melhorias, evitando que a *startup* falhe por não atender as necessidades e demandas de seus clientes.

Devido ao contexto ágil e de incertezas em que as *startups* estão inseridas, o registro do conhecimento de software é muitas vezes negligenciado. No *survey* apresentado no Capítulo 4, apenas 29 *startups* que responderam o questionário não fazem uso de ferramentas de gestão do conhecimento. A criação e a atualização de documentação é visto como um desperdício de tempo, esforço e recursos financeiros, arriscando que se alcance a produtividade esperada em troca de ciclos mais rápidos de desenvolvimento e de validação (Nascimento e Travassos, 2017). De acordo com Paternoster *et al.* (2014), *startups* favorecem produtividade sobre processos e documentação, confiando na habilidade de comunicação e competência da equipe para alcançar ciclos de produção mais rápidos.

O trabalho de Paternoster *et al.* (2014) aponta que as *startups* possuem uma falta de documentação e de processos quando transitam para a fase de crescimento devido a indisponibilidade de conhecimento de software para novos integrantes do time, o que aumenta o tempo dos ciclos de desenvolvimento e validação, podendo até mesmo dificultar a inclusão de novas *features* no produto. Sendo assim, a armazenagem do conhecimento de software é importante para a manutenção de um fluxo contínuo de produção, pois evita que ocorram gargalos durante o desenvolvimento do produto, onde um tempo que seria investido na geração de valor para o cliente seja na verdade desperdiçado.

Resultados esperados: Os resultados esperados da gestão do conhecimento do projeto de software são:

- manutenção do histórico de desenvolvimento do produto;
- visualização do estado das *features* através do tempo;
- tomada de decisões com base nos erros ou aprendizados passados;
- agilidade e facilidade na inserção de novos integrantes na equipe;

- fluxo contínuo de produção.

Ferramentas recomendadas: Nascimento e Travassos (2017) procuraram entender como as *startups* armazenam o conhecimento de ideias e de *features* do software. Entre as estratégias mais utilizadas estão o uso de protótipos de interface gráfica de usuário e *mockups*, seguidos pelo uso de texto livre e imagens. Os protótipos de interface gráfica e os *mockups* são visualmente atraentes e permitem observar as interações dos usuários com a interface gráfica do sistema. Texto e imagens também são interessantes porém sua falta de formalismo podem abrir caminho para diferentes interpretações, dessa forma seu uso é recomendado como forma de complemento.

Uma técnica interessante e simples de manter é a implementação de um *changelog* do produto, uma lista de mudanças. Esse *changelog* é escrito a cada iteração lançada e contém uma breve descrição das *features* que foram inseridas e das mudanças realizadas no software. Além de importante para a equipe de desenvolvimento, que pode consultar as alterações realizadas em cada versão do sistema de forma simplificada, o *changelog* pode ser disponibilizado para o usuário em conjunto com o lançamento da nova versão, permitindo que o mesmo também acompanhe as mudanças. Um usuário mais ativo pode até mesmo auxiliar o time de desenvolvimento na descoberta de defeitos com base em seu conhecimento a respeito das alterações realizadas no software.

No *survey* executado neste trabalho, 18 *startups* indicaram utilizar a ferramenta Atlassian Confluence para a gestão do conhecimento. Trello e Slack são ferramentas gratuitas que também podem desempenhar este papel. Ambas permitem que sejam anexados arquivos as conversas e possuem um sistema de busca. O Trello inclusive possui uma organização em listas, parecido com o Kanban, e pode servir como um meio de organizar os requisitos do sistema.

Fonte: Nascimento e Travassos (2017), Paternoster *et al.* (2014) e resultados do *survey* (questão 23, apenas 29% informou não utilizar).

Diretriz 16: Minimizar a dispersão das informações do projeto.

Descrição: Armazenar as informações do projeto em múltiplas ferramentas provoca resultados imprecisos e dificulta uma visibilidade em tempo real confortável (Taheri e Sadjadi, 2015). Espalhar as informações por várias ferramentas diferentes dificulta o trabalho da equipe de desenvolvimento, que terá que realizar várias consultas para

achar a informação necessária. Isso pode até mesmo resultar em um atraso no ciclo de desenvolvimento, visto que este é um tempo precioso que seria investido na construção e no aperfeiçoamento do produto.

Utilizar várias ferramentas para armazenar as informações do projeto faz com que exista uma grande quantidade de redundância sem necessidade, visto que muitas vezes a mesma informação estará alocada em duas ferramentas diferentes (Reichert *et al.*, 2015). Ao ter que gerenciar múltiplas ferramentas, o tempo de manutenção das mesmas, envolvendo atualizações e configurações, aumenta significativamente. Atualizar a informação também passa a ser custoso, pois será preciso acessar várias ferramentas diferentes para tal. Além de dispendioso em termos de tempo, as chances de que erros sejam cometidos também aumentam. Informações podem acabar ficando desatualizadas em uma ferramenta caso a equipe esqueça de alimentar a mesma, ou ainda podem ficar inconsistentes caso duas pessoas diferentes façam a atualização da mesma informação em ferramentas também diferentes. Ao passar do tempo, é esperado que as informações passem a ficar mais e mais diferentes umas das outras (Reichert *et al.*, 2015).

Com o crescimento do projeto, é esperado que a quantidade de informações se torne maior, e no caso do armazenamento em apenas uma ferramenta é possível que a manutenção e o compreendimento das informações armazenadas se torne difícil. Sendo assim, o uso de várias ferramentas não é descartado, porém a ideia é que as informações sejam centralizadas. Dessa forma, supondo que a *startup* utilize Base Camp, Trello ou outra ferramenta similar para gerenciar o projeto e, para comunicação, utilize Slack ou similares, é recomendável que as conversas que são relevantes ao projeto sejam posteriormente documentadas e transferidas para a ferramenta de gestão.

Resultados esperados: Ao minimizar a dispersão das informações do projeto tem-se como resultados esperados:

- rastreabilidade das informações do projeto;
- facilidade e agilidade no acesso as informações do projeto;
- recursos não desperdiçados em tarefas que não agregam valor ao produto.

Ferramentas recomendadas: A ferramenta irá depender do perfil da equipe, porém se aplicam as mesmas ferramentas recomendadas na Diretriz 1, sendo elas: Axosoft OnTime, Microsoft TFS, Rally Platform, Version One, Blossom.io, Base Camp, AgileZen,

Kanbanize e IceScrum. Outras ferramentas interessantes são Trello e Notion.

Fonte: Taheri e Sadjadi (2015) e Reichert *et al.* (2015).

Diretriz 17: Utilizar ferramentas de coleta de comportamento dos usuários.

Descrição: Coletar informações a respeito do comportamento do usuário durante a utilização do produto pode fornecer métricas importantes quanto ao desempenho do software, permitindo analisar se as expectativas em relação ao seu funcionamento estão sendo cumpridas. Além disso, as métricas coletadas podem ser um forte indicador para a implementação de novas *features* e de possíveis melhorias em funcionalidades já existentes. Estas métricas podem ajudar a entender de onde os usuários vieram e como eles interagiram com o produto (Hokkanen e Väänänen-Vainio-Mattila, 2015). Isto pode indicar informações importantes, como, por exemplo, uma *feature* secundária do produto que está sendo mais utilizada do que as *features* principais, podendo ser o gatilho para uma mudança no produto. Mudanças no produto também podem acarretar em mudanças no comportamento dos usuários, que poderão ser identificadas por meio da coleta dessas métricas. A partir deste conhecimento é possível analisar que tipo de comportamento ocasionou em um *feedback* positivo dos usuários (Hokkanen e Väänänen-Vainio-Mattila, 2015).

As informações coletadas também podem auxiliar no melhoramento da UX do produto. De acordo com Dande *et al.* (2014), a UX é muitas vezes o principal fator de sucesso para *startups* e por isso deve ser desenvolvida apropriadamente. Projetar uma boa UX envolve a escolha da funcionalidade adequada e o desenho da interface do produto (Hokkanen e Väänänen-Vainio-Mattila, 2015). Apesar dos benefícios do desenvolvimento de uma boa experiência de usuário serem conhecidos, o mesmo é muitas vezes negligenciado em detrimento de outras etapas do desenvolvimento do produto que são consideradas de maior importância (Hokkanen e Väänänen-Vainio-Mattila, 2015). No estudo realizado por Souza *et al.* (2017), apesar da qualidade do produto ter baixa prioridade para as *startups* avaliadas, a experiência de usuário foi o aspecto de qualidade mais citado, em particular a facilidade de uso e a atratividade da interface de usuário.

Recentemente surgiu na indústria o *Lean UX*, que empresta características da metodologia *Lean* e das metodologias ágeis (Viviano, 2014). Seu objetivo é produzir um produto com rapidez utilizando poucos recursos, porém sem comprometer a satisfação do usuário. São seis os princípios do *Lean UX*, sendo que os mesmos podem ser generalizados como

boas práticas para o desenvolvimento de uma boa experiência de usuário.[6]

Resultados esperados: Os resultados esperados ao utilizar ferramentas de coleta de comportamento dos usuários são:

- validação contínua por meio dos usuários;
- aprendizagem validada;
- aumento da satisfação dos usuários;
- retenção dos usuários;
- redução dos custos com suporte;
- aumento das vendas do produto.

Ferramentas recomendadas: Google Analytics, Flurry, Heap, Mixpanel e UXCam são algumas das ferramentas que auxiliam no rastreamento do comportamento do usuário. Também podem ser utilizados usuários de testes, que irão testar os protótipos ou versões iniciais do produto e relatar suas experiências.

Teste A/B também é uma opção de coleta, podendo ser feito através de dois protótipos ou até mesmo por meio de uma simples *landing page* ao alterar o seu layout após um certo período de tempo. Se o contato direto com o usuário for possível, outras ferramentas mais simples como questionários, entrevistas, análise documental e dinâmicas de grupo também podem ser utilizadas.

Após a coleta, algumas das ferramentas que podem ser utilizar para melhor a UX do produto são: Balsamiq, Sketch, MockFlow, Justinmind, Pixate, InVision, TypeForm, Usabilla, Lookback, Mixpanel e KISSmetrics.

Fonte: Giardino *et al.* (2014), Hokkanen e Väänänen-Vainio-Mattila (2015), Dande *et al.* (2014) e Ries (2011).

Diretriz 18: Praticar *code review*.

[6] `https://www.smashingmagazine.com/2014/01/lean-ux-manifesto-principle-driven-design/`. Acesso em 23 de abr. de 2018.

Descrição: O *code review* consiste na inspeção do código por outro desenvolvedor que não seja o autor do código em questão. Esta técnica é uma prática comum em organizações que seguem a metodologia XP, porém também é utilizada fora deste contexto. No questionário aplicado, 35 *startups* indicaram utilizar a técnica.

No passado, a técnica consistia em uma análise formal e altamente estruturada do código, onde análises linha por linha eram realizadas em grupo em longas inspeções (Bacchelli e Bird, 2013). Apesar dos benefícios, principalmente se tratando da detecção de problemas, o incômodo e demorado processo de análise dificultava em muito a adoção da técnica (Shull e Seaman, 2008). Atualmente, entretanto, as organizações estão adotando métodos mais leves e informais de praticar o *code review*, onde inclusive são utilizadas ferramentas que dão suporte à ocorrência da prática (Bacchelli e Bird, 2013).

O principal benefício do *code review* é o seu auxílio na detecção de defeitos, sendo assim o uso da técnica proporciona uma maior qualidade no estado do produto. Em *startups*, onde recursos como tempo e financiamento são escassos, a prática é ainda mais importante pois evita que sejam necessárias a realização de sessões de retrabalho, muito mais demoradas e dispendiosas do que curtas sessões de análise de código.

Resultados esperados: Os resultados esperados ao praticar *code review* são:

- auxílio na detecção de defeitos;
- aumento da qualidade do produto.

Ferramentas recomendadas: As ferramentas de controle de versão facilitam o *code review*, umas vez que o mesmo código está disponível para todos os desenvolvedores, e ferramentas como Github e Bitbucket disponibilizam comentários de código que facilitam este trabalho. Outras ferramentas específicas incluem: Collaborator, Atlassian Crucible e Reviewable.

Fonte: Bacchelli e Bird (2013), Shull e Seaman (2008) e resultados do *survey* (35% dos respondentes, questão 18).

Diretriz 19: Gerenciar a dívida técnica.

Descrição: *Startups* buscam produzir e lançar software rapidamente com o objetivo de encontrarem seu espaço no mercado. Devido a essa agilidade, é comum que o software

seja liberado com falhas (Curtis *et al.*, 2012). Isso faz com que a *startup* tenha que balancear entre lançar um software de menor qualidade mais cedo ou um software de maior qualidade mais tarde, ou seja, é preciso decidir qual nível de qualidade é aceitável e quais compromissos terão que ser feitos durante o processo de desenvolvimento (Terho *et al.*, 2016; Yli-Huumo *et al.*, 2014). Ao construir um MVP, conforme descrito na Diretriz 6, é comum que isso ocorra, visto que a *startup* está focada em entregar um produto funcional da forma mais rápida possível, ocasionando comprometimentos em relação a qualidade do código do produto.

Essa abordagem rápida no desenvolvimento do produto aumenta o conhecimento tácito, porém, diminui as chances de que as atividades de engenharia sejam documentadas (Souza *et al.*, 2017). Negligenciar a documentação torna ainda mais difícil o rastreamento dos pontos de dívida técnica que deverão ser tratados pela *startup*. Entre as *startups* analisadas por Souza *et al.* (2017) foi percebido que essas organizações gerenciam o projeto minimamente, realizam um estudo de viabilidade grosseiro (Diretriz 5), possuem uma arquitetura de projeto escassa e registram as especificações de maneira informal (Diretriz 21).

A partir disso a *startup* acumula dívida técnica. A dívida técnica é uma metáfora em relação ao comprometimento técnico em virtude de um benefício a curto prazo mas que poderá custar caro à saúde do software no longo prazo (Li *et al.*, 2015). O gerenciamento da dívida técnica busca identificar e monitorar os pontos de dívida técnica inseridos no projeto de forma que os mesmos estejam explícitos e sejam pagos em um momento oportuno, evitando transtornos maiores (Alves e Spínola, 2017). Ou seja, gerenciar ou reduzir a dívida técnica é de extrema importância para a organização, caso contrário a mesma poderá afetar negativamente o desempenho da *startup* no futuro, uma vez que implementar novas *features* e até mesmo manter o código atual se tornará uma tarefa difícil. *Startups* nesta situação correm sérios riscos de falhar (Chicote, 2017).

Chicote (2017) apresenta uma estratégia de gerenciamento da dívida técnica por meio de *Visual Thinking*, que seria "pensamento visual" em uma tradução literal. A técnica consiste em colar um pedaço de fita adesiva em algum lugar que seja constantemente visível para o time, como um quadro ou uma parede, toda vez que é feito um comprometimento em termos de código que irá resultar em dívida técnica. Dessa forma, cada pedaço de fita irá representar uma parte do código que foi desenvolvido com baixa qualidade ou de forma temporária. Chicote (2017) aponta que o tamanho do pedaço de fita deve ser correspondente ao tamanho da dívida técnica que está sendo assumida. Apesar de não ser uma representação fiel, o tamanho do pedaço de fita será um indicativo de quais partes

do código devem ter prioridade no momento de "pagar" a dívida.

Resultados esperados: Gerenciar a dívida técnica tem como resultados esperados:

- maior qualidade e produtividade no desenvolvimento do produto;
- rastreamento dos pontos de dívida técnica presentes no código;
- maior agilidade na implementação de novas *features*.

Ferramentas recomendadas: A refatoração de código é uma prática simples porém efetiva contra a dívida técnica, evitando que o código se torne defasado por meio do aprimoramento de código existente. A refatoração permite melhorar a estrutura interna do código sem que seu comportamento seja alterado. O uso de estilos de codificação e *code review* também auxiliam em manter a dívida técnica baixa.

As ferramentas indicadas na Diretriz 1, assim como outras ferramentas já comentadas, como Trello e Redmine, podem ser utilizadas como forma de gerenciar a dívida técnica, mantendo um registro dos pontos onde existe uma baixa qualidade de código, facilitando o seu rastreamento e consequentemente a sua manutenção.

Fonte: Njima e Demeyer (2017), Terho *et al.* (2016), Yli-Huumo *et al.* (2014), Souza *et al.* (2017), Chicote (2017) e Li *et al.* (2015).

Diretriz 20: Utilizar indicadores chave de desempenho.

Descrição: O uso de indicadores chave de desempenho são uma forma de mensurar o desempenho de alguma atividade em particular, visando enxergar as demandas do cliente. Por meio destes indicadores é possível determinar com que eficiência a *startup* está alcançando os pontos chave para o sucesso do produto. Estes indicadores permitem verificar pontos como o atrito do cliente, tempos de ciclo de utilização do produto e etc. (Ries, 2011). O uso de indicadores chave de desempenho é importante pois os mesmos auxiliam na tomada de decisão, permitindo identificar demandas específicas dos clientes que não seriam descobertas por meio de outras ferramentas. Os indicadores irão variar de acordo com o mercado que o produto busca atingir, sendo assim, os mesmos devem ser definidos de acordo com as necessidades da *startup*.

O *Balanced Scorecard* (BSC) pode ser utilizado para analisar aspectos chave do desempenho da organização. O BSC permite que a organização defina suas metas e

estratégias e então verifique o seu desempenho por meio de indicadores. A ferramenta possibilita a comunicação de estratégias e as ações necessárias para ter sucesso (Ratnavali e Murthy, 2016), auxilia no alinhamento da equipe por meio de objetivos e no reforço dos resultados por meio de recompensas e ajuda os gerentes a focarem nas áreas importantes e em métricas críticas (Kaplan e Norton, 2004). Para Hubbard (2009), o BSC unifica os interesses de todos os *stakeholders* por meio do alinhamento em atingir o desempenho organizacional.

O uso de sistemas de gestão de desempenho, em inglês *performance management system* (PMS), como o BSC, permitem mensurar os resultados para gerenciar o desempenho individual, medindo o comportamento dos desenvolvedores e os resultados que os mesmos produzem (Radnor e McGuire, 2004). Seu uso proporciona uma cultura de desempenho, trazendo alinhamento entre a organização, a equipe e as metas individuais, define padrões de desempenho e expectativas, facilita a comunicação e proporciona recompensas pela contribuição do membro da equipe (Ratnavali e Murthy, 2016), sendo assim estes sistemas estão fortemente ligados ao empoderamento da equipe de desenvolvimento (Diretriz 12).

Os indicadores podem ser gerados por meio de quaisquer dados que a *startup* possua, porém é provável que seja necessária uma abordagem mais ativa na obtenção destes dados como, por exemplo, os processos de obtenção de *feedback* descritos nas Diretrizes 14 e 18. Após definir de que forma irá obter os dados necessários para a alimentação dos indicadores, a *startup* deverá monitorar constantemente os seus estados, sendo que isso pode ser feito por meio de ferramentas automáticas ou de forma manual.

Resultados esperados: Os resultados esperados do uso de indicadores chave de desempenho são:

- apoio na tomada de decisão;
- aprendizagem validada;
- auxílio na realização dos objetivos da organização.

Ferramentas recomendadas: Entre as ferramentas recomendadas para o uso de indicadores chave de desempenho estão: ClearCompany, BambooHR, Namely, Bizmerlin, Lumesse, empXtrack, Oracle HCM e SuccessFactors. Também é possível rastrear os indicadores chave de desempenho por meio de ferramentas mais simples como planilhas ou papel e caneta.

Outra possibilidade é o uso do *Business Model Canvas*, ou, mais especificamente o *Lean Canvas*, que é uma versão adaptada para *startups* (Maurya, 2012). O canvas pode ser impresso e fixado em uma área visível para os membros da equipe, de forma que seja discutido constantemente.

Fonte: Ries (2011), Ratnavali e Murthy (2016) e resultados do *survey* (39% dos respondentes, questão 17).

Diretriz 21: Estabelecer um processo de gerência de requisitos do produto.

Descrição: Identificar os requisitos de um produto é uma tarefa difícil. Em *startups*, devido ao contexto de incertezas em que estão inseridas, mais do que em organizações tradicionais, os requisitos do produto muitas vezes não são claros, sendo necessário um grande esforço para que estas *startups* saibam o que devem desenvolver (Gonçalves, 2017; Melegati e Goldman, 2016). Por estarem envolvidas em um ambiente de mudanças frequentes os requisitos de software são difíceis de extrair e mudam constantemente (Paternoster *et al.*, 2014). Sendo assim, é grande a importância de gerenciar os requisitos nestas organizações, visando construir um produto que atenda as reais necessidades do cliente e evitar que a *startup* falhe pois não foi capaz de entendê-las. Apesar de existirem casos de *startups* de sucesso que não seguem um processo bem definido de gerência de requisitos, é constatado que informações importantes são constantemente perdidas e o processo de trabalho se torna mais longo, enquanto que *startups* com um processo bem definido conseguem entregar um produto de maior qualidade e evitam a dívida técnica (Diretriz 19) (Gralha *et al.*, 2018).

No estudo de caso desenvolvido por Leal (2015), a *startup* em foco destacou enfrentar dificuldades quanto a manutenção e a conversão dos requisitos identificados devido a falta de padronização quanto a documentação dos artefatos que guardam os requisitos. Isso gera uma documentação incompleta e inconsistente. Visto que muitas vezes estes requisitos demoram até começarem a ser desenvolvidos, muitas informações sobre a sua especificação são perdidas, sendo que até mesmo quem registrou o requisito acaba por esquecer das necessidades que foram levantadas. A *startup* relatou que isto acaba por gerar retrabalho, perda de produtividade e impacta negativamente na qualidade do produto. Segundo Paternoster *et al.* (2014) as práticas de engenharia de requisitos estão limitadas a algumas práticas principais, apresentam uma dificuldade crescente devido ao progressivo número de usuários e demanda um maior envolvimento dos mesmos. Entre os principais desafios estão a pouca documentação, disponibilidade do cliente, estimativa de tempo

e orçamento, negligência de requisitos não funcionais, incapacidade e concordância do cliente, limitações contratuais e mudança de requisitos e sua avaliação (Gonçalves, 2017).

Entender as necessidades do cliente é essencial, aponta o estudo realizado por Gordón e O'Connor (2016). Alguns dos desenvolvedores das organizações entrevistadas no estudo relatam que participar do processo de levantamento dos requisitos os dá conhecimento, sendo que quando não estão envolvidos precisam revisar a especificação para garantir que a mesma não está incompleta. Caso a especificação contendo as necessidades do cliente não exista, se torna muito difícil satisfazer o usuário e até mesmo finalizar o projeto se torna complicado. A elicitação dos requisitos é portanto uma etapa crucial do processo de gerência de requisitos (Rafiq *et al.*, 2017), entretanto, por sua característica dirigida a mercado, onde não há clientes ou usuários bem definidos até que o produto seja lançado (Dahlstedt *et al.*, 2003), *startups* possuem dificuldades para a execução desta etapa. Prototipagem, debates informais, questionários, uso de mídias sociais, análise de *feedback* e, estes mais inovadores, análise de produtos similares ou competidores, discussões colaborativas e uso de usuários modelo são algumas das técnicas utilizadas por *startups* durante a fase de elicitação (Rafiq *et al.*, 2017).

Estabelecer um processo de gerência de requisitos do produto consiste em definir como os requisitos serão coletados, analisados, documentados e validados. Na fase de coleta a *startup* deve consultar os possíveis usuários a fim de identificar quais problemas o produto se propõe a resolver para então definir quais serão os seus limites (Vajrapu e Kothwar, 2018). Após a coleta será necessário realizar uma análise destes requisitos, visando definir o que realmente será implementado no produto. Esse processo ocorre por meio da negociação com os *stakeholders*, resolvendo pontos de conflito e ajustando os limites do produto de acordo com o escopo do projeto. Os requisitos então são documentados e monitorados durante o desenvolvimento do produto. Estes passos são muito importantes, uma vez que uma documentação inconsistente pode dificultar a implementação dos requisitos e, monitorar os requisitos visa garantir que o produto está sendo desenvolvido conforme aquilo que foi definido. Por fim os requisitos devem ser validados, isto é, deve ser verificado se os requisitos estão completos e consistentes para que possam ser implementados. Após a implementação deve ser verificado se os requisitos atingiram os seus objetivos (Vajrapu e Kothwar, 2018).

Resultados esperados: Ao estabelecer um processo de gerência de requisitos do produto os resultados esperados são:

- garantir que o produto esteja de acordo com os requisitos;

- manter um histórico dos requisitos;
- evitar o retrabalho;
- maior produtividade.

Ferramentas recomendadas: Algumas ferramentas que podem auxiliar no gerenciamento dos requisitos são: Atlassian JIRA, Pencil Project, Caliber, codeBeamer, VersionOne, Axosoft, Balsamiq Mockups, Kanban Flow, Mingle, Pivotal Tracker, Trello e Redmine. Para a coleta dos requisitos podem ser utilizados questionários, entrevistas, análise documental, prototipagem, debates, uso de mídias sociais, análise de *feedback*, análise de produtos similares ou competidores, discussões colaborativas, uso de usuários modelo e cartões de estórias.

Fonte: Melegati e Goldman (2016), Gonçalves (2017), Paternoster *et al.* (2014), Gordón e O'Connor (2016), Vajrapu e Kothwar (2018) e resultados do *survey* (37% dos respondentes, questão 17).

