UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
COORDENADORIA DO CURSO DE ENGENHARIA DE SOFTWARE

JUAN FELIPE KRASSMANN

O CENÁRIO DAS PRÁTICAS ÁGEIS NAS EMPRESAS DE SOFTWARE DO SUDOESTE PARANAENSE

TRABALHO DE CONCLUSÃO DE CURSO

DOIS VIZINHOS

2019

JUAN FELIPE KRASSMANN

O CENÁRIO DAS PRÁTICAS ÁGEIS NAS EMPRESAS DE SOFTWARE DO SUDOESTE PARANAENSE

Trabalho de Conclusão de Curso apresentado como requisito parcial à obtenção do título de Bacharel em Engenharia de Software, da Universidade Tecnológica Federal do Paraná.

Orientador: Prof. Dr. Rafael A. P. de Oliveira

DOIS VIZINHOS

2019

TERMO DE APROVAÇÃO

O Cenário das Práticas Ágeis nas Empresas de Software do Sudoeste Paranaense

por

Juan Felipe Krassmann

Este Trabalho de Conclusão de Curso foi apresentado em 05 de Julho de 2019 como requisito parcial para a obtenção do título de Bacharel em Engenharia de Software. O(a) candidato(a) foi arguido(a) pela Banca Examinadora composta pelos professores abaixo assinados. Após deliberação, a Banca Examinadora considerou o trabalho aprovado.

Rafael Alves Paes de Oliveira
Presidente da Banca

Rene Pomilio de Oliveira
Membro Titular

Simone de Sousa Borges
Membro Titular

* A Folha de Aprovação assinada encontra-se na Coordenação do Curso

Dedico este trabalho a Deus, à minha família e a meus amigos.

AGRADECIMENTOS

Agradeço primeiramente a Deus que permitiu a minha jornada até aqui.

Minha intensa e eterna gratidão ao apoio de meus pais e de toda minha família, que nunca mediram esforços para me auxiliar no que precisei.

Agradeço fortemente aos meus amigos pelo apoio e amizade, e também aos colegas de universidade pelo companheirismo acadêmico.

Agradeço a todos os professores que fizeram parte da minha graduação, em especial ao meu orientador Rafael Alves Paes de Oliveira que muito contribuiu para o desenvolvimento desse trabalho.

"Não andem ansiosos por coisa alguma, mas em tudo, pela oração e súplicas, e com ação de graças, apresentem seus pedidos a Deus."

Filipenses 4:6

RESUMO

KRASSMANN, Juan F. O CENÁRIO DAS PRÁTICAS ÁGEIS NAS EMPRESAS DE SOFTWARE DO SUDOESTE PARANAENSE. 109 f. Trabalho de Conclusão de Curso – Coordenadoria do Curso de Engenharia de Software, Universidade Tecnológica Federal do Paraná. Dois Vizinhos, 2019.

Os métodos ágeis são resultados de uma evolução dos processos de desenvolvimento de software como alternativa aos métodos tradicionais pesados. Entretanto, um problema latente de adoção de métodos ágeis na prática é o fato que em cenários reais de desenvolvimento não há viabilidade para que organizações sigam um padrão sólido e sistematizado. Com isso, o presente estudo se insere no contexto de trabalhos que visam a identificar fenômenos acerca do uso de métodos ágeis para o desenvolvimento de software considerando aspectos regionais e humanos. Com o objetivo geral de identificar e apresentar características acerca do cenário atual de ágeis nas empresas de desenvolvimento de software do sudoeste do Paraná (Brasil), foi elaborada e aplicada uma *survey* exploratória em 25 organizações da região. Os resultados obtidos foram analisados e discutidos. Foram descobertos vários fenômenos presentes no cenário, bem como problemas e obstáculos que as empresas encontram. Algumas características identificadas envolvem imaturidade por parte dos profissionais em relação a ágeis, empresas com tamanhos variados porém nenhuma com mais de 500 funcionários, poucas equipes em cada organização com poucos membros em cada equipe, todos os 12 princípios praticados na maioria das organizações, a metodologia *Scrum* sendo a utilizada, cliente longe do desenvolvimento, muitas práticas ágeis realizadas, administração apoiando a equipe, identificação de muitas ferramentas de apoio à comunicação, incentivos a *feedback* rápido e prazos e entregas sendo um problema latente das organizações.

Palavras-chave: Engenharia de Software, Processos de Software, Métodos Ágeis de Desenvolvimento

ABSTRACT

KRASSMANN, Juan F. THE SCENARIO OF AGILE PRACTICES IN SOFTWARE COMPANIES IN PARANÁ SOUTHWESTERN. 109 f. Trabalho de Conclusão de Curso – Coordenadoria do Curso de Engenharia de Software, Universidade Tecnológica Federal do Paraná. Dois Vizinhos, 2019.

Agile methods are results of an evolution of software development processes as an alternative to traditional heavy methods. However, a latent problem of adopting agile methods in practice is the fact that in real development scenarios there is no viability for organizations to follow a solid and systematized pattern. Therefore, the present study is part of a series of studies aimed to identifying phenomena about the use of agile methods for software development considering regional and human aspects. With the overall objective of identifying and presenting characteristics about the current agile scenario in the software development companies of southwestern Paraná (Brazil), an exploratory survey was elaborated and applied in 25 organizations of the region. The results obtained were analyzed and discussed. Several phenomena were present in the scenario, as well as problems and obstacles that companies encounter. Some identified characteristics involve immaturity by professionals in relation to agile, companies with varied sizes but none with more than 500 employees, few teams in each organization with few members in each team, all 12 principles practiced in most organizations, methodology Scrum being the most used, customer far from development, many agile practices performed, administration supporting the team, identification of many communication support tools, incentives to fast feedback and deadlines and deliveries being a latent problem of organizations.

Keywords: Software Engineering, Software Process, Development Agile Methods

LISTA DE FIGURAS

LISTA DE SIGLAS

APL	Arranjo Produtivo Local
AUP	*Agile Unified Process* - Processo Ágil Unificado
CMMI	*Capability Maturity Model Integration* - Modelo Integrado de Maturidade em Capacitação
CP	*Chief Programmers* - Programadores Principais
DevOps	*Development and Operations* - Desenvolvimento e Operações
DSDM	*Dynamic System Development Method* - Metodologia de Desenvolvimento de Sistemas Dinâmicos
ES	Engenharia de Software
FDD	*Feature Driven Development* - Desenvolvimento Dirigido à Características
GQM	*Goals, Questions and Metrics* - Objetivos, Questões e Métricas
IEEE	Instituto de Engenheiros Eletricistas e Eletrônicos
JIT	*Just In Time* - Na Hora Certa
NTI	Núcleo de Tecnologia da Informação
PO	*Product Owner* - Dono do Produto
QP	Questão de Pesquisa
TDD	*Test Driven Development* - Desenvolvimento Dirigido à Testes
TI	Tecnologia da Informação
UML	*Unified Modeling Language* - Linguagem de Modelagem Unificada
UP	*Unified Process* - Processo Unificado
UTFPR	Universidade Tecnológica Federal do Paraná
XP	*Extreme Programming* - Programação Extrema

SUMÁRIO

1 INTRODUÇÃO

Este estudo consiste de um trabalho de conclusão do curso de Bacharel em Engenharia de Software e visa a fazer um levantamento das práticas ágeis de *software houses* do sudoeste paranaense.

1.1 CONTEXTO

Segundo Schwaber (2018), Engenharia de Software (ES) é uma disciplina da engenharia tradicional cujo foco está em sistematizar todos os aspectos da produção de software (SOMMERVILLE, 2011). Os estudos ocorrem desde os estágios iniciais da especificação do sistema até sua manutenção, quando o sistema já está sendo utilizado e mantido/atualizado. O Instituto de Engenheiros Eletricistas e Eletrônicos (IEEE) define Engenharia de Software como sendo "a aplicação de uma abordagem sistemática, disciplinada e quantificável para o desenvolvimento, operação e manutenção do software" (PRESSMAN, 2011). Pressman (2011) complementa que a definição de ES em torno de uma abordagem "sistemática, disciplinada e quantificável" a ser aplicada por uma equipe de desenvolvimento pode ter uma implementação prática complexa. Adicionalmente, é possível afirmar que além da disciplina, processos devem ter adaptabilidade e agilidade (PRESSMAN, 2011).

Dentre os principais recursos da Engenharia de Software, destacam-se os processos de desenvolvimento. Nesse contexto, processo de software é um conjunto de passos previsíveis, um roteiro que apoia a criação de um produto de software de alta qualidade e dentro do prazo estabelecido (PRESSMAN, 2011). A principal importância em se ter um processo bem definido é o controle obtido a partir dele, reduzindo as probabilidades de se ter cenários caóticos de desenvolvimento.

Diante da competitividade do setor produtivo de desenvolvimento de software, metodologias ágeis de desenvolvimento surgiram para propor a criação de software de qualidade de modo sistemático e produtivo, introduzindo uma série de práticas que

dinamizaram o processo de construção de software (HODA; SALLEH; GRUNDY, 2018). Os métodos ágeis são estratégias de desenvolvimento que promovem a implementação de projetos de software com pequenos incrementos (PRESSMAN, 2011). Em essência, os métodos ágeis se desenvolveram em um esforço para sanar fraquezas reais e perceptíveis da ES convencional (PRESSMAN, 2011).

Esforços de pesquisa na área de Processos, e mais objetivamente acerca das metodologias ágeis, são indispensáveis para o progresso científico desse setor (HODA; SALLEH; GRUNDY, 2018). É possível realizar estudos e investigações identificando tendências em relação ao desenvolvimento de software em determinados ambientes e assim propor estratégias e intervenções para facilitar a aplicação dessas metodologias em ambientes reais de desenvolvimento de software. Por meio de tais estudos e investigações, organizações podem se apoiar de modo a melhorar cenários defasados, encontrar recursos para tomadas de decisão e obter a referência de um cenário global no qual está inserido.

Processos tradicionais de desenvolvimento estabelecem etapas bem definidas que devem ser seguidas no processo de desenvolvimento e manutenção de software (SOMMERVILLE, 2011). Apesar de fornecerem um controle extremo nas atividades de desenvolvimento, esse paradigma de processo dificulta as mudanças no produto de software (PRESSMAN, 2011). Na economia moderna é difícil, ou impossível, prever a evolução de um sistema computacional. Regras de negócio, leis e ambições das organizações mudam rapidamente, levando à necessidade constante de evolução nos sistemas computacionais. Assim, pode-se afirmar que as condições de mercado estão em constante mudança, as necessidades do cliente se alteram e novas ameaças surgem sem aviso (PRESSMAN, 2011). Diante disso, as metodologias ágeis buscam trabalhar melhor com esses obstáculos, respondendo rapidamente às mudanças naturais em regras de negócios e correções de defeitos.

Um problema latente de adoção de métodos ágeis na prática é o fato que em cenários reais de desenvolvimento não há viabilidade para que organizações sigam um padrão sólido e sistematizado no contexto prático. Desse modo, empresas de desenvolvimento de software estão sempre adaptando os ágeis para o seu próprio contexto (RAMIREZ-MORA; OKTABA, 2017). Em diversas organizações, é comum notar que papéis e práticas ágeis são constantemente adaptadas para contextos particulares (HODA; SALLEH; GRUNDY, 2018).

Visando a contribuir com esse cenário, o presente trabalho de conclusão de curso consiste de um esforço de pesquisa que analisa o cenário ágil no sudoeste paranaense. A

partir desse cenário são identificadas tendências e perspectivas de ambientes de desenvolvimento que adotem práticas ágeis na região sudoeste do estado do Paraná. Discussões sobre a maturidade do cenário ágil das empresas também são apresentados.

Diante do cenário local do sudoeste paranaense, este estudo promove as seguintes contribuições:

- Um diagnóstico e uma visão geral sobre as práticas ágeis no sudoeste do Paraná;
- Análises sobre resultados qualitativos coletados;
- Uma pesquisa sobre métodos ágeis que pode ser replicada em intervalos determinados de tempo para identificar pontos de melhoria;
- Uma pesquisa sobre métodos ágeis que pode ser replicada em diversas regiões do país; e
- Uma série de discussões sobre perspectivas futuras e possíveis cenários de melhoria dos métodos ágeis das empresas participantes do pesquisa.

Diante do cenário apresentado, destaca-se que o presente estudo se insere no contexto de trabalhos que visam a identificar fenômenos acerca do uso de métodos ágeis para o desenvolvimento de software considerando aspectos regionais e humanos.

1.2 PROBLEMAS DE PESQUISA

Uma série de pesquisas ao longo dos anos aponta que a adaptação de métodos ágeis e suas práticas nas empresas é relativa e muito particular (HODA; SALLEH; GRUNDY, 2018). Essa adaptação geralmente varia de acordo com alguns campos como aspectos organizacionais e processuais, além de aspectos humanos e culturais das pessoas envolvidas no processo de desenvolvimento (LóPEZ-MARTíNEZ et al., 2016). Recursos financeiros disponíveis, recursos humanos disponíveis, capacitação, maturidade, comunicação e área de desenvolvimento são alguns dos fatores que geram dificuldade para se estabelecer um processo adequado.

Adicionalmente, aspectos organizacionais trazem diversas limitações para a adoção das metodologias ágeis. Diversas culturas organizacionais tradicionais são improdutivas para o desenvolvimento de software de modo rápido e não suportam formas de trabalho ágil (BERMEJO et al., 2014; ELORANTA et al., 2013; GANDOMANI et al., 2014).

Existem problemas da gerência, falta de apoio das organizações (ELORANTA et al., 2013; KAPITSAKI; CHRISTOU, 2014) e pressão externa para a utilização de práticas tradicionais (ELORANTA et al., 2013; MELO et al., 2013).

Estudos apontam que pessoas e seus comportamentos configuram um fator crítico no processo de mudança de um método de desenvolvimento tradicional para um método ágil (BOOTLA; ROJANAPORNPUN; MONGKOLNAM, 2015; GANDOMANI et al., 2014). Falta cooperação, comunicação constante com o cliente (MELO et al., 2013; IHME, 2013; KAPITSAKI; CHRISTOU, 2014) e formação profissional (MELO et al., 2013). O tamanho da equipe de desenvolvimento também pode resultar em obstáculos para a efetiva aplicação de um bom processo (KAPITSAKI; CHRISTOU, 2014; IHME, 2013). Muitas rotações de membros da equipe também podem acarretar em cenários problemáticos (IHME, 2013). Ainda, falta de experiência com os métodos ágeis (ELORANTA et al., 2013; MELO et al., 2013), disponibilidade de pessoal treinado (ELORANTA et al., 2013; KAPITSAKI; CHRISTOU, 2014), falta de comunicação efetiva e mal-entendidos (BOOTLA; ROJANAPORNPUN; MONGKOLNAM, 2015), falta de compreensão e conscientização dos valores ágeis (GANDOMANI et al., 2014; BOOTLA; ROJANAPORNPUN; MONGKOLNAM, 2015), treinamento inadequado e disfuncional (GANDOMANI et al., 2014), falta de compromisso com as decisões (MELO et al., 2013) e falta de participação constante com o cliente (BOOTLA; ROJANAPORNPUN; MONGKOLNAM, 2015).

O estudo apresentado por Lorber e Mish (2013), destaca alguns problemas que aparecem nos estágios iniciais da adoção da metodologia ágil. A saber:

- a falta de entrega de histórias do usuário;
- falta de confiança; e
- a definição de horários para reuniões de planejamento, diárias, e retrospectivas podem significar a perda de um tempo precioso para os participantes.

Diante do exposto, observam-se diversos aspectos problemáticos que cercam a adoção de métodos ágeis de modo efetivo. O diagnóstico de cenários ágeis em ecossistemas de desenvolvimento é essencial para a proposição de melhorias. O diagnóstico do cenário de desenvolvimento ágil das *software houses* do sudoeste paranaense é a problemática central tratada pelo presente estudo.

1.3 OBJETIVOS

Nessa seção são descritos os objetivos do trabalho, divididos em objetivo geral e objetivos específicos.

1.3.1 OBJETIVO GERAL

Este trabalho de conclusão de curso tem como objetivo geral identificar e apresentar características sobre o cenário atual das práticas ágeis nas empresas de desenvolvimento de software do sudoeste do Paraná.

1.3.2 OBJETIVOS ESPECÍFICOS

A partir do objetivo geral definido, derivam-se os seguintes objetivos específicos:

- Levantar as principais atividades e práticas ágeis de empresas da região sudoeste paranaense, bem como as metodologias utilizadas;
- Conhecer o grau de prática dos princípios ágeis em equipes de organizações do sudoeste paranaense;
- Identificar potenciais benefícios e limitações na adoção de metodologias ágeis em cenários práticos de desenvolvimento;
- Obter características acerca das estruturas organizacionais de empresas de desenvolvimento que praticam ágeis no sudoeste do Paraná; e
- Identificar o tempo de conhecimento e práticas de ágeis de profissionais que atuam nas organizações da região.

1.4 JUSTIFICATIVA

A literatura é carente de estudos que analisem e revelem os motivos da efetividade do usos de ágeis em regiões pontuais. Em particular na região sudoeste do estado do Paraná, onde diversas empresas de tecnologia estão situadas, a literatura não aponta estudos sistemáticos sobre métodos ágeis. Por isso, em linhas gerais, as empresas não possuem um plano de visão bem especificado para a adoção dos princípios em seu meio empresarial. Com isso, desde a aplicação das ideias ágeis ocorrem confusões e isso é estendido e refletido de forma direta no processo de desenvolvimento.

Esforços de estudos empíricos, *surveys* e outros tipos de pesquisas são de grande valor para mitigar os prejuízos dos cenários supracitados. É de legítima importância identificar, categorizar e evidenciar características de práticas ágeis em nível regional e nacional, sendo assim, o objeto de estudo do presente projeto e justificando a pesquisa realizada.

1.5 ESTRUTURAÇÃO DA MONOGRAFIA

Além da introdução, este trabalho apresenta os seguintes capítulos:

- Capítulo 2: apresenta tópicos sobre as áreas envolvidas no projeto, sendo elas Engenharia de Software, processos de produção de software e o cenário da região onde o estudo está sendo conduzido. Há conceitos, definições e explicações sobre cada item, proporcionando conhecimentos em torno das áreas que envolvem o trabalho;
- Capítulo 3: apresenta uma seção específica sobre os métodos ágeis de desenvolvimento, a história, os valores e princípios, comparativos com outros métodos e detalhamento de algumas metodologias. Conclui-se com uma rápida abordagem sobre DevOps;
- Capítulo 4: é composto pelos estudos empíricos. São descritas as questões de pesquisa, o *design*, os procedimentos e a condução experimental. Todas as fases de planejamento e execução do projeto estão descritas nesse capítulo;
- Capítulo 5: contém os resultados brutos obtidos na pesquisa, as associações desses resultados com as questões de pesquisa, as discussões acerca do que foi realizado, além de perspectivas futuras sobre ágeis e as ameaças à validade do trabalho;
- Capítulo 6: apresenta alguns trabalhos relacionados com o presente estudo e fornece breves descrições do que foi feito em cada um desses; e
- Capítulo 7: finaliza o projeto com considerações finais sobre o que foi realizado em todas as fases de trabalho, além de possíveis trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo tem como objetivo abordar conceitos fundamentais para o completo entendimento do trabalho desenvolvido. É apresentado um referencial teórico abrangendo conceitos da Engenharia de Software e processos de produção tradicionais e processos ágeis. Adicionalmente, o ecossistema de empresas ligadas à Tecnologias da Informação do sudoeste paranaense é apresentado de modo sucinto.

2.1 ENGENHARIA DE SOFTWARE

A engenharia tradicional aumenta a produtividade e qualidade dos produtos finais nos quais é aplicada. São aplicadas teorias consistentes e há uma constante busca por novas soluções para os problemas de acordo com as limitações impostas. A Engenharia de Software é uma disciplina que surgiu da engenharia tradicional e que cujo foco está em todos os aspectos da produção de software. Isso abrange aspectos desde os estágios iniciais do processo de desenvolvimento, quando se especifica um sistema por meio de requisitos, e se estende até a sua manutenção, quando o produto já está esse sendo usado pelo usuário final (SOMMERVILLE, 2011).

Segundo o Instituto de Engenheiros Eletricistas e Eletrônicos (IEEE), Engenharia de Software é a aplicação de uma abordagem sistemática, disciplinada e quantificável no desenvolvimento, na operação e na manutenção de software (PRESSMAN, 2011). ES pode ser concebida como uma ação, realizar algo para se obter melhorias e soluções de software.

Assim como todas as engenharias, a ES consiste de uma tecnologia em camadas, fundamentada em um comprometimento organizacional com a qualidade (PRESSMAN, 2011). A gestão da qualidade promove uma cultura de aperfeiçoamento contínuo de processos e é exatamente essa cultura que leva ao desenvolvimento de abordagens cada vez mais efetivas na Engenharia de Software (PRESSMAN, 2011).

A ES fornece insumos para o desenvolvimento de software de qualidade considerando diferentes aspectos. Tais insumos se baseiam não somente em aspectos técnicos/práticos,

mas também com gerenciamento de projetos, desenvolvimento de ferramentas, métodos e teorias para apoiar as fases de produção de software (PRESSMAN, 2011). Os objetos de estudo da ES apoiam-se nos meios para prover maior eficiência dos sistemas desenvolvidos, trabalhando nos aspectos organizacionais. Adicionalmente, pode-se afirmar que até os aspectos tecnológicos são abordados, fazendo com que os produtos de software evoluam naturalmente e tenham características próprias associadas à solução para as quais foram projetados.

A importância da Engenharia de Software está ligada ao crescimento acelerado de indivíduos e sociedades dependentes dos sistemas de software avançados (SOMMERVILLE, 2011). Diante desse cenário, Sommerville (2011) alerta que, ao se fazer valer dos conceitos de ES, profissionais da área de tecnologia devem ser capazes de produzir sistemas confiáveis de forma agilizada e econômica, favorecendo cenários de evolução/manutenção.

A Figura 1 apresenta as quatro camadas da Engenharia de Software segundo Pressman (2011). Foco na qualidade, processos, métodos e ferramentas formam os pilares e se interagem desde à fase de concepção da solução de software até o fim da vida útil de um sistema.

Figura 1: Camadas da Engenharia de Software
Fonte: Pressman (2011)

Como pode ser observado, a base para a Engenharia de Software é a camada de processos. Tal camada é a liga que mantém as camadas de tecnologia coesas e possibilita o desenvolvimento de software de forma racional e dentro do prazo (PRESSMAN, 2011). Com um processo bem definido, maduro e que apoie mudanças, as tecnologias podem ser trabalhadas e desenvolvidas com clareza, fazendo com que o software em produção atenda os requisitos a partir dos quais ele foi demandado.

Ainda segundo a Figura 1, nota-se a relevância dos métodos. A camada de métodos

fornece uma ampla gama de tarefas, que segundo Pressman (2011) incluem:

- comunicação;
- análise de requisitos;
- modelagem de projeto;
- construção do programa;
- testes; e
- suporte.

Por fim, nota-se que as ferramentas sustentam de forma automatizada ou semi automatizada os processos e métodos. Segundo Pressman (2011), quando as ferramentas são integradas e as informações criadas podem ser utilizadas por outra ferramenta, é estabelecida uma engenharia de software com o auxílio do computador.

Por fim, nota-se que os engenheiros de software devem escolher as melhores formas para realizar o trabalho proposto e chegar na solução adequada. Dependendo das circunstâncias, o trabalho envolvente na Engenharia de Software pode ser mais dinâmico e menos sistematizado, mais criativo e menos formal (SOMMERVILLE, 2011), cabe a essa disciplina estudar os casos e propor melhores soluções.

2.2 PROCESSOS DE DESENVOLVIMENTO DE SOFTWARE

Um processo de desenvolvimento de software é definido por Sommerville (2011) como um conjunto de atividades relacionadas que levam à produção de um produto de software. Pressman (2011) descreve que é uma abordagem adaptável que possibilita às pessoas envolvidas no processo de desenvolvimento a realizarem o trabalho e selecionarem o conjunto apropriado de ações e tarefas para o projeto. Pode-se então concluir que o processo de software não é rígido e imutável. Pelo contrário, um processo deve ser algo flexível e adaptável à mudanças.

Os processos podem ser complexos e para a evolução e o funcionamento dependem de pessoas e suas decisões. O ideal é evoluir um processo de acordo com as necessidades próprias da organização, tirando proveito das capacidades pessoais de todos os membros participantes. Logo, não existe um processo ideal, esses variam conforme o projeto, para

situações críticas deve-se ter um embasamento mais estruturado, para situações em que há muita atualização de requisitos, a estruturação deve ser flexível e de fácil manutenção.

Os processos podem ser dirigidos a planos ou ágeis. Segundo Sommerville (2011), nos dirigidos a planos todas as atividades possuem um planejamento inicial, e nos dirigidos a métodos ágeis o planejamento é gradativo. Porém Boehm e Turner (2003), explicam que cada abordagem possui seus melhores tipos de processos de software e geralmente é necessário equilibrar os conceitos para promover melhor atendimento dos requisitos como solução para o problema.

Sommerville (2011) aponta que todos os processos devem incluir quatro atividades fundamentais. A saber:

1. especificação de software;
2. projeto e implementação;
3. validação; e
4. evolução.

Durante a especificação as funcionalidades e as restrições devem ser definidas, no projeto e implementação ocorre a produção do sistema, na validação é checado se o software atendeu às demandas do cliente e na evolução ocorre o progresso e a melhoria do software.

Projetos de desenvolvimento de software são muito distintos, e variados projetos podem ser implementados. Com isso, há diferenças que podem ocorrer no processo que variam de projeto para projeto. Segundo Pressman (2011), se diversificam em vários itens, são eles:

- o fluxo geral das atividades;
- o grau pelo qual ações e tarefas são definidos;
- o grau pelo qual artefatos são identificados e exigidos;
- o modo de aplicar as atividades de garantia de qualidade e de acompanhamento do projeto;
- o grau geral de detalhamento e rigor do projeto;

- o grau de envolvimento com o projeto por parte do cliente, da gerência ou outros envolvidos;
- o nível de autonomia; e
- o grau de prescrição da organização da equipe.

Pressman (2011) conclui que a intenção de um processo de desenvolvimento de software é melhorar a qualidade final do produto de software. Com isso, é necessário tornar os projetos mais gerenciáveis, tornar as datas de entrega e os custos mais previsíveis e orientar as equipes de engenheiros de software conforme realizam o trabalho de desenvolvimento de um sistema.

As seções a seguir apresentam, de modo breve, modelos de processos para o desenvolvimento de software. Seu entendimento é fundamental para que seja compreendida a migração para métodos ágeis.

2.2.1 MODELO CASCATA

O modelo Cascata é o paradigma mais antigo da Engenharia de Software. Foi bem aceito até metade da década de 1980, porém sua eficácia para projetos de desenvolvimento nas últimas décadas vem sendo questionada até por seus principais defensores (PRESSMAN, 2011). Com abordagens para sistemas de software antigos e com poucos cenários de evolução, o modelo Cascata não se encaixa com as necessidades atuais do mercado de software. Por isso, por muitas *software houses* é um modelo considerado obsoleto.