Diretriz 22: Estabelecer uma política de manutenção do produto.

Descrição: A fase de manutenção do software é uma das mais críticas do seu ciclo de vida, sendo responsável por cerca de 50 a 90% dos custos envolvidos (April e Abran, 2012; Grubb e Takang, 2003). No caso de *startups*, devido a escassez de recursos que estas organizações enfrentam, planejar a manutenção do software de forma que a mesma não exceda os gastos é imprescindível. A manutenção do produto garante que o mesmo continuará funcionando, sendo que para *startups* isto é ainda mais importante, pois significa reter um usuário que foi difícil de adquirir. Para preservar a integridade do produto no decorrer das mudanças é preciso definir um processo onde as modificações possam ser registradas e monitoradas, onde o impacto das alterações propostas possa ser determinado e onde o código e outros artefatos de software possam ser modificados e testados (Moreira, 2015).

A manutenção do produto consiste em registrar e monitorar solicitações de mudanças, determinar o impacto das alterações, modificar e testar o produto e, por fim, liberar uma nova versão do produto (Moreira, 2015). As solicitações de mudanças nem sempre são explícitas, podendo surgir a partir de problemas encontrados no software (Diretriz 4), demandas não atendidas descobertas por meio da análise do comportamento dos usuários (Diretriz 18), problemas relacionados ao desempenho do software (Diretriz 23),

entre outros. Com as mudanças necessárias registradas, é preciso analisar quais os impactos relacionados a implementação dessas mudanças. Isto envolve analisar os custos e recursos que deverão ser empregados, quais serão os impactos para os usuários e como minimizá-los, etc. Em seguida serão realizadas as modificações necessárias visando atender as solicitações de mudanças que foram registradas. Após proceder com as modificações, é de extrema importância que a *startup* realize testes para verificar se as mudanças comprometeram de alguma forma a integridade ou funcionalidade do produto. Por fim ocorre o *release* de uma nova versão do produto contendo as modificações que foram implementadas. Nesta etapa a *startup* pode optar por utilizar seus canais de comunicação com os clientes para informá-los a respeito das mudanças, visando obter seu *feedback* (Diretriz 14).

Resultados esperados: Os resultados esperados ao estabelecer uma política de manutenção do produto são:

- economia de recursos;
- maior longevidade do produto;
- retenção dos usuários.

Ferramentas recomendadas: As ferramentas indicadas na Diretriz 1 devem ser suficientes para registrar e monitorar as mudanças decorrentes da manutenção. Outras ferramentas interessantes que não foram citadas na Diretriz 1 são: Atlassian JIRA, Trac e Redmine. Também é possível utilizar ferramentas mais especificas ao rastreamento de defeitos como Bugzilla e Mantis.

Fonte: April e Abran (2012) e Moreira (2015).

Diretriz 23: Analisar o desempenho do software.

Descrição: *Startups* lançam atualizações constantes de seus produtos em virtude dos ciclos rápidos em que estas organizações trabalham. A diminuição do desempenho do software após atualizações no software é um problema para qualquer organização que produza software, porém, no caso de *startups* pode ser um problema ainda maior devido a este ciclo constante de mudanças. A diminuição do desempenho do software tem efeitos negativos como o aumento dos custos com a manutenção e a insatisfação do

cliente (Khanna *et al.*, 2006; Nguyen, 2012), podendo até mesmo levar a grandes perdas financeiras.

Analisar o desempenho do software permite identificar essas diminuições assim que as atualizações ocorrem (Ahmed *et al.*, 2016). Por meio de ferramentas de análise de desempenho é possível verificar o desempenho do software em requisições individuais ou simultâneas, uso e desempenho de todas as dependências, rastreamento detalhado de transações até linhas de código específicas, utilização de memória e processador no servidor, erros da aplicação, métricas customizadas, etc.

Uma das formas de verificar o desempenho do software é fazê-lo antes do lançamento de uma atualização, possibilitando detectar qualquer queda de desempenho antes que o produto chegue ao cliente. Isto é feito por meio de testes de regressão de desempenho. Estes testes constituem no processo de aplicar uma carga de trabalho em duas versões diferentes do software, uma estável, onde é sabido que não há problemas de desempenho e outra que é a nova versão. Os testes permitem verificar se as mudanças do código introduziram regressões de desempenho ou não (Ahmed *et al.*, 2016).

Outra opção é o uso de ferramentas de gerenciamento de desempenho de aplicação, em inglês *Application Performance Management* (APM). Estas ferramentas são bastante utilizadas na prática (Ahmed *et al.*, 2016), principalmente para detectar anomalias no desempenho ao invés de detectar regressões. Isto é feito por meio de uma combinação de abordagens de mineração de dados de desempenho com ferramentas *off-the-shelf* de monitoramento. Uma APM é normalmente utilizada para monitorar o desempenho e a disponibilidade de aplicações web (Wang *et al.*, 2014). A ferramenta coleta várias métricas de desempenho, como tempo de resposta, da aplicação monitorada e minera essas métricas para medir a saúde da aplicação (Ahmed *et al.*, 2016). Apesar destas ferramentas serem primariamente voltadas para minerar grandes quantidades de dados de desempenho para detectar anomalias em códigos estáveis por meio de mudanças na carga de trabalho, o estudo de Ahmed *et al.* (2016) concluiu que as ferramentas APM são eficazes em detectar diminuições de desempenho causadas por mudanças no código por meio da aplicação de cargas de trabalho estáveis.

Resultados esperados: Os resultados esperados ao analisar o desempenho de software são:

- indicação de desempenho do produto;
- detecção de defeitos após atualizações;

- diminuição dos custos de manutenção.

Ferramentas recomendadas: As ferramentas recomendadas podem variar de acordo com a linguagem de programação, visto que, como em outros casos, algumas delas são específicas daquela linguagem. Entre as mais populares é possível citar Arm MAP e AppDynamics para as linguagens C e C++, inspectIT, JConsole, JProfiler e Pinpoint para a linguagem Java, Dbg e Xdebug para a linguagem PHP, Scout Devtrace e Rack trace para a linguagem Ruby e Firebug para a linguagem JavaScript.

Entre as ferramentas que suportam diversas linguagem de programação estão: New Relic APM, AppDynamics, Stackify Retrace, Dynatrace, TraceView, Riverbed SteelCentral, Dell Foglight e JenniferSoft.

Fonte: Ahmed *et al.* (2016) e Nguyen (2012).

Diretriz 24: Introduzir práticas de suporte à melhoria de processo.

Descrição: *Startups* geralmente precisam de investimentos externos para realizar as suas operações (Paternoster *et al.*, 2014; Sutton, 2000), sendo que *startups* que possuíram investimentos possuem uma taxa de *turnover* menor do que as demais (Suominen *et al.*, 2017). A melhoria de processo pode ser um diferencial na aquisição de recursos, visto que organizações de financiamento, como aceleradoras, investidores anjo e etc., provavelmente darão prioridade a organizações que possuam um processo melhor definido e de maior qualidade. A implementação de melhoria de processo em grandes organizações é algo comum, diferentemente de organizações menores, como *startups*. Estas organizações menores entendem que os esforços referentes a melhoria de processo são desenvolvidos por e para grandes organizações, são custosos e requerem muita documentação e burocracia (Gordón e O'Connor, 2016). Entretanto, modelos de melhoria de processo voltados para pequenas organizações já existem, como ISO/IEC 29110, Competisoft e ITmark, (Larrucea *et al.*, 2016) e até mesmo modelos tradicionais podem ser adotados com sucesso nestas organizações (Dangle *et al.*, 2005).

Gordón e O'Connor (2016) apontam que estas organizações menores estão interessadas e conhecem sobre processo de software e padrões de qualidade. As organizações entrevistadas demonstraram acreditar nos potenciais benefícios da adoção de um padrão de qualidade, particularmente do ISO/IEC 29110. Entre os benefícios estão um produto de qualidade, melhora da imagem da organização, melhora do processo de trabalho,

consistência no trabalho de desenvolvimento e maior lucratividade em virtude de menos tempo ser gasto em trabalho não produtivo.

A melhoria de processo auxilia a organização a visualizar as suas metas de negócio, sendo que isso será transferido para a qualidade do software, que estará melhor alinhado com os objetivos da *startup* e consequentemente irá atender melhor as necessidades dos usuários (Dangle *et al.*, 2005). A adoção de um processo de melhoria ajuda a identificar e organizar os processos de acordo com as suas áreas, tornado-os distintos, porém de uma forma que essas áreas possam trabalhar em uníssono em direção ao cumprimento dos objetivos da *startup*. Além disto, estabelecer um processo repetível e que pode ser executado por várias pessoas auxilia no entendimento das tarefas que devem ser executadas pelos membros da equipe, tornando o desenvolvimento do produto mais ágil e mais eficiente (Dangle *et al.*, 2005).

Resultados esperados: Os resultados esperados ao introduzir práticas de suporte à melhoria de processo são:

- padronização dos processos;
- redução dos custos;
- aumento da qualidade do software;
- agilidade na construção de software.

Ferramentas recomendadas: Larrucea *et al.* (2016) apontam o ISO/IEC 29110 como o padrão mais adotado pelas organizações menores, porém, outros padrões como Competisoft e ITmark também são focados nessas organizações. No cenário nacional existem alternativas como o PQ2S, desenvolvido em conjunto por Sebrae e Senai, e o MPS.br, que se mostrou satisfatório no contexto de *startups* (Silva e de Miranda Junior, 2016). Caso a *startup* possua os recursos e a estrutura necessária, práticas de modelos tradicionais como CMMI e ISO/IEC 15504 também podem ser adotadas.

Fonte: Gordón e O'Connor (2016), Dangle *et al.* (2005) e Larrucea *et al.* (2016).

5.5 Considerações Finais

Neste capítulo foram apresentadas as diretrizes propostas. Estas diretrizes são a parte mais importante deste trabalho, pois espera-se que as mesmas possam auxiliar as atividades de desenvolvimento de software em *startups* e, possivelmente, aumente as chances de sucesso dessas organizações, evitando que falhem e deixem de existir precocemente. Este, porém, não é um objetivo simples, portanto estas diretrizes foram subsidiadas por tudo que foi apresentado durante os Capítulos 2, 3 e 4 e validadas pelos processos que serão apresentados no Capítulos 6.

O capítulo também apresenta o formato de especificação das diretrizes, que visa estruturar as diretrizes de forma que facilite a sua leitura e compreensão. Apesar de trabalhos similares existirem, nenhum deles continha um formato de especificação considerado apropriada para a especificação das diretrizes, sendo o mais próximo do desejado o trabalho de Leal (2015), de onde o formato foi inspirado e adaptado.

6

Validação e Refinamento das Diretrizes Propostas

6.1 Considerações Iniciais

Neste capítulo é apresentada a fase de refinamento do trabalho. Esta etapa teve como objetivo avaliar as diretrizes propostas visando adquirir conhecimento sobre as suas aplicações e, com isto, identificar seus benefícios e oportunidades de melhoria. O refinamento é divido em três etapas: condução de um questionário confirmatório com as *startups* participantes do *survey* apresentado no Capítulo 4, condução de um grupo focal e condução de um painel com especialistas, os quais são descritos nas seções seguintes.

6.2 Survey Confirmatório

O *survey* confirmatório foi conduzido buscando obter a opinião das *startups* participantes do *survey*, apresentado no Capítulo 4, a respeito das diretrizes propostas. Uma vez que várias das ideias encontradas nas diretrizes são consequências de achados do *survey*, é importante entender o que as *startups* que proporcionaram essa informação tem a dizer a respeito das diretrizes. Este questionário pode ser classificado como de avaliação (Wohlin e Aurum, 2015), pois busca avaliar as diretrizes propostas sob a perspectiva das *startups*.

Em relação a técnica utilizada para a coleta dos dados, a mesma é caracterizada como de corte transversal, uma vez que a compilação dos dados ocorreu em um período específico de tempo (Sampieri *et al.*, 2010). O questionário foi conduzido de acordo com

o método proposto por Forza (2002), que é subdivido em quatro fases: Planejamento, Experimento Piloto, Coleta dos Dados e Análise dos Resultados.

6.2.1 Planejamento

A fase de Planejamento visa a prevenção de problemas durante as fases seguintes do questionário e busca garantir a qualidade do processo de pesquisa. Esta fase é composta pelas seguintes atividades: seleção de contexto, seleção dos participantes e desenvolvimento dos instrumentos de pesquisa.

6.2.1.1 Seleção de Contexto

O questionário considerou todas as *startups* que participaram do *survey* apresentado no Capítulo 4.

6.2.1.2 Seleção dos Participantes

A lista de *startups* foi obtida por meio do e-mail daquelas que participaram da pesquisa anterior. Todas as *startups* foram convidadas via e-mail a participarem da pesquisa.

6.2.1.3 Desenvolvimento dos Instrumentos de Pesquisa

O instrumento de pesquisa consiste de dois documentos, sendo eles a Carta de Apresentação e o Questionário. A Carta de Apresentação (verificar Apêndice D) buscou apresentar formalmente os pesquisadores e o propósito da pesquisa para as *startups*, visando obter sua colaboração para responder o questionário. O Questionário (verificar Apêndice E e L) apresenta as questões que compõem o *survey*, desenvolvidas de acordo o propósito da pesquisa, ele está estruturado em 2 seções: i) caracterização do respondente, buscando identificar o perfil do respondente do questionário; ii) diretrizes para o desenvolvimento de software em startups, que tem por finalidade verificar o grau de concordância do respondente a respeito de cada uma das diretrizes propostas.

6.2.2 Experimento Piloto

O Experimento Piloto visa analisar os processos de medição do questionário e verificar a viabilidade da sua aplicação, ou seja, seu objetivo é testar o que foi planejado. Sendo assim, o Experimento Piloto foi conduzido com o objetivo de analisar o comportamento do instrumento de pesquisa em uma situação real de coleta de dados, possibilitando verificar

se as questões estavam compreensíveis, se as opções de respostas estavam completas e se o mesmo estava suficientemente sucinto, de forma a facilitar que fosse respondido e assim aumentar a adesão. O experimento piloto foi conduzido em ambiente acadêmico com três pesquisadores da área de Engenharia de Software.

O primeiro dos pesquisadores, da Universidade Estadual de Maringá, com doze anos de experiência, atua na área de apoio à tomada de decisão, englobando as áreas de pesquisa operacional e gestão de tecnologia da informação, e possui interesse nas áreas de otimização, desenvolvimento distribuído de software, mapeamento e melhoria de processos, capitalização de conhecimento e qualidade de software. O segundo pesquisador, da Universidade do Pernambuco, também com doze anos de experiência, atua nas áreas de pesquisa, desenvolvimento e inovação em Engenharia de Software, melhoria de processos de software e gerência de projetos e métodos ágeis, e também atua como consultor, implementador e avaliador dos modelos MPT.BR e implementador dos modelos MR-MPS-SV. O terceiro dos pesquisadores, também da Universidade Estadual de Maringá, possui quinze anos de experiência, atua nas áreas de Engenharia de Software, computação aplicada à saúde e desenvolvimento distribuído de software, e atualmente é bolsista CNPq de Desenvolvimento Tecnológico e Científico, categoria DTC-A.

Após a condução do experimento piloto, alguns ajustes foram realizados para melhorar a legibilidade do questionário, adequando a descrição de algumas das diretrizes a fim de melhorar o compreendimento das mesmas. O experimento piloto também permitiu estimar o tempo médio para responder o questionário em 20 minutos.

6.2.3 Coleta de Dados

A Coleta de Dados tem por objetivo reunir as informações relevantes do questionário, proporcionando as informações necessárias para o cumprimento do objetivo da pesquisa. A Coleta de Dados foi realizada utilizando a ferramenta de formulários do Google (`https://forms.google.com`). A escolha de uma ferramenta on-line para a coleta dos dados permite uma maior abrangência, principalmente geográfica, pois qualquer *startup* com conexão a Internet tem acesso ao formulário.

O procedimento de coleta consistiu do envio de e-mail para as *startups* encontradas contendo a carta de apresentação e o link de acesso ao formulário (`https://goo.gl/forms/JlPfiegZqRChopXc2`). O formulário ficou disponível por um período de 90 dias e as solicitações para participar da pesquisa foram enviadas duas vezes para cada *startup*.

6.2.4 Análise dos Resultados

A Análise dos Resultados é fundamental para os objetivos da pesquisa, pois permite investigar os dados por diversos ângulos e também compará-los, de forma a produzir novos conhecimentos. Neste questionário, a Análise dos Resultados foi realizada por meio da ferramenta de planilhas LibreOffice Calc. A partir da ferramenta os dados foram estruturados e entrelaçados conforme as visões desejadas, permitindo destacar os fatos mais importantes e possibilitando o compreendimento das informações coletadas.

6.2.5 Resultados

A pesquisa foi conduzida com *startups* brasileiras de desenvolvimento de software, onde o total de participantes foi de 26 *startups*. O questionário foi enviado para 100 *startups* diferentes, que são aquelas que participaram do primeiro *survey*, descrito no Capítulo 4, dessa forma a taxa de resposta da pesquisa foi de 26%. Na condução do primeiro questionário as *startups* não foram informadas de que poderiam ser convidadas a participar de um segundo, portanto, não é possível dizer se isso influenciaria a taxa de resposta. O autor considerou que caso o respondente tivesse conhecimento prévio de que haveria um segundo questionário, a taxa de resposta seria ainda menor, pois o respondente ficaria receoso em se comprometer.

Como o objetivo do questionário confirmatório não é de caracterizar as *startups*, uma vez que essas já foram caracterizadas, questões com este objetivo não foram incluídas no *survey*. Em relação ao respondente, como não é possível afirmar que a pessoa que respondeu o primeiro *survey* é a mesma que iria responder este, considerou-se interessante incluir questões para caracterizá-lo novamente.

6.2.5.1 Caracterização do Respondente

Essa seção apresenta o tempo de experiência com desenvolvimento de software, o tempo de experiência com *startups* e o grau de formação do respondente. Esses elementos visam caracterizar o perfil do respondente do questionário. Por alguma razão desconhecida, um dos respondentes não respondeu ou não teve suas respostas registradas na caracterização, dessa forma, essa seção apresenta 25 respostas.

A Figura - 6.1 apresenta o tempo de experiência dos respondentes com desenvolvimento de software. É possível verificar que 84% possuem mais do que 5 anos de experiência, indicando que a maioria dos envolvidos nas *startups* possuem uma quantidade considerável de experiência com desenvolvimento de software. Do restante, 12% possuem entre 3 e 5

anos de experiência, 4% possuem entre 1 e 3 anos e 0 possuem menos do que 1 ano de experiência.

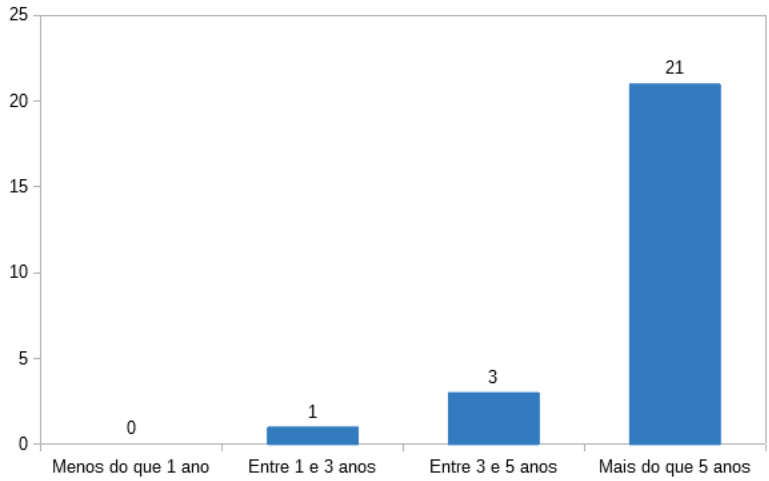

Figura 6.1: Tempo de experiência com desenvolvimento de software

Quanto ao tempo de experiência com *startups*, conforme demonstra a Figura - 6.2, 44% possuem mais do que 5 anos, 32% possuem entre 3 e 5 anos e 24% possuem entre 1 e 3 anos. Nenhum respondente possui menos do que 1 ano de experiência com *startups*.

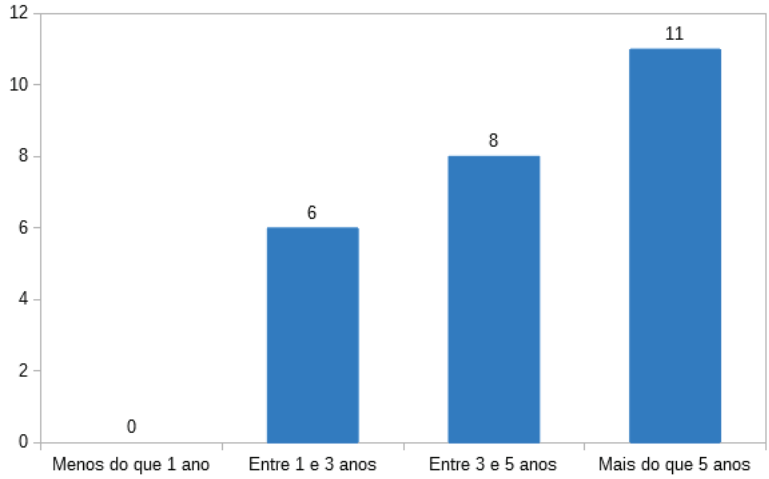

Figura 6.2: Tempo de experiência com *startups*

Em relação ao grau de formação, nenhum dos respondentes indicou Mestrando, Mestre ou Doutorando, sendo que 4% faz Curso Técnico, 4% é Graduando, 40% é Graduado, 44% é Especialista e 8% é Doutor, conforme ilustrado pela Figura - 6.3. Isto demonstra que a maioria dos respondestes são graduados ou especialistas.

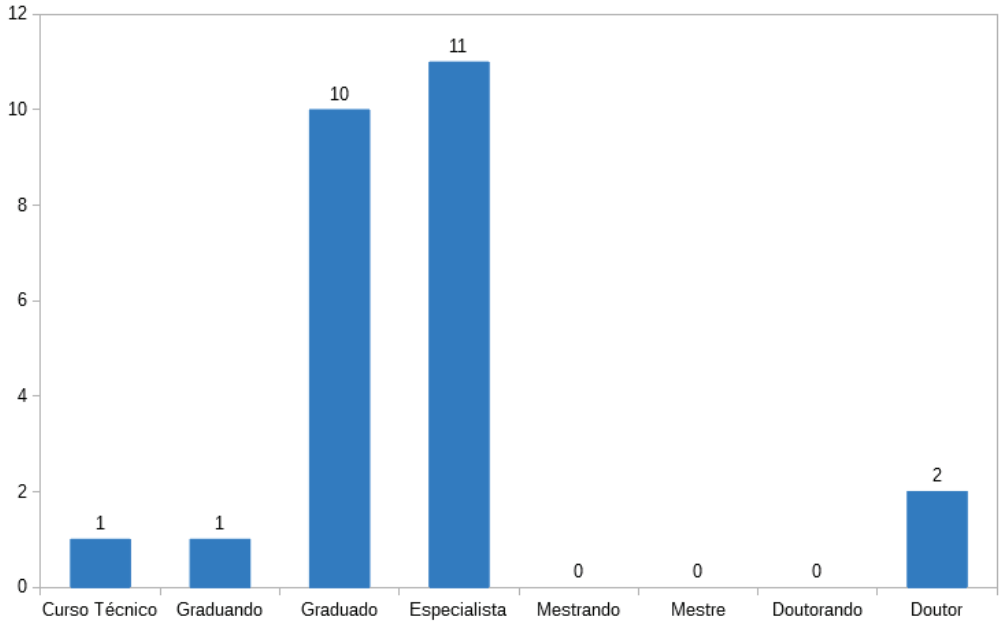

Figura 6.3: Grau de formação dos respondentes

Nas questões destinadas aos respondentes não há algo que específico que permita identificar se a mesma pessoa que respondeu o primeiro questionário é quem está respondendo desta vez, porém, por meio dos dados podemos induzir que pelo menos parte dos respondentes mudou.

6.2.5.2 Diretrizes para o Desenvolvimento de Software em Startups

Essa seção apresenta as 24 diretrizes que foram propostas no Capítulo 5 e demonstra o grau de concordância dos respondentes em relação as mesmas. Para isso, foi utilizado uma escala Likert com as opções: i) discordo totalmente; ii) discordo parcialmente; iii) indiferente; iv) concordo parcialmente e v) concordo totalmente. Em seguida, as diretrizes são apresentadas sequencialmente em conjunto com os percentuais de concordância registrados pelos respondentes do questionário. A Tabela - 6.1 demonstra o nível de concordância de forma resumida, onde cada linha representa uma diretriz e as colunas o percentual de concordância.

No que se refere a Diretriz 1, utilizar uma metodologia ágil para guiar o desenvolvimento de software, 73,1% concordam totalmente e 26,9% concordam parcialmente, demonstrando a forte ligação entre *startups* e o uso de métodos ágeis. Em relação a Diretriz 2, utilizar *frameworks open source* e/ou com grande adoção no mercado, o número de concordância total é ainda maior do que no caso da Diretriz 1, visto que 76,9% concordam totalmente, e 23,1% concordam parcialmente, sendo este um dos principais

Tabela 6.1: Nível de concordância das diretrizes

	Discordo totalmente	**Discordo parcialmente**	**Indiferente**	**Concordo parcialmente**	**Concordo totalmente**
D1	0%	0%	0%	26,9%	73,1%
D2	0%	0%	0%	23,1%	76,9%
D3	0%	0%	23,1%	30,8%	46,2%
D4	0%	0%	7,7%	30,8%	61,5%
D5	0%	0%	3,8%	34,6%	61,5%
D6	0%	0%	0%	19,2%	80,8%
D7	0%	3,8%	15,4%	23,1%	57,7%
D8	0%	3,8%	7,7%	34,6%	53,7%
D9	0%	0%	3,8%	30,8%	65,4%
D10	0%	0%	15,4%	19,2%	65,4%
D11	0%	0%	26,9%	30,8%	42,3%
D12	3,8%	3,8%	0%	34,6%	57,7%
D13	0%	0%	0%	34,6%	65,4%
D14	0%	0%	0%	26,9%	73,1%
D15	0%	3,8%	11,5%	26,9%	57,7%
D16	0%	0%	11,5%	38,5%	50%
D17	0%	3,8%	7,7%	26,9%	61,5%
D18	0%	3,8%	15,4%	34,6%	46,2%
D19	0%	7,7%	30,8%	19,2%	42,3%
D20	0%	0%	19,2%	19,2%	61,5%
D21	3,8%	0%	15,4%	26,9%	53,8%
D22	0%	0%	23,1%	30,8%	46,2%
D23	0%	11,5%	3,8%	26,9%	57,7%
D24	0%	7,7%	3,8%	23,1%	65,4%

pontos de sucesso para uma *startup* de acordo com Giardino *et al.* (2014). Quanto a Diretriz 3, usar *backlog* em conjunto com Kanban independente do método ágil utilizado, 46,2% concordam totalmente, 30,8% concordam parcialmente e 23,1% são indiferentes, sendo essa a primeira diretriz a demonstrar um grau de indiferença. Sobre a Diretriz 4, estabelecer uma rotina de testes, 61,5% concordam totalmente, 30,8% concordam parcialmente e 7,7% são indiferentes.

Em relação a Diretriz 5, validar a ideia antes de começar a desenvolver, 61,5% concordam totalmente, 34,6% concordam parcialmente e 3,8% são indiferentes. Sendo este um dos principais pontos defendidos por Ries (2011) no método *Lean Startup*, é surpreendente que o nível de concordância total não seja maior. No que se refere a Diretriz 6, construir um MVP, 80,8% concordam totalmente e 19,2% concordam parcialmente. Assim como a Diretriz 5, essa diretriz é fortemente relacionada com as propostas de

Ries (2011) e apresenta o maior nível de concordância total entre as diretrizes, com nenhuma indiferença ou discordância. Sobre a Diretriz 7, utilizar uma ferramenta de gerenciamento de versão descentralizada, 57,7% concordam totalmente, 23,1% concordam parcialmente, 15,4% são indiferentes e 3,8% discordam parcialmente, sendo esta a primeira diretriz que apresenta um nível de discordância. A Diretriz 8, priorizar a terceirização da infraestrutura possui números parecidos com a diretriz anterior onde, 53,8% concordam totalmente, 34,6% concordam parcialmente, 7,7% são indiferentes e 3,8% discordam parcialmente.

A respeito da Diretriz 9, modularizar a estrutura do produto, 65,4% concordam totalmente, 30,8% concordam parcialmente e 3,8% são indiferentes, demonstrando ser uma diretriz importante dado o baixo nível de indiferença. Em relação a Diretriz 10, utilizar estilos de codificação, o nível de indiferença é um pouco maior, com 15,4%, do restante, 65,4% concordam totalmente e 19,2% concordam parcialmente. Quanto a Diretriz 11, usar prototipagem evolucionária durante o desenvolvimento do produto, os níveis de concordância são equilibrados, onde, 42,3% concordam totalmente, 30,8% concordam parcialmente e 26,9% são indiferentes. Porém, assim como a Diretriz 2, Giardino *et al.* (2014) consideram este um dos principais fatores de sucesso para uma *startup*. É também o caso da Diretriz 12, empoderar a equipe de desenvolvimento, porém aqui 57,7% concordam totalmente, 34,6% concordam parcialmente, 3,8% discordam parcialmente e 3,8% discordam totalmente.

Sobre a Diretriz 13, realizar ciclos de *feedback* com *stakeholders* e clientes, a maioria concorda totalmente, com 65,4%, e 34,6% concordam parcialmente. Em relação a Diretriz 14, realizar entregas frequentes/*releases* curtos, que tem forte ligação com métodos ágeis e também é recomendada por Giardino *et al.* (2014), 73,1% concordam totalmente e 26,9% concordam parcialmente. A Diretriz 15, gerenciar o conhecimento do projeto de software, apresenta concordância da maioria, onde 57,7% concordam totalmente e 26,9% concordam parcialmente, 11,5% são indiferentes e 3,8% discordam parcialmente. No que se refere a Diretriz 16, minimizar a dispersão das informações do projeto, metade dos respondentes (50%) concordam totalmente, 38,5% concordam parcialmente e 11,5% são indiferentes.

A Diretriz 17, utilizar ferramentas de coleta de comportamento dos usuários, também está relacionada às indicações de Giardino *et al.* (2014), sendo que 61,5% dos respondentes concordam totalmente, 26,9% concordam parcialmente, 7,7% são indiferentes e 3,8% discordam parcialmente. Sobre a Diretriz 18, praticar *code review*, apesar da maioria não concordar totalmente, a maioria ainda concorda com a diretriz, visto que 46,2% concordam totalmente, 34,6% concordam parcialmente, apenas 15,4% são indiferentes e 3,8% discordam parcialmente. Em relação a Diretriz 19, gerenciar a dívida técnica,

os níveis de concordância são mais divididos, onde 42,3% concordam totalmente, 19,2% concordam parcialmente, 30,8% são indiferentes e 7,7% discordam parcialmente.

No que se refere a Diretriz 20, utilizar indicadores chave de desempenho, o nível dos que concordam parcialmente (19,2%) e que são indiferentes (19,2%) são idênticos, porém 61,5% concordam totalmente. A respeito da Diretriz 21, estabelecer um processo de gerência de requisitos do produto, 53,8% concordam totalmente, 26,9% concordam parcialmente, 15,4% são indiferentes e 3,8% discordam totalmente. A Diretriz 22, estabelecer uma política de manutenção do produto, apresenta dois níveis de concordância, sendo que 46,2% concordam totalmente, 30,8% concordam parcialmente e 23,1% são indiferentes. Em relação a Diretriz 23, analisar o desempenho do software, a maioria (57,7%) concordam totalmente, 26,9% concordam parcialmente, 3,8% são indiferentes e 11,5% discordam parcialmente. A Diretriz 24, introduzir práticas de suporte à melhoria de processo, apresenta números semelhantes a diretriz anterior, onde 65,4% concordam totalmente, 23,1% concordam parcialmente, 3,8% são indiferentes e 7,7% discordam parcialmente.

6.2.6 Ameaças à Validade e Limitações

Este questionário confirmatório foi conduzido seguindo os mesmos passos descritos no estudo empírico relatado no Capítulo 4, dessa forma, as mesmas ameaças à validade descritas no primeiro são consideradas neste.

A principal limitação deste questionário está relacionada ao baixo número de *startups* que se disponibilizaram a respondê-lo. O questionário ficou disponível para ser respondido por cerca de 90 dias, sendo que neste período as *startups* que haviam respondido o questionário anterior foram convidadas a responder o novo questionário em mais de uma ocasião. Ainda assim, apenas 26 de um total de 100 organizações o fizeram. Esta limitação se deve ao fato de que, por ser um questionário confirmatório, não foi possível convidar *startups* que não tivessem participado da primeira etapa, restringindo o número de potenciais respondentes.

6.3 Grupo Focal

O grupo focal é um tipo de entrevista em profundidade realizada em grupo onde as reuniões apresentam características definidas em relação a proposta, tamanho, composição e procedimentos de condução. Entre as características gerais de um grupo focal estão o envolvimento de pessoas, a realização de reuniões, homogeneidade dos participantes de

acordo com os aspectos de interesse da pesquisa, geração de dados, natureza qualitativa e, principalmente, discussão foca em um tópico, de acordo com o propósito da pesquisa. O objeto de análise é a interação dentro do grupo e seu uso é apropriado quando o objetivo é entender como as pessoas consideram uma experiência, ideia ou evento (Oliveira e Freitas, 1998).

A técnica consiste basicamente em três etapas: planejamento, condução das entrevistas e análise dos dados. A fase de planejamento é onde são definidas as questões a serem levantadas durante as reuniões e, também, onde é realizada a seleção dos participantes. A fase de condução consiste na moderação das reuniões. Na fase de análise é realizada a transcrição e tratamento dos dados obtidos, assim como a elaboração do relatório.

O objetivo do grupo focal realizado neste trabalho é coletar a opinião de profissionais com experiência em *startups* a respeito das diretrizes desenvolvidas no Capítulo 5 de forma a homologá-las a partir da visão da indústria.

6.3.1 Planejamento

O grupo focal foi desenvolvido de acordo com o plano cronológico proposto por Krueger e Casey (1994), seguindo as etapas i) planejamento, ocorrendo da semana 1 a 2, com as atividades de desenvolver o plano, elaborar as questões, identificar os participantes, definir o local da sessão e recrutar os participantes; ii) condução, ocorrendo na semana 3, com as atividades de conduzir a sessão e receber *feedback* do planejamento; iii) análise, ocorrendo da semana 4 a 6, com as atividades de transcrever os dados, tratar os dados, analisar os dados e redigir o relatório.

6.3.1.1 Questões Levantadas

A condução da sessão foi guiada a partir de um roteiro de questões que foram levantadas para que os participantes pudessem discutir. As questões foram formuladas seguindo as categorias propostas por Krueger e Casey (1994), dessa forma, as questões foram marcadas de acordo com qual categoria pertencem, sendo elas:

- **QA - Questões abertas:** questões rápidas que buscam identificar características que os participantes possuem em comum;
- **QI - Questões introdutórias:** questões que buscam fornecer uma visão geral do tópico em discussão;
- **QT - Questões de transição:** estas questões visam mover a conversação para as questões chave da pesquisa;

- **QC - Questões chave:** questões que direcionam o estudo, sendo as que requerem maior atenção e análise;
- **QE - Questões finais ou de encerramento:** buscam fechar a discussão e considerar aquilo que foi discutido nas questões anteriores;
- **QR - Questões resumo:** é realizado um resumo das questões chave e das grandes ideias que surgiram da discussão;
- **QF - Questão final:** questão padrão do final da sessão buscando destacar uma opinião geral dos participantes a respeito do grupo focal realizado.

As questões utilizadas para a condução do grupo focal foram:

- *QA1:* Qual(is) experiência(s) com *startups* os participantes possuem?
- *QA2:* Os participantes se conhecem? Se sim, a partir de onde?
- *QI1:* São apresentadas aos participantes as diretrizes propostas. Os participantes possuem familiaridade com os temas abordados?
- *QI2:* Do que é proposto nas diretrizes, os participantes possuem alguma experiência prática? Quais?
- *QT1:* Os participantes são convidados a discutirem sobre quaisquer aspectos que fazem parte das diretrizes. Quais são os pontos debatidos?
- *QC1:* Qual a importância das diretrizes? O que é proposto é relevante?
- *QC2:* Quais das diretrizes propostas os participantes acham que podem ser inseridas em experiências atuais e quais poderiam ter auxiliado em experiências passadas?
- *QC3:* Quais das diretrizes propostas os participantes acreditam que não possuem validade prática?
- *QC4:* O que os participantes alterariam em relação as diretrizes para que elas se tornassem relevantes ou práticas?
- *QE1:* De todas as diretrizes, qual é a mais importante para os participantes?
- *QR1:* É realizado um resumo das questões debatidas e então é questionado aos participantes se os mesmos acreditam que o resumo foi adequado.
- *QF1:* Os participantes são questionados se acreditam que faltou algo a ser debatido e quais conselhos os mesmos tem em relação a pesquisa.

6.3.1.2 Seleção dos Participantes

A seleção dos participantes foi realizada por meio do método não probabilístico, incluindo amostragem por conveniência e amostragem *snowball*, visando obter uma amostra de participantes válida.

A amostragem por conveniência refere-se ao recrutamento de participantes que possuem disponibilidade e estão dispostos a participar, sendo utilizada em virtude de vários potenciais candidatos serem convidados a participar (Kitchenham e Pfleeger, 2008).

A amostragem *snowball* refere-se a solicitar aos participantes para convidar outros possíveis participantes que os mesmos conheçam que atuem no mesmo segmento e que possuam disponibilidade para participar (Kitchenham e Pfleeger, 2008).

Ao todo, 15 atores da indústria foram convidados a participar do grupo focal por meio de uma Carta de Apresentação (verificar Apêndice F). Desses, dois mencionaram não poder participar, um não respondeu ao convite e quatro não puderam participar devido a conflito de horário. Sendo assim, o número de participantes do grupo focal foi de 8 pessoas. Todos os participantes assinaram um Termo de Consentimento Livre e Esclarecido (TCLE), (disponível no Apêndice H) e preencheram um Formulário de Caracterização que continha perguntas em relação ao tempo de experiência dos participantes em desenvolvimento de software, tempo de experiência em *startups* e grau de formação (Apêndice G). Aos participantes confirmados também foi enviada uma cópia das diretrizes conforme apresentadas no Capítulo 5.

6.3.2 Condução da Sessão

A sessão foi conduzida pelo próprio autor desta dissertação na Universidade Estadual de Maringá no dia 21 de agosto de 2018, entre às 18:30h e 20:00h. Assim que todos os participantes chegaram, a sessão foi iniciada por meio de uma saudação de boas vindas e de agradecimento pela participação. Em seguida foi disponibilizado para cada participante uma cópia de um documento contendo um resumo de cada diretriz (Apêndice I) para que os mesmos pudessem consultar durante as conversas. Logo após, foi apresentada a visão geral do tópico, as regras da reunião e então se deu início as questões conforme o roteiro definido.

Para Krueger e Casey (1994) é importante que os participantes tenham liberdade para falarem, visto que não existem respostas certas ou erradas, sendo assim, os mesmos foram encorajados a trazerem para discussão qualquer ponto de vista que tivessem, mesmo que opiniões contrárias já tivessem sido expostas.

6.3.3 Análise dos Dados

6.3.3.1 Caracterização dos Participantes

Essa seção apresenta o tempo de experiência com desenvolvimento de software, o tempo de experiência com *startups* e o grau de formação do participante. Esses elementos visam caracterizar o perfil do participante do grupo focal.

A Figura - 6.4 apresenta o tempo de experiência dos participantes com desenvolvimento de software. É possível verificar que todos possuem pelo menos mais do que 1 ano de experiência com desenvolvimento de software. Metade dos participantes possuem entre 1 e 3 anos de experiência, enquanto a outra metade está dividida entre as opções restantes, sendo que então 25% do total possuem entre 3 e 5 anos de experiência e outros 25% do total possuem mais do que 5 anos de experiência.

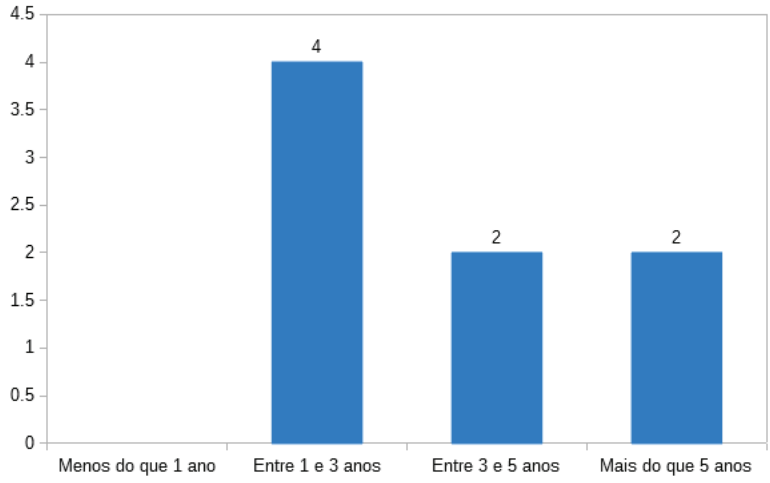

Figura 6.4: Tempo de experiência com desenvolvimento de software

Quanto ao tempo de experiência com *startups*, conforme demonstra a Figura - 6.5, a maior faixa de experiência é entre 1 e 3 anos, com 4 participantes que se encaixam na mesma. Do restante, apenas 1 possui menos do que 1 ano e apenas 1 possui mais do que 5 anos de experiência com *startups*. Os outros 2 possuem entre 3 e 5 anos de experiência.

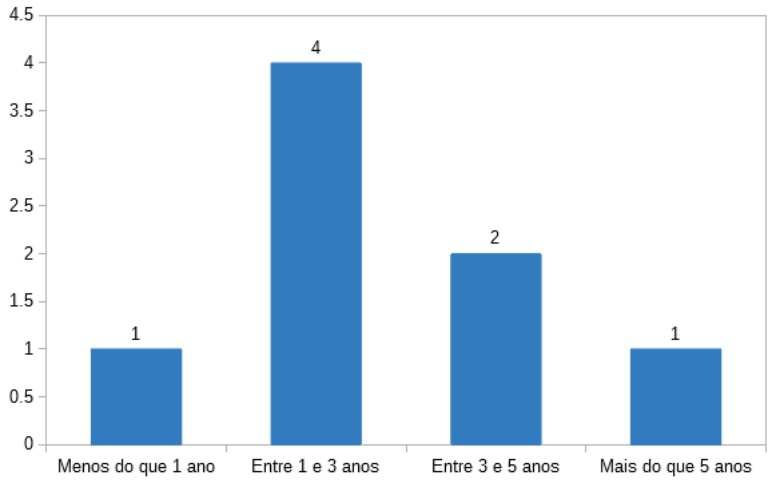

Figura 6.5: Tempo de experiência com *startups*

No que se refere ao grau de formação, Curso Técnico foi indicado por apenas 1 dos participantes, 2 indicaram Graduado, Especialista também foi indicado por apenas 1 dos participantes e 4 participantes indicaram Mestrando, conforme ilustrado pela Figura - 6.6. Nenhum dos participantes indicou ser Graduando, Mestre, Doutorando ou Doutor.

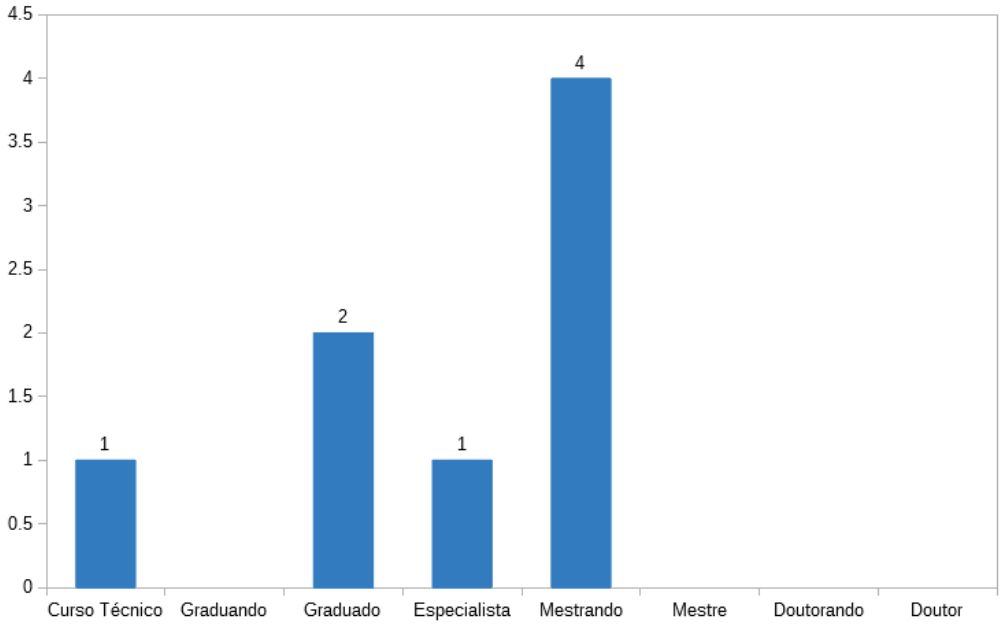

Figura 6.6: Grau de formação dos participantes

6.3.3.2 Discussão dos Dados

Essa seção apresenta uma análise do que foi discutido durante a condução da sessão do grupo focal. Como a sessão foi gravada, para interpretar e relatar o que foi discutido, o autor ouviu a sessão fazenda pausas quando era necessário fazer anotações. A análise é realizada de acordo com a ordem das questões definidas na seção 6.3.1.1. Para preservar a identidade dos participantes os mesmos serão identificados por meio das abreviações P1, P2, P3, P4, P5, P6, P7 e P8. A Figura - 6.7 apresenta um resumo da discussão dos dados por meio de um mapeamento dos tópicos debatidos pelos participantes. Este mapeamento utiliza o conceito de mapas mentais, feito por meio da ferramenta MindNode, que permite demonstrar o relacionamento entre vários conceitos de forma simples, sendo adequada para dados qualitativos como estes (Eppler, 2006; Wheeldon e Faubert, 2009).

A Figura - 6.7 permite verificar que algumas diretrizes não foram avaliadas ou discutidas durante a condução da sessão do grupo focal, entretanto, isso não implica que estas diretrizes não são importantes. Devido a limitação de tempo para a realização da sessão, é natural que algumas diretrizes não fossem comentadas, sendo que aquelas que chamaram mais a atenção em um primeiro momento acabaram sendo discutidas, inclusive algumas geraram longas discussões e outras apenas comentários breves.

Em relação a QA1, quais experiências com *startups* os participantes possuem, P1 relata que é sócio de uma *startup* há cerca de 4 anos, o participante saiu de um emprego de desenvolvedor em uma organização para se dedicar a criação do produto da *startup*. O mesmo relata que sua principal dificuldade no início foi a falta de conhecimento, isso acabou ocasionando várias mudanças de tecnologias e de *frameworks*, buscando melhorar a qualidade do produto. P1 acrescenta que possui ajuda de terceiros, porém, é o principal desenvolvedor. Além disso, o mesmo é responsável pelas atividades de levantamento de requisitos, testes, implantação e suporte. O produto possui foco em dispositivos móveis, sendo desenvolvido para a plataforma Android e por isso grande parte da codificação é realizada na linguagem de programação Java e Android.

P2 relata que trabalhou durante cerca de 1 ano em uma *startup* e durante o período participou do processo de decisão das tecnologias e de práticas que seriam utilizadas, como testes e construção do MVP. Acrescenta que, atualmente, possui em andamento a concepção de um novo produto com mais alguns amigos. P3 comenta que trabalha junto a P2 nesse produto, os mesmos fizeram as escolhas de tecnologias e processos, sendo que o MVP deve ser lançado em breve. P4 relata que começou a trabalhar em uma *startup* que estava em andamento a cerca de 1 ano e meio e portanto não participou das decisões referentes a tecnologias e processos. P5 comenta que seu caso é parecido com o

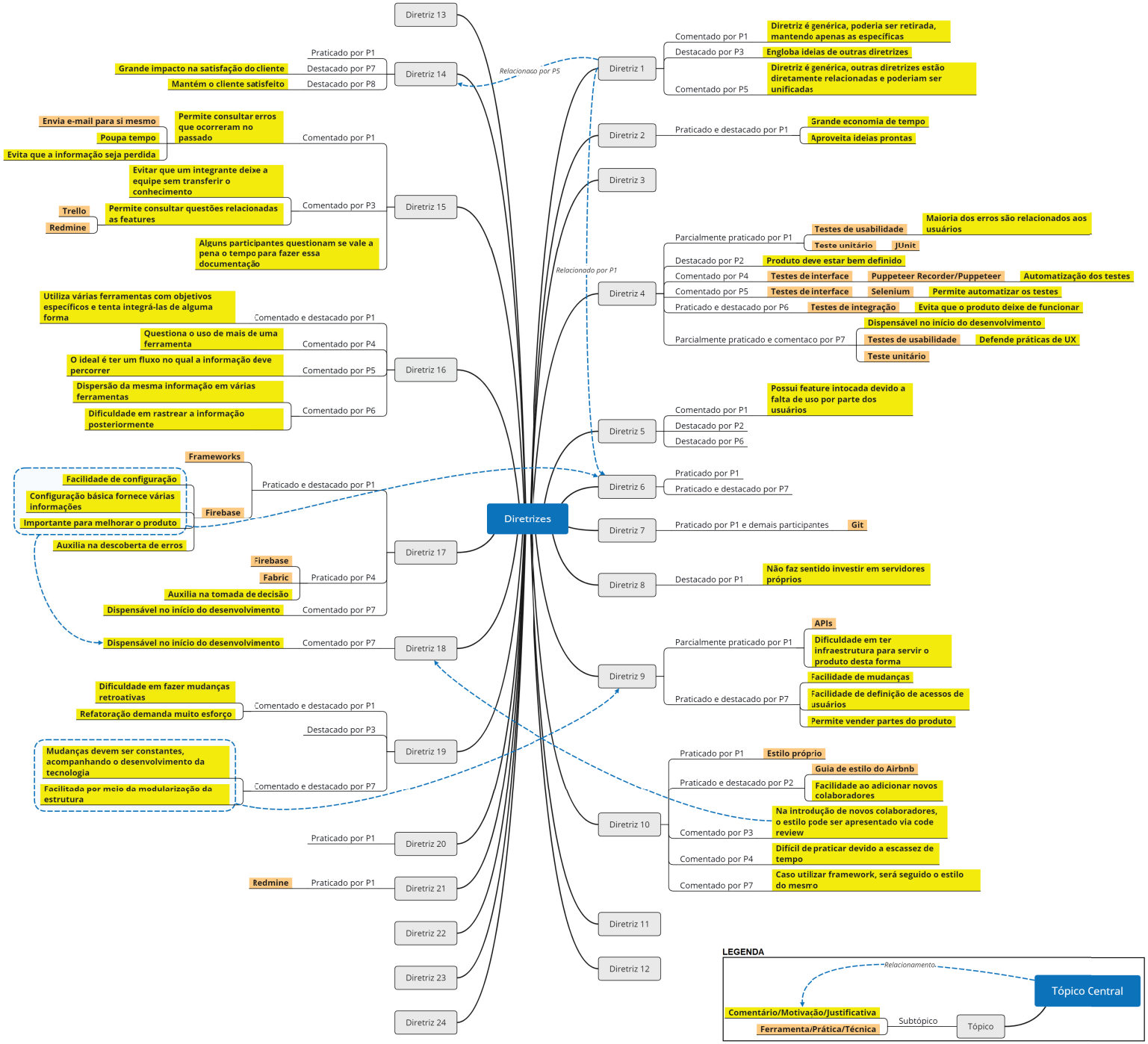

Figura 6.7: Mapeamento dos tópicos debatidos

de P4, porém saiu recentemente da *startup*. O participante relata que quando começou na *startup* já existia uma ideia de produto mas acabou participando de boa parte das decisões relacionadas ao desenvolvimento.