Esse modelo é assim conhecido por causa de sua ligação entre as fases. É formada uma sequência linear de atividades que devem ser seguidas uma após a outra, e a troca para o início da atividade seguinte só ocorre ao término da anterior. Porém, Sommerville (2011) aponta que, na prática, os estágios se sobrepõem e trocam informações. Até mesmo nesse modelo, ocorreram evoluções internas conforme os projetos se desenvolvem.

Quando os requisitos são estáveis e podem ser extraídos de forma consistente, o modelo Cascata pode ser uma boa alternativa. Esse cenário se difere muito dos principais sistemas em desenvolvimento na atualidade. Nos dias atuais, o software tem atualizações constantes, então o modelo Cascata pode não ser o ideal a ser seguido para o desenvolvimento. Ainda sim, ele serve como base para adaptações de novos modelos de desenvolvimento.

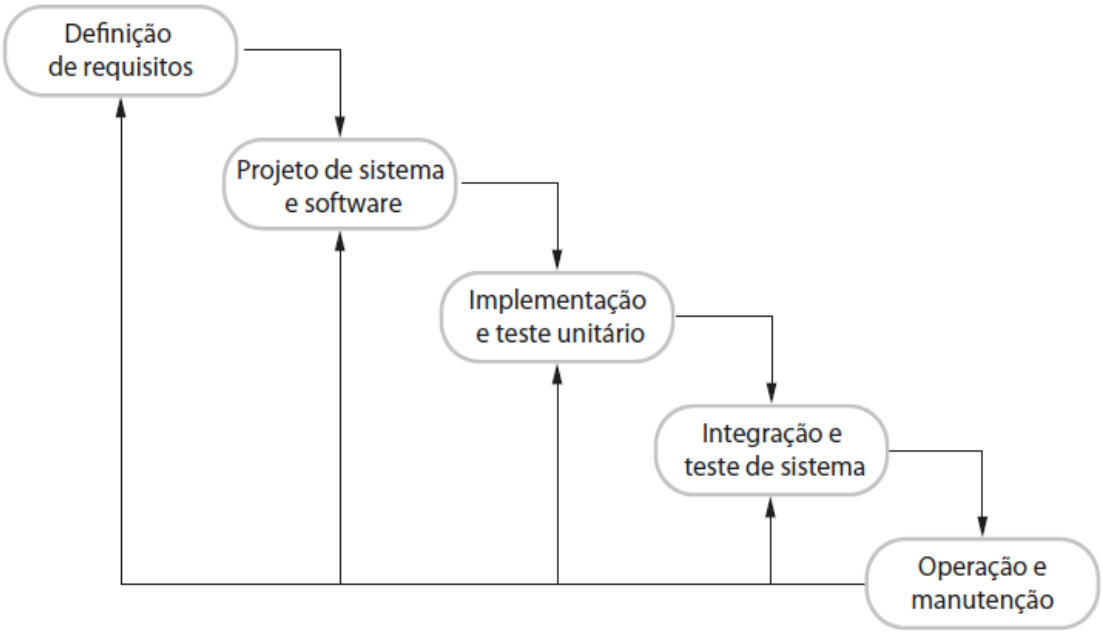


Figura 2: Fases do desenvolvimento do modelo Cascata
Fonte: Sommerville (2011)

Segundo Sommerville (2011), o modelo Cascata possui cinco fases de desenvolvimento, como pode ser observado na Figura 2. Em definição de requisitos, todos os serviços, restrições e metas são estabelecidos por meio de consulta aos usuários. Após a conclusão dessa etapa, são definidos aspectos de hardware e software assim como suas comunicações em um ambiente geral do sistema. Na implementação e testes unitários, são realizadas as execuções das partes únicas e a verificação de que sejam atendidas as necessidades de cada unidade. Após a fase de testes unitários, ocorre a integração e os testes completos, com as unidades integradas e a verificação do atendimento dos requisitos especificados. Por fim, ocorre a operação e manutenção, quando o sistema é instalado e utilizado, promovendo suporte e possíveis manutenções.

O resultado de cada estágio se dá pela aprovação de documentos assinados (SOMMERVILLE, 2011). Essa questão é levada com muito rigor no modelo Cascata, pois tem como objetivo a visualização e os acordos bem firmados entre as partes envolvidas, com uma visualização eficiente do andamento do projeto, principalmente por parte da alta gerência.

2.2.2 MODELOS INCREMENTAIS

Os modelos incrementais evoluíram da necessidade de controlar um projeto de desenvolvimento de software em relação aos riscos, às constantes alterações e incertezas que existem no decorrer da construção de um sistema (LARMAN; BASILI, 2003). Observou-se que essa é uma visão mais eficiente em determinados projetos do que planejar todo o desenvolvimento em uma única fase inicial.

O processo se baseia em ter um produto inicial já em funcionamento para o cliente e a partir disso, incrementar com outras funcionalidades. Primeiramente o cliente possui algo que contemple algumas de suas necessidades iniciais, depois ocorre a avaliação do estado do sistema com o auxílio do *feedback* do usuário. Por fim, é planejado um próximo incremento, sempre se adequando e evoluindo o sistema para a melhor avaliação a cada fase. O processo se repete até o produto estar completo.

As atividades de especificação, desenvolvimento e validação são intercalados e não separados, possuindo *feedback* do cliente em todas as atividades (SOMMERVILLE, 2011). Com o *feedback* constante e o cliente participando do ciclo de vida, frequentemente ocorre um melhor controle da evolução do sistema em fases de evolução e planejamento dos incrementos.

Desenvolvimento incremental de software é melhor que uma abordagem Cascata para a maioria dos sistemas de negócios, *e-commerce* e sistemas pessoais (SOMMERVILLE, 2011). Com essa metodologia, é elaborado um caminho para a solução de forma evolutiva, recuando quando ocorrem imprevistos e falhas. Desse modo, o processo fica mais dinâmico, influenciando para um melhor desenvolvimento nas condições especificadas.

Pressman (2011) indica utilizar o incremento quando não se há todo o pessoal necessário disponível para uma entrega completa até o vencimento do prazo estabelecido. Assim, torna-se essencial um primeiro sistema base com funções úteis de imediato ao cliente, e a partir disso os incrementos são elaborados com um pessoal extra ou com um tempo maior de implementação.

Os incrementos também podem ser planejados para administrar os riscos técnicos envolvidos (PRESSMAN, 2011). Cada incremento possui seus riscos, benefícios e dificuldades, e isso integrado à etapa de planejamento pode-se conduzir um projeto com mais controle desde os aspectos tecnológicos até os organizacionais, dividindo o trabalho que antes seria totalitário e massante em pequenos ciclos com mais clareza e planejamento do que vai ser desenvolvido.

Como vantagens dos modelos incrementais, Sommerville (2011) cita o menor custo, melhor captação do *feedback* dos clientes e o ganho que esse possui com as entregas rápidas. A redução do custo se dá pela quantidade inferior de documentação e análise envolvida nos métodos incrementais comparados ao Cascata, por exemplo. Em relação ao *feedback*, o cliente possui mais dificuldade em dar um retorno eficiente com documentação em vez de ver o sistema em funcionamento. Além disso, o cliente se beneficia atualizando o seu sistema aos poucos, tendo a resolução de seus problemas de negócios em evolução e em curtas faixas de tempo.

Sommerville (2011) também cita dois problemas dos métodos incrementais, o processo não é visível e a estrutura inicial do sistema tende a se degradar. A alta gerência não terá toda a documentação inicial na fase introdutória ao desenvolvimento, isso significa que para mensurar o progresso é necessário entregas regulares. Por consequência de amplas entregas, as soluções podem ser realizadas fora do foco do problema inicial, descaracterizando o projeto e denegrindo o sistema final.

Contudo, é possível desenvolver um sistema incremental sem realmente implantá-los constantemente (SOMMERVILLE, 2011). Desenvolvem-se os incrementos e entregam-se ao cliente, recebendo os comentários e realizando as avaliações normalmente, porém não se implanta enquanto não exista uma integração exata do que se pretende. A maior razão para realizar isso é não interromper processos cruciais de negócios em pouco tempo (SOMMERVILLE, 2011), correndo riscos de ocorrerem falhas e perdas organizacionais.

2.2.3 MODELO ESPIRAL

Proposto por Boehm (1988), o modelo Espiral é evolucionário e possui uma junção incremental com aspectos sistemáticos e controlados do modelo Cascata. Esse modelo fornece potencial para o rápido desenvolvimento de versões cada vez mais completas do software (PRESSMAN, 2011).

Considerando a representação abstrata do modelo de processos espiral, cada volta no espiral representa uma fase no processo de um sistema de software e é dividida em quatro setores (SOMMERVILLE, 2011), como é mostrado na Figura 3. O primeiro setor descreve os objetivos, define as restrições e traça um plano de gerenciamento. No segundo setor estão as avaliações para a redução de riscos, uma análise detalhada é realizada e medidas são tomadas para solucionar ameaças. O desenvolvimento e validação são realizados no terceiro setor, e no quarto ocorre o planejamento, no qual o projeto é revisão e é tomada uma decisão a respeito da continuidade do modelo com mais uma volta.

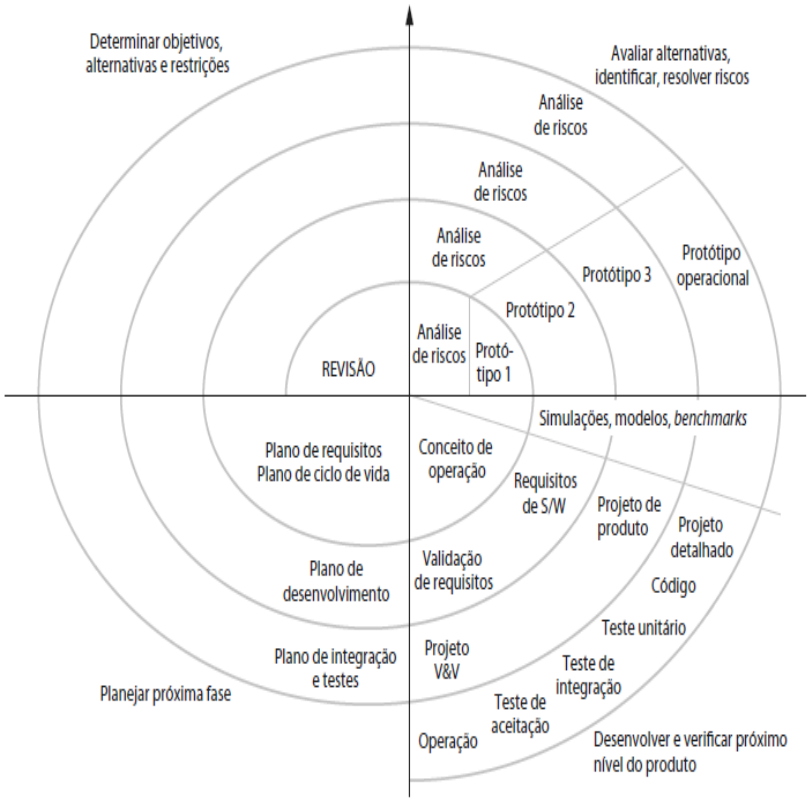


Figura 3: Representação Gráfica do Modelo Espiral
Fonte: Sommerville (2011)

O fluxo do processo é iniciado pelo centro e evolui no sentido horário passando pelas fases distintas. Há pontos âncora de controle que são combinações de produtos de trabalho e condições que são satisfeitas ao longo do trajeto (PRESSMAN, 2011). O custo e o cronograma são ajustados de acordo com o *feedback* do cliente após a entrega, após isso geralmente o gerente do projeto faz um ajuste no número de iterações planejadas para completar o sistema.

Se aplicados apropriadamente, os modelos espirais reduzem riscos antes de se tornarem problemáticos (PRESSMAN, 2011). O desenvolvedor e o cliente, juntos, compreendem melhor os riscos e ameaças em cada nível evolucionário. São utilizados protótipos como mecanismo para a redução dos riscos e é possível a aplicação de um protótipo em qualquer estágio do processo evolutivo do produto. Deste modo, o modelo mantém a abordagem em etapas de forma sistemática sugerida pelo Cascata e a incorpora em uma metodologia mais ágil (PRESSMAN, 2011).

Sommerville (2011) difere o modelo Espiral a outros modelos com seu reconhecimento explícito do risco. O ciclo começa definindo objetivos e apresentando formas de se atingi-los

levando em consideração as restrições do produto. Cada alternativa é validada e os riscos são identificados. O último passo é resolver os riscos por meio de atividades de coleta de informações, como análise mais detalhada, prototipagem e simulação (SOMMERVILLE, 2011).

2.2.4 *UNIFIED PROCESS* (UP)

O Processo Unificado surgiu da necessidade de possuir um método centrado a casos de uso (PRESSMAN, 2011). O UP é dirigido na arquitetura de forma iterativa e incremental. Com ele, obtém-se a tentativa de aproveitar as melhores características dos modelos tradicionais de processo mas conjuntamente implementando os princípios do desenvolvimento ágil (SOMMERVILLE, 2011).

O UP ajuda o arquiteto a manter o foco nas metas corretas, tais como compreensibilidade, confiança em mudanças futuras e reutilização (JACOBSON; BOOCH; RUMBAUGH, 2000). É reconhecida a importância da comunicação constante com o cliente e de métodos racionalizados para descrever a visão do usuário final sobre um sistema de software (PRESSMAN, 2011).

O UP possui cinco fases (PRESSMAN, 2011), que não ocorrem em sequência, mas de forma concorrente e escalonada, sendo elas:

1. concepção;
2. elaboração;
3. construção;
4. transição; e
5. produção.

A fase de concepção está concentrada nos aspectos iniciais de um desenvolvimento. Há a comunicação intensa com o cliente e um planejamento primário do sistema com todos os interessados no projeto. São identificadas as necessidades de negócios e com isso propõe-se uma arquitetura inicial e planeja-se um padrão de incrementos. Para isso, os requisitos iniciais são descritos juntamente com alguns recursos e funções. São avaliados os riscos e definidos cronogramas estabelecendo uma base inicial para o decorrer das tarefas de desenvolvimento.

Na fase de elaboração são refinados alguns tópicos iniciados na fase de concepção e são expandidos casos práticos preliminares. A arquitetura do sistema é ampliada e segundo Pressman (2011), inclui cinco visões diferentes do software:

- caso prático;
- requisitos;
- de projeto;
- de implementação; e
- de emprego.

Na fase de elaboração, ainda não são oferecidas todas as funções do software, mas já é demonstrada a viabilidade da arquitetura. O plano concebido anteriormente é revisado minuciosamente para assegurar que o escopo, riscos e datas permaneçam razoáveis.

No início da fase de construção, há o modelo de arquitetura. São desenvolvidos e adquiridos componentes de software fazendo com que cada caso de uso se torne operacional para os usuários finais. São complementados os modelos de requisitos e de projetos para refletir a versão final do incremento (PRESSMAN, 2011). Além da fase de programação implementando todas as funções para a versão, executam-se os teste unitários e são realizadas as atividades de integração para compor o sistema como um todo.

Juntamente com os últimos estágios das atividades de construção iniciais genéricas, está a fase de transição. Nessa etapa são realizadas as entregas iniciais ainda não completas, para se obter *feedback* dos usuários e clientes com defeitos e mudanças necessárias para sua evolução. A equipe também elabora os materiais de apoio para os usuários como passo a passo para instalações, manuais, como resolver possíveis problemas, entre outros. Na conclusão dessa fase, o incremento torna-se uma versão utilizável do sistema (PRESSMAN, 2011).

Por fim, na última fase, de produção, monitora-se o uso contínuo do software. São disponibilizados suportes para o ambiente e são realizados relatórios de defeitos e solicitações de mudanças para os próximos incrementos. Além disso é avaliado o uso qualitativo do sistema como um todo e discutido os aspectos positivos e negativos.

2.3 HISTÓRICO EVOLUTIVO DOS PROCESSOS

Desde a década de 60, com a definição da disciplina e do termo Engenharia de Software, tem-se aprimorado o desenvolvimento de sistemas de software. A crise de software foi um marco que obrigou as empresas e desenvolvedores a apontarem soluções para os problemas encontrados com um tratamento mais sistemático e controlado (HODA; SALLEH; GRUNDY, 2018).

Os processos começaram com suas sequências de práticas com o objetivo de desenvolver de forma mais eficiente os programas. As fases foram marcadas, estudadas e evoluídas, desde às atividades de especificações, de projeto até implementação e testes, englobando ferramentas, pessoas e métodos.

O modelo Cascata surgiu por volta de 1970, com um artigo de Winston Walker Royce (ROYCE, 1987). O mesmo já apontava falhas e discutia as suas fases e propostas de desenvolvimento. Em respostas às falhas e fraquezas do modelo Cascata, surgiram o desenvolvimento Iterativo e Incremental. Aperfeiçoando esse novo jeito de desenvolvimento, na década de 90, o desenvolvimento ágil surgiu contra os métodos pesados e burocráticos e ainda mais tarde o termo DevOps – Desenvolvimento e Operações – aparece com força, com a proposta de unir partes de desenvolvimento e operação.

2.4 O CENÁRIO DA REGIÃO SUDOESTE

O sudoeste do Paraná é uma importante região de desenvolvimento e fomento à tecnologia. Em quase todas as cidades da região sudoeste é possível estabelecer alguma organização de TI que auxilie a comunidade nas atividades tecnológicas (FERREIRA; PICININ, 2017). A região é destaque por demanda de profissionais qualificados na área, dados da Associação das Empresas Brasileiras de Tecnologia da Informação (ASSEPRO – PR) apontam que o estado é o quarto que mais gera empregos na área.[1] No sudoeste, os municípios onde encontram-se os maiores índices empregatícios e qualificação profissional correspondem a Dois Vizinhos, Francisco Beltrão e Pato Branco, e esses, mesmo com alguma atuação a nível nacional, distribuem seus serviços majoritariamente a cunho regional, contribuindo para a economia, renda e melhores condições de vida da região (FERREIRA; PICININ, 2017).

Segundo Ferreira e Picinin (2017), uma importante contribuição para o desenvolvi-

[1] Acesse: https://www.assespropr.org.br/parana-e-o-quarto-que-mais-gera-emprego-na-area-de-ti-no-brasil/

mento da região está a oferta de vários cursos tecnológicos prestados pelas instituições de ensino superior e às oportunidades de integração tecnológica de empresas de fora da região. Um dos principais agentes está a Universidade Tecnológica Federal do Paraná (UTFPR) que possui campus em várias cidades, incluindo nas três principais com relação à TI, sendo elas Dois Vizinhos, Francisco Beltrão e Pato Branco. Em paralelo a isso, as universidades particulares e a criação e a organização de hotéis tecnológicos, desenvolvimento de *startups*, incubadoras tecnológicas e outras ações de inovação se aliaram à essa oferta formando uma forte aliança de desenvolvimento.

É notório que as grandes organizações de TI do sudoeste utilizam práticas de gestão participativa, desenvolvimento de pessoas pelo viés democrático e o atendimento às normas de certificações (FERREIRA; PICININ, 2017). Essas características incentivam o desenvolvimento pessoal e profissional e mais investimentos nesse aspecto.

Segundo IPARDES (2006), o estado do Paraná tem-se mostrado fiel ao desenvolvimento tecnológico, especificamente no setor de TI, e a partir de 2005 ordenou seus arranjos produtivos de forma a compor as definições de APLs (Arranjo Produtivo Local). Esses aglomerados são importantes para o desenvolvimento do Estado como um todo. Como principais pontos estão os das cidades de Londrina, Maringá, Curitiba e Região Sudoeste em sua totalidade, cujo o último é o presente cenário de estudo desse projeto.

Nesse cenário, o APL e o NTI (Núcleo de Tecnologia da Informação) são as maiores e mais importantes organizações que trabalham para estimular os processos e contribuir para a evolução do setor na região. Definidos como *clusters* e também identificados como Sistemas Produtivos Inovativos Locais (SPILs), os APLs são uma oportunidade de desenvolvimento histórico e cultural, estimulando os processos de inclusão social, além de reduzir o estado agravante de falta de recursos às regiões suburbanas e periféricas (FERREIRA et al., 2015).

3 MÉTODOS ÁGEIS

Na presente seção, são abordados tópicos sobre metodologias ágeis de desenvolvimento de software. A seção apresenta aspectos históricos, os valores e princípios da metodologias ágeis, realizando um comparativo com as metodologias tradicionais. Adicionalmente, são descritas algumas características do *Extreme Programming*, FDD, *Lean*, e *Scrum* e, por fim, são abordadas as práticas DevOps.

3.1 HISTÓRIA

Os métodos ágeis são resultados de uma evolução de desenvolvimento de software como alternativa aos métodos tradicionais pesados (HODA; SALLEH; GRUNDY, 2018). Como características principais muitas documentações e exaustão em cada fase de desenvolvimento, os métodos pesados como o Cascata não estavam atendendo às demandas. Com isso, adaptaram-se os processos evoluindo para uma tentativa de desenvolvimento mais eficiente de software, com características distintas encontradas nos métodos tradicionais.

Em meados de 2001, membros proeminentes da comunidade de desenvolvimento se reuniram e criaram o Manifesto Ágil (BECK et al., 2001), um documento que possui os princípios, práticas e valores da metodologia ágil de desenvolvimento de software. Mais tarde, foi formada a Aliança Ágil (HUGHES et al., 2018), uma organização sem fins lucrativos que promove e busca evoluir o desenvolvimento ágil de software.

Normalmente, quando se constrói um manifesto, há a busca pela mudança e sugere-se alterações revolucionárias sobre algo (PRESSMAN, 2011). Isso aconteceu com esse manifesto, impondo novas formas de pensamentos e raciocínios no desenvolvimento de software. As *software houses* responderam de imediato às propostas do manifesto e, assim, foi iniciada uma revolução no modo como se desenvolve software.

3.2 VALORES E PRINCÍPIOS

O Manifesto Ágil enfatiza valores e princípios que devem ser utilizados e seguidos para um bom desenvolvimento de software com metodologia ágil. Mesmo havendo valor nos itens à direita, será valorizado mais ainda os da esquerda (BECK et al., 2001). Os valores são:

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

Essas diretrizes proporcionam aos envolvidos no projeto de software mais flexibilidade e eficiência no desenvolvimento. Os indivíduos são tratados como aspecto importante para a evolução de um sistema, enfatizando a opinião do cliente e a boa comunicação entre os membros. O software funcionando rapidamente proporciona melhor *feedback* do cliente em comparação com documentação abrangente. O software terá melhor qualidade de atendimento dos requisitos quando o cliente colabora diretamente com a equipe e não apenas pela formação de contratos. E as mudanças, como característica inevitável são levadas como um recurso para o desenvolvimento contínuo do sistema.

Os princípios oferecem aos membros mais clareza de como conduzir os projetos, o que priorizar, como se comportar e o como raciocinar no decorrer do desenvolvimento (BECK et al., 2001). Abaixo são explicitados cada um dos princípios.

1. Nossa maior prioridade é satisfazer o cliente, por meio da entrega adiantada e contínua de software de valor;
2. Aceitar mudanças de requisitos, mesmo no fim do desenvolvimento. Processos ágeis se adequam a mudanças, para que o cliente possa tirar vantagens competitivas;
3. Entregar software funcionando com frequência, na escala de semanas até meses, com preferência aos períodos mais curtos;
4. Pessoas relacionadas à negócios e desenvolvedores devem trabalhar em conjunto e diariamente, durante todo o curso do projeto;

5. Construir projetos ao redor de indivíduos motivados. Dando a eles o ambiente e suporte necessário, e confiar que farão seu trabalho;
6. O método mais eficiente e eficaz de transmitir informações para, e por dentro de um time de desenvolvimento, é por meio de uma conversa cara a cara;
7. Software funcional é a medida primária de progresso;
8. Processos ágeis promovem um ambiente sustentável. Os patrocinadores, desenvolvedores e usuários, devem ser capazes de manter indefinidamente, passos constantes;
9. Contínua atenção à excelência técnica e bom design, aumenta a agilidade;
10. Simplicidade: a arte de maximizar a quantidade de trabalho que não precisou ser feito;
11. As melhores arquiteturas, requisitos e designs emergem de times auto-organizáveis; e
12. Em intervalos regulares, o time reflete em como ficar mais efetivo, então, se ajustam e otimizam seu comportamento de acordo.

3.3 COMPARATIVOS ENTRE MÉTODOS ÁGEIS E TRADICIONAIS

O que diferencia as metodologias ágeis das tradicionais são o enfoque e os valores (SOARES, 2004). A metodologia ágil compreende que as pessoas envolvidas no projeto devem ter mais atenção e dedicação de estudos e trabalhos do que os processos burocráticos e os próprios algoritmos. As pessoas são quem coordenam o projeto, os valores pessoais são exaltados com comunicação eficiente entre as partes, deixando de lado regras contratuais e focando no objetivo principal que é o que o cliente realmente deseja e na solução para o problema que ele possui.

Nos projetos ágeis, o objetivo é gastar menos tempo com a documentação e mais com a implementação. Assim, as metodologias ágeis tornam-se adaptativas em vez de preditivas (SOARES, 2004). Elas se adaptam a novas situações que ocorrem no desenvolvimento do projeto, e não são previamente pensadas com exaustão a fim de tentar prever as mudanças antecipadamente. Essa análise inicial é difícil e apresenta alto custo (SOARES, 2004).

Uma das muitas melhorias que são apresentadas com as metodologias ágeis, é a entrega constante. O cliente possui o sistema utilizável em um curto espaço de

tempo. Com isso, ele pode contribuir ativamente e substancialmente no desenvolvimento do projeto, tornando-o mais fácil de finalizar com sucesso. Diferente das metodologias tradicionais, quando a entrega era realizada após todas as longas fases de desenvolvimento, e durante o processo de fabricação havia zero ou pouco contato com o cliente, entregando posteriormente, algo que não era esperado.

A integração e os testes são realizados de forma contínua, trazendo mais flexibilidade ao desenvolvimento (PETERSEN; WOHLIN, 2009). Nas metodologias tradicionais, há fases seccionadas de integração e testes, trazendo mais rigidez ao projeto. Com a contínua integração e testes unitários em curtos trechos de código, os problemas são detectados rapidamente e novas possíveis soluções são melhores propostas, encaixando as partes e resultando em um software com mais qualidade atendendo de forma mais fiel os requisitos.

Os processos tradicionais ainda são boas alternativas para diversos casos. Quando se há total consistência dos requisitos e há boa previsibilidade do sistema no futuro, os métodos pesados podem ser uma boa opção. Métodos ágeis podem levar a esses casos, desordem, acordos irregulares entre as partes e um projeto final com muitas brechas. É necessário avaliar os recursos e as regras de negócios envolvidas para se decidir como será realizado o andamento do projeto.

Métodos ágeis não significam desorganização. É inevitável que sejam decididos os papéis e o cumprimento de forma fiel deles, separando as funções e realizando uma boa interação entre os membros da equipe. Além disso, os eventos sugeridos em diversas abordagens devem ser realizados de forma integra e constante, captando os benefícios trazidos por eles. O projeto deve estar controlado, a hierarquia deve ser respeitada junto com uma comunicação mais descontraída em comparação aos métodos tradicionais. A equipe deve ser competente e séria, tendo como objetivo a dedicação total ao projeto e ao software desenvolvido.