P6 também trabalhou durante um período em uma *startup* e comenta que tinha liberdade para tomar decisões pois era um dos principais desenvolvedores, porém não participou da concepção da ideia visto que a mesma já havia sido elaborada. P7 responde que é sócio-fundador de uma *startup* há cerca de 4 anos, acrescenta que a mesma opera por meio do modelo de software como serviço, conhecido como SaaS, do inglês *Software as a Service*. Atualmente participa mais das decisões relacionadas a negócio do que ao desenvolvimento de software. P8 relata que é sócio-fundador de uma *startup* recém iniciada e que tenta seguir boa parte dos padrões de mercado, principalmente relacionados a validação.

Quanto a QA2, os participantes são questionados se conhecem e, se sim, a partir de onde. Todos os participantes se conhecem, alguns não de forma próxima, porém em vários casos chegaram a trabalhar ou estudar juntos em algum momento. Em seguida, referente a QI1, são apresentadas aos participantes as diretrizes propostas e os mesmos são questionados a respeito de sua familiaridade com os temas abordados. Essa questão ocasionou uma longa discussão entre os participantes e o que foi discutido acabou por responder o que se esperava encontrar por meio das questões QI2 e QT1.

P1 inicia o debate destacando que a metodologia ágil está muito ligada a construção do MVP e relata que quando precisa desenvolver uma nova *feature*, produz uma versão mínima da mesma e busca lançá-la o mais rápido possível utilizando algumas formas de medição. Essa medição são indicadores para entender se os usuários estão utilizando a *feature* ou não, visando validar a ideia. O participante acrescenta que leva cerca de um mês pra entregar a *feature* e portanto tenta trabalhar com essa periodicidade. P1 continua mencionando que trabalhou com Scrum no emprego anterior, porém como passou a trabalhar praticamente sozinho deixou de utilizar o modelo fechado. Atualmente utiliza o Redmine pra registrar as tarefas, utilizando descrições curtas para o que deve ser feito. Utiliza Git para fazer o controle de versão do produto e descreve que sua rotina de lançamento envolve subir as alterações para um servidor de testes, que o mesmo descreve como *beta*, e testar as alterações por meio da utilização do próprio aplicativo. P5 comenta que algumas diretrizes parecem se repetir, citando a Diretriz 1 e 14, pois acredita que a prática de realizar entregas frequentes está diretamente ligada a utilização de metodologias ágeis. O mesmo questiona a P1 se as *features* são liberadas para todos os usuários, sendo que P1 confirma que sim, pois a quantidade de usuários não é grande. P1 acrescenta que seria possível liberar de acordo com certos usuários pois utiliza uma estrutura modular

por meio de APIs. P5 justifica a pergunta mencionando que queria saber se P1 utilizava algum tipo de usuário de testes. P1 responde que não, o próprio é quem realiza todos os testes. O mesmo acrescenta que sobe as alterações para o servidor de testes e faz os testes necessários, além de verificações da própria plataforma, e em seguida sobe as alterações para o servidor de produção.

O moderador questiona P1 sobre a sua opinião em relação a ter utilizado SVN no passado e utilizar Git atualmente. P1 menciona que facilitou o seu trabalho, pois a criação e transição entre *branches* no Git é mais fácil e que o processo de voltar versões no SVN é difícil, pois é necessário fazer adequações de ambiente, algo que não é necessário no Git. P5 comenta que o SVN, por ser centralizado, faz com que muitos conflitos ocorram no momento do *merge*, principalmente quando se tem muitos desenvolvedores envolvidos no projeto, enquanto no Git os conflitos podem ser solucionados localmente antes de serem subidos para o servidor principal.

P7 comenta sobre a Diretriz 9, salientando que modularizar a estrutura do produto é uma tendência. Segundo o mesmo a *startup* deve pensar em uma estrutura de microsserviços e destaca que entre os benefícios estão, facilidade de mudanças, facilidade de definição de acessos de usuários, permite vender partes do produto. O participante também comenta que testes no início não são importantes e justifica que a *startup* deve focar na entrega do MVP, pois dificilmente a *startup* irá escalar em uma velocidade onde os testes passem a ser necessários. Outros participantes concordaram com o argumento. P7 conclui que mudar rápido é essencial, por isso testes não são prioridade. P2 acrescenta que os testes são fundamentais, porém, difíceis de implementar para *startups* por causa das mudanças que irão ocorrer, incluindo *features* que não serão mais usadas e demanda de novas *features*, consequentemente gerando um comprometimento de tempo em testes que não serão úteis. O participante conclui que os testes são bons depois que o produto estiver melhor definido. P1 comenta que no seu caso a cobertura de testes é pequena, mas atualmente utiliza os testes como forma de implementação de código por meio da ferramenta JUnit. Acrescenta que testes de interface são mais importantes do que testes unitários, justificando que boa parte dos erros estão relacionados ao usuário. P2 relata uma experiência que teve na *startup* em que trabalhou onde, após vários meses de desenvolvimento utilizando testes tudo foi descartado. Isso acabou atrasando o desenvolvimento do MVP, que demorou cerca de cinco meses para ficar pronto.

Ainda em relação a testes, P6 comenta que considera os testes de integração importantes, pois passou por experiências onde funcionalidades do produto deixaram de funcionar corretamente devido a falta de execução desses testes. P7 adiciona que UX é muito importante, pois usabilidade está entre as prioridades dos usuários e por isso

defende a execução de testes de usabilidade. O mesmo comenta que atualmente surgiu a necessidade de implementar testes de código, porém testes relacionados a UX sempre foram realizados. P5 relata que para testes de interface a ferramenta Selenium pode ser uma opção interessante, visto que a mesma permite automatizar os testes. P4 também sugere a ferramenta Puppeteer Recorder, que permite gravar as interações feitas no navegador e gerar um *script* para ser executado na ferramenta Puppeteer, automatizando o processo de teste.

P2 comenta que acha interessante utilizar estilos de codificação e relata que utiliza o guia de estilo do Airbnb. O mesmo justifica que facilita principalmente quando é preciso adicionar novos colaboradores ao projeto. P4 menciona que tentou utilizar e teve sucesso em projetos pessoais porém achou difícil utilizar na rotina de trabalho devido as demandas constantes e a falta de tempo para ajustar o código de acordo com o estilo. P1 adiciona que usa um estilo próprio que trouxe de sua experiência no emprego anterior. P7 comenta que no caso de utilizar algum *framework*, provavelmente será seguido o estilo do próprio framework, visto que haverá pouca margem para mudanças. P3 acrescenta que no caso da inclusão de novos desenvolvedores, o estilo pode ser apresentado por meio de sessões de *code review*. P2 comenta a respeito da importância da Diretriz 5, validar a ideia antes de começar a desenvolver. Sobre isso, P1 relata que possui uma *feature* que está intocada a algum tempo, pois não há clientes suficientes para justificar a expansão da mesma.

Em seguida, P8 trouxe um tópico que não faz parte das diretrizes, porém o moderador considerou que o mesmo poderia ser interessante e não interferiu. O participante comenta que trazer desenvolvedores qualificados é algo importante, pois caso contrário perde-se muito tempo com o treinamento do mesmo. P1 concorda e comenta que já passou por essa experiência, onde precisou acompanhar de perto um novo colaborador devido a falta de qualificação do mesmo. Acrescenta que o ideal é trazer colaboradores que saibam mais de desenvolvimento do que os atuais desenvolvedores da *startup*. P7 discorda parcialmente, pois dificilmente a *startup* terá os recursos para trazer um desenvolvedor qualificado. Acrescenta que para partes do produto que já estão bem resolvidas, um desenvolvedor, mesmo que inexperiente pode economizar tempo, deixando partes que precisam evoluir com desenvolvedores mais experientes.

Alguns participantes questionaram o que significa a Diretriz 15, gerenciar o conhecimento do projeto de software. Antes que o moderador pudesse intervir, P3 começou a explicar que, em suas palavras, "é não deixar alguém sair do projeto com um conhecimento que só aquela pessoa possui". O moderador acrescenta que é basicamente documentar as informações. P1 apresenta um exemplo de um erro que ocorre em uma ferramenta e salienta que se esse erro for documentado, o mesmo poderá ser consultado na próxima vez

que ocorrer, poupando tempo e também evitando que a informação seja perdida caso a pessoa que solucionou o erro na primeira vez tenha deixado a equipe. P3 da exemplo de ferramentas como Redmine e Trello, onde podem ser documentados questões relacionadas a *features*, sobre o que deu certo, o que deu errado e etc. Alguns participantes questionam se vale a pena o tempo para fazer essa documentação. P6 responde que é um *trade-off*, pois no futuro ter essa informação disponível pode economizar tempo e relata uma experiência própria onde tomou certas decisões relacionadas ao desenvolvimento, porém, após um tempo, quando precisou retomar os motivos dessas decisões, nem mesmo o próprio se lembrava o que levou a tomá-las. P7 e P4 comentam que essa situação nunca ocorreu com os mesmos. P5 relata uma experiência, de quando saiu da *startup* em que atuava, foi preciso treinar um novo desenvolvedor para que o mesmo tivesse conhecimento das decisões que o mesmo tomou e pudesse entender o que foi desenvolvido. Para P7 isso é algo que faz sentido para organizações maduras, sendo que em *startups* o próprio código deve ser capaz de fornecer essas informações. P3 responde que a questão não está apenas relacionada a código, mas principalmente as decisões que foram tomadas e, no caso seja feita a documentação, seria possível entender os *insights* das pessoas da época, mesmo que essas pessoas já tenham deixado a *startup*. P7 comenta que a maioria dos seus problemas são resolvidos por meio de uma busca no Stack Overflow. P1 adiciona que costuma mandar um e-mail para si mesmo quando resolve algo, para que possa servir como fonte de consulta posteriormente.

P4 questiona a respeito da Diretriz 16, minimizar a dispersão das informações do projeto, se seria o caso de usar só o Trello, ao contrário de usar, por exemplo, Trello e Redmine, e questiona se alguém usa mais de uma. P1 comenta que usa várias ferramentas, com seus objetivos específicos, e que tenta integrar elas de alguma forma. Para o mesmo, mais importante do que centralizar, é fazer uma conexão entre as informações nas ferramentas. P6 comenta que um problema recorrente é ter a mesma informação em várias ferramentas diferentes, dificultando rastrear essa informação posteriormente. Para P5 o ideal é ter um fluxo, ou seja, ao usar ferramentas diferentes seria interessante ter um caminho definido no qual a informação deve percorrer.

Em relação as questões QC1 e QC2, os participantes são questionados sobre o que acham da importância das diretrizes, se os mesmos consideram que o que é proposto é relevante e também quais das diretrizes propostas os participantes acham que podem ser inseridas em experiências atuais e quais poderiam ter auxiliado em experiências passadas. P1 comenta que acha a Diretriz 17, utilizar ferramentas de coleta de comportamento dos usuários, importante. O participante relata tenta utilizar *frameworks* para isso além de utilizar o Firebase, este inclusive auxilia o mesmo a encontrar erros ao analisar o

comportamento dos usuários. P4 comenta que isso é algo que estão tentando utilizar com maior frequência na *startup* em que trabalha, também menciona que utiliza o Firebase para isso e o Fabric, servindo como base para a tomada de decisão. Para P7 isso é interessante quando a *startup* está mais madura, na fase de início da *startup* o mesmo argumenta que investiria em *deploy* rápido, para entregar o produto com frequência, sem se importar com *code review* ou coleta de dados. P1 rebate destacando que atualmente esse processo é fácil e exemplifica que no Firebase é apenas uma questão de configuração, sendo que não é necessário gastar muito tempo para isso, pois mesmo uma configuração básica já fornece várias informações. Acrescenta que sem essa coleta, mesmo no MVP, não há como entender o que o usuário está buscando e consequentemente melhorar o produto.

P7 ressalta novamente a importância de modularizar a estrutura do produto, argumenta que isso é algo que deve ser pensado desde o início, pois será difícil alterar a estrutura depois que o produto escalar. Acrescenta que não daria muita prioridade para o restante das diretrizes, pois o mesmo acredita que podem ser pensadas conforme a *startup* amadurece. P8 destaca a Diretriz 14, realizar entregas frequentes, pois de acordo com o mesmo é isto que mantém o cliente satisfeito. P6 comenta que vê valor em todas as diretrizes, algumas podem ter mais prioridade que outras porém acredita que todas são importantes. P7, P3 e P1 comentam a respeito de gerenciar a divida técnica, destacando que é algo que deve ser pensado diariamente visto que é inevitável entregar um código de menor qualidade para garantir que o produto esteja disponível. P1 acrescenta que acha muito difícil fazer mudanças retroativas, pois demanda muito esforço para refatorar, e relata que seu produto contém vários trechos que precisam passar por essa refatoração, porém as faz assim que surge a demanda. P7 comenta que essas mudanças devem ser constantes, acompanhando o desenvolvimento da tecnologia e por isso defende a modularização da estrutura, pois facilita a aplicação dessas mudanças. P1 menciona que a dificuldade da modularização está na infraestrutura necessária para servir o produto dessa forma, porém relata que tenta codificar o produto de forma modular.

Quanto as questões QC3 e QC4, os participantes são convidados a elaborar a respeito das diretrizes propostas que os mesmos acreditam que não possuem validade prática e o que os mesmos alterariam em relação as diretrizes para que elas se tornassem relevantes ou práticas. Todos os participantes concordam que as diretrizes possuem validade prática. Sobre o que poderia ser alterado, P1 sugere que seria interessante definir as diretrizes de acordo com o nível de maturidade da *startup*, visto que algumas se aplicam mais a *startups* que estão iniciando e outras a *startups* que já estão em fase de crescimento. P3 adiciona que talvez seria interessante não restringir ao nível de maturidade, mas sim indicar para qual nível aquela diretriz é mais recomendada. P5 comenta que algumas diretriz são

muito próximas, como por exemplo a Diretriz 1 que é genérica, e que outras diretrizes estão diretamente ligadas a essa diretriz, e questiona se isso não poderia ser apenas uma diretriz. P1 adiciona que essa diretriz genérica poderia ser retirada, deixando apenas as diretrizes específicas. P7 relata que gostaria de consumir algo relacionado ao estado da prática em *startups* que são referências de mercado, porém não sabe se isso faz parte do escopo da pesquisa.

Em relação a QE1, os participantes são questionados sobre qual é a diretriz que consideram a mais importante. P3 destaca a Diretriz 1, já que ela engloba várias ideias das outras diretrizes. P1 destaca a Diretriz 2, em relação a utilizar *frameworks*, justificando que a economia de tempo trazida é enorme, pois, em suas palavras, "não é preciso reinventar a roda". P6 vê a Diretriz 5 como a mais importante. P7 destaca a Diretriz 14, realizar entregas frequentes, pois para o mesmo é a que mais impacta na satisfação do cliente. P1 destaca a terceirização da infraestrutura (Diretriz 8), pois acredita que não faz sentido para uma *startup* investir em servidores próprios.

Em relação as questões QR1 e QF1, visto que a condução da sessão já estava chegando ao limite de tempo, o resumo das questões debatidas não foi realizado, porém os participantes foram questionados se acreditavam que faltou algo a ser debatido, sendo que os mesmos declararam que não. Em relação a algum conselho que os mesmos poderiam ter em relação a pesquisa, os mesmos destacaram que acreditavam que já haviam exposto os conselhos e sugestões durante a discussão.

6.3.4 Ameaças à Validade

De acordo com Fern e Fern (2001), existem quatro processos grupais que representam ameças à validade na condução de grupo focais, sendo: bloqueio de produção, influência social, "pegar carona" e influência normativa.

O bloqueio de produção está relacionado à ativação simultânea de dois processos cognitivos distintos, o pensar e o ouvir. Essa condição da interação grupal torna difícil para que o participante consiga acompanhar a discussão e ao mesmo tempo consiga pensar no que vai dizer. Contornar esse problema é difícil do ponto de vista do moderador, pois depende intrinsecamente do participante, uma vez que nem todos estão preparados para lidar com a atenção difusa.

A influência social é subdividida em três processos: apreensão da avaliação, auto-consciência e influência normativa. A primeira está relacionada ao medo da desaprovação social, o que poderia comprometer a sinceridade das opiniões no grupo. Acredita-se que essa dificuldade foi contornada pois, no início da sessão o moderador deixou claro

a importância das manifestações individuais e ressaltou que não existiam respostas certas, sendo toda e qualquer opinião bem-vinda, inclusive as negativas. Em relação a autoconsciência, em contextos de discussões em grupo, os participantes comparam as suas opiniões e valores, e ao constatarem inconsistências podem assumir posições mais extremas, positivas ou negativas. Em virtude dos participantes constituírem um grupo homogêneo, acredita-se que este problema foi minimizado, visto que quando possuem experiências comuns, o ambiente do grupo torna-se mais propício à avaliação crítica dos posicionamentos internos. Por fim, a influência normativa está ligada à comparação feita sobre as normas ou padrões sociais, podendo também levar a adoção de posições mais extremas por parte do participante, com o objetivo de obter uma melhor avaliação do grupo. Por parte do moderador, posições divergentes foram evitadas quando acreditado que poderiam ser consideradas como um desvio por parte dos participantes. Entretanto, não é possível controlar que os participantes assumam posições controvertidas e, consequentemente, estimulem outros participantes a fazer o mesmo.

O "pegar carona" diz respeito àqueles participantes que se beneficiam do grupo, mas dão pouco em retorno. Este processo está diretamente ligado a apreensão da avaliação e, novamente, para lidar com o problema o moderador ressaltou no início da sessão a importância das manifestações pessoais, fossem elas opiniões negativas ou positivas.

Por fim, a influência normativa, pode afetar o grupo de duas formas: força do argumento e extensão do compartilhamento da informação no grupo. Estes processos estão relacionados à habilidade individual para persuadir ou influenciar na decisão do outro, sendo que a mudança de opinião ocorre com mais facilidade quando o indivíduo não possui um posicionamento bem definido sobre o assunto. Entretanto, essa divergência pode ser benéfica para o grupo, uma vez que o surgimento de uma opinião discordante pode provocar um redirecionamento dos posicionamentos até então compartilhados. Para Fern e Fern (2001), os argumentos persuasivos levam os participantes a integrarem novos elementos em suas avaliações, ampliando seu entendimento sobre o tema.

6.4 Painel com Especialistas

O painel com especialistas é um método de pesquisa onde é utilizado um grupo de indivíduos que são considerados experientes em uma determinada área, os quais irão contribuir com opiniões fundamentadas sobre um tópico específico (Okoli e Pawlowski, 2004). A partir da condução de um painel com especialistas busca-se obter as reflexões, ideias, suposições, especulações e estimativas de especialistas da área de pesquisa, visto que a absorção desse conhecimento pode ser de extrema importância para os objetivos

da pesquisa (Cooke *et al.*, 1991). O objetivo do painel com especialistas neste trabalho é avaliar as incertezas em relação as diretrizes propostas por meio de especialistas da área de pesquisa.

A condução do painel com especialistas envolveu as seguintes etapas, baseando-se nas fases estabelecidas por Okoli e Pawlowski (2004): i) fase 1, seleção dos especialistas, envolvendo as atividades de identificar habilidades relevantes, descrever o nome dos indivíduos com as habilidades relevantes e contatar os especialistas listados para convidá-los; ii) fase 2, *brainstorming*, envolvendo as atividades de enviar o primeiro questionário, solicitando aos especialistas listar os fatores relevantes para corrigir estrutura ou oportunidades, consolidar todas as listas enviadas pelos especialistas e unificar terminologias; iii) fase 3, alinhamento, envolvendo as atividades de apresentar o segundo questionário com as primeiras correções, especialistas classificam os fatores que cada um considera relevante, avaliar o consenso para cada sugestão apresentada dentro de cada pergunta do questionário e redigir o relatório final com as contribuições aderidas.

6.4.1 Seleção dos Participantes

Buscando atender aos critérios da primeira fase, foram buscados pesquisadores com conhecimentos e atividades relacionadas a área de *startups* de software. Essa seleção foi realizada por meio do método não probabilístico, incluindo amostragem por conveniência, visando obter uma amostra de participantes válida.

A amostragem por conveniência refere-se ao recrutamento de participantes que possuem disponibilidade e estão dispostos a participar, sendo utilizada em virtude de vários potenciais candidatos serem convidados a participar (Kitchenham e Pfleeger, 2008).

Ao todo, 28 pesquisadores foram convidados a participar do painel com especialistas por meio de uma Carta de Apresentação (J).

6.4.2 Brainstorming

Na segunda fase, os questionários foram enviados para uma pré-análise por parte de dois especialistas da área acadêmica, visando avaliar as terminologias e o entendimento das questões. Os especialistas sugeriram alguns ajustes nas questões e também de algumas terminologias, sendo todos implementados conforme apresentado no Apêndice L.

6.4.3 Alinhamento

Para a terceira fase, o questionário foi então enviado aos 28 especialistas para que pudesse ser respondido. O mesmo ficou disponível para ser respondido durante um período de 60 dias, sendo constantemente monitorado.

6.4.4 Resultados

Após o período estabelecido, dos 28 especialistas convidados, apenas 6 responderam o mesmo, sendo assim, a taxa de resposta foi de 21,43%.

6.4.4.1 Caracterização do Respondente

Essa seção apresenta o tempo de experiência com pesquisa em *startups*, qual área específica dentro do contexto de *startups* acontece a pesquisa dos especialistas e o seu grau de formação. Estes elementos visam caracterizar o perfil do especialista respondente do questionário.

A Figura - 6.8 apresenta o tempo de experiência dos especialistas com desenvolvimento de pesquisas no campo de *startups*. É possível verificar que todos possuem pelo menos mais do que 1 ano de experiência na área, sendo que a maioria (66,7%) possui entre 3 e 5 anos de experiência. Do restante, 16,7% possuem entre 1 e 3 anos e 16,7% possuem mais do que 5 anos.

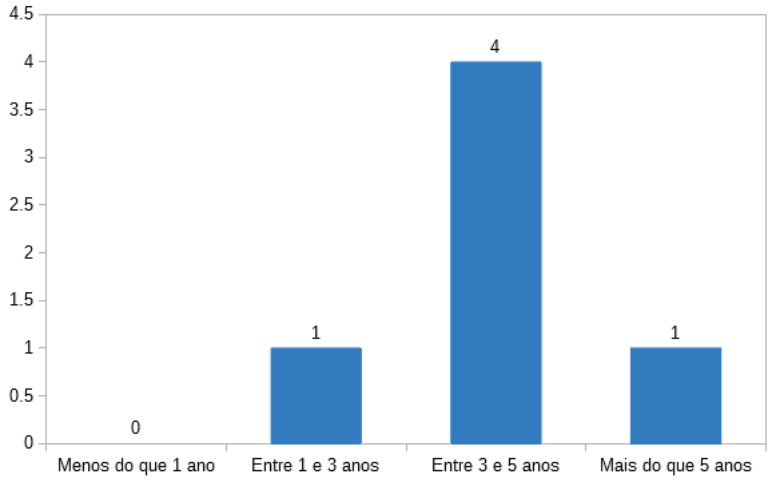

Figura 6.8: Tempo de experiência com pesquisa em *startups*

A Tabela - 6.2 traz os dados quanto as áreas específicas em que os especialistas atuam dentro do contexto de *startups*.

Tabela 6.2: Área de atuação dos especialistas

Especialista	Área(s) de atuação
Especialista 1	Métodos ágeis.
Especialista 2	Inovação, empreendedorismo, inteligência artificial, governo, judiciário e outras.
Especialista 3	Ecossistemas de startups de software.
Especialista 4	Negócios.
Especialista 5	Engenharia de requisitos.
Especialista 6	Engenharia de Software.

Em relação ao grau de formação, Doutor foi indicado por 1 dos participantes, 2 indicaram Mestre e 3 indicaram Doutorando, conforme ilustrado pela Figura - 6.9. Isto demonstra um alto nível de formação entre os especialistas participantes.

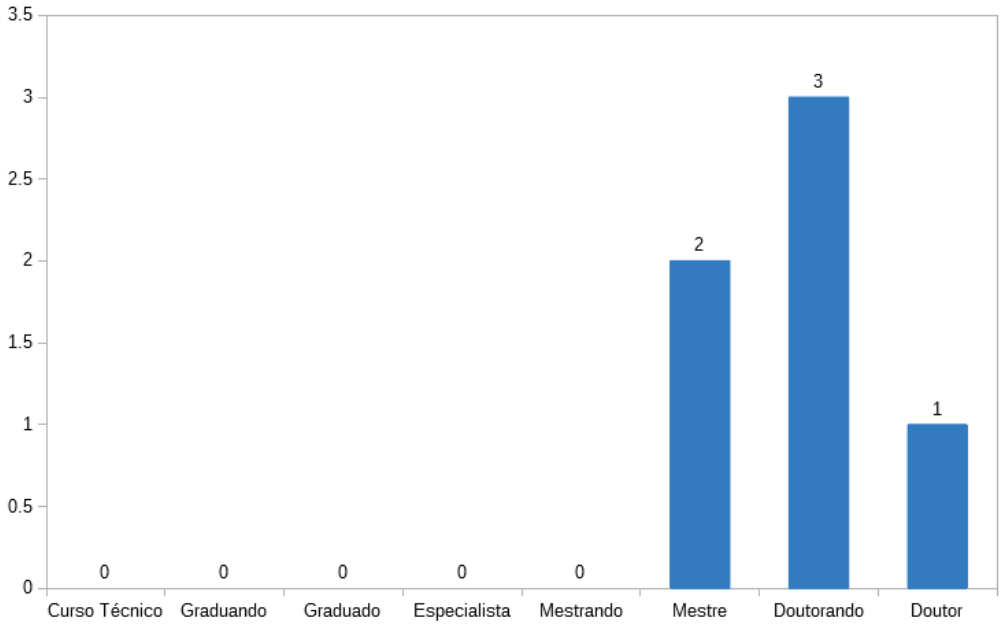

Figura 6.9: Grau de formação dos participantes

6.4.4.2 Diretrizes para o Desenvolvimento de Software em Startups

Essa seção apresenta as 24 diretrizes que foram propostas no Capítulo 5 e demonstra o grau de concordância dos especialistas em relação as mesmas. Para isso, foi utilizado uma escala Likert com as opções: i) discordo totalmente; ii) discordo parcialmente; iii) indiferente; iv) concordo parcialmente e v) concordo totalmente. Em seguida, as diretrizes são apresentadas sequencialmente em conjunto com os percentuais de concordância registrados

pelos especialistas do questionário. A Tabela - 6.3 demonstra o nível de concordância de forma resumida, onde cada linha representa uma diretriz e as colunas o percentual de concordância.

Tabela 6.3: Nível de concordância das diretrizes

	Discordo totalmente	**Discordo parcialmente**	**Indiferente**	**Concordo parcialmente**	**Concordo totalmente**
D1	0%	0%	0%	16,7%	83,3%
D2	0%	0%	0%	50%	50%
D3	0%	0%	16,7%	33,3%	50%
D4	0%	33,3%	0%	50%	16,7%
D5	0%	0%	0%	16,7%	83,3%
D6	0%	0%	0%	0%	100%
D7	0%	0%	33,3%	16,7%	50%
D8	16,7%	0%	0%	50%	33,3%
D9	0%	0%	16,7%	33,3%	50%
D10	0%	0%	16,7%	0%	83,3%
D11	0%	0%	16,7%	0%	83,3%
D12	0%	0%	0%	33,3%	66,7%
D13	0%	0%	0%	0%	100%
D14	0%	0%	0%	16,7%	83,3%
D15	0%	0%	0%	33,3%	66,7%
D16	0%	0%	0%	66,7%	33,3%
D17	0%	0%	0%	33,3%	66,7%
D18	16,7%	0%	16,7%	33,3%	33,3%
D19	0%	16,7%	33,3%	16,7%	33,3%
D20	0%	0%	33,3%	0%	66,7%
D21	0%	0%	0%	33,3%	66,7%
D22	0%	16,7%	16,7%	33,3%	33,3%
D23	0%	0%	33,3%	33,3%	33,3%
D24	0%	16,7%	16,7%	0%	66,7%

Em relação a Diretriz 1, utilizar uma metodologia ágil para guiar o desenvolvimento de software, 83,3% concordam totalmente e 16,7% concordam parcialmente. Em relação a Diretriz 2, utilizar *frameworks open source* e/ou com grande adoção no mercado, o número de concordância total é de 50% e 50% concordam parcialmente. Quanto a Diretriz 3, usar *backlog* em conjunto com Kanban independente do método ágil utilizado, 50% concordam totalmente, 33,3% concordam parcialmente e 16,7% são indiferentes, sendo essa a primeira diretriz a demonstrar um grau de indiferença. Sobre a Diretriz 4, estabelecer uma rotina de testes, 16,7% concordam totalmente, 50% concordam parcialmente e 33,3% são indiferentes, sendo esta a primeira diretriz a demonstra um nível de discordância.

A respeito da Diretriz 5, validar a ideia antes de começar a desenvolver, 83,3% concordam totalmente e 16,7% concordam parcialmente. No que se refere a Diretriz 6, construir um MVP, a mesma é considerada de grande importância, visto que 100% concordam totalmente. Sobre a Diretriz 7, utilizar uma ferramenta de gerenciamento de versão descentralizada, 50% concordam totalmente, 16,7% concordam parcialmente e 33,3% são indiferentes. Em relação a Diretriz 8, priorizar a terceirização da infraestrutura, 33,3% concordam totalmente, 50% concordam parcialmente e 16,7% discordam totalmente, sendo a primeira diretriz a apresentar um nível de discordância total.

No que se refere a Diretriz 9, modularizar a estrutura do produto, 50% concordam totalmente, 33,3% concordam parcialmente e 16,7% são indiferentes. Em relação a Diretriz 10, utilizar estilos de codificação, 83,3% concordam totalmente e 16,7% são indiferentes. Os mesmos números aparecem na Diretriz 11, usar prototipagem evolucionária durante o desenvolvimento do produto, onde, 83,3% concordam totalmente e 16,7% são indiferentes. No que se refere a Diretriz 12, empoderar a equipe de desenvolvimento, 66,7% concordam totalmente e 33,3% concordam parcialmente.

Sobre a Diretriz 13, realizar ciclos de *feedback* com *stakeholders* e clientes, todos concordam totalmente, com 100%, demonstrando ser uma diretriz de grande relevância. Em relação a Diretriz 14, realizar entregas frequentes/*releases* curtos, 83,3% concordam totalmente e 16,7% concordam parcialmente. A Diretriz 15, gerenciar o conhecimento do projeto de software, apresenta total concordância, onde 66,7% concordam totalmente e 33,3% concordam parcialmente. No que se refere a Diretriz 16, minimizar a dispersão das informações do projeto, 33,3% concordam totalmente e 66,7% concordam parcialmente.

A Diretriz 17, utilizar ferramentas de coleta de comportamento dos usuários, também apresenta total concordância, sendo que 66,7% dos respondentes concordam totalmente e 33,3% concordam parcialmente. Sobre a Diretriz 18, praticar *code review*, apesar da maioria não concordar totalmente, a maioria ainda concorda com a diretriz, visto que 33,3% concordam totalmente e 33,3% concordam parcialmente, apenas 16,7% são indiferentes e 16,7% discordam totalmente. Em relação a Diretriz 19, gerenciar a dívida técnica, os níveis de concordância também são divididos, onde 33,3% concordam totalmente, 16,7% concordam parcialmente, 33,3% são indiferentes e 16,7% discordam parcialmente.

Em relação a Diretriz 20, utilizar indicadores chave de desempenho, 66,7% concordam totalmente e 33,3% são indiferentes. A respeito da Diretriz 21, estabelecer um processo de gerência de requisitos do produto, 66,7% concordam totalmente e 33,3% concordam parcialmente. A Diretriz 22, estabelecer uma política de manutenção do produto, apresenta dois níveis de concordância, sendo que 33,3% concordam totalmente e 33,3% concordam

parcialmente, 16,7% são indiferentes e 16,7% discordam parcialmente. Em relação a Diretriz 23, analisar o desempenho do software, os números são bem divididos, onde 33,3% concordam totalmente, 33,3% concordam parcialmente e 33,3% são indiferentes. A Diretriz 24, introduzir práticas de suporte à melhoria de processo, apresenta concordância da maioria, onde 66,7% concordam totalmente, 16,7% são indiferentes e 16,7% discordam parcialmente.

6.4.5 Ameaças à Validade e Limitações

A principal ameaça à validade está relacionada a um possível viés de confirmação por parte dos especialistas. Isto é, de acordo com o foco de pesquisa que possuem, seria possível que apresentassem uma visão enviesada ao responderem o questionário, indicando maior nível de concordância para aquelas diretrizes que estão fortemente relacionadas a sua área de pesquisa e discordância para aquelas que não estão. Entretanto, acredita-se que essa ameaça foi mitigada uma vez que, considerando a amostragem baixa, as áreas de pesquisa são bastante heterogêneas.

A principal limitação deste painel com especialistas está relacionada ao baixo número de pessoas que se disponibilizaram a respondê-lo. Isto está ligado à dificuldade de entrar em contato com os pesquisadores da área. Geralmente a única fonte de contato possível de ser encontrada é o e-mail informado em suas publicações, entretanto, não houve retorno de muitos dos pesquisadores pois acredita-se que, como estes endereços são institucionais, boa parte deles estão desatualizados. Além disto, em virtude da distância geográfica, ou seja, falta de amigos em comum, é difícil entrar em contato por meio de redes sociais, visto que estas redes, como por exemplo o Facebook, possuem mecanismos que tratam mensagens de desconhecidos como *spam*.

6.5 Considerações Finais

O *survey* confirmatório tem como contribuição, as opiniões de atores da indústria a respeito das diretrizes propostas. O questionário permitiu verificar qual o nível de concordância dos respondentes em relação as diretrizes, sendo um indicator importante de qual a relevância das mesmas para essas pessoas. Uma vez que a maioria concordou com as diretrizes, é possível afirmar que as ideias expressas nas mesmas estão na direção correta. O questionário também permitiu checar novamente o perfil dos atores ao levantar o tempo de experiência que possuem com desenvolvimento de software, o tempo de experiência que possuem com *startups* e qual o nível de escolaridade que possuem.

A principal dificuldade está relacionada ao baixo número de *startups* que se disponibilizaram a responder o *survey*, visto a taxa de resposta de 26%. Esta limitação se deve ao fato de que, por ser um questionário confirmatório, não foi possível convidar *startups* que não tivessem participado da primeira etapa, restringindo o número de potenciais participantes.

Como contribuição, assim como o *survey* confirmatório, o grupo focal conduzido permite analisar a opinião de atores da indústria em relação as diretrizes propostas. Este *feedback* é importante pois permite verificar qual a relevância e a importância dessas diretrizes de acordo com profissionais que enfrentam as dificuldades e desafios de gerir ou trabalhar em uma *startup* diariamente. A partir disso, é possível analisar se as diretrizes estão de acordo com a realidade desses profissionais e realizar os ajustes necessários para adequá-las. O grupo focal também permite verificar o perfil de uma parcela das pessoas envolvidas no cenário por meio do tempo de experiência que possuem com desenvolvimento de software, o tempo de experiência que possuem com *startups* e qual o nível de escolaridade que possuem.

Entre as dificuldades encontradas estão a limitação geográfica, visto que o grupo focal deve ser realizado presencialmente, fazendo com que apenas profissionais de Maringá ou região pudessem participar da sessão e, com isso, potencialmente restringindo o número de potenciais participantes; e o agendamento da sessão, visto que encontrar um horário para realizar a sessão que fosse confortável para todos os participantes não foi trivial, uma vez que todos possuem rotinas diferentes, com alguns inclusive dividindo estudo e trabalho.

Assim como o *survey* confirmatório, o painel com especialistas tem como contribuição as opiniões de pessoas que estão diretamente relacionadas ao contexto de *startups* em relação as diretrizes propostas. Porém, dessa vez, ao invés de atores da indústrias, as opiniões são provenientes de integrantes da academia. Os resultados demonstraram que, assim como para a indústria, as diretrizes são importantes e possuem relevância para a academia, visto o alto grau de concordância coletado no painel. Além disto, o painel também permitiu verificar o perfil destes especialistas, ao apresentar o tempo de experiência que possuem com pesquisa em *startups*, qual a sua área de atuação dentro do contexto e o seu grau de formação.

A principal dificuldade encontrada foi entrar em contato com os pesquisadores da área, dado isso a taxa de resposta baixa. Em boa parte dos casos, a única fonte de contato possível de ser encontrada foi o e-mail informado em suas publicações, entretanto, não houve retorno de muitos dos pesquisadores, inclusive, em muitos casos, logo após o envio, era retornada um mensagem do serviço de e-mail informando que aquele e-mail não

existia. Acredita-se que, como estes endereços são institucionais, boa parte deles estão desatualizados. A distância geográfica também prejudicou, visto que a falta de amigos em comum em redes sociais dificulta o contato, visto que estas redes, como por exemplo o Facebook, possuem mecanismos que tratam mensagens de desconhecidos como *spam*.

7

Conclusões

7.1 Considerações Iniciais

Neste capítulo são apresentadas as considerações finais deste trabalho de pesquisa, destacando as contribuições, dificuldades e limitações, oportunidades de trabalhos futuros e a disseminação dos resultados obtidos.

Ao retomar a questão levantada no Capítulo 1, de que forma a ES pode apoiar as atividades de desenvolvimento de software em *startups*, este trabalho permite concluir que as diretrizes propostas são adequadas para proporcionar esse suporte. Por serem subsidiadas por estudos empíricos, estas diretrizes estão de acordo com a realidade destas organizações e, portanto, condizem com as suas necessidades. Sendo assim, novas propostas provenientes da ES que sejam empiricamente sustentadas podem trazer inúmeros benefícios para auxiliar na sobrevivência das *startups*.

7.2 Contribuições

A principal contribuição deste trabalho refere-se a proposta das diretrizes para o desenvolvimento de software em *startups*. Estas diretrizes são importantes pois configuram um ponto de partida para aquelas *startups* que estão iniciando suas atividades, sendo este um ponto crítico para o sucesso da mesma. Por sua falta de maturidade, as diretrizes podem ser uma fonte de informações e conhecimento crucial para o desempenho destas organizações. *Startups* que se encontram em fase de estabilização ou crescimento também podem se beneficiar das diretrizes, visto que as mesmas não fazem distinção de nível de maturidade.

As diretrizes configuram um guia acessível, pois descrevem detalhadamente os problemas e apresentam soluções para os mesmos. Além disso, apresentam diversas referências que podem ser consultadas, quais são os resultados esperados ao colocá-las em prática e auxilia para que isso ocorra por meio da sugestão de diversas ferramentas. Ao consultar as diretrizes é esperado que as *startups* tenham mais qualidade e maior produtividade durante as suas fases, evitando que falhem e deixem de existir.

Outra contribuição refere-se a execução do mapeamento sistemático, que permitiu obter um panorama do estado de pesquisa sobre desenvolvimento de software em *startups*. Por meio do mesmo pôde-se observar que a lacuna de pesquisa apresentada por Sutton (2000) ainda existe, visto que o número de estudos que apresentam resultados transferíveis para a indústria ainda é baixo. Por outro lado, o mapeamento permitiu identificar um conjunto considerável de técnicas, práticas e ferramentas por meio dos estudos encontrados, podendo servir como um ponto de partida para organizações que ainda não possuem bem definidos estes aspectos técnicos.

Em comparação aos mapeamentos apresentados por Paternoster *et al.* (2014) e Klotins *et al.* (2015), o mapeamento realizado buscou apresentar seus achados de uma forma que *startups*, sejam em fase inicial, de estabilização ou crescimento, possam utilizar este conhecimento como uma base para a seleção de quais técnicas, práticas e ferramentas serão utilizadas em suas atividades e a partir disto evitar que escolhas sejam feitas sem conhecimento de sua validade e aplicação.

Uma outra contribuição refere-se a condução do *survey* com *startups* brasileiras que tenham entre suas atividades principais o desenvolvimento de software. Este estudo empírico possibilitou caracterizar o estado do desenvolvimento de software nestas *startups*, a partir das metodologias, ferramentas, técnicas e práticas utilizadas. O estudo também permitiu verificar o perfil das pessoas envolvidas no cenário por meio do tempo de experiência que possuem com desenvolvimento de software, o tempo de experiência que possuem com *startups* e qual o nível de escolaridade que possuem. Além disto, a condução do *survey* do estudo possibilitou traçar o perfil das *startups* brasileiras, sendo que por meio do mesmo é possível indicar qual a média de colaboradores que estas *startups* possuem, em qual nível de maturidade se encontram, quais delas possuem algum tipo de investimento externo e quais fazem parte de algum grupo ou rede de colaboração.

Como contribuição secundária destaca-se a interação com a indústria, visto que em diversos momentos durante a pesquisa foi possível ter um contato direto com profissionais que estão diretamente relacionados a *startups*, destacando a realização do *survey*, questionário confirmatório e o grupo focal. Esta interação proporcionou uma melhoria na relevância e qualidade dos estudos realizados neste trabalho (Rodríguez *et al.*, 2014;

Sjoberg *et al.*, 2007) e contribuiu para uma aproximação entre a academia e a indústria, de forma que as *startups* possam fazer uso do conhecimento gerado pela academia de forma ágil e que a academia possa aproveitar o ambiente dinâmico e desafiador das *startups* para a aplicação de novas pesquisas.

Por fim, este trabalho permite fornecer um conjunto de evidências que foram obtidas por meio do mapeamento sistemático e do *survey* para os pesquisadores e profissionais da indústria que buscam entender o contexto de desenvolvimento de software em *startups*. A partir disto, espera-se que seja fomentado o interesse na realização de outros trabalhos que abordem o desenvolvimento de software em *startups*.

7.3 Dificuldades e Limitações

A principal dificuldade encontrada refere-se a não encontrar um modelo adequado para a especificação das diretrizes. Sendo assim, o autor precisou propor um formato de especificação que pudesse proporcionar os detalhes necessários para o compreendimento das diretrizes propostas, definindo seus elementos e o relacionamento entre eles. Em relação a composição das diretrizes, a principal dificuldade foi encontrar os materiais necessários para subsidiar as ideias. Apesar do mapeamento sistemático e do *survey* terem fornecido vários argumentos que foram utilizados, muitas das diretrizes ainda careciam de fontes de referência. Apesar de estudos que pudessem contribuir existirem, os mesmos são difusos, fazendo com que vários materiais tivessem que ser consultados para que fosse possível formular um argumento.

Outra dificuldade encontrada durante a elaboração deste trabalho está relacionada principalmente àquelas atividades no qual era necessário o envolvimento de terceiros, destacando a condução do *survey* exploratório, do *survey* confirmatório, do grupo focal e do painel com especialistas.

Em relação a condução do *survey*, a principal dificuldade refere-se a baixa taxa de resposta, sendo este um problema conhecido na realização deste tipo de estudo. Para minimizar este efeito, o *survey* foi desenvolvido de forma que pudesse ser respondido rapidamente, sendo estimado o tempo médio para responder o mesmo em 15 minutos. Ainda assim, a taxa de resposta foi baixa, com apenas 16,58%, mesmo após ter enviado o convite para participar do questionário mais de uma vez para todas as *startups*. O mesmo pode ser dito quanto ao questionário confirmatório, que apresentou taxa de resposta de 26%.

Quanto a condução do grupo focal estão a limitação geográfica, visto que o grupo focal deve ser realizado presencialmente, potencialmente restringindo o número de possíveis

participantes e o agendamento da sessão, pois encontrar um horário de realização da sessão que fosse aceito por todos dependeu de inúmeros fatores.

No que se refere à condução do painel com especialistas, a principal limitação foi entrar em contato com os pesquisadores da área. Geralmente a única fonte de contato possível de ser encontrada é o e-mail informado em suas publicações, entretanto, não houve retorno de muitos dos pesquisadores pois acredita-se que, como estes endereços são institucionais, boa parte deles estão desatualizados. Além disto, em virtude da distância geográfica, ou seja, falta de amigos em comum, é difícil entrar em contato por meio de redes sociais, visto que estas redes, como por exemplo o Facebook, possuem mecanismos que tratam mensagens de desconhecidos como *spam*.

Em virtude das dificuldades encontradas durante a realização do questionário confirmatório, grupo focal e painel com especialistas, a principal limitação deste trabalho está relacionada ao refinamento das diretrizes propostas. Esperava-se que fosse possível encontrar um número maior de participantes nessas etapas, o que geraria uma quantidade maior de *feedback* e potencialmente de mais qualidade.

7.4 Trabalhos Futuros

De acordo com as limitações e lacunas encontradas durante a execução deste trabalho, é possível destacar algumas oportunidades de trabalhos futuros.

A partir do mapeamento sistemático é possível verificar que ainda existe uma lacuna em relação a quantidade de estudos empíricos a ser preenchida (Sutton, 2000). Dessa forma, é sugerido a execução de estudos que visem criar cenários onde técnicas, práticas e ferramentas possam ser validadas e, principalmente, comparadas. Possivelmente, a partir dos dados levantados, criarem-se guias de métodos confiáveis que possam ser adotados.

Devido a baixa taxa de resposta do *survey*, seria interessante realizar um novo estudo, talvez com uma janela de tempo maior, para identificar o perfil de mais *startups* brasileiras. Outra possibilidade seria aprofundar as questões que foram realizadas, buscando, por exemplo, as motivações envolvidas nas escolhas de cada prática, técnica ou ferramenta. Na mesma linha, uma possibilidade de trabalho futuro seria a expansão do grupo focal, realizando sessões em diferentes regiões do país, principalmente aquelas ainda não estudadas, permitindo identificar o perfil das *startups* brasileiras por meio da visão de seus profissionais.

Em relação às diretrizes propostas, uma possibilidade seria definir um guia de prioridade para a adoção das mesmas de acordo com o nível de maturidade das *startups*. Outra possibilidade está relacionada a buscar o estado da prática em *startups* consideradas

de sucesso e que são referência de mercado e, a partir deste conhecimento, realizar alterações nas diretrizes propostas. Além disto, investigar a respeito da interoperabilidade de ferramentas para *startups* também é uma oportunidade de pesquisa.

Referências

Aaen, I. *Essence-pragmatic software innovation.* 2015.

Ahmad, M. O.; Markkula, J.; Oivo, M. Insights into the perceived benefits of kanban in software companies: Practitioners' views. In: *International Conference on Agile Software Development*, Springer, 2016, p. 156–168.

Ahmed, T. M.; Bezemer, C.-P.; Chen, T.-H.; Hassan, A. E.; Shang, W. Studying the effectiveness of application performance management (apm) tools for detecting performance regressions for web applications: An experience report. In: *Proceedings of the 13th International Conference on Mining Software Repositories*, ACM, 2016, p. 1–12.

Al-Baik, O.; Miller, J. The kanban approach, between agility and leanness: a systematic review. *Empirical Software Engineering*, v. 20, n. 6, p. 1861–1897, 2015.

Alves, C.; Pereira, S.; Castro, J. A study in market-driven requirements engineering. In: *WER*, 2006.

Alves, N. S. R.; Spínola, R. O. Organização do corpo de conhecimento sobre dívida técnica: tipos, indicadores, estratégias de gerenciamento e causas. In: *XVI Simpósio Brasileiro de Qualidade de Software*, SBC, 2017, p. 340–354.

April, A.; Abran, A. *Software maintenance management: evaluation and continuous improvement*, v. 67. John Wiley & Sons, 2012.

Azizyan, G.; Magarian, M. K.; Kajko-Matsson, M. Survey of agile tool usage and needs. In: *Agile Conference (AGILE), 2011*, IEEE, 2011, p. 29–38.

Azizyan, G.; Magarian, M. K.; Kajko-Matsson, M. The dilemma of tool selection for agile project management. In: *ICSEA 2012, The Seventh International Conference on Software Engineering Advances*, 2012, p. 605–614.

Bacchelli, A.; Bird, C. Expectations, outcomes, and challenges of modern code review. In: *Proceedings of the 2013 international conference on software engineering*, IEEE Press, 2013, p. 712–721.

Ball, T.; Kim, J.-M.; Porter, A. A.; Siy, H. P. If your version control system could talk. In: *ICSE Workshop on Process Modelling and Empirical Studies of Software Engineering*, 1997.

Berner, S.; Weber, R.; Keller, R. K. Observations and lessons learned from automated testing. In: *Software Engineering, 2005. ICSE 2005. Proceedings. 27th International Conference on*, IEEE, 2005, p. 571–579.