3.4 METODOLOGIAS

3.4.1 *EXTREME PROGRAMMING* (XP)

A base para o XP bem sucedido é formada por cinco valores definidos por Beck e Gamma (2000), sendo eles: comunicação; simplicidade; *feedback*; coragem; e respeito. No valor de comunicação, é exaltada a comunicação informal e verbal entre os envolvidos. Os clientes e fornecedores devem ter uma boa relação e segundo Pressman (2011), podem estabelecer metáforas eficazes para comunicar conceitos importantes. A comunicação

verbal é menos propícia a desentendimentos do que a comunicação escrita (ROSENBERG; STEPHENS, 2008). É indicado *feedback* contínuo de todas as partes e deve-se evitar documentação volumosa.

Com a simplicidade, é adequado apenas projetar as necessidades imediatas. Geralmente, desenvolvedores tendem a pensar no futuro, com implementações extras, logo, isso deve ser evitado na metodologia XP. Se o software tiver que ser melhorado depois, será refabricado (PRESSMAN, 2011). Porém, Rosenberg e Stephens (2008) alerta que problemas complexos requerem soluções complexas, então deve-se levar em consideração o negócio e a dificuldade na implementação antes de propor uma simples solução.

O *feedback* vem de três fontes: do software funcionando, do cliente, e dos membros da equipe (BECK; GAMMA, 2000). Testes de unidade dão bons retornos sobre o sistema, assim como os usuários utilizando algo real também impacta na avaliação e na evolução do sistema. Casos de uso são utilizados como testes de aceitação (PRESSMAN, 2011). Assim, como o usuário retorna seu ponto de vista à equipe de desenvolvimento, essa também deve retornar ao cliente impacto dos custos e cronogramas que evoluem conforme o desenvolvimento baseados nas codificações e nos incrementos entregues.

Para conseguir realizar todas as práticas, é necessário coragem. Beck e Gamma (2000) exemplificam que "projetar para o amanhã" é natural dos desenvolvedores, com o objetivo de poupar tempo e esforço ao longo prazo, então, a equipe deve possuir coragem e disciplina para codificar pra hoje, levando a risca o que é proposto na metodologia.

Como finalidade, deve-se possuir respeito entre todos os envolvidos, além de com o próprio projeto. Conforme consegue-se entregas mais eficientes e volumosas, a equipe ganha respeito pelo trabalho desenvolvido, e isso agrega a todos (PRESSMAN, 2011).

Um conjunto de 12 atividades é indicado por Rosenberg e Stephens (2008) em uma metodologia XP ideal. São elas:

1. Desenvolvimento orientado por testes (por meio de testes unitários e de aceitação);
2. O jogo do planejamento;
3. Equipe inteira;
4. Pequenos lançamentos;
5. Metáforas;
6. Design simples;

7. Refatorar impiedosamente;
8. Propriedade coletiva;
9. Programação em par;
10. Integração contínua;
11. Ritmo sustentável; e
12. Padrões de codificação.

Como um conjunto de regras e práticas constantes no XP, Pressman (2011) cita planejamento, projeto, codificação e testes. Essas atividades se relacionam com as 12 indicadas por Rosenberg e Stephens (2008).

Na fase de planejamento, deve-se ouvir, entender e perceber o ambiente de negócios, avaliar os fatores e as funcionalidades, ouvir as histórias de usuários, as características e entender as funcionalidades requisitadas para o software ser construído com eficiência.

Cada história é escrita pelo cliente em uma ficha e após isso, atribuído uma nota de prioridade pelos integrantes da equipe. Posteriormente, cada membro atribui um custo medido em semanas de desenvolvimento (PRESSMAN, 2011). Se a história requerer mais de três semanas, é requisitado ao cliente que divida em histórias menores, em pequenas partes para a implementação, e aí o processo ocorre novamente. Isso é realizado por todos os membros da equipe, de forma conjunta decidem como agrupar as histórias para as versões. Depois de versões entregues, por meio das histórias é possível estimar a velocidade do projeto e dar retorno mais preciso ao cliente (PRESSMAN, 2011).

Na fase de projeto, a base é a simplicidade. Nessa etapa é oferecida um guia de implementação para uma história, e a medida que é escrita, é desencorajado o projeto com funcionalidades extras (PRESSMAN, 2011). Se um difícil problema de projeção for encontrado como parte do projeto de uma história, é recomendada a criação, implementação e avaliação de um pequeno protótipo operacional com o objetivo de reduzir o risco quando a verdadeira implementação iniciar (PRESSMAN, 2011).

Depois de as histórias serem desenvolvidas, não é realizada a codificação direta, mas passa por uma fase de testes de unidade que exercitarão cada história que serão inclusas na versão oficial do software (PRESSMAN, 2011). Com isso, o desenvolvedor foca realmente no que deve ser implementado. Após o código completo, ele passa por um novo teste de unidade e aí é retornado aos desenvolvedores. Corrigir pequenos problemas em

intervalos de poucas horas leva menos tempo do que corrigir problemas enormes próximo ao prazo de entrega (WELLS, 1999). Rosenberg e Stephens (2008) ordena que deve-se escrever o teste, para então escrever o código.

Na fase de codificação ocorre a programação em pares. Recomendado que duas pessoas trabalhem juntas para a criação dos códigos, assim, esse dará em tempo real uma revisão de tudo que é criado (PRESSMAN, 2011). Além disso, mantém os desenvolvedores focados. Na prática, cada desenvolvedor tem uma função ligeiramente diferente, mas sempre no mesmo código e em tempo real com o desenvolvimento.

A Figura 4 ilustra o processo XP, destacando conceitos e tarefas chaves associados a cada uma das atividades.

Figura 4: ***Extreme Programming***
Fonte: Pressman (2011)

Pelo fato de muitas organizações adotarem apenas um subconjunto de práticas do *Extreme Programming*, elas enfraquecem o processo como um todo (ROSENBERG; STEPHENS, 2008). Rosenberg e Stephens (2008) cita alguns itens que são alvos dos críticos: volatilidade de requisitos; necessidades conflitantes de clientes; requisitos levantados informalmente; e falta de projeto formal.

O escopo do sistema pode mudar radicalmente quando os requisitos mudam a todo momento, porém, defensores indicam que isso pode acontecer utilizando qualquer processo

de desenvolvimento. Os requisitos também podem ser conflitantes, e diferentes clientes podem ter visões distintas, e cabe a equipe decidir o que é mais importante, mesmo talvez sendo incapacitada pra realizar tal ação.

Os requisitos levantados informalmente podem resultar em omissões e inconsistências. Isso pode ser um problema se detectado apenas após a construção do sistema. Por isso, a falta de projeto formal envolve elaborar o projeto e a estrutura geral para garantir o software com qualidade. Além de prover melhor manutenção. Defensores do XP rebatem com o argumento de que a metodologia limita a complexidade e prega um projeto simples, reduzindo a necessidade de o software extenso (ROSENBERG; STEPHENS, 2008).

3.4.2 *FEATURE DRIVEN DEVELOPMENT* (FDD)

FDD é uma metodologia ágil de desenvolvimento de software criado por Jeff de Luca e Peter Code (LUCA; CODE, 2002). Desde 1997 essa é uma metodologia reconhecida. Tem como filosofia incentivar a colaboração entre pessoas da equipe, gerenciar problemas e complexidade de projetos utilizando a decomposição baseada em funcionalidades, seguida pela integração dos incrementos e a comunicação de detalhes técnicos usando meios verbais, gráficos e de texto (PRESSMAN, 2011).

O desenvolvimento consiste em especificar uma lista de recursos, e implementar os recursos especificados (KHRAMTCHENKO, 2004). Na especificação da lista de recursos ou funcionalidades são definidos vários aspectos do software. Essa etapa forma um processo crítico e resultará em quão sustentável e expansível será o código futuro dos incrementos (KHRAMTCHENKO, 2004). É requerido a participação integral dos clientes.

Na lista de funcionalidades, obtém-se informações pertinentes tanto para os clientes e usuários como para os membros da equipe. Os clientes veem as funcionalidades que estarão disponíveis na aplicação assim como os desenvolvedores e outros membros do projeto conseguem captar unidades de trabalhos (KHRAMTCHENKO, 2004).

Reunir todos os requisitos e funcionalidades do sistema é a etapa mais importante do FDD. Chamado de “Processo um”, tem como resultado um diagrama UML (*Unified Modeling Language* ou Linguagem de Modelagem Unificada) e uma lista de recursos. Os resultados definem o domínio do problema que o sistema busca a resolução. Se bem realizado, cobre todas as áreas relevantes de negócio e permite adicionar facilmente funcionalidades para versões seguintes. A lista serve como base para os cronogramas do projeto e é utilizada para o rastreamento do progresso.

A implementação inicia-se com o agrupamento das funcionalidades em um pacote de trabalho. Esse pacote contém todos os recursos que estarão presentes em uma iteração. Cada iteração é curta, portante os pacotes devem suportar o desenvolvimento rápido e eficiente.

Cada iteração possui algumas atividades desde a escolha das funcionalidades até a implantação. A reunião inicial é realizada para formar um pacote de trabalho, com detalhes incluídos como *design* de classes, métodos e criação de documentações. Após isso é realizada a revisão do *design*, com a aprovação ou rejeição do que foi realizado anteriormente. O desenvolvimento, implementação e testes de unidades são as etapas posteriores, realizando o processo de desenvolvimento por meio do código e dos testes unitários. Posterior a essa fase, há a reunião de revisão do código e reunião de lançamento da versão desenvolvida.

FDD possui uma semelhante dependência dos requisitos comparando com o Cascata (KHRAMTCHENKO, 2004). Porém, possui uma coleta interativa, com relacionamento ativo da equipe de desenvolvimento e do cliente.

Como requisição para a metodologia, está um papel chamado CP. Geralmente são mais experientes, e desempenham o papel de líderes técnicos. Em uma hierarquia, eles estão entre os desenvolvedores e os gerentes de projeto (KHRAMTCHENKO, 2004). Os *Chief Programmers* preparam a reunião de planejamento de iteração, agrupando uma quantidade de recursos em um pacote de trabalho. Uma equipe tem mais de um CP, e cada um deles realiza iterações de forma independente. Cada iteração possui um número de desenvolvedores selecionado pelo CP. Cada desenvolvedor recebe um subconjunto de recursos e a equipe examina esses com base no "Processo um". Pode haver uma comunicação com o cliente caso haja dificuldades em especificar alguns requisitos.

A fase de codificação se torna um processo mecânico. Isso acontece porque as funcionalidades já foram exaustivamente discutidas durante o "Processo Um", na reunião de início de iteração, nas reuniões de revisão, e todas as classes, métodos e diagramas já foram montados. A finalidade então é apenas tornar operacional o que já foi projetado.

O FDD desencoraja a mudança do código, a refatoração (KHRAMTCHENKO, 2004). O argumento principal é o tempo que é necessário para realizar essas ações comparando com o não valor levado ao cliente. Além disso, é incentivado a propriedade de código forte. Cada programador entende de seu código, conhece ele como consequência de seu desenvolvimento, logo, consegue perceber melhor a consequência que pode ocorrer com alguma mudança.

Como conclusão, é possível notar que o FDD não é muito bom para resolver um problema com constantes alterações. As funcionalidades formam uma base que sustenta todo o desenvolvimento do projeto. Em contrapartida, quando há requisitos estáveis comparado com o Cascata, obtém-se melhor taxas de sucesso (KHRAMTCHENKO, 2004). Além disso, o FDD oferece melhor rastreamento do progresso, em relação a custos e cronograma, que conforta os gerentes e torna mais atraente para grandes empresas.

3.4.3 *LEAN SOFTWARE DEVELOPMENT*

A abordagem *Lean*, em português "enxuta", surgiu do processo de produção industrial e posteriormente no desenvolvimento na década de 1980 (POPPENDIECK; CUSUMANO, 2012). Juntamente com outro termo comum chamado JIT (*Just In Time* ou Na Hora Certa), a intenção era a produção de pequenos lotes de componentes para montagem e envio final aos revendedores.

Segundo Petersen e Wohlin (2011), *Lean* compartilha princípios de agilidade na gestão de pessoas, o foco na qualidade e excelência técnica, e o foco na entrega frequente e rápida de valor ao cliente. Enfatiza-se o desenvolvimento agilizado, processos mais flexíveis, contato com o cliente e melhora contínua.

Lean é utilizado para descrever qualquer prática eficiente com foco no desperdício mínimo. No desenvolvimento de produtos, isto ficou claro quando montadoras japonesas alcançaram tempos de liderança significativamente menores e menos horas de engenharia comparando com projetos dos americanos e europeus (POPPENDIECK; CUSUMANO, 2012).

O gerenciamento japonês e o desenvolvimento de softwares começavam a apresentar semelhanças na década de 1990. Poppendieck e Cusumano (2012) cita a correção dos *bugs* diariamente na Microsoft comparando com a filosofia de produção da Toyota, onde os trabalhadores também paravam as linhas de montagem para a detecção de problemas e correções imediatas.

As abordagens *Lean* no desenvolvimento de produtos enfatizava a reutilização de componentes principais, fases curtas e sobrepostas de forma simultânea, reduzindo assim o tempo de entregas (POPPENDIECK; CUSUMANO, 2012). Essas abordagens se entrelaçam com os princípios ágeis de desenvolvimento de software, onde o *feedback* é constante e as alterações são incentivadas.

Poppendieck e Cusumano (2012) cita sete princípios do desenvolvimento enxuto,

sendo eles:

1. Otimizar tudo;
2. Eliminar desperdícios;
3. Construir qualidade;
4. Aprender constantemente;
5. Entregar rapidamente;
6. Envolver todos; e
7. Melhorar continuamente.

Middleton e Joyce (2012) definem que a forma de pensar de forma enxuta é importante porque pode reduzir taxas de erro para um por milhão de unidades. E fora isso, cita três benefícios reais do desenvolvimento *Lean*, a saber:

1. O controle estatístico do processo no software *Lean* permite que eles quantifiquem seu processo de desenvolvimento de software;
2. A fabricação e desenvolvimento de software fornece uma abordagem e linguagem comuns, simplificando, portanto, o gerenciamento de suas operações; e
3. Promovendo uma abordagem mais produtiva e riscos mais baixos, isso aumentará os lucros.

3.4.4 *SCRUM*

Scrum é um *framework* no qual é possível empregar vários processos (SCHWABER; SUTHERLAND, 2017). É um ambiente onde se torna possível adaptações para o contexto do negócio, melhorando continuamente o produto, o time e o trabalho. É basicamente uma estrutura leve e ágil que fornece etapas para gerenciar e controlar o processo de desenvolvimento de software e produto (SRIVASTAVA; BHARDWAJ; SARASWAT, 2017).

Criado nos anos 1990, o *Scrum* foi inicialmente desenvolvido para gerenciar e desenvolver produtos (SCHWABER; SUTHERLAND, 2017). Foi criado para aumentar a velocidade de desenvolvimento, alinhar lemas individuais e das organizações, definir uma cultura focada no desempenho, apoiar a criação de valor para acionistas, ter uma boa

comunicação e melhorar o desenvolvimento individual e a qualidade de vida (SRIVASTAVA; BHARDWAJ; SARASWAT, 2017).

Atualmente, a abordagem tem sido usado extensivamente para pesquisar e identificar mercados viáveis, tecnologias e produtos; desenvolver produtos e melhorias; liberar produtos e melhorias frequentes; desenvolver e sustentar a nuvem; e sustentar e renovar produtos (SCHWABER; SUTHERLAND, 2017).

O *Scrum* é fundamentado nas teorias empíricas de controle de processo (SCHWABER; SUTHERLAND, 2017), ou seja, no conhecimento que vem da experiência. Há três pilares que apoiam a implementação de controle de processo empírico: transparência, inspeção e adaptação (SCHWABER; SUTHERLAND, 2017).

Scrum possui time, eventos e artefatos. Esses possuem características de organização e atividades que devem ser desempenhadas.

No time, há três papeis: *Product Owner* (PO), *Scrum Master* e a equipe de desenvolvimento.

O PO é o dono do produto, o responsável por gerenciar os requisitos do que é desenvolvido. Como atividades, e segundo Schwaber e Sutherland (2017) envolve:

- Expressar claramente o que será desenvolvido;
- Ordenar itens a ser desenvolvidos de forma eficaz para alcançar metas;
- Otimizar o valor do trabalho do time;
- Garantir que os itens a serem desenvolvidos estejam visíveis e transparentes para todos; e
- Garantir que a equipe de desenvolvimento entenda os itens a serem desenvolvidos.

O *Scrum Master* promove o processo como um todo. Ele ajuda todos a entenderem o funcionamento geral das atividades, dos papéis, do fluxo de trabalho e as demais regras e valores do *Scrum*. O *Guia Scrum* (SCHWABER; SUTHERLAND, 2017), separa as atividades deste papel se relacionando com o *Product Owner*, com a equipe de desenvolvimento e com a organização (SCHWABER; SUTHERLAND, 2017). Um *Scrum Master* não é semelhante a um gerente de projeto, mas a um treinador no jogo esportivo dando sugestões, ajudando a equipe a entregar o software a tempo (OUNSRIMUANG; NOOTYASKOOL, 2017).

Segundo o *Guia Scrum* (SCHWABER; SUTHERLAND, 2017), o *Scrum Master* serve o PO:

- Garantindo que objetivos, escopo e domínio do produto sejam entendidos o melhor possível por todos do time;
- Encontrando técnicas para o gerenciamento efetivo dos itens a ser desenvolvidos;
- Ajudando o time a entender as necessidades para ter itens de desenvolvimento mais claro e concisos;
- Compreendendo o planejamento do produto em um ambiente empírico;
- Garantindo que o *Product Owner* saiba como organizar os itens de desenvolvimento para maximizar valor; e
- Compreendendo e praticando a agilidade;

Segundo o *Guia Scrum* (SCHWABER; SUTHERLAND, 2017), o *Scrum Master* serve a equipe de desenvolvimento:

- Treinando a equipe em auto gerenciamento e interdisciplinaridade;
- Ajudando a equipe na criação de produtos de alto valor;
- Removendo impedimentos para o progresso;
- Facilitando os eventos conforme exigidos ou necessários; e
- Treinando a equipe em ambientes organizacionais.

Segundo o *Guia Scrum* (SCHWABER; SUTHERLAND, 2017), o *Scrum Master* serve a Organização:

- Liderando e treinando a organização na adoção do *Scrum*;
- Planejando implementações *Scrum* dentro da organização;
- Ajudando funcionários e partes interessadas a compreender o e tornar aplicável o *Scrum* e o desenvolvimento empírico;
- Causando mudanças que aumentam a produtividade da equipe; e

- Trabalhando com outros *Scrum Masters* para aumentar a eficácia da aplicação do *Scrum*.

A equipe de desenvolvimento é formada pelos profissionais que realizam o trabalho prático para tornar um incremento apto a ser entregue. Segundo Schwaber e Sutherland (2017) os times são estruturados e autorizados pela organização para se auto gerenciarem e eles devem ser auto organizáveis, multifuncionais, sem títulos, sem sub-times e com habilidades individuais mas com responsabilidade da equipe como um todo. É aqui que estão os programadores, testadores, entre outros papéis que realizam as atividades práticas de desenvolvimento.

As atividades presentes no *Scrum* são formadas pela *Sprint*, Planejamento da *Sprint*, Reunião Diária, Revisão da *Sprint* e Retrospectiva da *Sprint*. Todos os cenários se concentram no incremento entregável de software.

A *Sprint* é o um incremento, uma fase do desenvolvimento total do sistema. Sempre há uma em execução, quando termina a antiga, logo se inicia uma nova. Cada incremento pode ser considerado um projeto com duração curta, não maior que um mês (SCHWABER; SUTHERLAND, 2017). Para se iniciar um ciclo, necessita-se de planejamento, reuniões diárias durante o desenvolvimento, uma revisão e uma retrospectiva.

O Planejamento da *Sprint* é realizado antes do início das atividades. Se fazem presente todo o Time *Scrum*. Tem duração de no máximo oito horas, e consiste em um evento onde os participantes entendem o propósito da *Sprint* e respondem questões como o que será entregue e o trabalho necessário para a entrega (SCHWABER; SUTHERLAND, 2017).

A Reunião Diária abrange um planejamento para as próximas 24 horas. Dura em média 15 minutos e quem participa é a equipe de desenvolvimento. Isso otimiza a colaboração e a performance do time por meio da inspeção do trabalho desde a última reunião e é mantido o local e o horário todos os dias para reduzir a complexidade (SCHWABER; SUTHERLAND, 2017). Aqui é visto o progresso do desenvolvimento em relação ao objetivo. Todos respondem três perguntas:

1. O que eu fiz ontem que ajudou a atingir a meta da *Sprint*?
2. O que eu farei hoje para ajudar a atingir a meta da *Sprint*?
3. Eu vejo algum obstáculo que impeça o atingimento da meta da *Sprint*?

Reuniões diárias melhoram a comunicação e gerenciam impedimentos (SCHWABER; SUTHERLAND, 2017). Com isso o andamento do processo como um todo se torna mais eficiente, promovendo um bom andamento geral das atividades e do desenvolvimento.

A Revisão da *Sprint* ocorre sempre ao final da *Sprint*. É discutido o que foi feito no tempo de desenvolvimento. É uma reunião informal e a apresentação do incremento destina-se a motivar e obter *feedback*, promovendo a colaboração (SCHWABER; SUTHERLAND, 2017). Dura no máximo quatro horas em relação a uma um incremento de um mês, e se este for menor, esta reunião será menor também.

Segundo o *Guia Scrum* (SCHWABER; SUTHERLAND, 2017), os seguintes elementos são incluídos na reunião de revisão de *Sprint*:

- Os participantes incluem o Time *Scrum* e os envolvidos chaves convidados pelo *Product Owner*;
- O PO esclarece quais itens do *Backlog* do Produto ficaram prontos e quais não ficaram;
- O Time de Desenvolvimento discute o que foi bem durante a *Sprint*, quais problemas ocorreram e como estes problemas foram resolvidos;
- O Time de Desenvolvimento demonstra o trabalho que está pronto e responde as questões sobre o incremento;
- O PO discute o *Backlog* do Produto tal como está. Ele projeta os prováveis alvos e datas de entrega baseado no progresso até a data;
- O grupo todo colabora sobre o que fazer a seguir, e é assim que a Revisão da *Sprint* fornece valiosas entradas para o planejamento do próximo incremento;
- Revisão de como o mercado ou o uso potencial do produto pode ter mudado e o que é a coisa mais importante a se fazer a seguir; e
- Revisão da linha do tempo, orçamento, potenciais capacidades, e mercado para a próxima versão esperada de funcionalidade ou de capacidade do produto.

Como resultado da união de revisão, cria-se uma lista do produto revisado e define os potenciais itens de desenvolvimento do produto para o próximo incremento (SCHWABER; SUTHERLAND, 2017).

A Retrospectiva da *Sprint* serve para a equipe se auto avaliar e melhorar para os próximos passos do desenvolvimento. Ocorre depois da revisão da *Sprint* e antes do planejamento para a próxima. Tem como propósito inspecionar como a última *Sprint* foi em relação às pessoas, relacionamentos, processos e ferramentas, identificar e ordenar os principais itens que foram bem e as potenciais melhorias e criar um plano para implementar as melhorias (SCHWABER; SUTHERLAND, 2017).

Como Artefatos da *Sprint*, temos o *Product Backlog*, o *Sprint Backlog* e o próprio incremento de software.

O *Product Backlog* é simplesmente uma lista de itens de tudo que é necessário estar no produto (SCHWABER; SUTHERLAND, 2017). É o único repositório geral dos requisitos do sistema para qualquer possível mudança a ser realizada. O *Product Owner* é o responsável direto por essa lista e ele é quem ordena as prioridades de desenvolvimento.

Esse artefato nunca está completo e não é fixo. Resultados dos *feedbacks* dos usuários, mudanças de mercados e outras variáveis que ocorrem naturalmente no ambiente de negócio e de desenvolvimento, a lista altera, evolui e é desenvolvida.

Sprint Backlog é uma lista dos itens do *Product Backlog* que são selecionados para a *Sprint*. Nela contém as funcionalidades entregáveis para o próximo incremento. Nesse artefato contém os itens a serem desenvolvidos que são revistos nas reuniões diárias para o acompanhamento do progresso do incremento.

O incremento é a soma de todos os itens do *Product Backlog* completados durante a *Sprint* (SCHWABER; SUTHERLAND, 2017). É uma parte do sistema inteiro. Um passo na direção de uma visão ou de um objetivo (SCHWABER; SUTHERLAND, 2017). É exigente de que um incremento seja utilizável, e é o PO que decide se será liberado ou não para a utilização.

Na Figura 5 (SCHWABER, 2018) é ilustrado todo o fluxo de atividades que compõe o *Scrum*. O ciclo começa considerando os requisitos e dividindo eles no *Product Backlog*. Mais tarde ocorre a classificação por prioridade e é completado o *Sprint Backlog* que será movido para o processo de desenvolvimento de até 30 dias. Todos os dias ocorrem reuniões diárias. Quando encerrado o prazo da *Sprint* ocorre a revisão e a retrospectiva, gerando um incremento ou não e aí realizando um novo planejamento.

Como finalidade, o *Scrum* apresenta como principais vantagens a redução de custos devido à comunicação constante e ao aumento da qualidade, garantindo que todas as equipes estejam cientes dos problemas e das mudanças (SRIVASTAVA; BHARDWAJ;

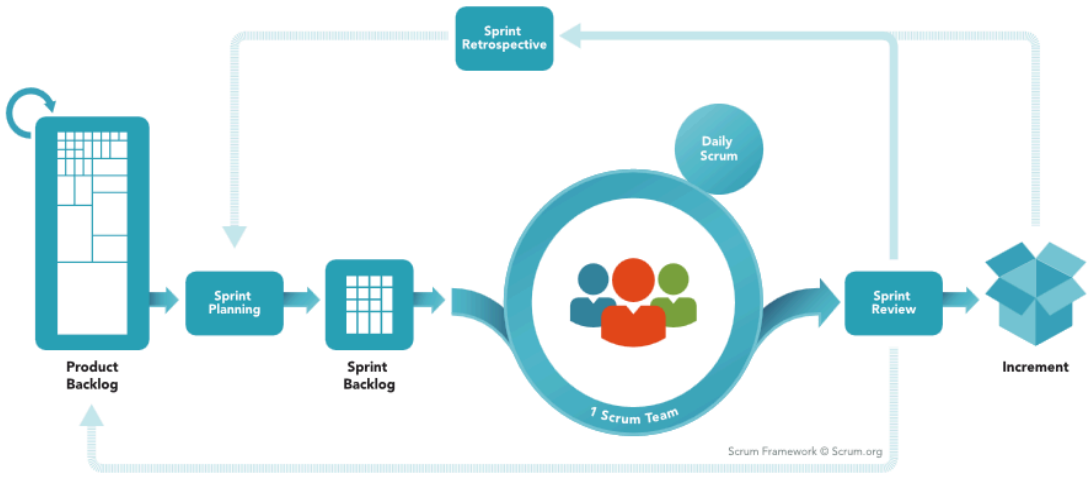


Figura 5: Fluxo do *Scrum*
Fonte: Schwaber (2018)

SARASWAT, 2017).

3.5 DEVOPS

Lustenberger (2016) afirma que a metodologia ágil de software otimizou o fluxo dos processos dentro das equipes de desenvolvimento, e Virmani (2015) cita que DevOps tem como objetivo estender a agilidade já presente nas equipes de software para as equipes de operações. Logo, Kim et al. (2016) descreve que DevOps é uma continuação do movimento ágil. As características de métodos ágeis estão presentes no movimento DevOps, como a valorização de indivíduos e interações, software em funcionamento, colaboração com o cliente e respostas às mudanças. Portanto, as duas filosofias são compatíveis e são trabalhadas e desenvolvidas em conjunto.

DevOps, segundo Davis e Daniels (2016), é um movimento cultural que muda a maneira como os indivíduos pensam sobre seu trabalho. Valoriza a diversidade do trabalho feito, dá suporte aos processos que aceleram os negócios pelo qual as empresas percebem o valor, e medem o efeito das mudanças sociais e técnicas. É um marco cultural para o compartilhamento de histórias e desenvolvimento de empatia, permitindo que as pessoas e as equipes sejam eficazes e duradouras em suas práticas na organização.

Segundo Httermann (2012), DevOps é considerada a maneira mais eficaz para eliminar as barreiras de desentendimentos que existem entre os desenvolvedores e equipes de operações, que tem como responsabilidades implantar, monitorar e suportar o sistema

(VIRMANI, 2015). Essas barreiras podem ser ultrapassadas por meio do compartilhamento de experiências e soluções sugeridas (HTTERMANN, 2012).

3.5.1 PILARES DO DEVOPS

Com o objetivo de auxiliar no desenvolvimento de software e contribuir com soluções para os problemas culturais e técnicos, Davis e Daniels (2016) apontam quatro pilares que sustentam a implantação e a prática eficiente do DevOps. Mesmo que não exista uma única maneira correta de praticar DevOps, esses quatro pilares identificados são comuns em qualquer equipe ou organização, com a qual a organização precisará investir com tempo e recursos.

- Colaboração: é o processo de construção em direção a um resultado específico por meio de interações de apoio e contribuições de várias pessoas. Um princípio orientador que moldou o movimento dos levantamentos foi a cooperação das equipes de desenvolvimento e operação de software. Antes que uma equipe possa trabalhar com sucesso com outra equipe com um foco diferente, os indivíduos de uma equipe precisam trabalhar juntos. Equipes que não funcionam bem em um nível individual ou em conjunto têm pouca esperança de trabalhar bem no nível de interação;

- Afinidade: além do crescimento e da manutenção de relacionamentos colaborativos entre indivíduos, equipes e departamentos dentro de uma organização e em toda a indústria, é necessário ter fortes relacionamentos. A afinidade é o processo de construir esses relacionamentos entre as equipes, navegar por metas ou métricas diferentes, ao mesmo tempo que se mantém metas organizacionais compartilhadas, e fomentar a empatia e o aprendizado entre diferentes grupos de pessoas. A afinidade também pode ser aplicada entre organizações, permitindo que as empresas compartilhem histórias e aprendam umas com as outras à medida que constroem um corpo coletivo de conhecimento cultural e técnico dentro da indústria;

- Ferramentas: são um acelerador, impulsionando a mudança com base na cultura e direção atuais. As escolhas de ferramentas podem ser vistas como vitórias fáceis. Compreender porque eles são vitórias e seu impacto nas estruturas existentes é importante para evitar problemas obscuros em equipes e organizações. A falha em examinar os problemas em valores, normas e estrutura organizacional leva a condições de falha invisíveis à medida que a dívida cultural se acumula. Se as ferramentas, ou a falta delas, atrapalharem os indivíduos ou as equipes que trabalham bem juntas,

suas iniciativas não serão bem-sucedidas. Se o custo da colaboração for alto, o mal investimento em ferramentas aumenta esse custo; e

- Dimensionamento: é um foco nos processos e pivôs que as organizações devem adotar ao longo de seus ciclos de vida. Além de simplesmente considerar o que abordar nas grandes organizações empresariais, esse pilar leva em conta como os outros pilares do DevOps podem ser aplicados nas organizações à medida que crescem, amadurecem e até encolhem. Existem diferentes considerações, tanto técnicas quanto culturais, para organizações que operam em diferentes escalas.

4 ESTUDOS EMPÍRICOS

Neste capítulo serão descritas as Questões de Pesquisa, o *design*, os procedimentos e condução experimental do trabalho.

Foi utilizado o paradigma GQM (*Goals*, *Question* e *Metric*, ou Objetivo, Questões e Métrica) (BASILI; CALDIERA; ROMBACH, 1994) como diretriz para estruturar as metas do estudo. O GQM, de acordo com Wohlin et al. (2012), é uma abordagem de cima para baixo *(top-down)* para estabelecer um sistema de medição direcionado a metas para o desenvolvimento de software. A partir do GQM, a pesquisa começa com metas gerais, define a medição das metas, levanta questões a abordar os objetivos e identifica as métricas que proporcionem.

Desse modo, a partir do objetivo geral (*Goal*) – *"identificar e apresentar características sobre o cenário atual das práticas ágeis nas empresas de desenvolvimento de software do sudoeste do Paraná"* – derivaram-se as questões apresentadas a seguir. Por meio das questões, métricas foram estabelecidas e uma estrutura de *survey* (WOHLIN et al., 2012) foi criada para coleta de informações.

A Figura 6 ilustra graficamente a utilização do paradigma GQM para o presente estudo. A Nível Conceitual, representando nossos objetivos (*goal*), está o objetivo geral do trabalho, como já descrito anteriormente. A Nível Operacional estão as questões (*question*) que são as QPs definidas, perguntas para caracterizar o objeto de medição. Concluindo, a Nível Quantitativo, estão todas as métricas (*metric*) que estão dissolvidas em todas as questões da *survey*, onde possuem as medidas necessárias para responder as questões e assim chegar no objetivo.

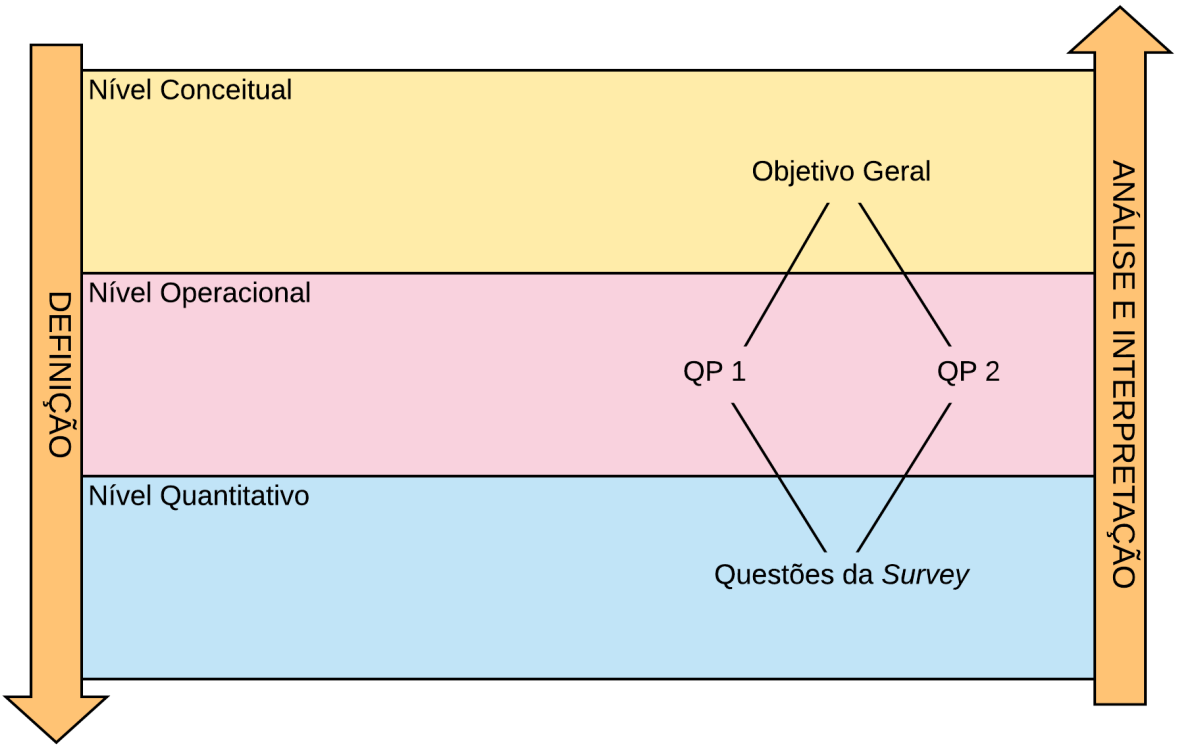


Figura 6: Representação do GQM

Fonte: Autoria Própria

4.1 QUESTÕES DE PESQUISA

Conforme apresentado, este trabalho foi conduzido de modo a responder algumas questões de pesquisa ao final da condução dos estudos a serem desenvolvidos. As questões são descritas a seguir:

- **QP 1:** Qual o cenário de metodologias ágeis nas empresas de desenvolvimento de software do sudoeste do Paraná?
- **QP 2:** Quais os obstáculos e dificuldades as empresas de desenvolvimento de software do sudoeste paranaense encontram em relação às metodologias ágeis?

4.2 *DESIGN* EXPERIMENTAL

A partir do estabelecimento das questões de pesquisa descritas na seção anterior, foram planejadas e executadas ações para respondê-las. O *design* do experimento empírico se dá por seis fases principais:

1. a própria definição e construção das QPs;
2. a decisão se realizar uma *survey* exploratória;
3. a construção dos questionários e a definição do grupo de participantes;
4. a validação por um profissional dos questionários e do grupo de participantes definidos;
5. a identificação das empresas participantes; e
6. o contato com as empresas identificadas.

Após todas essas atividades, têm-se como material os dados coletados. Um diagrama ilustra todas as fases descritas e pode ser visualizado na Figura 7.

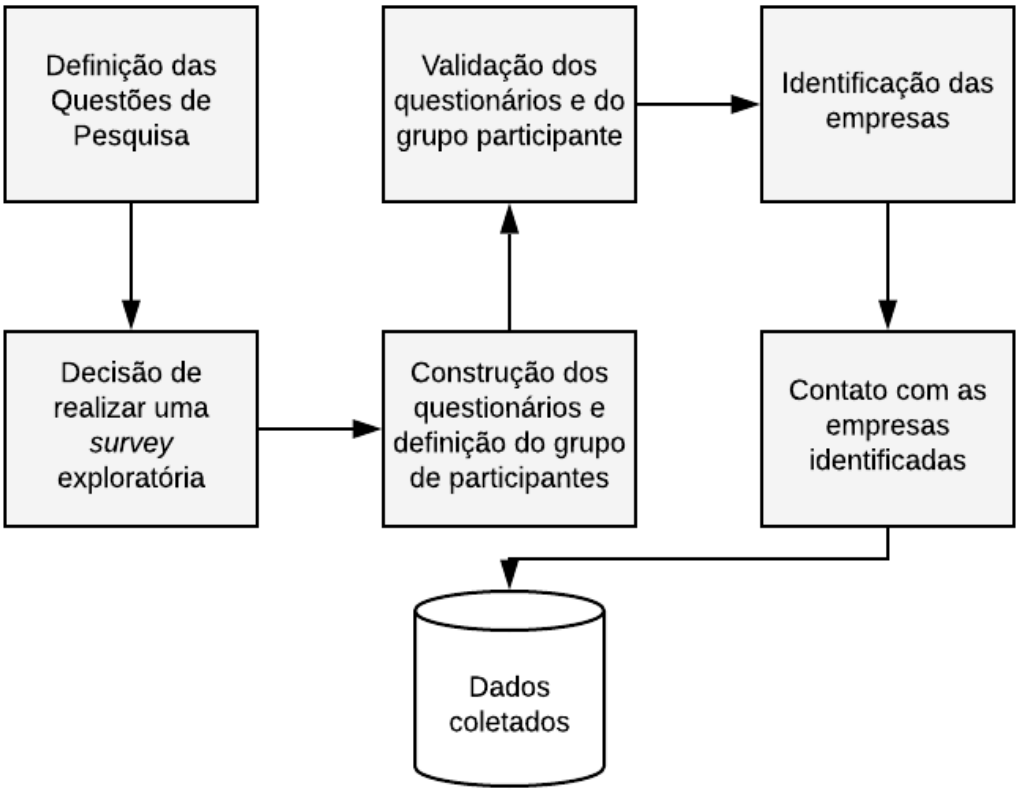


Figura 7: *Design* do experimento

Fonte: Autoria Própria

4.3 PROCEDIMENTOS E CONDUÇÃO EXPERIMENTAL

Os procedimentos iniciaram na definição das QPs usando o GQM. O anseio no estudo da região sudoeste paranaense motivou a abordagem e aplicação de estratégias para identificar o cenário e perspectivas sobre as práticas ágeis de desenvolvimento na região, resultados da carência de estudos desse tipo no contexto local. Com isso, as questões focaram em obter dados e informações acerca de seus métodos de desenvolvimento, além de identificar potenciais problemas e obstáculos que as empresas possuem.

Após as definições das QPs, os procedimentos práticos iniciaram pela decisão de realizar uma *survey* exploratória, conduzida por meio de um questionário online. Segundo Babbie (1999), as *surveys* exploratórias estão relacionados como uma forma de investigação e fornecem pistas para futuros estudos. Com isso, concluiu-se que esse tipo de estudo era viável para as intenções e contribuições propostas ao presente trabalho.

Posteriormente a essa decisão, foi realizada a fase de construção dos questionários que serviria como base para a condução do trabalho. Para dar confidencialidade e legitimidade, foram criados dois questionários que podem ser visualizados no Apêndice A e B. O primeiro composto de informações sobre quem estava respondendo (nome, organização e contato) além de questionar ao respondente se havia interesse em receber os dados finais da pequisa e se havia desejo de participar de uma possível entrevista. O segundo questionário continha informações pertinentes sobre as práticas ágeis de desenvolvimento na equipe e na organização.

O questionário principal, sobre as práticas ágeis, foi dividido em seis seções:

1. tinha como objetivo obter informações preliminares sobre o participante e a organização na qual estava inserido e era composta por oito questões;
2. indagava o grau de prática das organizações sobre cada um dos doze princípios ágeis, composto de uma questão para cada princípio;
3. questionava sobre as atividades realizadas na equipe e na organização e era composta por duas questões;
4. se referia a pessoas e papéis e suas relações, formada por onze questões;
5. buscava analisar as ferramentas e *frameworks* utilizadas pelas equipes com cinco questões; e

6. finalizava, interrogando sobre prazos e entregas com oito questões.

Todo o questionário foi finalizado e composto por 46 questões, que variavam em múltipla escolha, graus de concordância e questões descritivas.

Após a construção dos questionários, foi definido o grupo de participantes para os questionários a serem aplicados. O questionário destinou-se a quem exerce um cargo de gerência, seja de equipe ou da empresa, podendo ser líderes de projetos, gerentes funcionais, *scrum masters*, líderes de equipes, entre outros que obtivessem o conhecimento técnico e tivessem um papel de gerência ou liderança. Era permitida a resposta por mais de uma pessoa de uma mesma equipe.

Seguindo o trabalho, foi realizada uma etapa de validação dos documentos e definições até o momento. O responsável por essa validação foi o professor e profissional da área de processos de desenvolvimento de software Pedro Henrique de Alencar Machado, que com relação aos objetivos propostos pelo trabalho, aprovou as questões e o modelo dos questionários juntamente com o grupo responsável por respondê-los, além de assentir com a forma de abordagem e planejamento das etapas posteriores.

Após à validação, as organizações participantes foram identificadas. Inicialmente, havia como base apenas as empresas associadas ao APL NTI. Foram identificadas 36 empresas associadas. Um filtro dessas empresas foi realizado, e por meio de pesquisas online sobre cada empresa, seis dessas não foram encontradas, sem informações de contato, endereço ou responsáveis, e três não desenvolvem software, apenas são de TI. Com isso, um total de nove empresas foram retiradas do escopo do projeto. Com um baixo número de empresas identificadas, 27 no total, foi decidido também incluir empresas não associadas ao NTI. Foram buscadas mais empresas das cidades de Dois Vizinhos e Francisco Beltrão, inclusive as incubadas na Sudotec, uma incubadora tecnológica sem fins lucrativos situada na cidade de Dois Vizinhos. Com isso, mais 22 empresas foram incluídas como participantes da pesquisa, com 49 em sua totalidade.

O primeiro contato com as empresas foi feito por duas vias, e-mails e pelos seus próprios sites em seções de contato. Nesse momento, foi identificado uma falha no processo de abordagem às empresas. A maioria dos endereços de e-mail eram de suporte e talvez não fossem direcionados para as pessoas corretas na organização com perfis definidos para responderem. Para resolver esse conflito, foram realizadas ligações telefônicas, e assim foi até o final da captação de respostas. Com o contato via ligação telefônica aplicativos de mensagens, uma comunicação mais direta e eficiente foi possível, possibilitando que os

questionários chegassem aos verdadeiros interessados.

Os questionários ficaram disponíveis para respostas em um período de 3 meses, de março de 2019 até maio do mesmo ano. A composição dos dados terminou com 25 respostas. Houveram 22 respostas de 16 empresas distintas no questionário de contato, concluindo uma adesão de pelo menos 32.66% em relação às 49 empresas identificadas inicialmente. A Tabela 1 mostra as cidades onde moram os participantes que responderam o primeiro questionário de contato (22 respostas), e também o número e a porcentagem com base em todas às respostas recolhidas pelo mesmo questionário. Deve-se considerar que nem todos os participantes moram nas mesmas cidades onde trabalham.

Cidades	**Número de Participantes**	**Porcentagem**
Dois Vizinhos	10	45,46%
Francisco Beltrão	5	22,73%
Pato Branco	5	22,73%
Itapejara D'Oeste	1	4,55%
Verê	1	4,55%

Tabela 1: Cidades dos participantes

Fonte: Autoria Própria

5 RESULTADOS E DISCUSSÕES

Nesse capítulo serão mostrados os resultados da *survey*, em seguida essas respostas serão associadas às questões de pesquisa definidas inicialmente, comentando e discutindo sobre o que foi descoberto.

5.1 RESULTADOS

Nessa seção, serão mostrados e comentados os dados brutos recebidos por meio da *survey*, separados pelas próprias seções que constituíam os questionários.

5.1.1 SOBRE OS PARTICIPANTES

Nessa seção inicial o objetivo foi obter informações sobre o perfil dos participantes. Foram formuladas questões sobre os funcionários, sobre as equipes, sobre as metodologias utilizadas, sobre o tempo de prática e conhecimento dessas metodologias e sobre os papéis que os respondentes exerciam na organização.

Na Figura 8, observa-se que há muita variação no número de funcionários nas organizações. 36% dos respondentes fazem parte de uma organização de 101 a 500 funcionários. Por outro lado, 44% estão em empresas de até 50 funcionários. Nota-se também que 16% fazem parte de organizações com até 10 pessoas, consideradas microempresas de desenvolvimento. Conclui-se que há muita diversidade nas empresas identificadas, entretanto nenhuma possui mais de 500 funcionários.

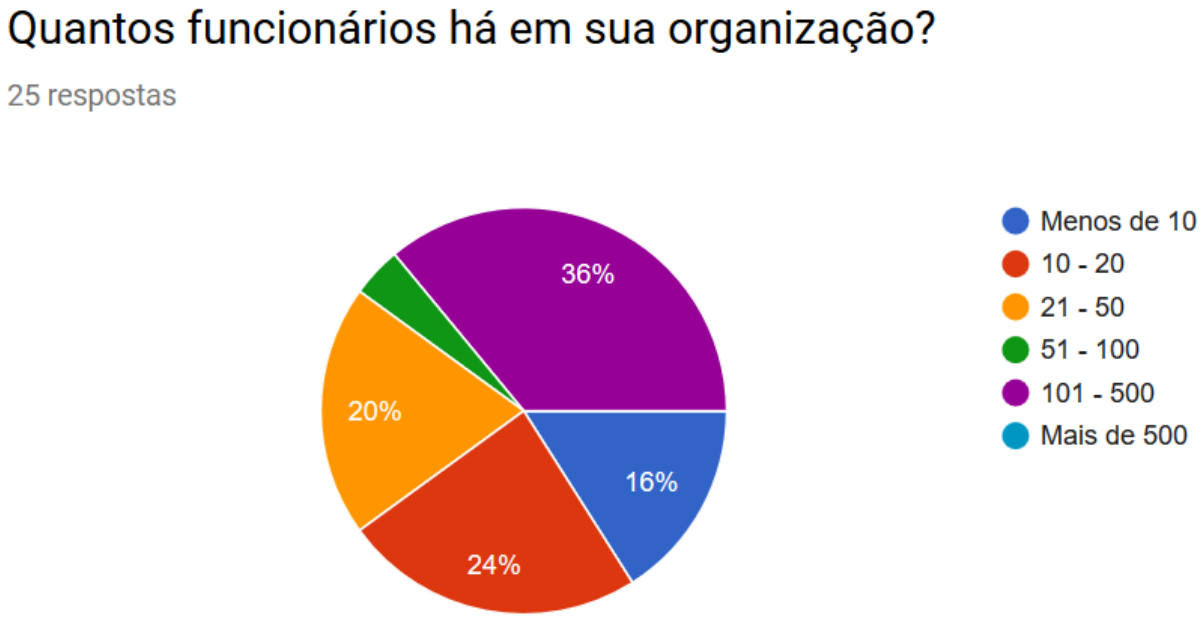


Figura 8: Quantidade de funcionários nas organizações
Fonte: Autoria Própria

Podemos visualizar na Figura 9, que mais de 2/3 dos participantes (68%) fazem parte de organizações com até 5 equipes de desenvolvimento. 20% das empresas possuem de 6 a 20 equipes e apenas 12% são formadas por mais de 20 equipes em suas organizações.

Figura 9: Quantidade de equipes nas organizações
Fonte: Autoria Própria

Na questão mostrada na Figura 10 foi perguntado sobre as metodologias ágeis utilizadas pelas empresas. 92% dizem utilizar o *Scrum*, sendo expressamente destacada das demais metodologias. Em segundo lugar estão o TDD (*Test Driven Development*, ou Desenvolvimento Dirigido a Testes) e o XP (*Extreme Programming*, ou Programação Extrema) com 24% cada. Além dessas, aparecem o FDD (*Feature Driven Development*, ou Desenvolvimento Dirigido à Funcionalidades) e o *Lean* com 8% cada e o AUP (*Agile Unified Process*, ou Processo Ágil Unificado) com 4%. O DSDM (*Dynamic Systems Development Method*, ou Metodologia de Desenvolvimento de Sistemas Dinâmicos) não obteve seleção. Além das metodologias sugeridas, foi permitido a resposta de outra que não estivesse nas opções e a metodologia "*Go Horse*" foi informada por um participante, essa metodologia é baseada na rapidez de desenvolvimento e não há muito planejamento inicial. Também foi permitida a seleção de mais de uma metodologia.

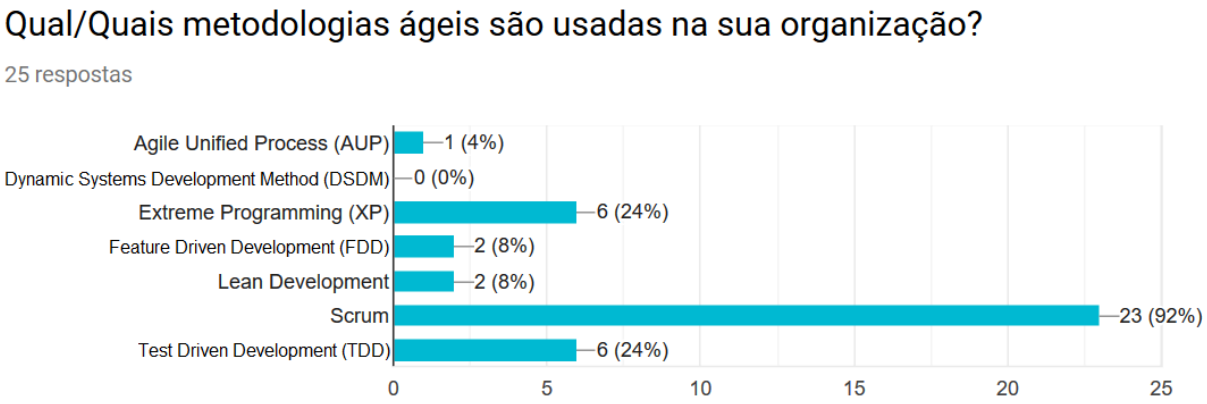


Figura 10: Metodologias ágeis utilizadas

Fonte: Autoria Própria

Na Figura 11, temos informações do tempo de conhecimento sobre as práticas ágeis de desenvolvimento. Todos os participantes conhecem as práticas ágeis. Mais da metade (52%) conhecem as práticas de 4 a 5 anos, 20% de 1 a 3 e a mesma porcentagem de 6 a 10. Apenas 8% conhecem a mais de 10 anos. Nesse cenário, podemos concluir que o conhecimento sobre as práticas ágeis ainda são recentes, com 92% dos participantes conhecendo há no máximo 10 anos.

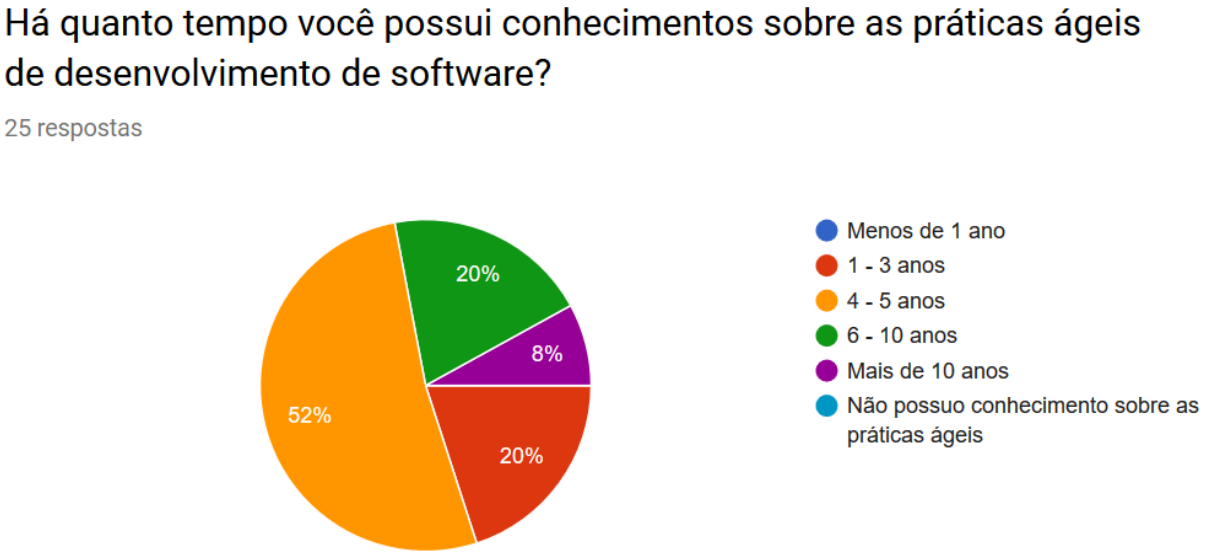


Figura 11: Tempo de conhecimento sobre as práticas ágeis

Fonte: Autoria Própria

Com relação ao tempo de conhecimento, na Figura 12 há informações sobre o tempo de desenvolvimento utilizando os princípios ágeis. Todos os participantes desenvolvem utilizando princípios ágeis. 56% desenvolvem há até 5 anos, 36% há até 3 anos e 8% a mais de 10 anos.