Björk, J.; Ljungblad, J.; Bosch, J. Lean product development in early stage startups. In: *IW-LCSP@ ICSOB*, 2013, p. 19–32.

Blischak, J. D.; Davenport, E. R.; Wilson, G. A quick introduction to version control with git and github. *PLoS computational biology*, v. 12, n. 1, p. 1–18, 2016.

Bosch, J.; Olsson, H. H.; Björk, J.; Ljungblad, J. The early stage software startup development model: a framework for operationalizing lean principles in software startups. In: *Lean Enterprise Software and Systems*, Springer, p. 1–15, 2013.

Bosch-Sijtsema, P.; Bosch, J. User involvement throughout the innovation process in high-tech industries. *Journal of Product Innovation Management*, v. 32, n. 5, p. 793–807, 2015.

Bourque, P.; Fairley, R. E. *Guide to the software engineering body of knowledge (swebok (r)): Version 3.0*. IEEE Computer Society Press, 2014.

Brereton, P.; Kitchenham, B. A.; Budgen, D.; Turner, M.; Khalil, M. Lessons from applying the systematic literature review process within the software engineering domain. *Journal of systems and software*, v. 80, n. 4, p. 571–583, 2007.

Budgen, D.; Turner, M.; Brereton, P.; Kitchenham, B. Using mapping studies in software engineering. In: *Proceedings of PPIG*, Lancaster University, 2008, p. 195–204.

Cannon-Bowers, J. A.; Salas, E. Reflections on shared cognition. *Journal of organizational behavior*, v. 22, n. 2, p. 195–202, 2001.

Carmel, E. Time-to-completion in software package startups. In: *1994 Proceedings of the Twenty-Seventh Hawaii International Conference on System Sciences*, 1994, p. 498–507.

Carmines, E. G.; Zeller, R. A. *Reliability and validity assessment*, v. 17. Sage publications, 1979.

Chicote, M. Startups and technical debt: managing technical debt with visual thinking. In: *Proceedings of the 1st International Workshop on Software Engineering for Startups*, IEEE Press, 2017, p. 10–11.

Claps, G. G.; Svensson, R. B.; Aurum, A. On the journey to continuous deployment: Technical and social challenges along the way. *Information and Software technology*, v. 57, p. 21–31, 2015.

Cohen, D.; Lindvall, M.; Costa, P. Agile software development. *DACS SOAR Report*, v. 11, 2003.

Coleman, G.; O'Connor, R. V. An investigation into software development process formation in software start-ups. *Journal of Enterprise Information Management*, v. 21, n. 6, p. 633–648, 2008.

Conboy, K.; Fitzgerald, B. Method and developer characteristics for effective agile method tailoring: A study of xp expert opinion. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, v. 20, n. 1, p. 2, 2010.

Cooke, R.; Shrader-Frechette, K.; *et al.* *Experts in uncertainty: opinion and subjective probability in science*. Oxford University Press on Demand, 1991.

Crocker, L.; Algina, J. *Introduction to classical and modern test theory*. ERIC, 1986.

Crowne, M. Why software product startups fail and what to do about it. evolution of software product development in startup companies. In: *Engineering Management Conference, 2002. IEMC'02. 2002 IEEE International*, IEEE, 2002, p. 338–343.

Crunchbase *Crunchbase - The Business Graph*. `http://www.crunchbase.com`, 2014.

Cukier, D. *Software startup ecosystems evolution: a maturity model*. Tese de Doutoramento, Universidade de São Paulo, 2017.

Cukier, D.; Kon, F.; Krueger, N. Designing a maturity model for software startup ecosystems. In: *International Conference on Product-Focused Software Process Improvement*, Springer, 2015a, p. 600–606.

Cukier, D.; Kon, F.; Krueger, N. *Towards a software startup ecosystems maturity model.* Relatório Técnico, Department of Computer Science, University of São Paulo, Tech. Rep. RT-MAC-2015–03, 2015b.

Curtis, B.; Sappidi, J.; Szynkarski, A. Estimating the size, cost, and types of technical debt. In: *Proceedings of the Third International Workshop on Managing Technical Debt*, IEEE Press, 2012, p. 49–53.

Dahlstedt, A.; Karlsson, L.; Persson, A.; NattochDag, J.; Regnell, B. Market-driven requirements engineering processes for software products–a report on current practices. In: *International Workshop on COTS and Product Software: Why Requirements Are So Important (RECOTS)*, 2003.

Dande, A.; Eloranta, V.-P.; Kovalainen, A.-J.; Lehtonen, T.; Leppänen, M.; Salmimaa, T.; Sayeed, M.; Vuori, M.; Rubattel, C.; Weck, W.; *et al. Software startup patterns - an empirical study.* Relatório Técnico, Tampere University of Technology, 2014.

Dangle, K. C.; Larsen, P.; Shaw, M.; Zelkowitz, M. V. Software process improvement in small organizations: a case study. *IEEE software*, v. 22, n. 6, p. 68–75, 2005.

De Alwis, B.; Sillito, J. Why are software projects moving from centralized to decentralized version control systems? In: *Proceedings of the 2009 ICSE Workshop on cooperative and human aspects on software engineering*, IEEE Computer Society, 2009, p. 36–39.

Duc, A. N.; Abrahamsson, P. Minimum viable product or multiple facet product? the role of mvp in software startups. In: *International Conference on Agile Software Development*, Springer, 2016, p. 118–130.

Dunkin, D. *Applying lean startup methodology to software startup: Case study-company "x".* Dissertação de Mestrado, 2017.

Dybå, T.; Dingsøyr, T. Strength of evidence in systematic reviews in software engineering. In: *Proceedings of the Second ACM-IEEE international symposium on Empirical software engineering and measurement*, ACM, 2008, p. 178–187.

Dyba, T.; Dingsoyr, T. What do we know about agile software development? *IEEE software*, v. 26, n. 5, p. 6–9, 2009.

Dyba, T.; Dingsoyr, T.; Hanssen, G. K. Applying systematic reviews to diverse study types: An experience report. In: *Empirical Software Engineering and Measurement, 2007. ESEM 2007. First International Symposium on*, IEEE, 2007, p. 225–234.

Dyba, T.; Kitchenham, B. A.; Jorgensen, M. Evidence-based software engineering for practitioners. *IEEE software*, v. 22, n. 1, p. 58–65, 2005.

El-Attar, M.; Miller, J. A subject-based empirical evaluation of ssucd's performance in reducing inconsistencies in use case models. *Empirical Software Engineering*, v. 14, n. 5, p. 477, 2009.

Eppler, M. J. A comparison between concept maps, mind maps, conceptual diagrams, and visual metaphors as complementary tools for knowledge construction and sharing. *Information visualization*, v. 5, n. 3, p. 202–210, 2006.

Fayad, M. E.; Laitinen, M.; Ward, R. P. Thinking objectively: software engineering in the small. *Communications of the ACM*, v. 43, n. 3, p. 115–118, 2000.

Fern, E. F.; Fern, E. E. *Advanced focus group research*. Sage, 2001.

Fischer, G.; Nakakoji, K.; Ye, Y. Metadesign: Guidelines for supporting domain experts in software development. *IEEE software*, v. 26, n. 5, p. 37–44, 2009.

Fischer, M.; Pinzger, M.; Gall, H. Populating a release history database from version control and bug tracking systems. In: *Software Maintenance, 2003. ICSM 2003. Proceedings. International Conference on*, IEEE, 2003, p. 23–32.

Fitzgerald, B.; Stol, K.-J. Continuous software engineering and beyond: trends and challenges. In: *Proceedings of the 1st International Workshop on Rapid Continuous Software Engineering*, ACM, 2014, p. 1–9.

Forza, C. Survey research in operations management: a process-based perspective. *International journal of operations & production management*, v. 22, n. 2, p. 152–194, 2002.

Giardino, C.; Bajwa, S. S.; Wang, X.; Abrahamsson, P. Key challenges in early-stage software startups. In: *International Conference on Agile Software Development*, Springer, 2015, p. 52–63.

Giardino, C.; Paternoster, N.; Unterkalmsteiner, M.; Gorschek, T.; Abrahamsson, P. Software development in startup companies: the greenfield startup model. *IEEE Transactions on Software Engineering*, v. 42, n. 6, p. 585–604, 2016.

Giardino, C.; Unterkalmsteiner, M.; Paternoster, N.; Gorschek, T.; Abrahamsson, P. What do we know about software development in startups? *IEEE software*, v. 31, n. 5, p. 28–32, 2014.

Gil, A. C. Como elaborar projetos de pesquisa. *São Paulo*, v. 5, n. 61, p. 16–17, 2002.

Gliem, J. A.; Gliem, R. R. Calculating, interpreting, and reporting cronbach's alpha reliability coefficient for likert-type scales. In: *SPSS for Windows step by step: A simple guide and reference*, Midwest Research-to-Practice Conference in Adult, Continuing, and Community Education, 2003, p. 82–88.

van Gompel, M.; Noordzij, J.; Huygens, I.; de Valk, R.; Scharnhorst, A. *Guidelines for software quality.* Relatório Técnico, Technical Report. CLARIAH, 2016.

Gonçalves, J. A. M. *Requirements engineering in software startups: a qualitative investigation.* Tese de Doutoramento, Universidade de São Paulo, 2017.

Gordón, M.-L. S.; O'Connor, R. V. Understanding the gap between software process practices and actual practice in very small companies. *Software Quality Journal*, v. 24, n. 3, p. 549–570, 2016.

Gralha, C.; Damian, D.; Wasserman, A. I. T.; Goulão, M.; Araújo, J. The evolution of requirements practices in software startups. In: *Proceedings of the 40th International Conference on Software Engineering*, ACM, 2018, p. 823–833.

Grubb, P.; Takang, A. A. *Software maintenance: concepts and practice.* World Scientific, 2003.

Hartmann, A. The role of organizational culture in motivating innovative behaviour in construction firms. *Construction innovation*, v. 6, n. 3, p. 159–172, 2006.

Highsmith, J.; Cockburn, A. Agile software development: The business of innovation. *Computer*, v. 34, n. 9, p. 120–127, 2001.

Hokkanen, L.; Väänänen-Vainio-Mattila, K. Ux work in startups: current practices and future needs. In: *International Conference on Agile Software Development*, Springer, 2015, p. 81–92.

Hubbard, G. Measuring organizational performance: beyond the triple bottom line. *Business strategy and the environment*, v. 18, n. 3, p. 177–191, 2009.

Ivarsson, M.; Gorschek, T. A method for evaluating rigor and industrial relevance of technology evaluations. *Empirical Software Engineering*, v. 16, n. 3, p. 365–395, 2011.

Jansen, S.; Brinkkemper, S.; Hunink, I.; Demir, C. Pragmatic and opportunistic reuse in innovative start-up companies. *IEEE software*, v. 25, n. 6, p. 42–49, 2008.

Järvinen, J.; Huomo, T.; Mikkonen, T.; Tyrväinen, P. From agile software development to mercury business. In: *International Conference of Software Business*, Springer, 2014, p. 58–71.

Jorgensen, M. Practical guidelines for expert-judgment-based software effort estimation. *IEEE software*, v. 22, n. 3, p. 57–63, 2005.

Kaplan, R. S.; Norton, D. P. The strategy map: guide to aligning intangible assets. *Strategy & leadership*, v. 32, n. 5, p. 10–17, 2004.

Karlsson, L.; Dahlstedt, Å.; och Dag, J. N.; Regnell, B.; Persson, A. Challenges in market-driven requirements engineering-an industrial interview study. In: *Eighth International Workshop on Requirements Engineering: Foundation for Software Quality*, 2002, p. 37–49.

Kaysen, A. C.; Mikkelsen, F. B.; Aaen, I. *Entrepreneurial software innovation.* Tese de Doutoramento, Aalborg University, 2016.

Keil, M.; Carmel, E. Customer-developer links in software development. *Communications of the ACM*, v. 38, n. 5, p. 33–44, 1995.

Kernighan, B. W.; Plauger, P. J. *The elements of programming style 2nd edition.* McGraw Hill, 1978.

Khanna, G.; Beaty, K.; Kar, G.; Kochut, A. Application performance management in virtualized server environments. In: *Network Operations and Management Symposium, 2006. NOMS 2006. 10th IEEE/IFIP*, IEEE, 2006, p. 373–381.

Kirkman, B. L.; Rosen, B. Beyond self-management: Antecedents and consequences of team empowerment. *Academy of Management journal*, v. 42, n. 1, p. 58–74, 1999.

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

Kitchenham, B. A.; Charters, S. Guidelines for performing systematic literature reviews in software engineering. In: *Technical report, Ver. 2.3 EBSE Technical Report. EBSE*, sn, 2007.

Kitchenham, B. A.; Dyba, T.; Jorgensen, M. Evidence-based software engineering. In: *Proceedings of the 26th international conference on software engineering*, IEEE Computer Society, 2004, p. 273–281.

Kitchenham, B. A.; Pfleeger, S. L. Personal opinion surveys. In: *Guide to advanced empirical software engineering*, Springer, p. 63–92, 2008.

Kitchenham, B. A.; Pfleeger, S. L.; Pickard, L. M.; Jones, P. W.; Hoaglin, D. C.; El Emam, K.; Rosenberg, J. Preliminary guidelines for empirical research in software engineering. *IEEE Transactions on software engineering*, v. 28, n. 8, p. 721–734, 2002.

Klepper, S. Entry, exit, growth, and innovation over the product life cycle. *The American economic review*, v. 86, n. 3, p. 562–583, 1996.

Klotins, E.; Unterkalmsteiner, M.; Gorschek, T. Software engineering knowledge areas in startup companies: a mapping study. In: *International Conference of Software Business*, Springer, 2015, p. 245–257.

Kon, F.; Cukier, D.; Melo, C.; Hazzan, O.; Yuklea, H. *A panorama of the israeli software startup ecosystem.* Relatório Técnico, Department of Computer Science, University of São Paulo, 2014.

Kon, F.; Cukier, D.; Melo, C.; Hazzan, O.; Yuklea, H. *A conceptual framework for software startup ecosystems: the case of israel.* Relatório Técnico, Department of Computer Science, University of São Paulo, Tech. Rep. RT-MAC-2015-01, 2015.

Kon, F.; Lyons, T. S. Software startup ecosystems evolution: The new york case study. In: *2nd International Workshop on Software Startups*, IEEE, 2016.

Kon, F.; Monteiro, J. Empreendedorismo em computação e startups de software. In: *Atualizações em Informática*, cap. 5, XXXIV Congresso da Sociedade Brasileira de Computação, p. 176–216, 2014.

Krueger, R. A.; Casey, M. A. *Focus groups: A practical guide for applied research.* Thousand Oaks: Sage, 1994.

Land, L. P. W.; Tan, B.; Bin, L. Investigating training effects on software reviews: a controlled experiment. In: *Empirical Software Engineering, 2005. 2005 International Symposium on*, IEEE, 2005, p. 356–366.

Landis, J. R.; Koch, G. G. The measurement of observer agreement for categorical data. *biometrics*, v. 33, n. 1, p. 159–174, 1977.

Langer, T.; Vaněček, P. Agile methods in tech-startup. In: *IMEA 2012*, 2012, p. 63–68.

Larrucea, X.; O'Connor, R. V.; Colomo-Palacios, R.; Laporte, C. Y. Software process improvement in very small organizations. *IEEE Software*, v. 33, n. 2, p. 85–89, 2016.

Leal, G. C. L. *Know-cap: um método para capitalização de conhecimento no desenvolvimento de software*. Dissertação de Mestrado, 2015.

Lee, G. K.; Cole, R. E. From a firm-based to a community-based model of knowledge creation: The case of the linux kernel development. *Organization science*, v. 14, n. 6, p. 633–649, 2003.

Li, Z.; Avgeriou, P.; Liang, P. A systematic mapping study on technical debt and its management. *Journal of Systems and Software*, v. 101, p. 193–220, 2015.

Lindgren, E.; Münch, J. Raising the odds of success: the current state of experimentation in product development. *Information and Software Technology*, v. 77, p. 80–91, 2016.

Marmer, M.; Herrmann, B.; Dogrultan, E.; Berman, R.; Eesley, C.; Blank, S. The startup ecosystem report 2012. *Startup Genome*, 2012.

Martin, D.; Rooksby, J.; Rouncefield, M.; Sommerville, I. 'good'organisational reasons for'bad'software testing: An ethnographic study of testing in a small software company. In: *Proceedings of the 29th international conference on Software Engineering*, IEEE Computer Society, 2007, p. 602–611.

Martin, R. L. The innovation catalysts. *Harvard Business Review*, v. 89, n. 6, p. 82–87, 2011.

Maurya, A. *Running lean: iterate from plan a to a plan that works*. "O'Reilly Media, Inc.", 2012.

Melegati, J.; Goldman, A. Seven patterns for software startups. In: *Proceedings of the 22nd Conference on Pattern Languages of Programs*, The Hillside Group, 2015, p. 20.

Melegati, J.; Goldman, A. Requirements engineering in software startups: a grounded theory approach. *2nd Int. Work. Softw. Startups, Trondheim, Norw*, 2016.

Melo, C. d. O.; Santos, V.; Katayama, E.; Corbucci, H.; Prikladnicki, R.; Goldman, A.; Kon, F. The evolution of agile software development in brazil. *Journal of the Brazilian Computer Society*, v. 19, n. 4, p. 523–552, 2013.

Miski, A. Development of a mobile application using the lean startup methodology. *International Journal of Scientific & Engineering Research*, v. 5, n. 1, p. 1743–1748, 2014.

Moreira, R. T. *Um perfil de capacidade para a melhoria do processo em micro e pequenas organizações orientadas à manutenção e evolução de produtos de software*. Dissertação de Mestrado, Universidade Federal de Pernambuco, 2015.

de Moura, E. S.; Herrera, M. R.; Santos, L.; Conte, T. When software impacts the economy and environment. *IEEE Software*, v. 33, n. 6, p. 23–26, 2016.

Nanavati, R. Experience report: a pure shirt fits. In: *ACM Sigplan Notices*, ACM, 2008, p. 347–352.

Nascimento, L. M. A.; Ribeiro, T. V. É possível balancear qualidade e time-to-deliver em ambientes de desenvolvimento de software para inovação? In: *XVI Simpósio Brasileiro de Qualidade de Software*, SBC, 2017.

Nascimento, L. M. A.; Travassos, G. H. Software knowledge registration practices at software innovation startups: Results of an exploratory study. In: *Proceedings of the 31st Brazilian Symposium on Software Engineering*, ACM, 2017, p. 234–243.

Nguyen, T. H. Using control charts for detecting and understanding performance regressions in large software. In: *Software Testing, Verification and Validation (ICST), 2012 IEEE Fifth International Conference on*, IEEE, 2012, p. 491–494.

Nguyen-Duc, A.; Shah, S. M. A.; Ambrahamsson, P. Towards an early stage software startups evolution model. In: *Software Engineering and Advanced Applications (SEAA), 2016 42th Euromicro Conference on*, IEEE, 2016, p. 120–127.

Njima, M.; Demeyer, S. Evolution of software product development in startup companies. In: *Proceedings of the 16th edition of the BElgian-NEtherlands software eVOLution symposium*, CEUR, 2017, p. 10–12.

Nobel, C. *Why companies fail–and how their founders can bounce back*. Harvard Business School, 2011.

Och Dag, J. N. *Elicitation and management of user requirements in market-driven software development*. Dissertação de Mestrado, 2002.

Okoli, C.; Pawlowski, S. D. The delphi method as a research tool: an example, design considerations and applications. *Information & management*, v. 42, n. 1, p. 15–29, 2004.

Oliveira, M.; Freitas, H. Focus group, método qualitativo de pesquisa: resgatando a teoria, instrumentalizando o seu planejamento. *RAUSP*, v. 33, n. 3, p. 83–91, 1998.

Paternoster, N.; Giardino, C.; Unterkalmsteiner, M.; Gorschek, T.; Abrahamsson, P. Software development in startup companies: A systematic mapping study. *Information and Software Technology*, v. 56, n. 10, p. 1200–1218, 2014.

Petersen, K.; Vakkalanka, S.; Kuzniarz, L. Guidelines for conducting systematic mapping studies in software engineering: An update. *Information and Software Technology*, v. 64, p. 1–18, 2015.

Pohja, M. J. *Quality control in a startup software project*. Tese de Doutoramento, Tampere University of Technology, 2016.

Pompermaier, L.; Chanin, R.; Sales, A.; Fraga, K.; Prikladnicki, R. An empirical study on software engineering and software startups: findings from cases in an innovation ecosystem. In: *The 29th International Conference on Software Engineering and Knowledge Engineering*, 2017, p. 48–51.

Potts, C. Invented requirements and imagined customers: requirements engineering for off-the-shelf software. In: *Requirements Engineering, 1995., Proceedings of the Second IEEE International Symposium on*, IEEE, 1995, p. 128–130.

Qumer, A.; Henderson-Sellers, B. An evaluation of the degree of agility in six agile methods and its applicability for method engineering. *Information and software technology*, v. 50, n. 4, p. 280–295, 2008.

Radnor, Z.; McGuire, M. Performance management in the public sector: fact or fiction? *International Journal of Productivity and Performance Management*, v. 53, n. 3, p. 245–260, 2004.

Rafiq, U.; Bajwa, S. S.; Wang, X.; Lunesu, I. Requirements elicitation techniques applied in software startups. In: *Software Engineering and Advanced Applications (SEAA), 2017 43rd Euromicro Conference on*, IEEE, 2017, p. 141–144.

Ramler, R.; Wolfmaier, K. Economic perspectives in test automation: balancing automated and manual testing with opportunity cost. In: *Proceedings of the 2006 international workshop on Automation of software test*, ACM, 2006, p. 85–91.

Ratnavali, R.; Murthy, S. Balanced scorecard and performance management system: A study of four indian small and medium size enterprises. *Asian Journal of Management Research*, v. 7, n. 1, p. 52–66, 2016.

Reichert, M.; Hallerbach, A.; Bauer, T. Lifecycle management of business process variants. In: *Handbook on Business Process Management 1*, Springer, p. 251–278, 2015.

Ren, M.; Dong, Z. *What do we know about testing practices in software startups?* Dissertação de Mestrado, Blekinge Institute of Technology, 2017.

Ribeiro, T. V.; Travassos, G. H. On the alignment of source code quality perspectives through experimentation: an industrial case. In: *Proceedings of the Third International Workshop on Conducting Empirical Studies in Industry*, IEEE Press, 2015, p. 26–33.

Ries, E. *The lean startup*. New York: Crown Business, 2011.

Rodríguez, P.; Haghighatkhah, A.; Lwakatare, L. E.; Teppola, S.; Suomalainen, T.; Eskeli, J.; Karvonen, T.; Kuvaja, P.; Verner, J. M.; Oivo, M. Continuous deployment of software intensive products and services: A systematic mapping study. *Journal of Systems and Software*, v. 123, p. 263–291, 2017.

Rodríguez, P.; Kuvaja, P.; Oivo, M. Lessons learned on applying design science for bridging the collaboration gap between industry and academia in empirical software engineering. In: *Proceedings of the 2nd International Workshop on Conducting Empirical Studies in Industry*, ACM, 2014, p. 9–14.

Rose, J.; Jones, M.; Furneaux, B. An integrated model of innovation drivers for smaller software firms. *Information & Management*, v. 53, n. 3, p. 307–323, 2016.

Runeson, P. A survey of unit testing practices. *IEEE software*, v. 23, n. 4, p. 22–29, 2006.

Salerno, M. S.; de Vasconcelos Gomes, L. A.; da Silva, D. O.; Bagno, R. B.; Freitas, S. L. T. U. Innovation processes: Which process for which project? *Technovation*, v. 35, p. 59–70, 2015.

Sampieri, R. H.; Collado, C. F.; Lucio, P. B.; Pérez, M. d. l. L. C. *Metodología de la investigación*, v. 5. Mcgraw-hill México, 2010.

Santos, M. C. F. R. d. *O ecossistema de startups de software da cidade de são paulo*. Tese de Doutoramento, Universidade de São Paulo, 2016.

Sarasvathy, S. D. Entrepreneurship as a science of the artificial. *Journal of Economic Psychology*, v. 24, n. 2, p. 203–220, 2003.

Sawyer, P.; Sommerville, I.; Kotonya, G. Improving market-driven re processes. In: *VTT SYMPOSIUM*, VTT; 1999, 1999, p. 222–236.

Shah, D. *On startups: patterns and practices of contemporary software entrepreneurs*. Tese de Doutoramento, Massachusetts Institute of Technology, 2006.

Shahzad, F.; Xiu, G.; Shahbaz, M. Organizational culture and innovation performance in pakistan's software industry. *Technology in Society*, v. 51, p. 66–73, 2017.

Shull, F.; Seaman, C. Inspecting the history of inspections: An example of evidence-based technology diffusion. *IEEE software*, v. 25, n. 1, p. 88–90, 2008.

Silva, J. D.; de Miranda Junior, P. O. Utilização do mps-br para análise do processo de desenvolvimento de software em startups. *Abakós*, v. 5, n. 1, p. 18–33, 2016.

Singer, J. Using the american psychological association (apa) style guidelines to report experimental results. In: *Proceedings of workshop on empirical studies in software maintenance*, 1999, p. 71–75.