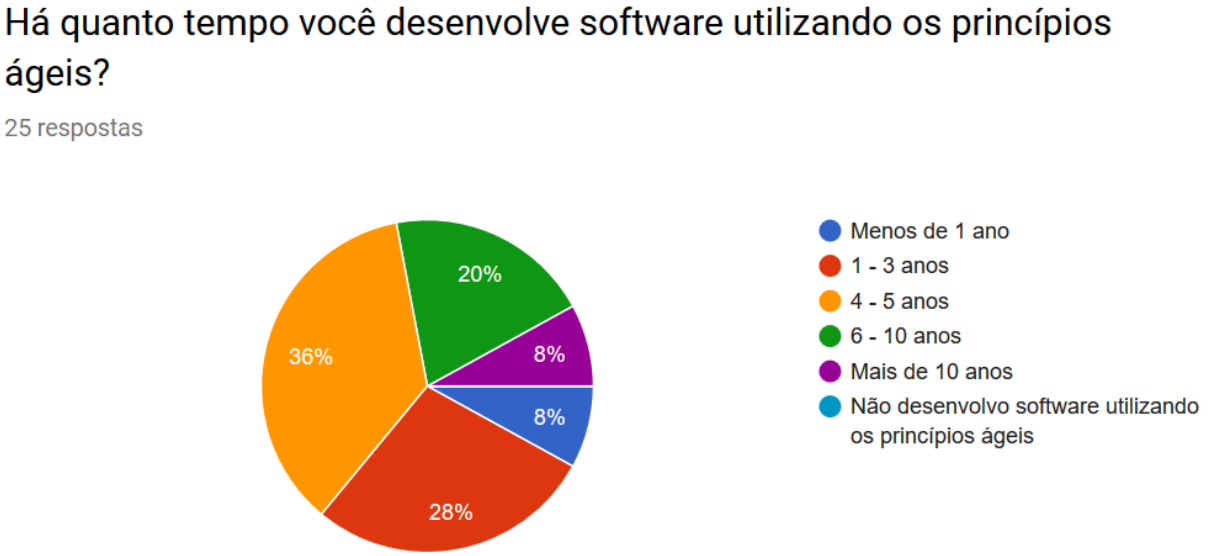


Figura 12: Tempo de desenvolvimento utilizando princípios ágeis

Fonte: Autoria Própria

A questão referente aos papéis exercidos na organização pelo respondente pode ser

visualizado na Figura 13. A maioria, com 60%, são líderes de equipe. Em segundo lugar, com 52%, são desenvolvedores, em terceiro, com 44%, gestores de projetos, e em quarto, quinto e sexto respectivamente, são gerentes funcionais (32%), analistas de negócios (16%) e testadores (12%). Nessa questão era permitida a múltipla seleção de papéis. Além das alternativas sugeridas, foi permitida a resposta de outras, e o papel de "analista de sistemas" foi inserido por um participante.

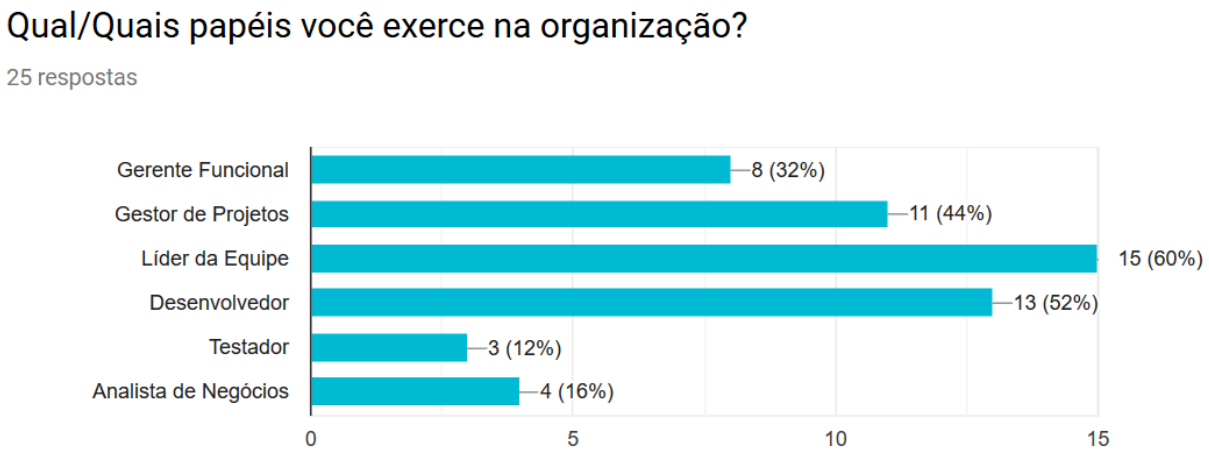


Figura 13: Papéis dos participantes exercidos nas suas organizações
Fonte: Autoria Própria

Na Figura 14 e 15 podem ser observadas informações sobre as equipes. Foi perguntado se o respondente participava de mais de uma equipe de desenvolvimento na organização, 56% não participava, 20% participava de duas, 16% de três e 8% de quatro ou mais. 60% dos respondentes estão em equipes pequenas, com até 5 pessoas. 32% das equipes possuem de 6 a 10 membros. Apenas 1 respondente informou que a equipe de desenvolvimento na qual ele participava possui de 11 a 20 e também apenas 1 com mais de 20. Há evidências de as empresas possuírem equipes pequenas de desenvolvimento e priorizarem um número pequenos de pessoas em seus times.

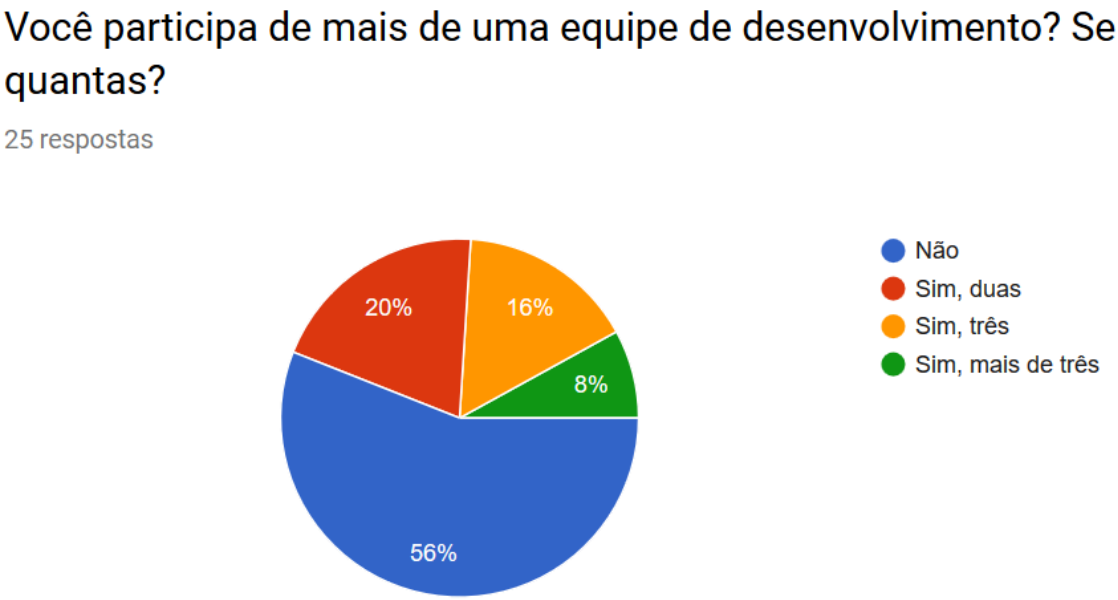


Figura 14: Participação de uma ou mais equipes de desenvolvimento

Fonte: Autoria Própria

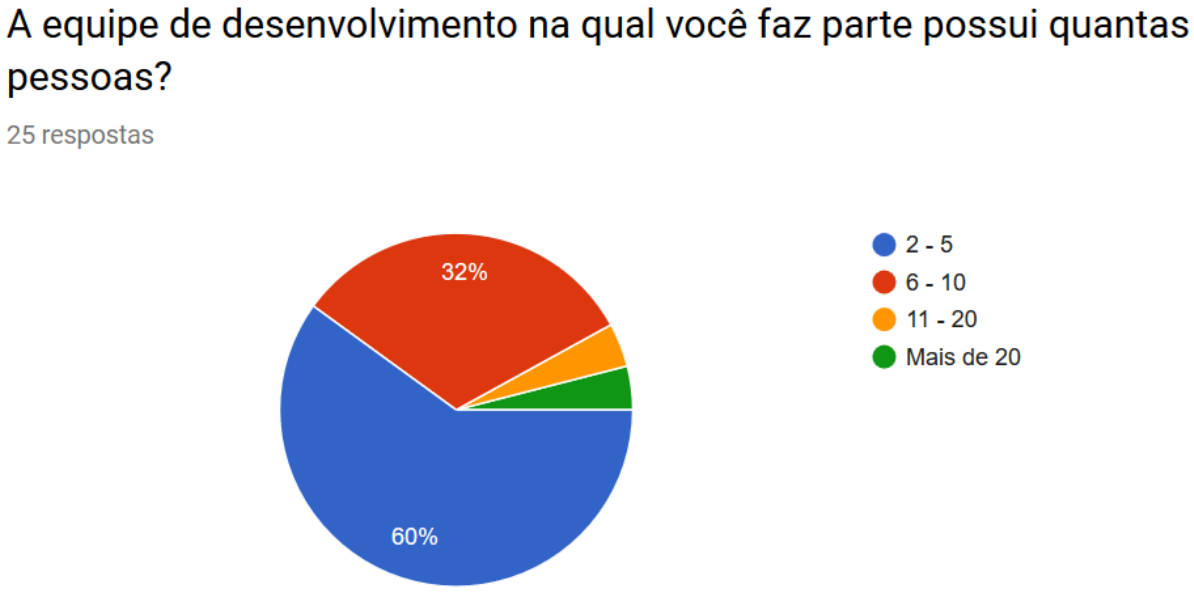


Figura 15: Quantidade de pessoas na equipe de desenvolvimento

Fonte: Autoria Própria

5.1.2 SOBRE AS PRÁTICAS DAS ORGANIZAÇÕES COM RELAÇÃO AOS PRINCÍPIOS ÁGEIS

Nessa seção, o objetivo foi obter o grau de prática nas organizações de cada um dos princípios ágeis já descritos no Capítulo 3. Segundo Beck et al. (2001), os princípios oferecem aos membros mais clareza de como conduzir os projetos, então foi considerado como um fator importante para se conhecer o perfil das empresas nesse quesito. Nas Figuras 16, 17, 18 e 19 são mostrados os resultados dessa seção.

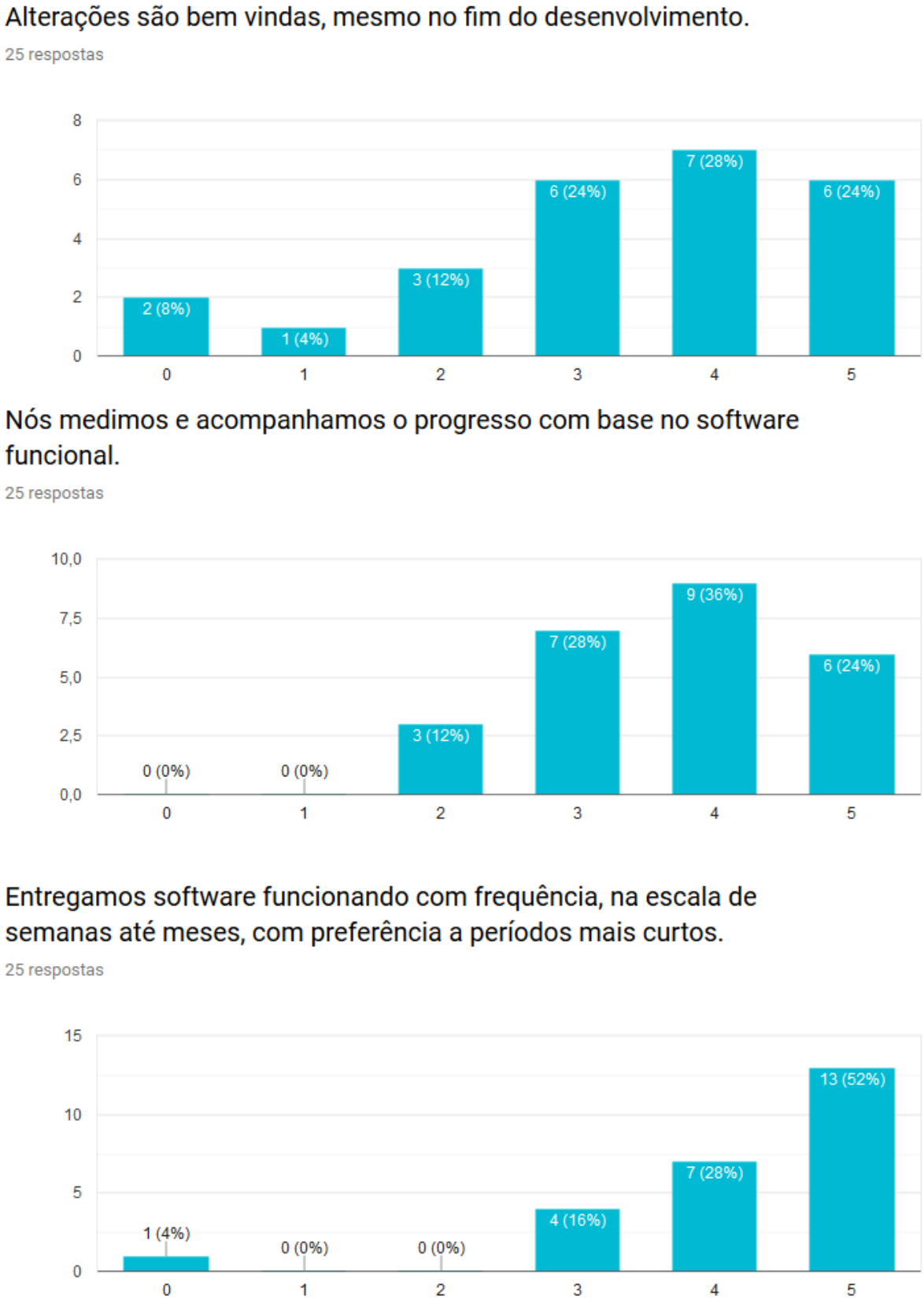


Figura 16: Prática dos princípios ágeis (parte 1)

Fonte: Autoria Própria

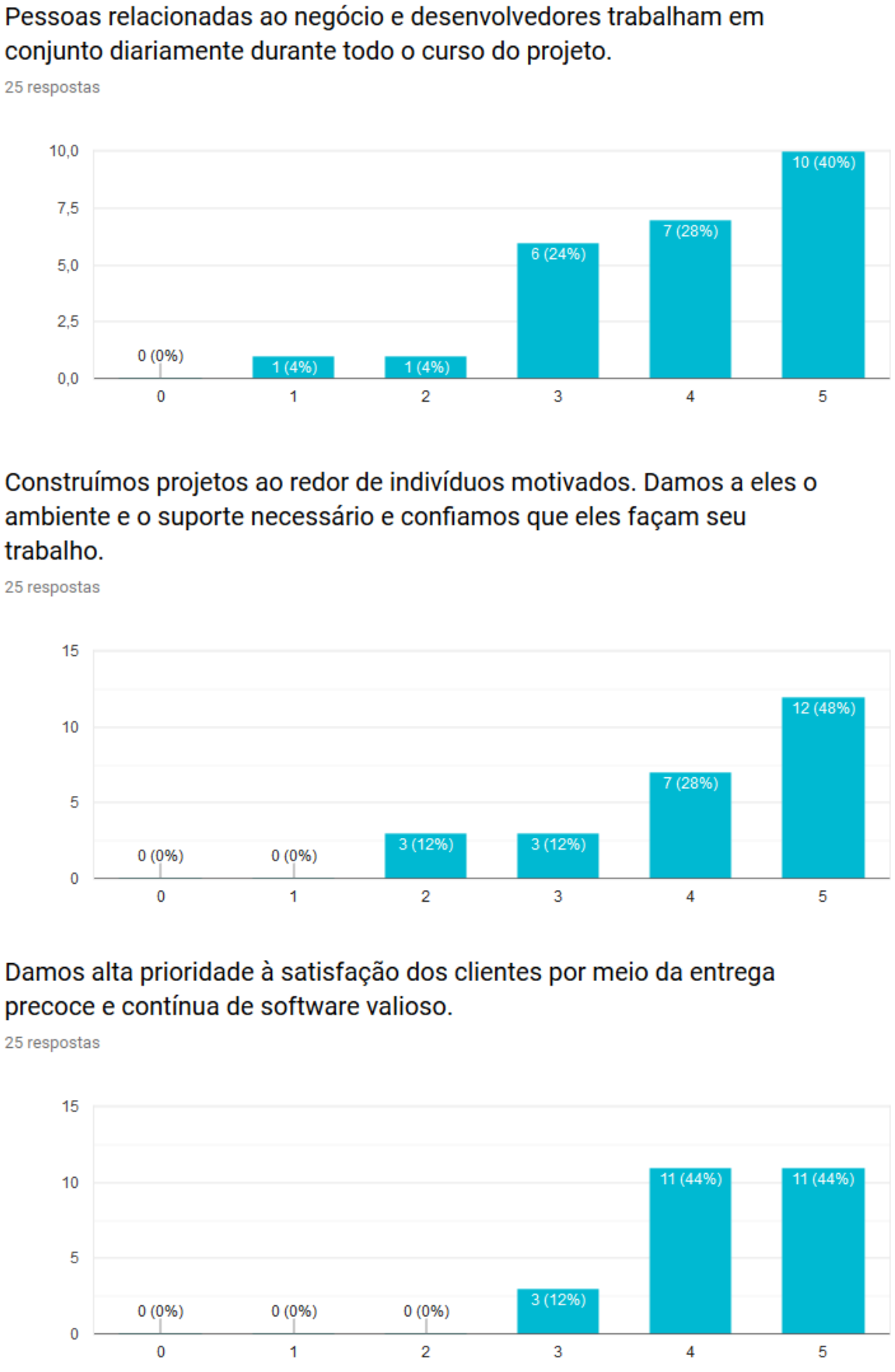
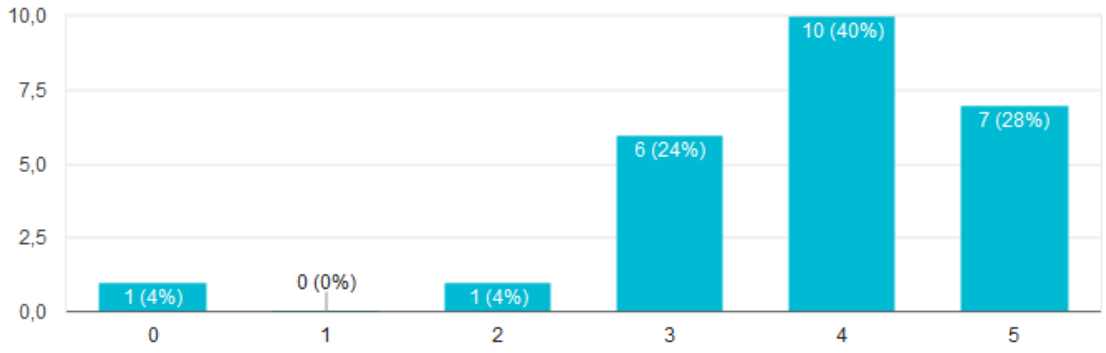


Figura 17: Prática dos princípios ágeis (parte 2)

Fonte: Autoria Própria

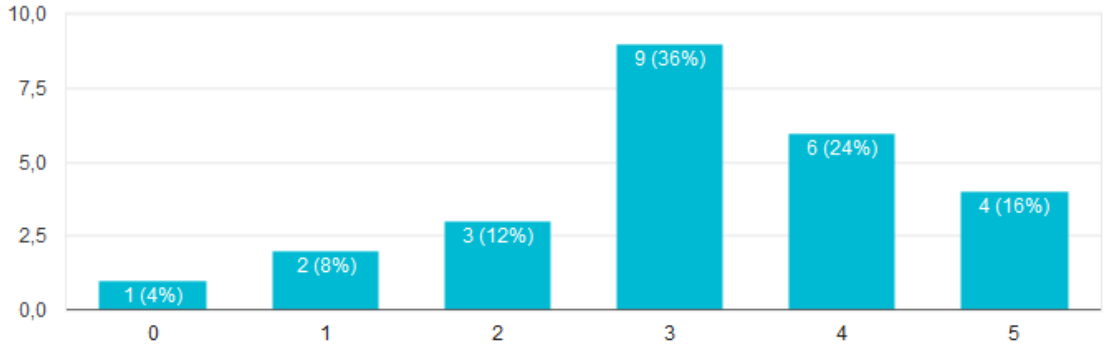
O método mais eficiente para se transmitir informações em uma equipe de desenvolvimento é cara a cara.

25 respostas



Promovemos um ambiente sustentável. Nossos patrocinadores, desenvolvedores e usuários mantém um ritmo constante indefinidamente.

25 respostas



Nossa equipe de projeto de desenvolvimento de software possui atenção contínua para excelência técnica e bom design para desenvolvimento.

25 respostas

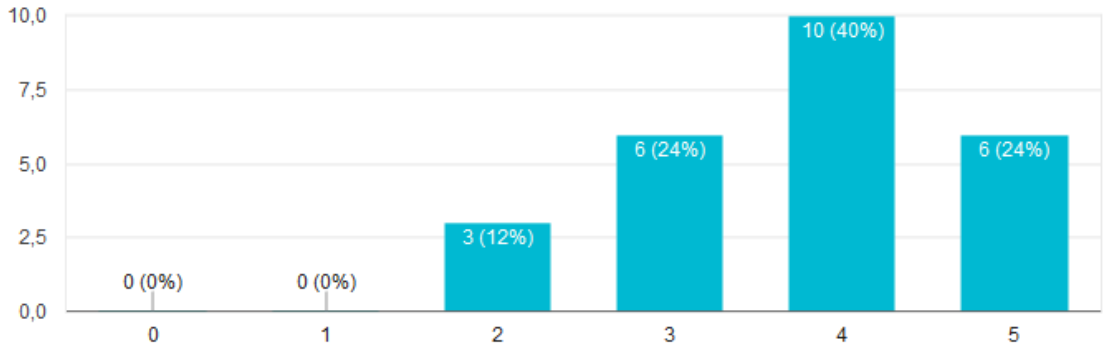
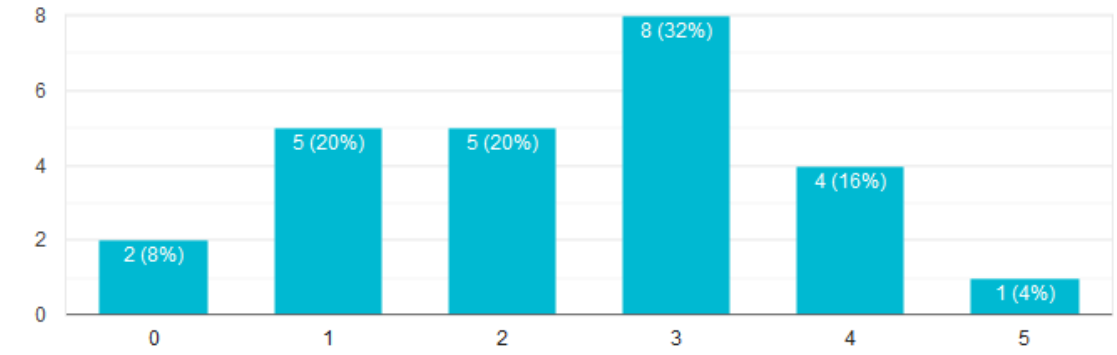


Figura 18: Prática dos princípios ágeis (parte 3)

Fonte: Autoria Própria

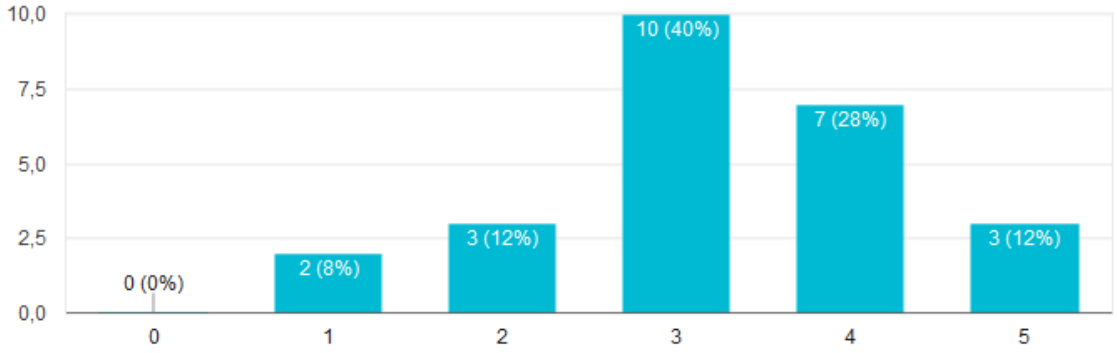
Praticamos projetos, processos e abordagens simples em nossas metodologias de desenvolvimento de software. Nós implementamos recursos que são exigidos pelos clientes, nada mais.

25 respostas



Nossas equipes de desenvolvimento são auto gerenciáveis. Elas têm a capacidade de se organizar continuamente em diferentes configurações para atender aos requisitos em constante mudança e aos novos desafios que surgem.

25 respostas



Em intervalos regulares, nossa equipe reflete em como se tornar mais eficiente e realiza ajustes.

25 respostas

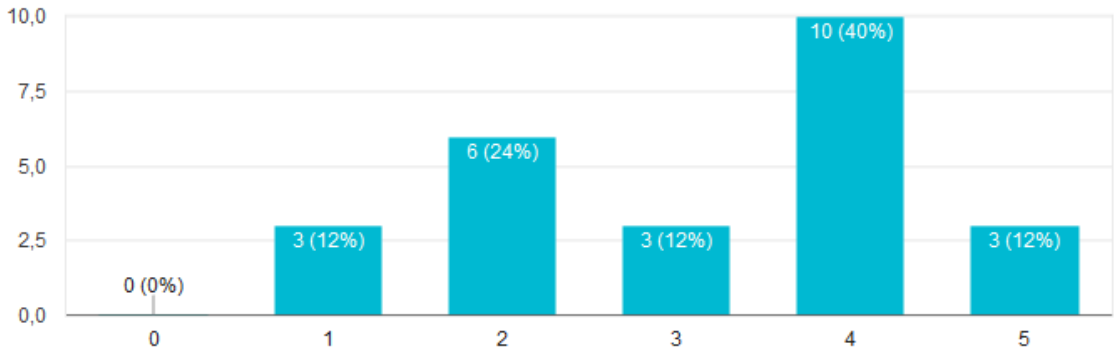


Figura 19: Prática dos princípios ágeis (parte 4)

Fonte: Autoria Própria

De modo geral, as empresas têm práticas que condizem com os princípios ágeis. Todas as questões tiveram a maioria das respostas com grau 3 ou superior. O princípio que mais é praticado nas empresas é o 6º: "Damos alta prioridade à satisfação dos clientes por meio da entrega precoce e contínua de software valioso." com 100% de respostas positivas. Em contraste, o menos praticado é o 10º: "Praticamos projetos, processos e abordagens simples em nossas metodologias de desenvolvimento de software. Nós implementamos recursos que são exigidos pelos clientes, nada mais." com 52% de respostas positivas.

5.1.3 SOBRE AS ATIVIDADES REALIZADAS

Na seção sobre as atividades, foram elaboradas duas questões. A primeira pede que o participante aponte as atividades praticadas pela equipe, ilustrada na Figura 20, e na segunda é perguntado se as atividades são realizadas com data e horas marcadas, ilustrada na Figura 21.

Na primeira questão foram apontadas atividades ágeis pré determinadas. Era possível a seleção de mais de uma atividade, além de possibilitar que o participante inserisse novas atividades. A atividade "*Grooming*", ou "Preparação" foi inserida por um participante. A atividade *Grooming* é uma reunião realizada perto do final de uma *sprint*, para garantir que o *product backlog* esteja pronto para a próxima fase (COHN, 2015). Durante essa reunião, a equipe discute os principais itens do *product backlog* e há o momento de realizar perguntas que normalmente surgiriam na fase de planejamento.

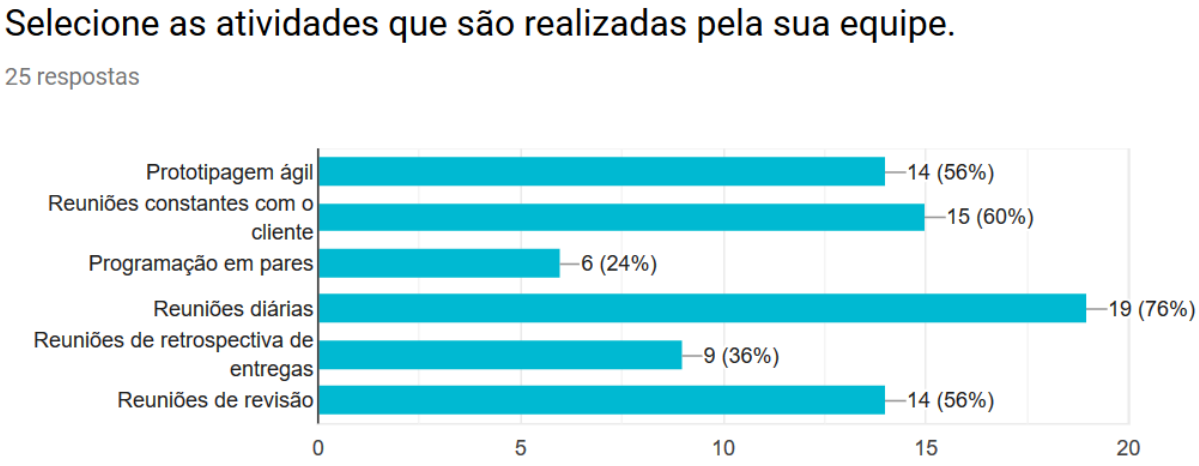


Figura 20: Atividades realizadas pelas equipes

Fonte: Autoria Própria

Na questão ilustrada na Figura 20, foi questionado se as atividades são realizadas

com data e horas marcadas e 68% responderam de forma positiva. Essas respostas traduzem que as empresas tendem a planejar a maioria de suas atividades antes de realizá-las.

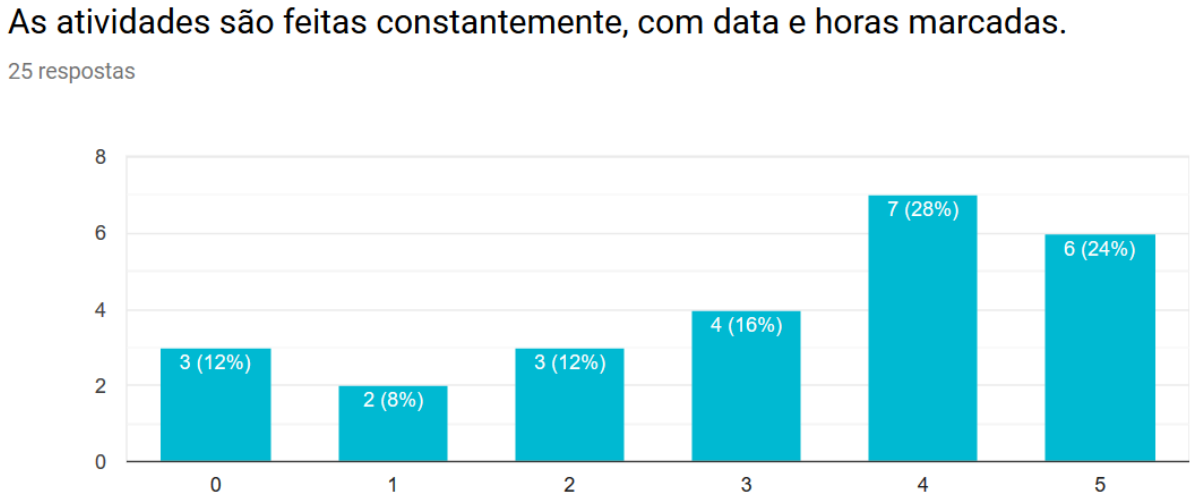


Figura 21: Planejamento das atividades

Fonte: Autoria Própria

5.1.4 SOBRE AS PESSOAS E OS PAPÉIS

Na seção sobre pessoas e papéis, há oito questões com premissas à serem avaliadas de 0 a 5 devido ao grau de concordância dos participantes, além de três questões descritivas buscando conhecer os papéis definidos na equipe e pontos positivos e negativos sobre esse setor.

Na questão da Figura 22, foi abordada a interação do cliente com os membros da equipe de desenvolvimento. A maioria das respostas foram negativas (56%), com a classificação 1 com mais seleções. Podemos considerar que nas empresas identificadas, não há a interação recomendada entre os clientes e os profissionais. A colaboração é feita de longe, sem acompanhamento contínuo e estreito. Muitos problemas podem ser criados por conta disso, sendo a insatisfação do cliente a principal.

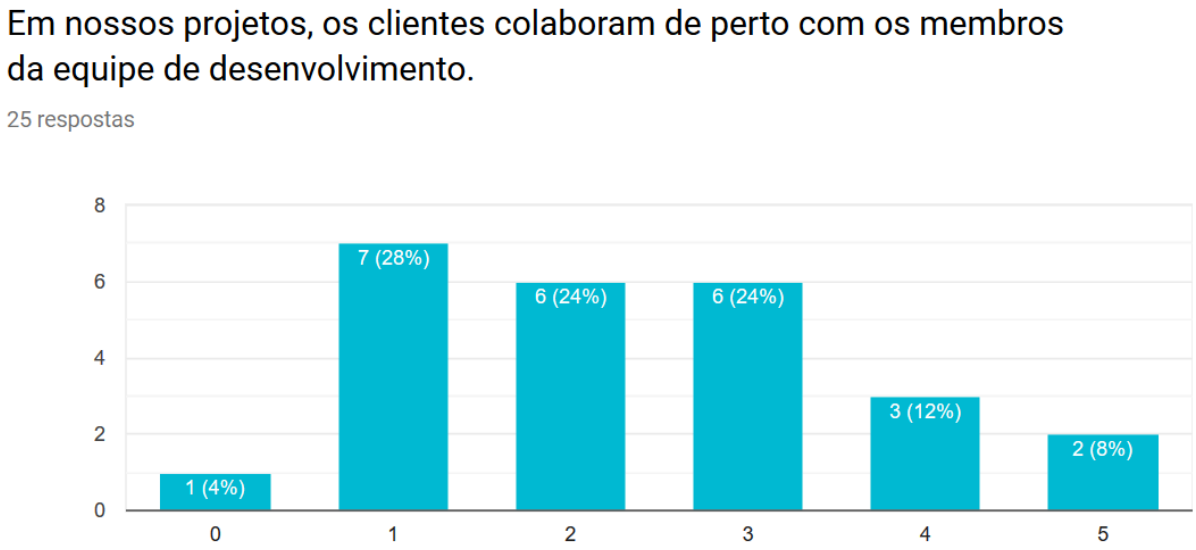


Figura 22: Colaboração entre clientes e membros da equipe

Fonte: Autoria Própria

Na sequência, na Figura 23 podemos observar as respostas se os participantes acreditam que os clientes são comprometidos com os projetos de desenvolvimento, e se eles são motivados e se consideram importantes personagens. Novamente as questões foram negativas, com 52% recebendo classificação 2 ou menor. O grau mais selecionado foi o 2, com 44%. Essa questão está relacionada com a da Figura 21, onde o cliente permanece longe do desenvolvimento.

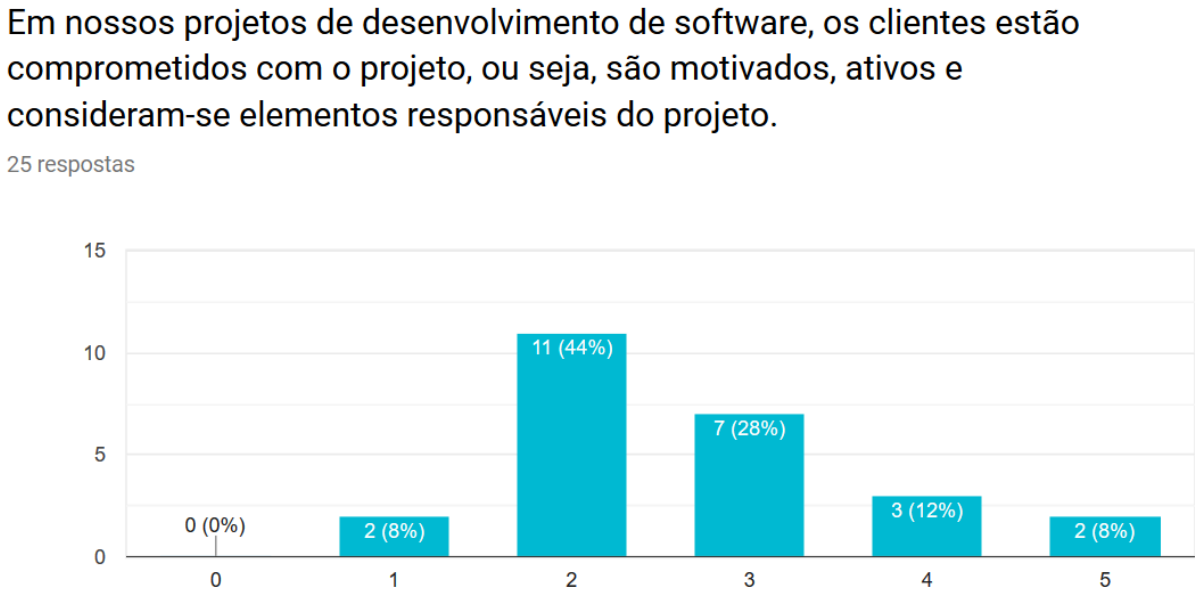


Figura 23: Comprometimento dos clientes nos projetos

Fonte: Autoria Própria

Foi perguntado sobre a posição geográfica dos membros dos projetos desenvolvidos, e as respostas podem ser visualizadas na Figura 24. 48% selecionaram o grau 5, o que traduz que os envolvidos trabalham juntos num mesmo ambiente. A maioria das questões foram positivas, o que conclui que a grande maioria não pratica *home office* e nem mora em outros países ou territórios.

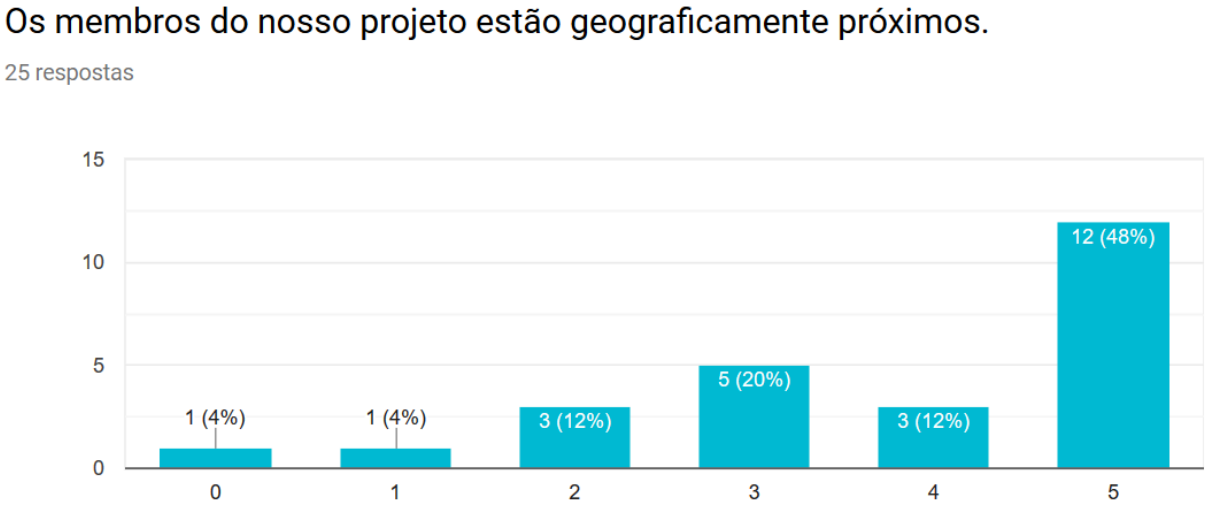


Figura 24: Posição geográfica dos membros do projeto

Fonte: Autoria Própria

Foi abordado como as decisões dos desenvolvedores são tratadas pela administração. 88% marcaram como positivas, ou seja, a administração apoia as decisões dos desenvolvedores. As repostas podem ser visualizadas na Figura 25. O grau 4 foi o mais selecionado, com 44%, e não houve seleção do grau 0 nessa questão.

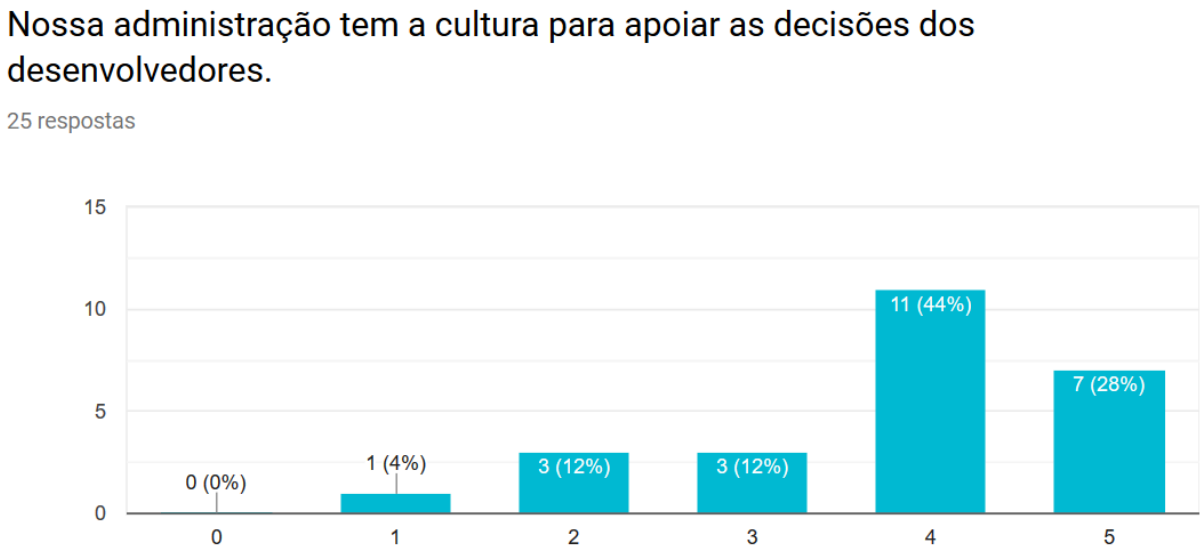


Figura 25: Apoio às decisões dos desenvolvedores

Fonte: Autoria Própria

Duas perguntas foram elaboradas para buscar dados sobre onde é centrada a cultura organizacional. A primeira premissa a ser avaliada a concordância dos participantes falava sobre a cultura organizacional ser centrada na equipe, e a segunda no cliente. As respostas podem ser observadas nas Figuras 26 e 27 respectivamente. Nas duas questões as respostas foram semelhantes. 88% de respostas positivas sobre a interpretação de que a cultura é centrada na equipe, e 80% de positivas sobre a cultura ser centrada no cliente. Concluímos que ambas as figuras, tanto do cliente quanto a equipe, são valorizadas no cenário organizacional das empresas identificadas.

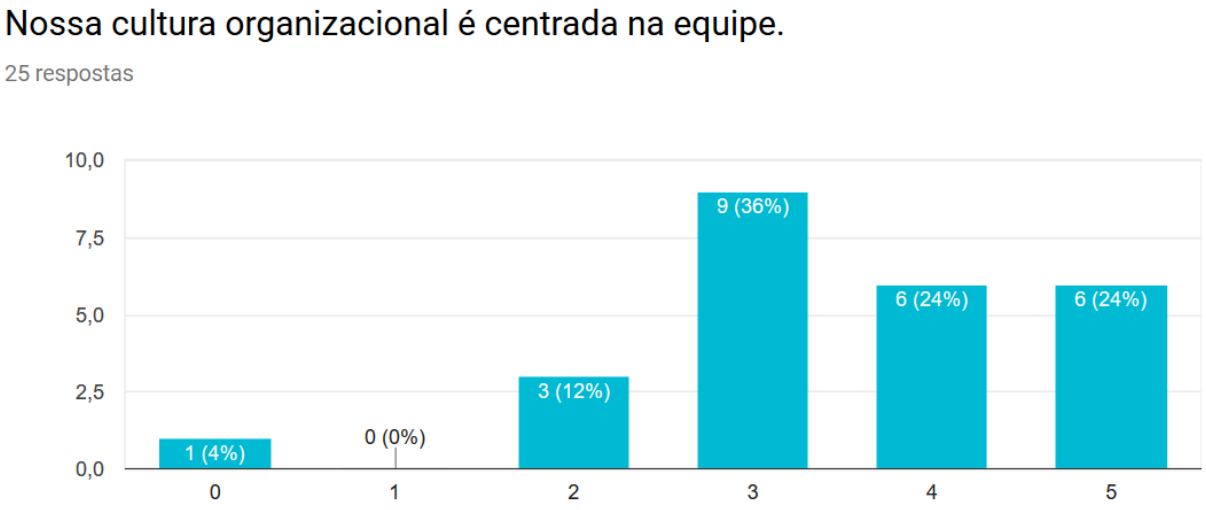


Figura 26: Cultura organizacional centrada na equipe

Fonte: Autoria Própria

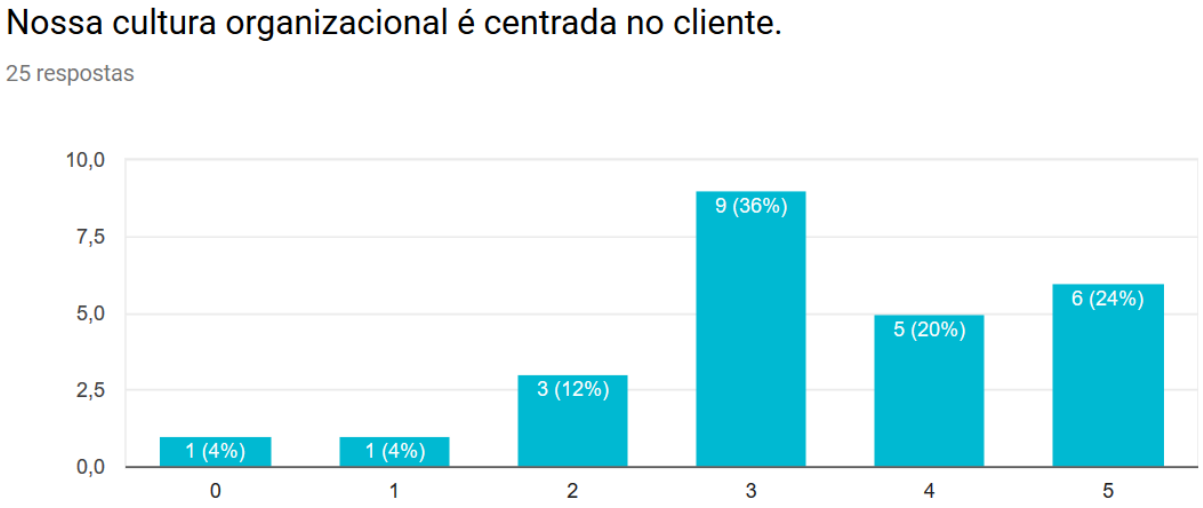


Figura 27: Cultura organizacional centrada no cliente

Fonte: Autoria Própria

Nessa seção, também foi buscado saber se as pessoas são competentes e experientes, ou seja, têm domínio sobre as tecnologias utilizadas e experiências semelhantes no passado. A respostas podem ser visualizadas na Figura 28, que obtém informações positivas sobre as questões técnicas. 80% responderam de forma positiva, porém as respostas tendem a ter um grau baixo, com a maioria no grau 3 (32%).

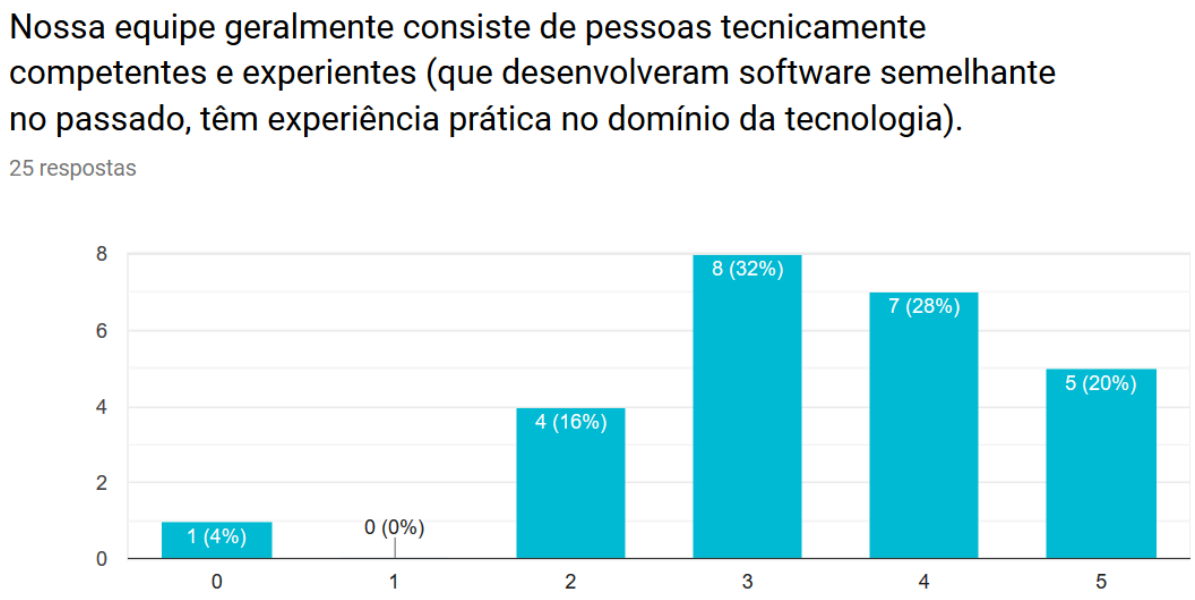


Figura 28: Competência e experiência da equipe

Fonte: Autoria Própria

Sobre a burocracia na estrutura organizacional, na Figura 29 podemos ver as respostas. 64% avaliam que as organizações não são burocráticas, com grau negativos em suas respostas. A maioria está concentrada nos graus 2 e 3 com 24% cada. Mesmo sendo a maioria negativa, a diferença não é tão grande quanto se espera de uma organização com processos ágeis de desenvolvimento.

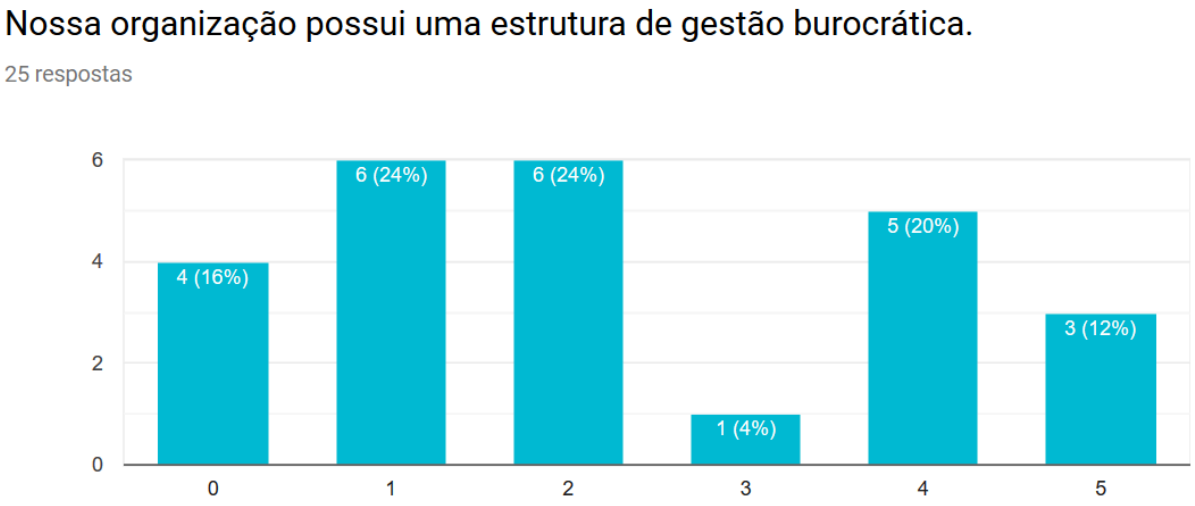


Figura 29: Burocracia da organização

Fonte: Autoria Própria

Três questões discursivas foram aplicadas para completar a seção sobre pessoas e papéis. Essas não foram marcadas como obrigatórias. A primeira pede os papéis definidos nas equipes e obteve 13 respostas que estão descritas a seguir:

1. "Diretor geral, gerente funcional, gerente de contas, *designers* e desenvolvedores."
2. "Desenvolvedor *front-end*."
3. "*Scrum Master* e desenvolvedores."
4. "Analista de sistemas, analista de negócios, analista de requisitos, testador, desenvolvedor, líder técnico, *Product Owner* e gerente de projetos."
5. "*Scrum Master*, *Product Owner*, desenvolvedor e testador."
6. "Desenvolvedores."
7. "Programação, prototipação, *designer*, requisitos e testes."
8. "Programador, testador, automação de testes, analista de negócios e gerente de projetos."
9. "Desenvolvedores, testadores e suporte."
10. "Programador, analista, teste e suporte."
11. "Desenvolvedores."
12. "*Product Owner*, testadores, desenvolvedores e *Scrum Master*."
13. "*Scrum Master*, *Product Owner*, desenvolvedores e testadores."

A penúltima e a última questão dessa seção focaram em obter aspectos positivos e negativos relacionados às pessoas e aos papéis. Essas questões também não foram marcadas como obrigatórias e obtiveram 11 e 10 respostas respectivamente. Os aspectos positivos obtidos são descritos a seguir:

1. "Time entrosado e unido, com valores e princípios semelhantes, o que facilita na comunicação interna."
2. "Bom engajamento, pessoas sem medo de aprender."
3. "Equipe proativa, participativa, experiente, excelente conhecimento técnico."