Sjoberg, D. I.; Dyba, T.; Jorgensen, M. The future of empirical methods in software engineering research. In: *Future of Software Engineering, 2007. FOSE'07*, IEEE, 2007, p. 358–378.

Sjøberg, D. I.; Hannay, J. E.; Hansen, O.; Kampenes, V. B.; Karahasanovic, A.; Liborg, N.-K.; Rekdal, A. C. A survey of controlled experiments in software engineering. *IEEE transactions on software engineering*, v. 31, n. 9, p. 733–753, 2005.

Škrabálek, J.; Böhm, C. Why modern mobile and web-based development need a lean agile web approach (lawa). In: *IDIMT-2013*, 2013, p. 225–232.

Smagalla, D. The truth about software startups: it's not the size of the budget but how it is used that determines success or failure of the enterprise. *MIT Sloan Management Review*, v. 45, n. 2, p. 7–8, 2004.

Song, X. M.; Montoya-Weiss, M. M. Critical development activities for really new versus incremental products. *Journal of product innovation management*, v. 15, n. 2, p. 124–135, 1998.

Souza, R.; Malta, K.; De Almeida, E. S. Software engineering in startups: a single embedded case study. In: *Proceedings of the 1st International Workshop on Software Engineering for Startups*, IEEE Press, 2017, p. 17–23.

Stagars, M. How universities can support their startups today. In: *University Startups and Spin-Offs*, Springer, p. 165–170, 2015.

Ståhl, D.; Bosch, J. Modeling continuous integration practice differences in industry software development. *Journal of Systems and Software*, v. 87, p. 48–59, 2014.

Steinmacher, I.; Treude, C.; Gerosa, M. Let me in: Guidelines for the successful onboarding of newcomers to open source projects. *IEEE Software*, v. PP, n. 99, 2018.

Streiner, D. L. Being inconsistent about consistency: When coefficient alpha does and doesn't matter. *Journal of personality assessment*, v. 80, n. 3, p. 217–222, 2003.

Suominen, A.; Hyrynsalmi, S.; Seppänen, M.; Still, K.; Aarikka-Stenroos, L. Software start-up failure. In: *9th International Workshop on Software Ecosystems (IWSECO 2017)*, 2017, p. 55–64.

Sutton, S. M. The role of process in software start-up. *IEEE Software*, v. 17, n. 4, p. 33–39, 2000.

Taheri, M.; Sadjadi, S. M. A feature-based tool-selection classification for agile software development. In: *SEKE*, 2015, p. 700–704.

Taipale, M. Huitale–a story of a finnish lean startup. In: *Lean Enterprise Software and Systems*, Springer, p. 111–114, 2010.

Tanveer, B. Guidelines for utilizing change impact analysis when estimating effort in agile software development. In: *Proceedings of the 21st International Conference on Evaluation and Assessment in Software Engineering*, ACM, 2017, p. 252–257.

Taromirad, M.; Ramsin, R. An appraisal of existing evaluation frameworks for agile methodologies. In: *Engineering of Computer Based Systems, 2008. ECBS 2008. 15th Annual IEEE International Conference and Workshop on the*, IEEE, 2008, p. 418–427.

Tegegne, E. W. *Software development methodologies and practices in startups-systematic literature review.* Dissertação de Mestrado, University of Oulu, 2018.

Terho, H.; Suonsyrjä, S.; Jaaksi, A.; Mikkonen, T.; Kazman, R.; Chen, H.-M. Lean startup meets software product lines: Survival of the fittest or letting products bloom? In: *SPLST*, 2015, p. 134–148.

Terho, H.; Suonsyrjä, S.; Systä, K. The developers dilemma: Perfect product development or fast business validation? In: *International Conference on Product-Focused Software Process Improvement*, Springer, 2016, p. 571–579.

Tingling, P.; Saeed, A. Extreme programming in action: a longitudinal case study. In: *Human-Computer Interaction. Interaction Design and Usability*, Springer, 2007, p. 242–251.

Travassos, G. H.; Gurov, D.; Amaral, E. *Introdução à engenharia de software experimental.* UFRJ, 2002.

Unterkalmsteiner, M.; Abrahamsson, P.; Wang, X.; Nguyen-Duc, A.; Shah, S.; Bajwa, S. S.; Baltes, G. H.; Conboy, K.; Cullina, E.; Dennehy, D.; *et al.* Software startups–a research agenda. *e-Informatica Software Engineering Journal*, v. 10, n. 1, p. 89–123, 2016.

Vajrapu, R. G.; Kothwar, S. *Software requirements prioritization practices in software start-ups: A qualitative research based on start-ups in india.* Dissertação de Mestrado, Blekinge Institute of Technology, 2018.

Vasilescu, B.; Yu, Y.; Wang, H.; Devanbu, P.; Filkov, V. Quality and productivity outcomes relating to continuous integration in github. In: *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering*, ACM, 2015, p. 805–816.

Visser, P. S.; Krosnick, J. A.; Lavrakas, P. J. *Survey research*, cap. 9 Cambridge University Press, p. 223–252, 2000.

Viviano, A. The lean ux manifesto: Principle-driven design. 2014.

Wang, C.; Su, L.; Zhao, X.; Zhang, Y. Application performance monitoring and analyzing based on bayesian network. In: *Web Information System and Application Conference (WISA), 2014 11th*, IEEE, 2014, p. 61–64.

Wang, X.; Edison, H.; Bajwa, S. S.; Giardino, C.; Abrahamsson, P. Key challenges in software startups across life cycle stages. In: *International Conference on Agile Software Development*, Springer, 2016, p. 169–182.

West, D.; Grant, T.; Gerush, M.; D'silva, D. Agile development: Mainstream adoption has changed agility. *Forrester Research*, v. 2, n. 1, p. 41, 2010.

Wheeldon, J.; Faubert, J. Framing experience: Concept maps, mind maps, and data collection in qualitative research. *International Journal of Qualitative Methods*, v. 8, n. 3, p. 68–83, 2009.

Williams, L.; Cockburn, A. Guest editors' introduction: Agile software development: It's about feedback and change. *Computer*, v. 36, n. 6, p. 39–43, 2003.

Wohlin, C. Guidelines for snowballing in systematic literature studies and a replication in software engineering. In: *Proceedings of the 18th international conference on evaluation and assessment in software engineering*, ACM, 2014, p. 38.

Wohlin, C.; Aurum, A. Towards a decision-making structure for selecting a research design in empirical software engineering. *Empirical Software Engineering*, v. 20, n. 6, p. 1427–1455, 2015.

Wohlin, C.; Runeson, P.; Höst, M.; Ohlsson, M. C.; Regnell, B.; Wesslén, A. *Experimentation in software engineering*. Springer Science & Business Media, 2012.

Yli-Huumo, J.; Maglyas, A.; Smolander, K. The sources and approaches to management of technical debt: a case study of two product lines in a middle-size finnish software company. In: *International Conference on Product-Focused Software Process Improvement*, Springer, 2014, p. 93–107.

Apêndices

A

Protocolo de Pesquisa - Formulário de Avaliação do Protocolo do Mapeamento Sistemático

Após a leitura do Protocolo do Mapeamento Sistemático da Literatura sobre Técnicas, Práticas e Ferramentas no Desenvolvimento de Software em Startups, por gentileza, responda as questões a seguir considerando os seguintes níveis de concordância para cada uma das questões:

- **Discordo totalmente (Peso 1):** quando o protocolo não atende de nenhuma forma os critérios da questão.
- **Discordo parcialmente (Peso 2):** quando o protocolo não atende os critérios da questão.
- **Neutro (Peso 3): quando** o protocolo não deixa claro se atende ou não os critérios da questão.
- **Concordo parcialmente (Peso 4):** quando o protocolo atende parcialmente os critérios da questão.
- **Concordo totalmente (Peso 5):** quando o protocolo atende totalmente os critérios da questão.

No caso de questões avaliadas como 'Discordo totalmente', 'Discordo parcialmente' ou 'Neutro', mas não limitando-se a elas, sugerimos que justifique a resposta.

Questões:

1. Pode ser encontrada uma questão importante de Engenharia de Software que o mapeamento se dedica a resolver.
2. A string de busca é adequadamente derivada da pergunta de pesquisa.
3. As fontes e os tipos de busca escolhidos provavelmente cobrirá os estudos relevantes.
4. Os critérios para inclusão e exclusão dos estudos primários são descritos e adequados.
5. A pesquisa conseguirá avaliar adequadamente a qualidade/validade dos estudos incluídos.
6. O procedimento de extração dos dados aborda adequadamente a pergunta de pesquisa.
7. O procedimento de análise dos dados é apropriado para responder a pergunta de pesquisa.
8. Questão aberta para sugestões/comentários.

B

Protocolo de Pesquisa - Carta de Apresentação

Prezado,

Venho, por meio desta, solicitar sua autorização para a condução de uma pesquisa de campo da dissertação de mestrado do aluno Bruno Henrique Cavalcante, que está sendo desenvolvida sob minha orientação no Programa de Pós-Graduação em Ciência da Computação (PCC) da Universidade Estadual de Maringá (UEM).

A pesquisa será realizada por meio de um questionário *on-line* visando entender o desenvolvimento de software em *startups*.

As informações prestadas serão tratadas de forma a preservar a privacidade do respondente e da empresa. Nenhuma informação será publicada de forma individualizada.

Aguardamos o seu retorno e agradecemos antecipadamente pela colaboração.

Atenciosamente,
Gislaine Camila Lapasini Leal

Programa de Pós-Graduação em Ciência da Computação (PCC)
Universidade Estadual de Maringá

C

Protocolo de Pesquisa - Survey Exploratório

C.1 Caracterização do Respondente

1) Tempo de experiência com desenvolvimento de software:

( ) Menos do que 1 ano

( ) Entre 1 e 3 anos

( ) Entre 3 e 5 anos

( ) Mais do que 5 anos

2) Tempo de experiência com *startups*:

( ) Menos do que 1 ano

( ) Entre 1 e 3 anos

( ) Entre 3 e 5 anos

( ) Mais do que 5 anos

3) Grau de formação

( ) Curso Técnico

( ) Graduando

( ) Graduado

( ) Especialista

( ) Mestrando

( ) Mestre

( ) Doutorando

( ) Doutor

C.2 Caracterização da Startup

4) Cidade

_____

5) Estado

_____

6) Ano de fundação

_____

7) Nível de maturidade da *startup*
Nota: Em fase de início: é aquela que ainda está buscando uma oportunidade de mercado. Em fase de estabilização: é aquela que já entregou o produto para o primeiro cliente, sendo que esta fase dura até o momento em que o produto está estável o suficiente para ser entregue a um novo cliente. Em fase de crescimento: a fase de crescimento começa exatamente no momento em que a fase de estabilização termina.

( ) Em fase de início

( ) Em estabilização

( ) Em fase de crescimento

8) A *startup* faz parte de algum grupo ou rede de colaboração (ecossistema, incubadora, arranjos produtivos, etc)?

( ) Não

( ) Sim

9) Se sim, de qual(is)?

_____

10) A *startup* possui algum tipo de investidor externo?

( ) Não

( ) Sim

11) Se sim, qual(is)?

_____

12) Quantos colaboradores a *startup* possui?

_____

C.3 Desenvolvimento de Software

13) A *startup* utiliza metodologias ágeis?

( ) Não

( ) Sim

14) Se sim, qual(is)?

( ) Scrum

( ) Lean/Lean Startup

( ) Extreme Programming (XP)

( ) Experiment Driven Development (EDD)

( ) Test Driven Development (TDD)

( ) Feature Driven Development (FDD)

( ) Outro _____

15) A *startup* utiliza metodologias tradicionais?

( ) Não

( ) Sim

16) Se sim, qual(is)?

( ) Modelo em cascata

( ) Rational Unified Process (RUP)

( ) Outro _____

17) Selecione as práticas que são utilizadas pela *startup*:

( ) Integração contínua

( ) Padrões de código

( ) Uso de frameworks

( ) Uso de soluções COTS ou open source

( ) Reuso de software

( ) Cliente no local de desenvolvimento

( ) Releases curtos

( ) Bug-tracking

( ) Capacitação dos membros da equipe

( ) Empoderamento da equipe de desenvolvimento

( ) Uso de indicadores de desempenho

( ) Prototipagem evolucionária

( ) Entregas frequentes/contínuas

( ) Gerenciamento de requisitos

( ) Backlog

( ) Uso de logs e estatísticas

( ) Outro _____

18) Selecione as técnicas que são utilizadas pela *startup*:

( ) Planning Game

( ) Code review

( ) Programação em pares

( ) MVP (Minimum Viable Product)

( ) Kanban

( ) Outro _____

19) Selecione os tipos de testes que são utilizadas pela *startup*:

( ) Testes de aceitação

( ) Testes de integração

( ) Teste contínuo

( ) Teste unitário

( ) Testes de usabilidade

( ) Usuários de testes

( ) Outro _____

20) Selecione as ferramentas de controle de versão e configuração que são utilizadas pela *startup*:

( ) Github

( ) Bitbucket

( ) Gitlab

( ) Subversion

( ) Mercurial

( ) Outro _____

21) Selecione as ferramentas de gerenciamento do projeto que são utilizadas pela *startup*:

( ) Atlassian Jira

( ) Redmine

( ) Pivotal Tracker

( ) Mingle

( ) Asana

( ) Trello

( ) Basecamp

( ) Outro _____

22) Selecione as suítes/plataformas de desenvolvimento que são utilizadas pela *startup*:

( ) Eclipse

( ) NetBeans

( ) Visual Studio

( ) JetBrains

( ) Xcode

( ) PyCharm

( ) Outro _____

23) Selecione as ferramentas de gestão do conhecimento que são utilizadas pela *startup*:

( ) Atlassian Confluence

( ) Freshdesk

( ) eXo Platform

( ) OpenKM

( ) Collective Knowledge

( ) Outro _____

24) Comentário geral.
Fique a vontade para expressar sua opinião.

_____

D

Refinamento - Carta de Apresentação do Survey Confirmatório

Prezado,

Venho, por meio desta, solicitar sua autorização para a condução de uma pesquisa de campo da dissertação de mestrado do aluno Bruno Henrique Cavalcante, que está sendo desenvolvida sob minha orientação no Programa de Pós-Graduação em Ciência da Computação (PCC) da Universidade Estadual de Maringá (UEM).

Essa pesquisa é uma continuação do questionário enviado anteriormente. A partir dos resultados do levantamento inicial foram criadas diretrizes com o objetivo de apoiar o desenvolvimento de software em startups. Este questionário visa obter a opinião dos respondentes em relação as diretrizes propostas.

As diretrizes são apresentadas por meio de um título, que apresenta um resumo da diretriz e propõe o que deve ser feito, e uma descrição, que apresenta maiores detalhes sobre a mesma. A leitura da descrição é opcional.

As informações prestadas serão tratadas de forma a preservar a privacidade do respondente e da empresa. Nenhuma informação será publicada de forma individualizada.

Aguardamos o seu retorno e agradecemos antecipadamente pela colaboração.

Atenciosamente,
Gislaine Camila Lapasini Leal

Programa de Pós-Graduação em Ciência da Computação (PCC)
Universidade Estadual de Maringá

E

Refinamento - Caracterização do Respondente do Survey Confirmatório

1) Tempo de experiência com desenvolvimento de software:

( ) Menos do que 1 ano

( ) Entre 1 e 3 anos

( ) Entre 3 e 5 anos

( ) Mais do que 5 anos

2) Tempo de experiência com *startups*:

( ) Menos do que 1 ano

( ) Entre 1 e 3 anos

( ) Entre 3 e 5 anos

( ) Mais do que 5 anos

3) Grau de formação

( ) Curso Técnico

( ) Graduando

( ) Graduado

( ) Especialista

( ) Mestrando

( ) Mestre

( ) Doutorando

( ) Doutor

F

Refinamento - Carta de Apresentação do Grupo Focal

Prezado,

Meu nome é Bruno H. Cavalcante e sou mestrando em Ciência da Computação no Programa de Pós-Graduação em Ciência da Computação (PCC) da Universidade Estadual de Maringá (UEM) sob a orientação da Prof. Dr. Gislaine Camila Lapasini Leal. Venho, por meio desta, solicitar sua participação para a condução de uma pesquisa de campo como parte da dissertação. A pesquisa será realizada por meio de uma reunião com data e local ainda a serem definidos.

O objetivo da reunião é conduzir uma sessão de conversação e troca de ideias a respeito das diretrizes para o desenvolvimento de software em *startups* que estão sendo propostas na dissertação.

As informações prestadas serão tratadas de forma a preservar a privacidade dos participantes. Nenhuma informação será publicada de forma individualizada.

Aguardamos o seu retorno e agradecemos antecipadamente pela colaboração.

Atenciosamente,
Bruno Henrique Cavalcante

Programa de Pós-Graduação em Ciência da Computação (PCC)
Universidade Estadual de Maringá

G

Refinamento - Caracterização do Participante do Grupo Focal

1) Tempo de experiência com desenvolvimento de software:

( ) Menos do que 1 ano

( ) Entre 1 e 3 anos

( ) Entre 3 e 5 anos

( ) Mais do que 5 anos

2) Tempo de experiência com *startups*:

( ) Menos do que 1 ano

( ) Entre 1 e 3 anos

( ) Entre 3 e 5 anos

( ) Mais do que 5 anos

3) Grau de formação

( ) Curso Técnico

( ) Graduando

( ) Graduado

( ) Especialista

( ) Mestrando

( ) Mestre

( ) Doutorando

( ) Doutor

H

Refinamento - Termo de Consentimento do Grupo Focal

Título do Projeto: Diretrizes para Desenvolvimento de Software em Startups
Pesquisador Responsável: Bruno Henrique Cavalcante
Nome do participante: _____
Data de nascimento: _____
R.G.: _____

Você está sendo convidado(a) para participar, como voluntário, do projeto de pesquisa Diretrizes para Desenvolvimento de Software em Startups, de responsabilidade do pesquisador Bruno Henrique Cavalcante.

Leia cuidadosamente o que segue e qualquer dúvida não hesite em perguntar. Após ser esclarecido(a) sobre as informações a seguir e aceitar fazer parte do estudo, assine ao final deste documento, que consta em duas vias. Uma via pertence a você e a outra ao pesquisador responsável. Em caso de recusa você não sofrerá nenhuma penalidade.

Declaro ter sido esclarecido sobre os seguintes pontos:

1. O trabalho tem por objetivo obter a visão dos profissionais da indústria a respeito das diretrizes para o desenvolvimento de software em *startups* propostas neste trabalho.

2. A minha participação nesta pesquisa consistirá em participar da(s) reunião(ões) para discutir sobre as diretrizes. A(s) reunião(ões) será(ão) realizada(s) no local: _____ com duração prevista entre ____ e ____ horas a partir das _____ do dia ____ de _____ de 2018. O pesquisador responsável estará presente para conduzir e moderar a(s) reunião(ões). A(s) reunião(ões) será(ão) gravada(s) em áudio visto que o pesquisador precisará transcreve-la(s) após o seu término.

3. Ao participar desse trabalho estarei contribuindo diretamente para a área acadêmia por meio do meu conhecimento e experiência e indiretamente para a indústria, uma vez que o objetivo da pesquisa é proporcionar novos conhecimentos para a mesma.

4. A minha participação neste projeto deverá ter a duração de ____ hora(s) durante o período de ____ dia(s).

5. Não terei nenhuma despesa financeira ao participar da pesquisa e poderei deixar de participar ou retirar meu consentimento a qualquer momento, sem precisar justificar, e não sofrerei qualquer prejuízo.

6. Fui informado e estou ciente de que não há nenhum valor econômico, a receber ou a pagar, por minha participação, no entanto, caso eu tenha qualquer despesa decorrente da participação na pesquisa, serei ressarcido.

7. Caso ocorra algum dano comprovadamente decorrente de minha participação no estudo, poderei ser compensado conforme determina a Resolução 466/12 do Conselho Nacional de Saúde.

8. Meu nome será mantido em sigilo, assegurando assim a minha privacidade, e se eu desejar terei livre acesso a todas as informações e esclarecimentos adicionais sobre o estudo e suas consequências, enfim, tudo o que eu queira saber antes, durante e depois da minha participação.

9. Fui informado que os dados coletados serão utilizados, única e exclusivamente, para fins desta pesquisa, e que os resultados poderão ser publicados.

10. Qualquer dúvida, pedimos a gentileza de entrar em contato com Bruno Henrique Cavalcante, pesquisador responsável pela pesquisa, telefone: (44) 99982-1148, e-mail: bruno.h.cavalcante@gmail.com, com os pesquisadores Gislaine Camila Lapasini Leal, telefone: (44) 3011-5840, email: gclleal@uem.br e Renato Balancieri, telefone: (44) 3011-5123, email: rbalancieri2@uem.br.

Eu, _____, RG nº _____ declaro ter sido informado e concordo em participar, como voluntário, do projeto de pesquisa acima descrito.

Maringá, ____ de _____ de 2018.

_____	_____
Assinatura do participante	Assinatura do pesquisador responsável

I

Refinamento - Resumo das Diretrizes

Diretriz 1: Utilizar uma metodologia ágil para guiar o desenvolvimento de software.

As metodologias ágeis são utilizadas pela maioria das *startups* pois permitem um ciclo de desenvolvimento mais rápido. Estes métodos permitem que o produto seja desenvolvido em pequenas iterações, e isto, se alinhado a uma prática de entregas frequentes, possibilita que sejam obtidos *feedbacks* constantes dos usuários. Estes *feedbacks* são importantes pois ajudam a visualizar se o desenvolvimento do produto está indo na direção certa, permitindo que sejam realizados ajustes quando este não é o caso.

Diretriz 2: Utilizar *frameworks open source* e/ou com grande adoção no mercado.

O uso de *frameworks* é de extrema importância para uma *startup*, principalmente para aquelas que estão em fase inicial, pois permite que mais tempo seja investido no desenvolvimento das *features* do produto, já que facilitam o trabalho dos desenvolvedores ao providenciar uma estrutura já pronta. Isto permite que o produto seja lançado no mercado mais cedo, permitindo que o *feedback* dos usuários seja recebido imediatamente e assim as mudanças necessárias podem ser implementadas antes que o produto falhe.

Diretriz 3: Usar *backlog* em conjunto com o Kanban independente do método ágil utilizado.

O Kanban é uma técnica versátil que pode ser utilizada em qualquer ambiente, independentemente da metodologia ágil que for adotada. Seu uso é recomendado independente

do estágio de maturidade do processo de desenvolvimento da *startup*, porém, no início do processo de adoção de uma nova metodologia pode ser ainda mais importante, pois permite que a equipe identifique de maneira fácil o status das tarefas e auxilia na manutenção de um fluxo contínuo do processo de desenvolvimento, acelerando o ciclo de produção.

Diretriz 4: Estabelecer uma rotina de testes.

Os testes são de suma importância pois garantem a qualidade do produto, permitindo que o melhor produto possível seja entregue ao usuário, aumentando as suas chances de aceitação. A rotina de testes deve acontecer de acordo com as necessidades do produto, mas também deve respeitar as capacidades da *startup*. Se possível, definir uma rotina de testes automatizados é a melhor opção, pois evita que muitos recursos sejam alocados na execução dos testes, principalmente se tratando dos testes que não dependam da interação dos usuários.

Diretriz 5: Validar a ideia antes de começar a desenvolver.

É natural que a partir do momento que a ideia surge, exista um desejo de começar a desenvolvê-la imediatamente. Dessa forma, é comum que a ideia não seja validada a princípio e que o produto comece a ser desenvolvido sem que se saiba se haverá demanda no mercado para o mesmo. Sendo assim, é essencial para as *startups* que a ideia seja validada antes, evitando que tempo e dinheiro sejam desperdiçados no desenvolvimento de um produto que não irá vender.

Diretriz 6: Construir um MVP.

A construção de um MVP permite testar o produto com o menor investimento possível. Por meio desta prática é possível verificar se o produto cumpre a função para qual foi projetado ao desenvolver apenas as funcionalidades essenciais e minimamente necessárias para a sua operação. O MVP foca em minimizar o tempo para a aprendizagem do conjunto de características que compõem as requisitos do cliente.

Diretriz 7: Utilizar uma ferramenta de gerenciamento de versão descentralizada.

O controle de versão descentralizado permite acesso de primeira classe a todos os desenvolvedores, isto é, não é necessário ser um contribuidor direto para ter acesso as versões.

Atomicidade nas alterações e uma maneira automática de fundi-las também são vantagens que ajudam a manter a consistência do projeto, além de prevenir a ocorrência de erros e de código desatualizado em alguma das estações de trabalho.

Diretriz 8: Priorizar a terceirização da infraestrutura.

Grande parte das *startups* de software precisam de algum tipo de infraestrutura para construir e, principalmente, para distribuir seu produto. Isto inclui a necessidade de servidores altamente disponíveis, capacidade de armazenamento de informação e velocidade de acesso. Porém, até mesmo pela falta de recursos, proporcionar este tipo de infraestrutura é muito difícil para *startups*, devido a complexidade e custos envolvidos. Além disso, há a necessidade de pessoas capacitadas para gerenciar esta complexidade, aumentando ainda mais os custos.

Diretriz 9: Modularizar a estrutura do produto.