4. “Equipes pequenas, fácil comunicação e organização das atividades.”
5. “Pessoal jovem e cheio de energia para aprender.”
6. “Equipe unida que consegue definir bem esses pontos, e pensá-los com maior acertividade pela experiencia de alguns dos envolvidos.”
7. “Agilidade, e informalidade.”
8. “Conhecimento.”
9. “Responsabilidade nas tarefas assumidas, comprometimento e agilidade.”
10. “Pessoal motivado e querendo aprender.”
11. “Todos os integrantes da equipe sabem lidar com os problemas do dia a dia, e estão sempre se ajudando para resolver os impedimentos e conseguir entregar todas as atividades durante o prazo pré-estabelecido.”

Os aspectos negativos obtidos são descritos a seguir:

1. “Divergências criativas, o que gera discussões sobre o rumo de certos projetos.”
2. “Falta de conhecimento tanto técnico quanto de regras de negócio.”
3. “Pouco interesse em gestão, isso devido a cultura da empresa.”
4. “Pouca ou nenhuma experiência de negócio, o que dificulta e/ou toma mais tempo para introduzir um problema/solução no planejamento da *sprint* ou *grooming*.”
5. “Pouco experientes, pouco tempo como programadores, limitando o conhecimento técnico deles.”
6. “Falha em alguns pontos, talvez por arrogância de conhecer a regra de negócio como um todo.”
7. “Falta de documentação.”
8. “Não saber sobre o tema envolvido.”
9. “Falta de conhecimento técnico e experiência.”
10. “Não vejo como pontos negativos a forma que a gente trabalha. Só tem vantagens.”

5.1.5 SOBRE AS FERRAMENTAS E OS *FRAMEWORKS*

Na seção sobre ferramentas e *frameworks*, foram abordados os controles qualitativos e quantitativos do software, além do tipo de comunicação e as ferramentas utilizadas pelas organizações. Cinco questões foram elaboradas para obter essas informações, quatro são de classificação dos graus de prática nas empresas de 0 à 5 e uma é discursiva, com o objetivo de os participantes informarem as principais ferramentas e *frameworks* utilizados.

As respostas sobre as duas primeiras questões podem ser visualizadas nas Figuras 30 e 31. Sobre a importância dada para os controles qualitativos, 76% marcaram graus positivos, sendo o grau 4 como o mais selecionado (32%). Em sequência, acerca dos dados quantitativos, 88% selecionaram graus positivos, sendo novamente o grau 4 como o mais selecionado (44%). Dentre os dois controles indicados, as análises quantitativas são mais levadas em consideração pelas organizações do que as análises qualitativas.

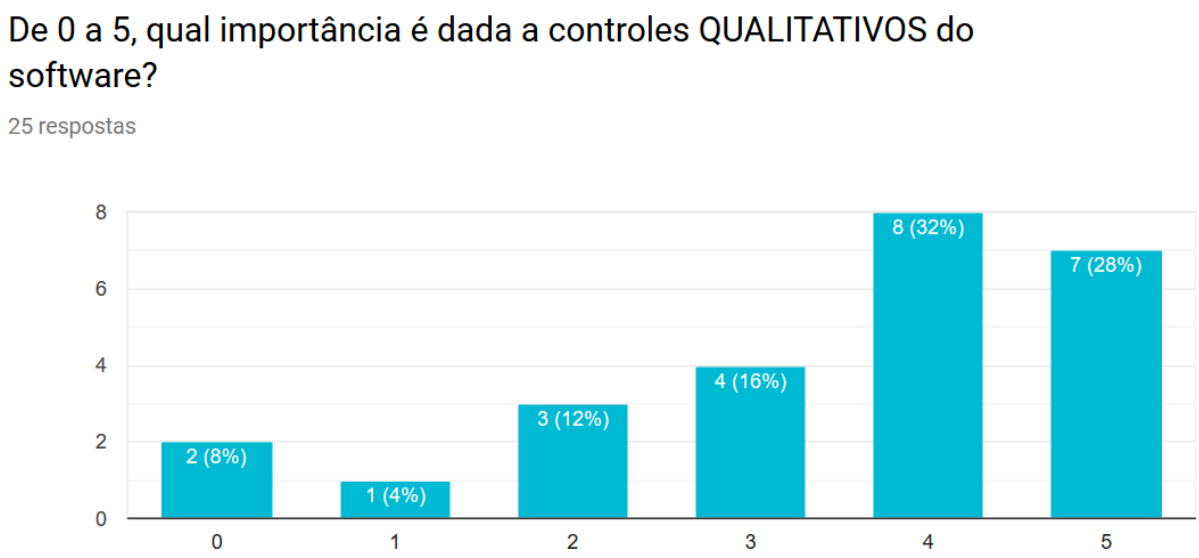


Figura 30: Importância dada a controles qualitativos

Fonte: Autoria Própria

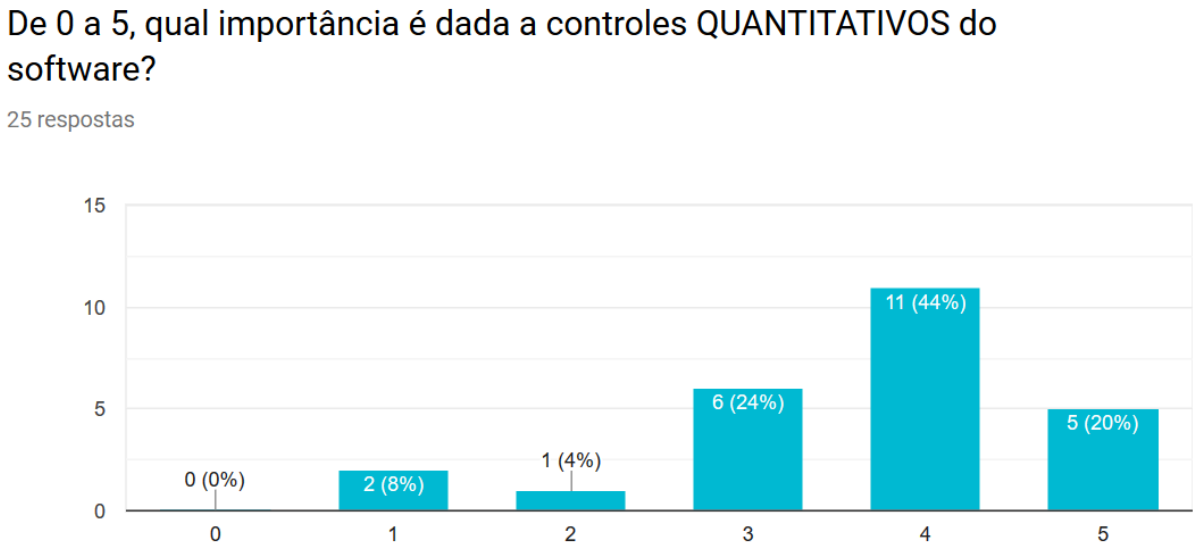


Figura 31: Importância dada a controles quantitativos

Fonte: Autoria Própria

Na questão ilustrada na Figura 32, os participantes informaram se há mecanismos de comunicações rápidas nas diversas áreas de desenvolvedores, operações, suporte, cliente, gerenciamento e negócios. 80% avaliaram com graus positivos. O grau 4 foi o mais selecionado, com 36% das respostas, e em segundo o grau 5 com 28%. Podemos classificar que as empresas possuem rápidas comunicações entre seus setores organizacionais e há ferramentas objetivadas à isso.

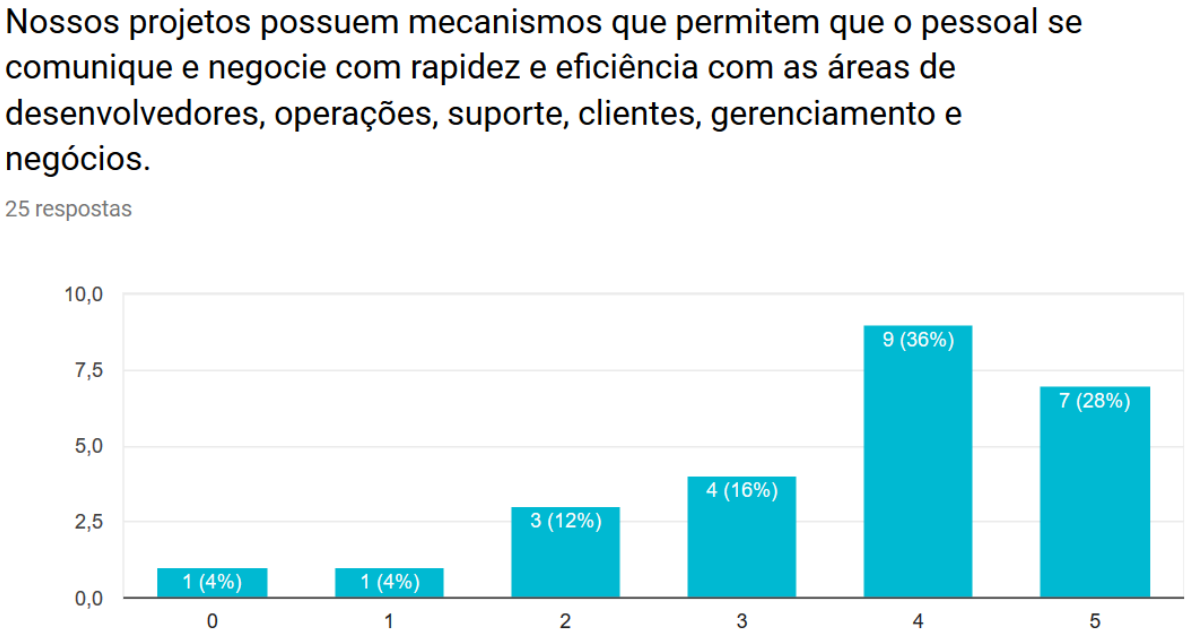


Figura 32: Mecanismos de comunicação e negociação
Fonte: Autoria Própria

Para saber se as comunicações e negociações são realizadas de maneira pessoal, as respostas da questão ilustrada na Figura 33 foi formulada. As práticas ágeis priorizam a comunicação cara a cara nos projetos, onde o processo se torna mais efetivo com indicações maiores de satisfação entre as partes. Nesse contexto, 88% classificaram com grau 3 ou superior que as comunicações são assim realizadas, porém essas respostas tendem a se concentrar no grau 3 (48%). Apenas três participantes informaram grau 0 e 1, o que expõe que essas priorizam a comunicação e negociação por outras vias.

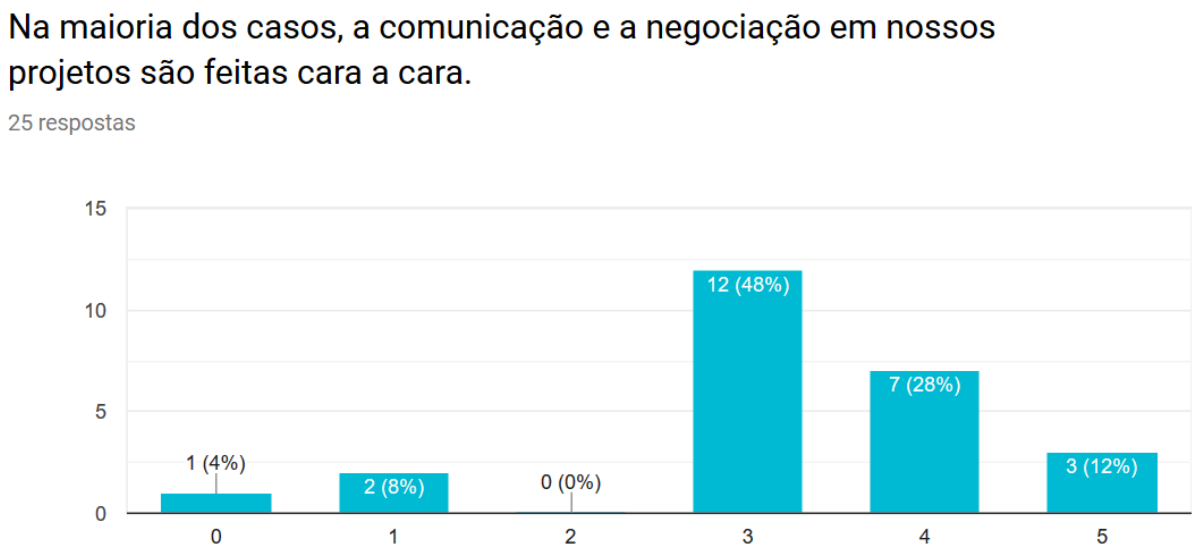


Figura 33: Comunicação e negociação cara a cara

Fonte: Autoria Própria

Completando essa seção, foi buscado descobrir as ferramentas e os *frameworks* utilizados pelas organizações participantes. A questão não obrigatória requisitava que o respondente escrevesse de forma livre as ferramentas e *frameworks* utilizados pelas suas equipes. Foram obtidas 16 respostas. A Tabela 2 apresenta os dados coletados, na coluna direita há a contagem de vezes com que determinada ferramenta ou *framework* foi citada expontaneamente pelos participantes. Jira e Kanban foram as que mais apareceram, quatro vezes, seguidas pelo Trello três vezes, Burndown, C#, Delphi, Laravel, PostgreSQL e Skype duas vezes e as demais apenas uma vez.

Ferramentas e *Frameworks*	Contagem
Jira	4
Kanban	4
Trello	3
Burndown	2
C#	2
Delphi	2
Java	2
Laravel	2
PostgreSQL	2
Skype	2
Adobe XD	1
Angular JS	1
Angular Material	1
Asada	1
Azure	1
Balsamic	1
Bizzagi	1
Bootstrap	1
Confluence	1
DevOps	1
Eclipse	1
E-mail	1
Git	1

(a) Início

Ferramentas e *Frameworks*	Contagem
Google Docs	1
Hibernate	1
Ibexpert	1
Ilustrator	1
Intellij	1
Javascript	1
Jazz	1
JSF	1
MongoDB	1
Photoshop	1
React	1
Redis	1
Ruby Rails	1
Scrum	1
Slack	1
Spark	1
Spring	1
SVN	1
Target Process	1
Teamwork	1
TOAD	1
VueJS	1
WhatsApp	1

(b) Continuação

Tabela 2: Ferramentas e *Frameworks*
Fonte: Autoria Própria

5.1.6 SOBRE OS PRAZOS E AS ENTREGAS

Finalizando o questionário, foi elaborada uma seção abordando prazos e entregas nas organizações. O objetivo foi obter informações acerca desse setor com questões relacionadas às práticas ágeis. Quatro questões são de classificação de 0 à 5 conforme a prática na organização, três questões são descritivas e uma de seleção.

A primeira questão dessa seção pedia o grau de incentivo de *feedback* rápido por parte dos clientes, avaliando os produtos desenvolvidos ou os materiais de planejamento sobre o produto. 88% dos participantes marcaram do grau 3 pra cima resultando em respostas positivas, de que o *feedback* é realmente incentivado pela organização. O grau 4 foi o que mais obteve seleção (40%), seguido pelo grau 5 (36%). As respostas com a pergunta podem ser visualizadas na Figura 34.

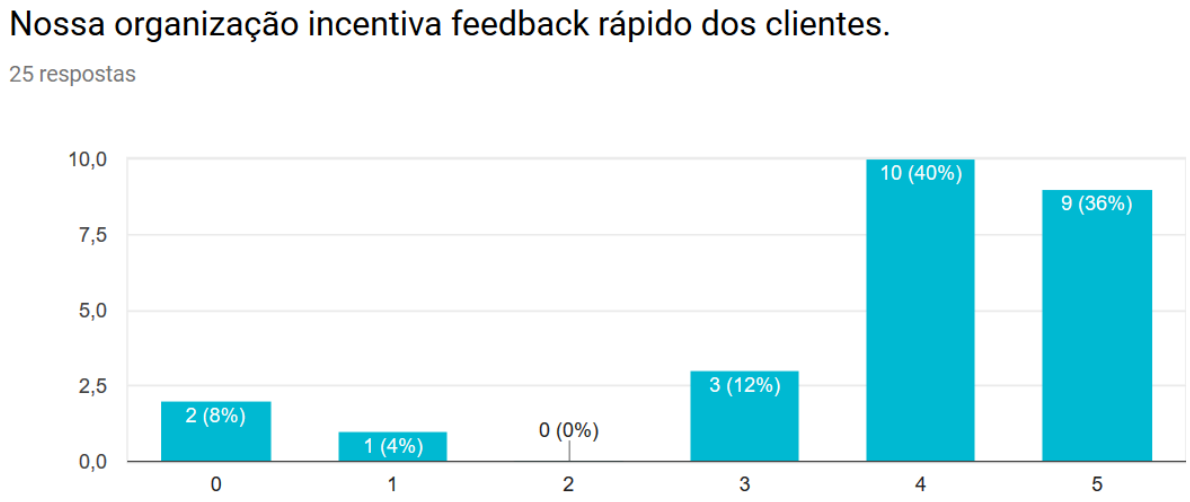


Figura 34: Incentivos a *feedbacks* rápidos do cliente
Fonte: Autoria Própria

Uma das características já comentadas sobre as práticas ágeis é o incentivo por mudanças nos requisitos. A questão ilustrada na Figura 35 requisitava o grau de prática desse incentivo pela organização. Os resultados mostram positivas respostas sobre esse incentivo (64%), porém essas tendem ao centro, no grau 3, com 32% sendo a maioria das seleções. O cenário que podemos observar é de que existe algum incentivo, mas não total, há um receio por alterações por parte das equipes e das organizações.

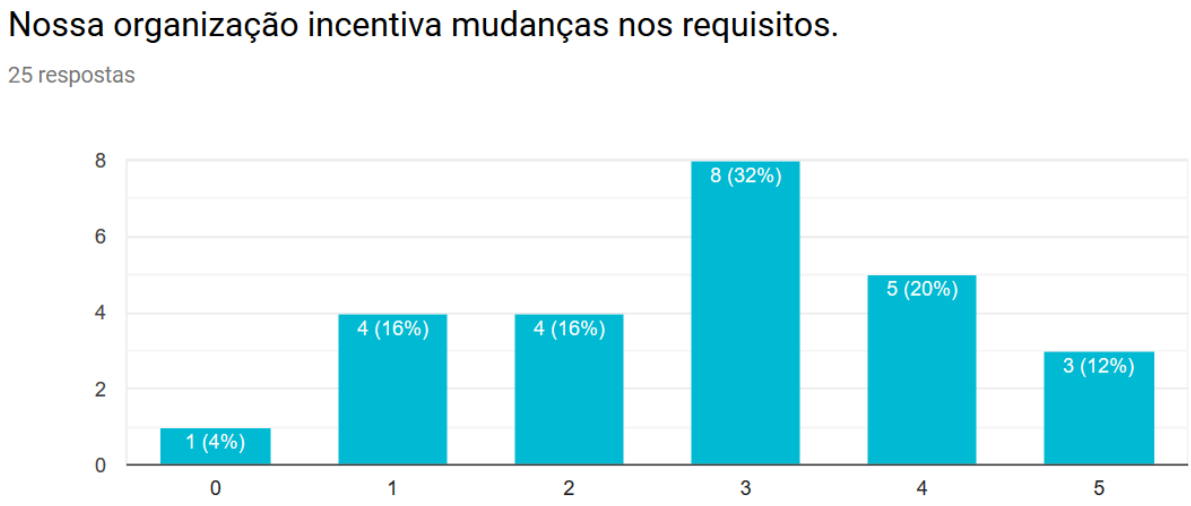


Figura 35: Incentivo à mudanças nos requisitos
Fonte: Autoria Própria

Três questões foram formuladas para descobrir se os prazos e entregas são bem definidos, quem é que define e quando são definidos. 84% dos participantes responderam de forma positiva sobre a definição, uma empresa respondeu como grau 0 e duas no grau 2, e podem ser observadas todas as respostas na Figura 36. Sobre quem define os prazos e entregas, foram dados quatro responsáveis para seleção: a equipe, o cliente, o responsável pela equipe e o responsável pelo projeto, além de uma opção de que os prazos e entregas não eram definidos, havia a possibilidade de os participantes inserirem novos responsáveis e selecionarem mais de uma opção. Nessa questão, ilustrada na Figura 37, a maioria respondeu a equipe (60%), seguido pelo responsável pelo projeto (56%) e o responsável pela equipe (44%). O cliente recebeu 36% e ninguém selecionou que os prazos e entregas não são definidos. Não houveram novas inserções de respostas.

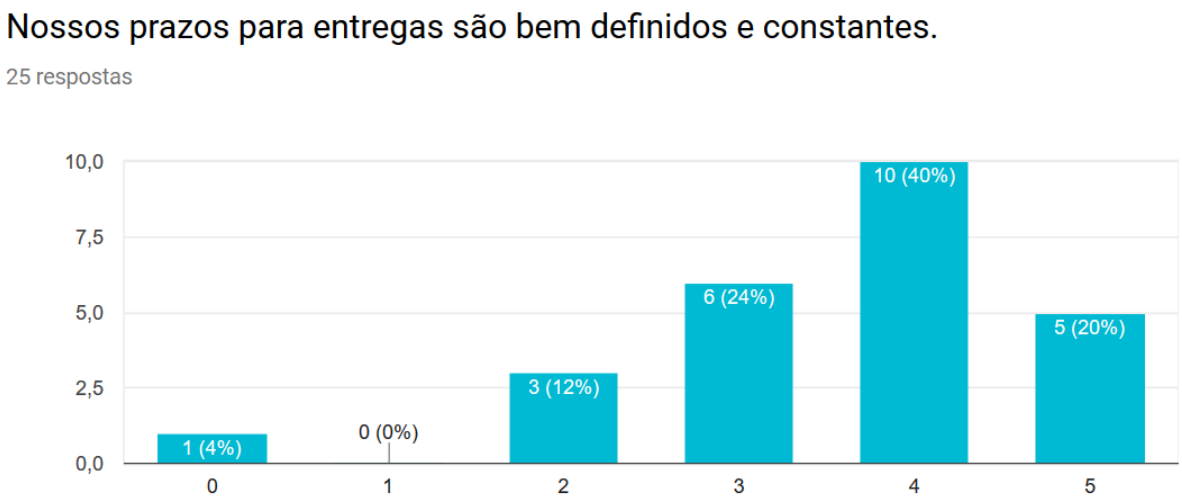


Figura 36: Prazos e entregas definidos e constantes

Fonte: Autoria Própria

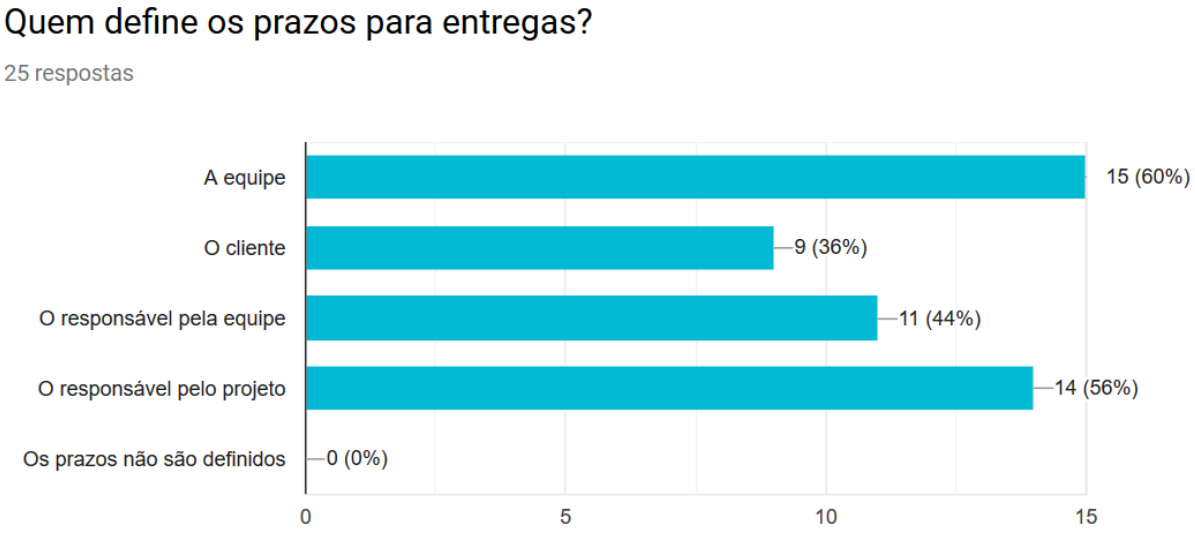


Figura 37: Quem define os prazos e as entregas

Fonte: Autoria Própria

Na questão sobre quando são definidos os prazos e entregas houveram duas respostas personalizadas pela opção "outros". As opções pré estabelecidas foram de que essa definição acontecia no início do desenvolvimento, durante o desenvolvimento, e ao término do desenvolvimento. A maioria das organizações definem no início do desenvolvimento (52%), seguido por durante o desenvolvimento (32%) e ao término do desenvolvimento (8%). Dois participantes não escolheram nenhuma das opções e detalharam como isso era feito nas suas organizações, as respostas selecionadas podem ser visualizadas na Figura 38

e as respostas personalizadas a seguir:

- "Definimos prazo somente para chamados pagos e o prazo é definido no momento da orçamentação."
- "Temos calendários de entregas, chamados ficam no *backlog* e são alocados dentro das versões disponíveis antes do desenvolvimento do mesmo."

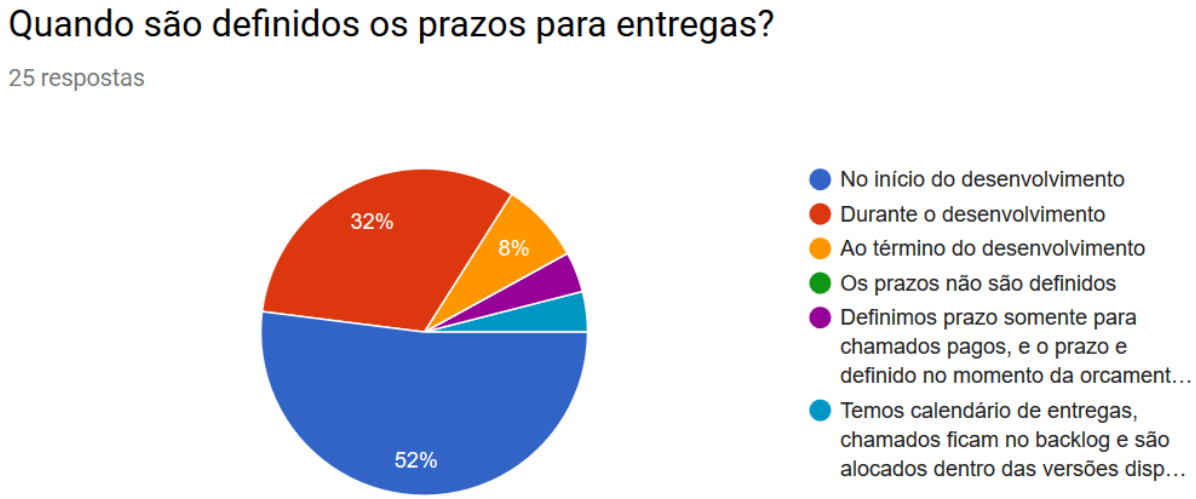


Figura 38: Quando são definidos os prazos e entregas
Fonte: Autoria Própria

Duas questões focaram em obter os aspectos positivos e negativos relacionados aos prazos e entregas. Essas questões não foram marcadas como obrigatórias e obtiveram 10 e 11 respostas respectivamente. Os aspectos positivos são descritos a seguir:

1. "É importante definir prazos para estabelecimento de metas e para mensurar a duração de projetos."
2. "Quando há clareza na necessidade a ser atendida e não ocorrem mudanças, temos conseguido cumprir os prazos."
3. "Prazos geralmente são atendidos, apesar de não ser muito assertivo nas estimativas. Isso acontece pois sempre é deixado uma margem de erro."
4. "Visibilidade da evolução do produto para a gestão e principalmente para o cliente."
5. "São negociados com base no escopo."