Conforme o produto evolui é necessário remover ou adicionar *features*. Essas mudanças devem ocorrer rapidamente, de forma a não atrasar o desenvolvimento do produto, porém, se a estrutura do produto não foi construída com essas mudanças em mente, as alterações podem se tornar complexas. Em *startups*, a adição e a remoção de *features* ocorrem frequentemente, visto que o produto está em constante mudança, modularizar a estrutura do produto facilita que estas ações ocorram, pois as *features* se tornam independentes.

Diretriz 10: Utilizar estilos de codificação.

Em *startups*, conforme o produto escala para atender novos clientes e compreender novas funcionalidades, muitas vezes há a necessidade de que novos desenvolvedores sejam inseridos no projeto. Isto traz problemas em relação a leitura e a compreensão do código, uma vez que estes novos desenvolvedores não estão familiarizados com o projeto, implicando em uma constante necessidade de reescrita de código, que por sua vez aumenta o *time-to-market*. Dessa forma, o uso de estilos de codificação é uma estratégia viável para evitar a reescrita de código.

Diretriz 11: Usar prototipagem evolucionária durante o desenvolvimento do produto.

A prototipagem é uma forma de experimentação durante o desenvolvimento do produto e um ponto importante para a inovação. A inovação cria oportunidades para o crescimento de mercado, diferenciação e vantagem competitiva. A prototipagem é uma estratégia viável para *startups*, visto que os custos para sua realização são baixos e não demanda tecnologia de ponta.

Diretriz 12: Empoderar a equipe de desenvolvimento.

A equipe de desenvolvimento deve estar apta para absorver e aprender por meio de tentativa e erro de forma rápida o suficiente para se adaptar às novas práticas emergentes no mercado. Empoderar a equipe é importante, pois diminui a interdependência e também a dependência dos membros com cargos superiores, deixando a estrutura do time mais horizontal, onde decisões importantes podem ser tomadas por qualquer um dos integrantes da equipe.

Diretriz 13: Realizar ciclos de *feedback* com *stakeholders* e clientes.

O contato direto com os usuários permite que os desenvolvedores obtenham um melhor compreendimento em relação as ideias dos mesmos, fazendo com que os desenvolvedores foquem nos aspectos certos imediatamente. Além disto, ao apresentar novas *features* aos usuários para que possam testar e experimentar, permite receber um *feedback* imediato quanto a satisfação dos mesmos em relação as soluções desenvolvidas. Para os desenvolvedores isto é importante pois valida o seu trabalho e os dá motivação.

Diretriz 14: Realizar entregas frequentes/*releases* curtos.

Para manter a sua competitividade, organizações intensivas de software, como *startups*, precisam entregar *features* valiosas de forma muito rápida para o seus clientes, muitas vezes próximo de ser em tempo real. Ao liberar novas versões constantemente é possível obter *feedbacks* mais precisos e tomar medidas mais rápidas em relação aos mesmos, gerando um ciclo de aprendizagem contínua e de melhoramento do produto.

Diretriz 15: Gerenciar o conhecimento do projeto de software.

Armazenar o conhecimento de software permite manter um histórico do desenvolvimento do produto, possibilitando visualizar o estado das suas *features* através do tempo. A

possibilidade de consultar as decisões que foram tomadas no passado é importante até mesmo para decisões futuras, pois podem indicar o que se esperava do produto e em qual direção o mesmo se encontra atualmente. A partir disto ajustes podem ser realizados para colocar o produto de volta ao seu caminho original ou para a inserção de melhorias, evitando que a startup falhe por não atender as necessidades e demandas de seus clientes.

Diretriz 16: Minimizar a dispersão das informações do projeto.

Armazenar as informações do projeto em múltiplas ferramentas provoca resultados imprecisos e dificulta uma visibilidade em tempo real confortável. Espalhar as informações por várias ferramentas diferentes dificulta o trabalho da equipe de desenvolvimento, que terá que realizar várias consultas para achar a informação necessária. Isso pode até mesmo resultar em um atraso no ciclo de desenvolvimento, visto que este é um tempo precioso que seria investido na construção e no aperfeiçoamento do produto.

Diretriz 17: Utilizar ferramentas de coleta de comportamento dos usuários.

Coletar informações a respeito do comportamento do usuário durante a utilização do produto pode fornecer métricas importantes quanto ao desempenho do software, permitindo analisar se as expectativas em relação ao seu funcionamento estão sendo cumpridas. Além disso, as métricas coletadas podem ser um forte indicador para a implementação de novas *features* e de possíveis melhorias em funcionalidades já existentes. Estas métricas podem ajudar a entender de onde os usuários vieram e como eles interagiram com o produto.

Diretriz 18: Praticar *code review*.

O principal benefício do *code review* é o seu auxílio na detecção de defeitos, sendo assim o uso da técnica proporciona uma maior qualidade no estado do produto. Em *startups*, onde recursos como tempo e financiamento são escassos, a prática é ainda mais importante pois evita que sejam necessárias a realização de sessões de retrabalho, muito mais demoradas e dispendiosas do que curtas sessões de análise de código.

Diretriz 19: Gerenciar a dívida técnica.

A dívida técnica é uma metáfora em relação ao comprometimento técnico em virtude de um benefício a curto prazo mas que poderá custar caro à saúde do software no longo

prazo. O gerenciamento da dívida técnica busca identificar e monitorar os pontos de dívida técnica inseridos no projeto de forma que os mesmos estejam explícitos e sejam pagos em um momento oportuno, evitando transtornos maiores.

Diretriz 20: Utilizar indicadores chave de desempenho.

O uso de indicadores chave de desempenho são uma forma de mensurar o desempenho de alguma atividade em particular, visando enxergar as demandas do cliente. Por meio destes indicadores é possível determinar com que eficiência a *startup* está alcançando os pontos chave para o sucesso do produto. O uso de indicadores chave de desempenho é importante pois os mesmos auxiliam na tomada de decisão, permitindo identificar demandas específicas dos clientes que não seriam descobertas por meio de outras ferramentas.

Diretriz 21: Estabelecer um processo de gerência de requisitos do produto.

Em *startups*, devido ao contexto de incertezas em que estão inseridas, mais do que em organizações tradicionais, os requisitos do produto muitas vezes não são claros, sendo necessário um grande esforço para que estas *startups* saibam o que devem desenvolver. Por estarem envolvidas em um ambiente de mudanças frequentes os requisitos de software são difíceis de extrair e mudam constantemente. Sendo assim, é grande a importância de gerenciar os requisitos nestas organizações, visando construir um produto que atenda as reais necessidades do cliente e evitar que a *startup* falhe pois não foi capaz de entendê-las.

Diretriz 22: Estabelecer uma política de manutenção do produto.

A manutenção do produto garante que o mesmo continuará funcionando, sendo que para *startups* isto é ainda mais importante, pois significa reter um usuário que foi difícil de adquirir. Para preservar a integridade do produto no decorrer das mudanças é preciso definir um processo onde as modificações possam ser registradas e monitoradas, onde o impacto das alterações propostas possa ser determinado e onde o código e outros artefatos de software possam ser modificados e testados.

Diretriz 23: Analisar o desempenho do software.

A diminuição do desempenho do software após atualizações no software é um problema para qualquer organização que produza software, porém, no caso de *startups* pode

ser um problema ainda maior devido ao ciclo constante de mudanças. A diminuição do desempenho do software tem efeitos negativos como o aumento dos custos com a manutenção e a insatisfação do cliente, podendo até mesmo levar a grandes perdas financeiras. Analisar o desempenho do software permite identificar essas diminuições assim que as atualizações ocorram e tomar as medidas necessárias para corrigi-las.

Diretriz 24: Introduzir práticas de suporte à melhoria de processo.

A melhoria de processo pode ser um diferencial na aquisição de recursos, visto que organizações de financiamento, como aceleradoras, investidores anjo e etc., provavelmente darão prioridade a organizações que possuam um processo melhor definido e de maior qualidade. A melhoria de processo auxilia a organização a visualizar as suas metas de negócio, sendo que isso será transferido para a qualidade do software, que estará melhor alinhado com os objetivos da *startup* e consequentemente irá atender melhor as necessidades dos usuários.

J

Refinamento - Carta de Apresentação do Painel com Especialistas

Prezado,

Meu nome é Bruno H. Cavalcante e sou mestrando em Ciência da Computação no Programa de Pós-Graduação em Ciência da Computação (PCC) da Universidade Estadual de Maringá (UEM) sob a orientação da Prof. Dr. Gislaine Camila Lapasini Leal e coorientação do Prof. Dr. Renato Balancieri. Venho, por meio desta, solicitar sua participação para a condução de uma pesquisa de campo como parte da dissertação.

A pesquisa será realizada por meio de um questionário *on-line* visando obter sua opinião sobre as diretrizes para o desenvolvimento de software em *startups* que estão sendo propostas na dissertação. As diretrizes são apresentadas por meio de um título, que apresenta um resumo da diretriz e propõe o que deve ser feito, e uma descrição, que apresenta maiores detalhes sobre a mesma.

As informações prestadas serão tratadas de forma a preservar a privacidade dos participantes. Nenhuma informação será publicada de forma individualizada.

Aguardamos o seu retorno e agradecemos antecipadamente pela colaboração.

Atenciosamente,
Bruno Henrique Cavalcante

Programa de Pós-Graduação em Ciência da Computação (PCC)
Universidade Estadual de Maringá

K

Refinamento - Caracterização do Respondente do Painel com Especialistas

1) Tempo de experiência com pesquisa em *startups*:

( ) Menos do que 1 ano

( ) Entre 1 e 3 anos

( ) Entre 3 e 5 anos

( ) Mais do que 5 anos

2) Qual área específica dentro do contexto de *startups* é realizada a sua pesquisa?

_____

3) Grau de formação

( ) Curso Técnico

( ) Graduando

( ) Graduado

( ) Especialista

( ) Mestrando

( ) Mestre

( ) Doutorando

( ) Doutor

L

Refinamento - Survey Confirmatório e Questionário Painel com Especialistas

Diretriz 1: Utilizar uma metodologia ágil para guiar o desenvolvimento de software.

As metodologias ágeis são utilizadas pela maioria das *startups* pois permitem um ciclo de desenvolvimento mais rápido. Estes métodos permitem que o produto seja desenvolvido em pequenas iterações, e isto, se alinhado a uma prática de entregas frequentes, possibilita que sejam obtidos *feedbacks* constantes dos usuários. Estes *feedbacks* são importantes pois ajudam a visualizar se o desenvolvimento do produto está indo na direção certa, permitindo que sejam realizados ajustes quando este não é o caso.

( ) Discordo totalmente

( ) Discordo parcialmente

( ) Indiferente

( ) Concordo parcialmente

( ) Concordo totalmente

Diretriz 2: Utilizar *frameworks open source* e/ou com grande adoção no mercado.

O uso de *frameworks* é de extrema importância para uma *startup*, principalmente para aquelas que estão em fase inicial, pois permite que mais tempo seja investido no desenvolvimento das *features* do produto, já que facilitam o trabalho dos desenvolvedores ao

providenciar uma estrutura já pronta. Isto permite que o produto seja lançado no mercado mais cedo, permitindo que o *feedback* dos usuários seja recebido imediatamente e assim as mudanças necessárias podem ser implementadas antes que o produto falhe.

( ) Discordo totalmente

( ) Discordo parcialmente

( ) Indiferente

( ) Concordo parcialmente

( ) Concordo totalmente

Diretriz 3: Usar *backlog* em conjunto com o Kanban independente do método ágil utilizado.

O Kanban é uma técnica versátil que pode ser utilizada em qualquer ambiente, independentemente da metodologia ágil que for adotada. Seu uso é recomendado independente do estágio de maturidade do processo de desenvolvimento da *startup*, porém, no início do processo de adoção de uma nova metodologia pode ser ainda mais importante, pois permite que a equipe identifique de maneira fácil o status das tarefas e auxilia na manutenção de um fluxo contínuo do processo de desenvolvimento, acelerando o ciclo de produção.

( ) Discordo totalmente

( ) Discordo parcialmente

( ) Indiferente

( ) Concordo parcialmente

( ) Concordo totalmente

Diretriz 4: Estabelecer uma rotina de testes.

Os testes são de suma importância pois garantem a qualidade do produto, permitindo que o melhor produto possível seja entregue ao usuário, aumentando as suas chances de aceitação. A rotina de testes deve acontecer de acordo com as necessidades do produto, mas também deve respeitar as capacidades da *startup*. Se possível, definir uma rotina de testes automatizados é a melhor opção, pois evita que muitos recursos sejam alocados na execução dos testes, principalmente se tratando dos testes que não dependam da interação dos usuários.

( ) Discordo totalmente

( ) Discordo parcialmente

( ) Indiferente

( ) Concordo parcialmente

( ) Concordo totalmente

Diretriz 5: Validar a ideia antes de começar a desenvolver.

É natural que a partir do momento que a ideia surge, exista um desejo de começar a desenvolvê-la imediatamente. Dessa forma, é comum que a ideia não seja validada a princípio e que o produto comece a ser desenvolvido sem que se saiba se haverá demanda no mercado para o mesmo. Sendo assim, é essencial para as *startups* que a ideia seja validada antes, evitando que tempo e dinheiro sejam desperdiçados no desenvolvimento de um produto que não irá vender.

( ) Discordo totalmente

( ) Discordo parcialmente

( ) Indiferente

( ) Concordo parcialmente

( ) Concordo totalmente

Diretriz 6: Construir um MVP.

A construção de um MVP permite testar o produto com o menor investimento possível. Por meio desta prática é possível verificar se o produto cumpre a função para qual foi projetado ao desenvolver apenas as funcionalidades essenciais e minimamente necessárias para a sua operação. O MVP foca em minimizar o tempo para a aprendizagem do conjunto de características que compõem as requisitos do cliente.

( ) Discordo totalmente

( ) Discordo parcialmente

( ) Indiferente

( ) Concordo parcialmente

( ) Concordo totalmente

Diretriz 7: Utilizar uma ferramenta de gerenciamento de versão descentralizada.

O controle de versão descentralizado permite acesso de primeira classe a todos os desenvolvedores, isto é, não é necessário ser um contribuidor direto para ter acesso as versões. Atomicidade nas alterações e uma maneira automática de fundi-las também são vantagens que ajudam a manter a consistência do projeto, além de prevenir a ocorrência de erros e de código desatualizado em alguma das estações de trabalho.

( ) Discordo totalmente

( ) Discordo parcialmente

( ) Indiferente

( ) Concordo parcialmente

( ) Concordo totalmente

Diretriz 8: Priorizar a terceirização da infraestrutura.

Grande parte das *startups* de software precisam de algum tipo de infraestrutura para construir e, principalmente, para distribuir seu produto. Isto inclui a necessidade de servidores altamente disponíveis, capacidade de armazenamento de informação e velocidade de acesso. Porém, até mesmo pela falta de recursos, proporcionar este tipo de infraestrutura é muito difícil para *startups*, devido a complexidade e custos envolvidos. Além disso, há a necessidade de pessoas capacitadas para gerenciar esta complexidade, aumentando ainda mais os custos.

( ) Discordo totalmente

( ) Discordo parcialmente

( ) Indiferente

( ) Concordo parcialmente

( ) Concordo totalmente

Diretriz 9: Modularizar a estrutura do produto.

Conforme o produto evolui é necessário remover ou adicionar *features*. Essas mudanças devem ocorrer rapidamente, de forma a não atrasar o desenvolvimento do produto, porém, se a estrutura do produto não foi construída com essas mudanças em mente, as alterações podem se tornar complexas. Em *startups*, a adição e a remoção de *features* ocorrem frequentemente, visto que o produto está em constante mudança, modularizar a estrutura do produto facilita que estas ações ocorram, pois as *features* se tornam independentes.

( ) Discordo totalmente

( ) Discordo parcialmente

( ) Indiferente

( ) Concordo parcialmente

( ) Concordo totalmente

Diretriz 10: Utilizar estilos de codificação.

Em *startups*, conforme o produto escala para atender novos clientes e compreender novas funcionalidades, muitas vezes há a necessidade de que novos desenvolvedores sejam inseridos no projeto. Isto traz problemas em relação a leitura e a compreensão do código, uma vez que estes novos desenvolvedores não estão familiarizados com o projeto, implicando em uma constante necessidade de reescrita de código, que por sua vez aumenta o *time-to-market*. Dessa forma, o uso de estilos de codificação é uma estratégia viável para evitar a reescrita de código.

( ) Discordo totalmente

( ) Discordo parcialmente

( ) Indiferente

( ) Concordo parcialmente

( ) Concordo totalmente

Diretriz 11: Usar prototipagem evolucionária durante o desenvolvimento do produto.

A prototipagem é uma forma de experimentação durante o desenvolvimento do produto e um ponto importante para a inovação. A inovação cria oportunidades para o crescimento de mercado, diferenciação e vantagem competitiva. A prototipagem é uma estratégia viável para *startups*, visto que os custos para sua realização são baixos e não demanda tecnologia de ponta.

( ) Discordo totalmente

( ) Discordo parcialmente

( ) Indiferente

( ) Concordo parcialmente

( ) Concordo totalmente

Diretriz 12: Empoderar a equipe de desenvolvimento.

A equipe de desenvolvimento deve estar apta para absorver e aprender por meio de tentativa e erro de forma rápida o suficiente para se adaptar às novas práticas emergentes no mercado. Empoderar a equipe é importante, pois diminui a interdependência e também a dependência dos membros com cargos superiores, deixando a estrutura do time mais horizontal, onde decisões importantes podem ser tomadas por qualquer um dos integrantes da equipe.

( ) Discordo totalmente

( ) Discordo parcialmente

( ) Indiferente

( ) Concordo parcialmente

( ) Concordo totalmente

Diretriz 13: Realizar ciclos de *feedback* com *stakeholders* e clientes.

O contato direto com os usuários permite que os desenvolvedores obtenham um melhor compreendimento em relação as ideias dos mesmos, fazendo com que os desenvolvedores

foquem nos aspectos certos imediatamente. Além disto, ao apresentar novas *features* aos usuários para que possam testar e experimentar, permite receber um *feedback* imediato quanto a satisfação dos mesmos em relação as soluções desenvolvidas. Para os desenvolvedores isto é importante pois valida o seu trabalho e os dá motivação.

( ) Discordo totalmente

( ) Discordo parcialmente

( ) Indiferente

( ) Concordo parcialmente

( ) Concordo totalmente

Diretriz 14: Realizar entregas frequentes/*releases* curtos.

Para manter a sua competitividade, organizações intensivas de software, como *startups*, precisam entregar *features* valiosas de forma muito rápida para o seus clientes, muitas vezes próximo de ser em tempo real. Ao liberar novas versões constantemente é possível obter *feedbacks* mais precisos e tomar medidas mais rápidas em relação aos mesmos, gerando um ciclo de aprendizagem contínua e de melhoramento do produto.

( ) Discordo totalmente

( ) Discordo parcialmente

( ) Indiferente

( ) Concordo parcialmente

( ) Concordo totalmente

Diretriz 15: Gerenciar o conhecimento do projeto de software.

Armazenar o conhecimento de software permite manter um histórico do desenvolvimento do produto, possibilitando visualizar o estado das suas *features* através do tempo. A possibilidade de consultar as decisões que foram tomadas no passado é importante até mesmo para decisões futuras, pois podem indicar o que se esperava do produto e em qual direção o mesmo se encontra atualmente. A partir disto ajustes podem ser realizados para colocar o produto de volta ao seu caminho original ou para a inserção de melhorias, evitando que a startup falhe por não atender as necessidades e demandas de seus clientes.

( ) Discordo totalmente

( ) Discordo parcialmente

( ) Indiferente

( ) Concordo parcialmente

( ) Concordo totalmente

Diretriz 16: Minimizar a dispersão das informações do projeto.

Armazenar as informações do projeto em múltiplas ferramentas provoca resultados imprecisos e dificulta uma visibilidade em tempo real confortável. Espalhar as informações por várias ferramentas diferentes dificulta o trabalho da equipe de desenvolvimento, que terá que realizar várias consultas para achar a informação necessária. Isso pode até mesmo resultar em um atraso no ciclo de desenvolvimento, visto que este é um tempo precioso que seria investido na construção e no aperfeiçoamento do produto.

( ) Discordo totalmente

( ) Discordo parcialmente

( ) Indiferente

( ) Concordo parcialmente

( ) Concordo totalmente

Diretriz 17: Utilizar ferramentas de coleta de comportamento dos usuários.

Coletar informações a respeito do comportamento do usuário durante a utilização do produto pode fornecer métricas importantes quanto ao desempenho do software, permitindo analisar se as expectativas em relação ao seu funcionamento estão sendo cumpridas. Além disso, as métricas coletadas podem ser um forte indicador para a implementação de novas *features* e de possíveis melhorias em funcionalidades já existentes. Estas métricas podem ajudar a entender de onde os usuários vieram e como eles interagiram com o produto.

( ) Discordo totalmente

( ) Discordo parcialmente

( ) Indiferente

( ) Concordo parcialmente

( ) Concordo totalmente

Diretriz 18: Praticar *code review*.

O principal benefício do *code review* é o seu auxílio na detecção de defeitos, sendo assim o uso da técnica proporciona uma maior qualidade no estado do produto. Em *startups*, onde recursos como tempo e financiamento são escassos, a prática é ainda mais importante pois evita que sejam necessárias a realização de sessões de retrabalho, muito mais demoradas e dispendiosas do que curtas sessões de análise de código.

( ) Discordo totalmente

( ) Discordo parcialmente

( ) Indiferente

( ) Concordo parcialmente

( ) Concordo totalmente

Diretriz 19: Gerenciar a dívida técnica.

A dívida técnica é uma metáfora em relação ao comprometimento técnico em virtude de um benefício a curto prazo mas que poderá custar caro à saúde do software no longo prazo. O gerenciamento da dívida técnica busca identificar e monitorar os pontos de dívida técnica inseridos no projeto de forma que os mesmos estejam explícitos e sejam pagos em um momento oportuno, evitando transtornos maiores.

( ) Discordo totalmente

( ) Discordo parcialmente

( ) Indiferente

( ) Concordo parcialmente

( ) Concordo totalmente

Diretriz 20: Utilizar indicadores chave de desempenho.

O uso de indicadores chave de desempenho são uma forma de mensurar o desempenho de alguma atividade em particular, visando enxergar as demandas do cliente. Por meio destes indicadores é possível determinar com que eficiência a *startup* está alcançando os pontos chave para o sucesso do produto. O uso de indicadores chave de desempenho é importante pois os mesmos auxiliam na tomada de decisão, permitindo identificar demandas específicas dos clientes que não seriam descobertas por meio de outras ferramentas.

( ) Discordo totalmente

( ) Discordo parcialmente

( ) Indiferente

( ) Concordo parcialmente

( ) Concordo totalmente

Diretriz 21: Estabelecer um processo de gerência de requisitos do produto.

Em *startups*, devido ao contexto de incertezas em que estão inseridas, mais do que em organizações tradicionais, os requisitos do produto muitas vezes não são claros, sendo necessário um grande esforço para que estas *startups* saibam o que devem desenvolver. Por estarem envolvidas em um ambiente de mudanças frequentes os requisitos de software são difíceis de extrair e mudam constantemente. Sendo assim, é grande a importância de gerenciar os requisitos nestas organizações, visando construir um produto que atenda as reais necessidades do cliente e evitar que a *startup* falhe pois não foi capaz de entendê-las.

( ) Discordo totalmente

( ) Discordo parcialmente

( ) Indiferente

( ) Concordo parcialmente

( ) Concordo totalmente

Diretriz 22: Estabelecer uma política de manutenção do produto.

A manutenção do produto garante que o mesmo continuará funcionando, sendo que para *startups* isto é ainda mais importante, pois significa reter um usuário que foi difícil de adquirir. Para preservar a integridade do produto no decorrer das mudanças é preciso definir um processo onde as modificações possam ser registradas e monitoradas, onde o impacto das alterações propostas possa ser determinado e onde o código e outros artefatos de software possam ser modificados e testados.

( ) Discordo totalmente

( ) Discordo parcialmente

( ) Indiferente

( ) Concordo parcialmente

( ) Concordo totalmente

Diretriz 23: Analisar o desempenho do software.

A diminuição do desempenho do software após atualizações no software é um problema para qualquer organização que produza software, porém, no caso de *startups* pode ser um problema ainda maior devido ao ciclo constante de mudanças. A diminuição do desempenho do software tem efeitos negativos como o aumento dos custos com a manutenção e a insatisfação do cliente, podendo até mesmo levar a grandes perdas financeiras. Analisar o desempenho do software permite identificar essas diminuições assim que as atualizações ocorram e tomar as medidas necessárias para corrigi-las.

( ) Discordo totalmente

( ) Discordo parcialmente

( ) Indiferente

( ) Concordo parcialmente

( ) Concordo totalmente

Diretriz 24: Introduzir práticas de suporte à melhoria de processo.

A melhoria de processo pode ser um diferencial na aquisição de recursos, visto que organizações de financiamento, como aceleradoras, investidores anjo e etc., provavelmente darão prioridade a organizações que possuam um processo melhor definido e de maior qualidade. A melhoria de processo auxilia a organização a visualizar as suas metas de negócio, sendo que isso será transferido para a qualidade do software, que estará melhor alinhado com os objetivos da *startup* e consequentemente irá atender melhor as necessidades dos usuários.

( ) Discordo totalmente

( ) Discordo parcialmente

( ) Indiferente

( ) Concordo parcialmente

( ) Concordo totalmente