6. “Na grande maioria o prazo é extenso, tendo bastante tempo para um desenvolvimento calmo e seguro da aplicação.”
7. “E dado prioridade aos chamados pagos, e trabalhamos com agenda.”
8. “Descanso.”
9. “Comprometimento e agilidade.”
10. “Planejamento e antecipação de eventualidades.”

Os aspectos negativos obtidos são descritos a seguir:

1. “Ponto negativo é quando existem muitas alterações por parte dos clientes e atrasa os entregáveis.”
2. “Mudanças e interferências internas e não do cliente tem feito com que tempo e trabalho sejam desperdiçados. Alguns projetos são encarados como laboratórios, quando deveria se tomar mais tempo antes de uma decisão arquitetural que impactará profundamente no ciclo de desenvolvimento.”
3. “Erro de 40% acima do estimado. Geralmente os erros acontecem em atividades maiores/complexas.”
4. “Escopo pouco maleável.”
5. “Quando o escopo muda altera o prazo, mas não a data negociada para entrega.”
6. “Constante entrada de outras tarefas não previstas na *sprint*.”
7. “Em dados momentos os prazos são muito apertado, gerando um desenvolvimento apressado e com pouca ou má qualidade.”
8. “Dificuldades em alocar novos projetos a curto prazo.”
9. “A pressão.”
10. “Há muita mudança no decorrer do projeto em relação ao que foi planejado.”
11. “Tudo é urgente para cliente, a determinação de um prazo pode levar o cliente a pensar que não está sendo dada a devida importância para sua dor.”

Finalizando a seção, uma simples questão foi aplicada para saber se os prazos e entregas são problemas dentro da organização. As repostas foram bem divididas e podem ser visualizadas na Figura 39. 56% dos participantes responderam de forma positiva, ou seja, prazos e entregas se configuram problemas na organização. Os graus 3 e 4 são os mais selecionados, com 24% cada. Isso conclui de que nas empresas identificadas, os prazos e as entregas resultam também em problemas e obstáculos a serem superados no que se diz respeito ao sucesso dos produtos e dos processos.

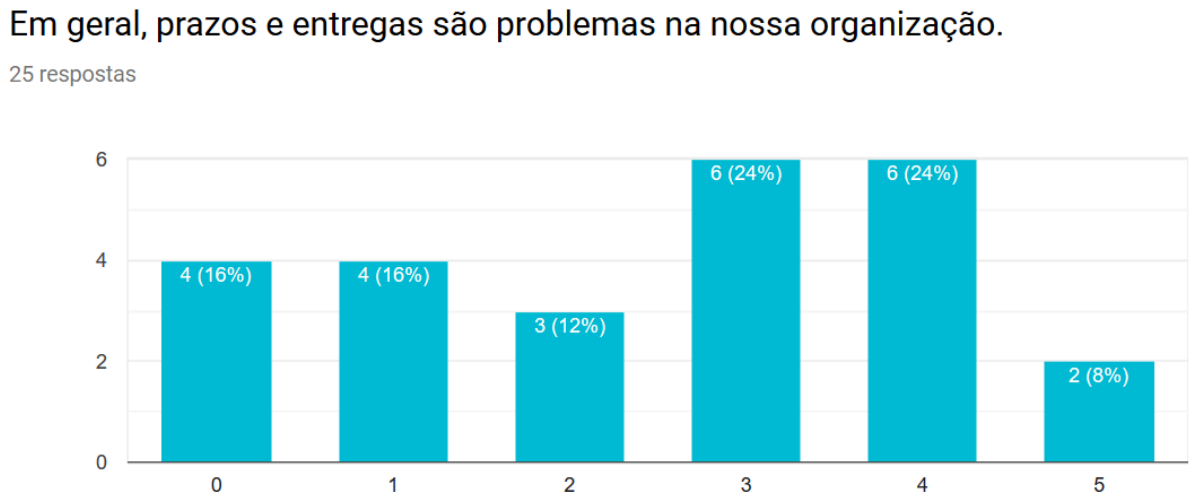


Figura 39: Prazos e entregas são problemas ou não para as organizações
Fonte: Autoria Própria

5.2 RESPOSTAS ÀS QUESTÕES DE PESQUISA E DISCUSSÕES

A *survey* desejava obter informações valiosas acerca das questões de pesquisa definidas inicialmente na Seção 4.1. A grande variedade de participantes e seus perfis dentro das organizações impulsionou o sucesso da pesquisa, contribuindo para o enriquecimento das informações recolhidas. Todos os participantes exerciam um cargo de liderança dentro das organizações, e muitos também desenvolviam. A maioria era composta por líderes de equipe, seguido por desenvolvedores e gestores de projetos. A liberdade e a confidencialidade foram características chaves para prover aos partícipes e às organizações o compromisso com a verdade e a transparência de respostas. De modo geral, as atividades desenvolvidas nessa pesquisa puderam contribuir de forma significativa para responder as QP's.

5.2.1 QUESTÃO DE PESQUISA 1

A primeira QP objetivava indagar sobre o cenário de metodologias ágeis nas empresas de desenvolvimento de software do sudoeste do Paraná. Todas as 46 questões, divididas em suas seções, apontaram e recolheram informações sobre as práticas ágeis das organizações da região. O termo "cenário" é amplo e não há uma resposta objetiva para clarificar e abordar a questão, mas sim, vários fenômenos descobertos por meio de todos estudo empírico realizado que formam uma parcela descoberta desse contexto.

Não há empresas com mais de 500 funcionários na região, facilitando a aplicação das metodologias ágeis em seus desenvolvimentos. Quase metade das empresas possuem até 50 funcionários, e de modo geral, há poucas equipes em cada organização com poucas pessoas em cada equipe e além disso, a maioria dos respondentes não participa de mais do que uma equipe de desenvolvimento. Esse é um cenário produtivo para o cumprimento das atividades ágeis. Quase todos os participantes da *survey* conhecem as metodologias ágeis há até 10 anos, e mais da metade há apenas 4 ou 5 anos, evidenciando uma baixa maturidade acerca desses conhecimentos.

As práticas exercidas pelas organizações condizem com os princípios ágeis. Todas as questões que envolviam a associação desses princípios com as suas práticas foram respondidas de forma positiva. A satisfação do cliente é prioridade nessas organizações, em contra partida observa-se que recursos além dos exigidos pelo cliente são desenvolvidos, prática que não é apoiada pelos ideais ágeis no desenvolvimento de software, e significam complexidade acima do essencial, tornando seus projetos e processos menos simples do que poderiam ser.

Todas as atividades que fazem parte de alguma metodologia ágil imposta previamente como opção de escolha aos participantes foram selecionadas. As reuniões diárias são as mais adeptas pelas organizações, além de reuniões constantes com os clientes e prototipagem ágil. Observa-se que todas as organizações praticam atividades ágeis em seus desenvolvimentos e os resultados mostram que essas atividades tem sido planejadas com datas e horas marcadas constantemente.

Uma característica das organizações que não condiz com os princípios ágeis é a não interação do cliente com os membros das equipes de desenvolvimento. A colaboração entre as partes citadas é realizada sem acompanhamento contínuo e muitos problemas podem resultar dessa prática. Do mesmo modo, a maioria dos participantes não enxerga na figura dos seus clientes motivação e a autoavaliação como importantes personagens

nos processos de desenvolvimento. Em contraposição, a grande maioria dos membros dos projetos trabalham juntos num mesmo ambiente geográfico, seguindo o que é recomendado pelas suas metodologias.

Uma característica interessante no cenário ágil na região é a forma com que as decisões dos desenvolvedores são tratadas pela administração das organizações. A administração tende a apoiar os desenvolvedores. Existem evidências de que os desenvolvedores têm voz ativa nos debates sobre as decisões a serem tomadas acerca dos projetos desenvolvidos. Do mesmo modo, os participantes apontaram que seus colaboradores possuem competência e experiência positivas e qualificadas.

A cultura organizacional tende a ser centrada mais na equipe do que no cliente, porém essa diferença é mínima e ambos os personagens são vistos com grande importância pelas organizações. Além disso, com base na visão dos participantes, a maioria das estruturas de gestão são pouco burocráticas, porém, a diferença poderia ser maior, mais de 1/3 dos participantes consideram as estruturas de gestão organizacionais da empresas burocráticas.

No cenário ágil, há mais interesse em avaliar os quesitos qualitativos em vez de quantitativos, entretanto no cenário da região a importância aos controles quantitativos são maiores com uma pequena margem de diferença. Deve-se levar em consideração a comparação entre os dois controles para buscar futuras alternativas de desenvolvimento.

Nos projetos e processos de desenvolvimento há mecanismos de apoio à comunicação rápida e eficaz entre as áreas de desenvolvimento, operações, suporte, clientes, gerenciamento e negócios. Na maioria dos casos essa comunicação é realizada pessoalmente, cara a cara, mesmo que a maioria das opções selecionadas sejam minimamente positivas. Essas são características muito importantes em relação aos princípios ágeis de desenvolvimento de software e pelas descobertas, as atividades são realizadas em sua maior parte de forma bem sucedida.

Os *feedbacks* rápidos são incentivados por quase a totalidade das organizações, isso aumenta o fluxo de troca de informações entre os envolvidos e leva a um bom andamento do projeto, contribuindo para a qualidade do produto final. Além disso, há incentivos das organizações para a mudança nos requisitos.

Os prazos e as entregas são bem definidos e constantes. Essa definição é realizada em sua maioria pela equipe e pelo responsável pelo projeto. O momento que ocorre a definição é em sua maioria no início do desenvolvimento, porém há uma grande parcela

que realiza durante a fase de produção.

As questões descritivas nos proporcionaram explorar alguns pontos positivos e negativos sobre as pessoas e papéis e prazos e entregas, porém não é possível confirmar a sua aplicabilidade de forma geral nas empresas. Todos esses aspectos descritos só são possíveis de confirmar no cenário específico da empresa que apontou. Todavia, os dados são interessantes e podem ser utilizados em futuros estudos. No geral, sobre os prazos e entregas constituírem um problema para as organizações, há um grande embate, divididas quase que na metade entre respostas positivas e negativas, o que concluí-se que há muitos problemas e obstáculos a serem combatidos nessa área.

5.2.2 QUESTÃO DE PESQUISA 2

A partir da análise sobre o cenário das práticas ágeis de desenvolvimento de software nas empresas identificadas, na seção anterior, nesta serão apontados os problemas e obstáculos que as empresas enfrentam. Assim sendo, objetiva-se contribuir para responder a segunda questão de pesquisa. Assim como na QP 1, essa questão é ampla e não pretende-se dar uma resposta totalmente objetiva, mas sim apontar os fenômenos descobertos no estudo.

É possível observar que os clientes estão afastados da equipe de desenvolvimento. Não há acompanhamento contínuo por parte dos futuros usuários no desenvolvimento do software. Os participantes demonstraram que não enxergam em seus clientes motivação sobre o produto e eles não se consideram personagens importantes nos processos de desenvolvimento. Essa questão é de extrema importância pois pode faltar com qualidade nos processos de correção e refinamento dos requisitos. Uma posição pode estar sendo tomada, indicando isso a questão sobre o incentivo aos *feedbacks* por parte da organização, porém podemos concluir que apenas esse incentivo não está sendo suficiente.

Os perfis dos participantes da pesquisa podem ser considerados como os mais providos de conhecimento técnico em suas equipes. A maioria é líder de equipe, gestor de projetos, gerente funcional e praticam em conjunto atividades de desenvolvimento. Com esse fato, podemos relacionar esses dados com os de tempo de conhecimento e prática dos mesmos participantes. 72% conhecem e praticam ágeis a 5 anos ou menos, demonstrando pouca maturidade. Pressupõe-se que os outros integrantes da equipe devam conhecer e praticar há ainda menos tempo. Com isso, podemos perceber que esse é um problema que tende a ser superado apenas com mais experiência e acúmulo de conhecimento por parte dos profissionais da região.

Na comparação dos princípios ágeis com as práticas nas organizações, a questão sobre propor um ambiente sustentável com patrocinadores, desenvolvedores e usuários manterem um ritmo constante indefinidamente, não resultaram em respostas tão positivas como se é esperado em uma organização que pratica os métodos ágeis. As respostas para essa questão se acumularam no nível 3, considerado positivo porém com possibilidade de melhorar. Essa é uma questão que merece atenção das organizações, o ambiente ágil sempre pode estar sendo estimulado e evoluído.

Na mesma seção sobre os princípios, foram notadas muitas respostas negativas sobre a prática de projetos, processos e abordagem simples nas metodologias de desenvolvimento, e que são implementados apenas os recursos exigidos pelos clientes. Quase metade avaliaram de forma negativa e 64% dentre as respostas positivas estão concentradas apenas no grau 3. Simplicidade tanto nas práticas de processos quanto nos produtos desenvolvidos é uma das principais características sobre ágeis e a baixa taxa de associação dessas características com as práticas reais na organização pode ser considerada um problema e um obstáculo a ser trabalho e superado.

Alguns dados que também chamam atenção é sobre as organizações possuírem uma estrutura de gestão burocrática. Mesmo que a maioria tenha selecionado graus negativos para essa afirmação, 36% respondeu de forma positiva. Essa questão merece atenção por parte dos administradores e estão relacionadas com a questão do ambiente sustentável não estar tão positivo quanto poderia. Agilidade, facilidade e eficácia são alguns dos antônimos da burocracia, e esses processos muito regrados podem impedir uma boa evolução da equipe, do desenvolvimento e das demais atividades.

Concluindo, um problema direto foi identificado sobre prazos e entregas. A maioria dos participantes concordou que os prazos e entregas são um problema nas suas organizações. Esses prazos e entregas são definidos na maioria dos produtos, no início ou durante o desenvolvimento, e geralmente é definido pela própria equipe. Esse obstáculo pode estar relacionado com todos os problemas e obstáculos até aqui encontrados, muitas más práticas em seus processos impactam diretamente nessa fase. Tendo em vista esse problema, alternativas devem ser tomadas e consequentemente a melhora dos produtos poderão ser percebidas significativamente. Prazos apertados e entregas má realizadas são características diretas de um produto sem qualidade, que não atende aos requisitos do cliente.

5.3 PERSPECTIVAS FUTURAS

Com relação aos estudos desenvolvidos, as informações captadas na *survey* e a experiência adquirida nas fases do projeto, algumas perspectivas futuras são apontadas sobre as práticas ágeis de desenvolvimento de software na região sudoeste paranaense. Alguns tópicos são explorados com a criação de questões discutindo possíveis avanços sobre as práticas na região.

- Maturidade dos profissionais: constatada uma imaturidade de ágeis na região, essa questão envolve o futuro dos profissionais com o ganho de experiência e o desenvolvimento de seus conhecimentos técnicos. Como será a evolução intelectual e o ganho de experiência dos profissionais da região? Qual será o impacto das instituições de ensino e das demais organizações para o desenvolvimento da área?

- Relação entre a academia e a indústria: visto a grande dificuldade em investigar as práticas ágeis da indústria na região, percebe-se o distanciamento entre as duas áreas. Como é possível unir a academia e a indústria, impulsionando o desenvolvimento intelectual e prático nas empresas? Como as universidades e outras organizações como APLs e incubadoras podem contribuir para melhorar essa relação?

- Mais pesquisas e o auxílio no contexto prático: havendo mais estudos e investigações, bem como utilizando da literatura para o auxílio do desenvolvimento de software na região, como esses trabalhos afetarão as organizações? As conclusões e intervenções propostas em estudos serão adotados pelas empresas? Como esse processo será realizado?

- Avanço tecnológico: como as tecnologias emergentes, como inteligência artificial e *big data* serão adotadas pelas empresas no uso das ferramentas de apoio aos processos ágeis? As empresas da região estão preparadas para receber essas intervenções?

- Aplicação do DevOps: como esforços para aplicar a agilidade nos times de operações, o DevOps ainda é recente no contexto prático de desenvolvimento de software no cenário mundial. Como essa prática será conduzida nas empresas da região sudoeste paranaense?

5.4 AMEAÇAS À VALIDADE

Se tratando desta uma pesquisa qualitativa na área de processos de desenvolvimento de software, Feldt e Magazinius (2010) cita que não há resultados específicos de engenharia de software e é necessário voltar para outros campos de estudo. A pesquisa qualitativa se estende entre o positivismo como um extremo e o "interpretativismo" como o outro (BELL; BRYMAN; HARLEY, 2018).

A diferença fundamental entre o positivismo e o "interpretativismo" reside na crença interpretivista de que os humanos diferem dos objetos de estudo nas ciências naturais, uma vez que a realidade social tem um maior significado para os seres humanos (BELL; BRYMAN; HARLEY, 2018). Os positivistas, por outro lado, veem os seres humanos como qualquer outra fonte de dados que pode ser sentida, medida e positivamente verificada (BELL; BRYMAN; HARLEY, 2018).

Para este trabalho, as ameaças à validade são baseadas na análise de Feldt e Magazinius (2010) acerca do pensamento positivista. Os critérios destacados para essa análise são: transferibilidade, credibilidade, confiabilidade e confirmabilidade.

- Transferibilidade: observa a relação de causa e efeito demonstrada, se esta é válida em outras situações e se pode generalizar os resultados e aplicá-los em outros contextos. Os resultados podem ser generalizados no ambiente de estudo, o sudoeste paranaense, pois os participantes foram diversos, de micro a grandes empresas, e os resultados são aplicados neste contexto. Houve grande variedade de respostas, englobando todo tipo de organização e fatores humanos e técnicos.

- Credibilidade: leva em consideração a veracidade das descobertas. Há uma legítima confiança de que as descobertas são verdadeiras pois em todo o processo de captação das informações houve liberdade individual de cada participante a contribuir de forma efetiva, sem pressão ou obrigação imposta. Todas as respostas foram feitas de forma livre e confidencial, propiciando credibilidade ao estudo.

- Confiabilidade: é questionado se as descobertas são consistentes e podem ser repetidas. A aplicação da pesquisa foi feita em um território limitado, no sudoeste paranaense, e a consistência das descobertas se comprovam pela metodologia com que a *survey* foi construída e aplicada. Além da anonimidade e não associação das informações captadas e quem concedeu, as empresas apontaram vários itens negativos em seus processos. Além disso, esta mesma pesquisa pode ser realizada em outro ambiente de

desenvolvimento, não havendo necessidade de alterar ou criar critérios de localidade, pessoas ou organizações.

- Confirmabilidade: é indagado sobre as descobertas serem moldadas pelos entrevistados em vez de pelo pesquisador. Para cobrir esse requisito, a *survey* para a captação das informações foi construída de forma neutra, dando liberdade ao respondente em concordar ou não com os pontos, além de não indicar uma resposta "correta" antecipadamente. Todas as questões são interpretativas por parte do entrevistado e não há nenhuma tendência às respostas.

6 TRABALHOS RELACIONADOS

Nesse capítulo serão apresentados alguns trabalhos relacionados à proposta principal desse projeto. Foram buscados em repositórios trabalhos que buscassem estudar e pesquisar o desenvolvimento de software ágil em algum meio de desenvolvimento, contribuindo de alguma forma para a qualidade dos produtos finais.

Em (DIEL et al., 2015) foi buscado descobrir quais as práticas e as habilidades necessárias de acordo com profissionais brasileiros. Foram identificados os entendimentos que os profissionais de TI têm sobre a metodologia ágil de desenvolvimento além de como e quais práticas ágeis estão sendo utilizadas e quais as habilidades necessárias para trabalhar com os métodos. Uma pesquisa baseada em questionários foi realizada e os resultados mostram que há fortes evidências de uma boa compreensão dos valores fundamentais de ágil, sendo *Scrum* e XP as metodologias mais populares e de que fatores culturais e comportamentais têm maior influência sobre a dificuldade de adotar agilidade do que fatores técnicos.

Nazir, Hasteer e Bansal (2016) organizaram uma pesquisa online na *web* para investigar algumas áreas da adoção de metodologias ágeis em empresas na Índia. Este trabalho direcionou seus objetivos cinco áreas, a saber: adoção, auto-organização, efeito do ágil, domínio da aplicação de práticas ágeis e interação entre as partes interessadas. Foi descoberto que a adoção da metodologia ágil afeta diretamente os custos, reduzindo-os e aumenta a produtividade e a satisfação das partes interessadas com os produtos desenvolvidos.

Foi feita uma revisão na literatura e uma pesquisa online com 52 empresas ágeis no Paquistão em (IQBAL; OMAR; YASIN, 2019). Foram descobertos alguns fatores principais que afetam as equipes de desenvolvimento ágil durante o processo de evolução do software, a saber: os membros da equipe e o líder, o relacionamento entre as equipes, requisitos de manipulação, velocidade da equipe, qualidade de conformidade e visão da equipe. O pesquisador utilizou de técnicas estatísticas para examinar os resultados, que

mostraram que a produtividade se correlaciona positiva e significativamente com a maioria dos fatores descobertos, exceto à aqueles com a correlação negativa.

Em (QUELAL; VILLAVICENCIO; MENDOZA, 2018) foi feito um estudo realizado no Equador sobre o uso, a utilidade e as causas de parar de usar as metodologias ágeis em organizações de médio e grande porte. Os resultados mostraram que muitos profissionais não recebem treinamentos antes da adoção das metodologias e que 56% das organizações decide abandonar. Algumas das razões percebidas para continuar usando metodologias ágeis são: a necessidade de controlar os processos de desenvolvimento e manutenção de software, a garantia de satisfação dos requisitos e a versatilidade, facilidade de entendimento e adaptabilidade das metodologias ágeis. Quelal, Villavicencio e Mendoza (2018) também recomendam uma estratégia para primeiramente assimilar as práticas ágeis por todos os *stakeholders*, fornecendo treinamento e acompanhamento.

A pesquisa em (BIN-HEZAM; BIN-ESSA; ABUBACKER, 2018) tentou estudar o atual estado do gerenciamento de projetos em pequenas e micro empresas na Arábia Saudita. Foi aplicado um questionário em 26 empresas que revelou que a maioria está seguindo os conceitos de ágil apesar de a metade não conhecerem os conceitos e as melhores práticas. Como resultado, foi descoberto que as metodologias ágeis são adequadas para esse tipo de empresa na Arábia Saudita.

Uma investigação por meio de pesquisa foi feita em (BOLLATI et al., 2017), com o objetivo de analisar o estado da prática ágil em empresas na Argentina. Foi comprovado que muitas empresas da região não adotavam práticas ágeis e muitas estão em fases iniciais de adoção. Uma das razões pela baixa adoção está a má interpretação de o que significa desenvolver com agilidade e que não existe muita capacitação disponível.

Foi realizada uma pesquisa para avaliar o nível de conhecimento e uso das metodologias ágeis e *Lean Startup* em (IBBA et al., 2018). A pesquisa foi feita em um contexto acadêmico industrial dentro de um projeto empreendedor da Universidade de Cagliari. Foram coletados dados de 30 estudos que envolviam mais de 151 pessoas. Como resultados foi mostrado que o conceito de "*Lean Startup*" é conhecido por uma alta porcentagem, e o conceito de ágil apenas por um quarto das *startups* envolvidas. Apesar disso, a maioria dos membros concordavam com os 12 princípios ágeis, e que essas metodologias, no contexto analisado, podem contribuir para o sucesso dos negócios.

7 CONSIDERAÇÕES FINAIS

Os métodos ágeis surgiram como uma alternativa aos métodos tradicionais pesados, e podem ser considerados resultados de uma evolução no desenvolvimento de software. O Manifesto Ágil, que destaca os valores e princípios proporcionam aos profissionais uma base para levarem aos seus projetos mais flexibilidade e eficiência em seus desenvolvimentos. Entretanto, muitos problemas têm sido identificados em cenários reais de desenvolvimento. Assim sendo, esforços nessa área são indispensáveis para diminuir esses problemas, identificar e superar os obstáculos.

O presente estudo foi realizado identificando sinais e fenômenos das práticas ágeis de desenvolvimento em empresas de uma região específica. O reconhecimento do cenário e a indicação de possíveis problemas e obstáculos são ideais para posteriormente construir alternativas para evolução dos processos de desenvolvimento como um todo, resultando assim na melhoria da qualidade final dos produtos desenvolvidos.

Outros estudos podem se apoiar em trabalhos como esse e continuar contribuindo na área de processos de desenvolvimento de software. A partir dos estudos empíricos e os resultados obtidos, as contribuições desse trabalho envolvem uma análise geral e um levantamento inicial sobre o cenário prático de desenvolvimento ágil do sudoeste paranaense, além da formação de recursos humanos na qual agregou aos envolvidos, adquirindo conhecimento técnico e prático.

Como trabalhos futuros, sugere-se mais estudos como esse em outras regiões, o aprimoramento da *survey* abordando mais áreas de conhecimento no contexto real de desenvolvimento das empresas, comparativos entre vários estudos da mesma espécie, um catálogo de recomendações na tentativa de sanar os problemas identificados e desenvolver ainda mais os processos, além da realização de publicações científicas em eventos de Engenharia de Software. Adicionalmente, deseja-se uma maior parceria entre a indústria e a academia.

REFERÊNCIAS

BABBIE, E. **Métodos de pesquisas de survey**. [S.l.]: Ed. da UFMG Belo Horizonte, 1999.

BASILI, V. R.; CALDIERA, G.; ROMBACH, D. H. The Goal Question Metric Approach. In: _____. [S.l.]: John Wiley & Sons, 1994. I.

BECK, K. et al. **Manifesto for Agile Software Development**. 2001. Disponível em: <http://agilemanifesto.org/iso/en/manifesto.html>.

BECK, K.; GAMMA, E. **Extreme programming explained: embrace change**. [S.l.]: addison-wesley professional, 2000.

BELL, E.; BRYMAN, A.; HARLEY, B. **Business research methods**. [S.l.]: Oxford university press, 2018.

BERMEJO, P. H. de S. et al. Agile Principles and Achievement of Success in Software Development: A Quantitative Study in Brazilian Organizations. **Procedia Technology**, v. 16, p. 718–727, 2014.

BIN-HEZAM, R.; BIN-ESSA, A.; ABUBACKER, N. F. Is the agile development method the way to go for small to medium enterprises (smes) in saudi arabia? In: **2018 21st Saudi Computer Society National Computer Conference (NCC)**. [S.l.: s.n.], 2018. p. 1–6.

BOEHM, B.; TURNER, R. **Balancing agility and discipline: A guide for the perplexed**. [S.l.]: Addison-Wesley Professional, 2003.

BOEHM, B. W. A spiral model of software development and enhancement. **Computer**, IEEE, v. 21, n. 5, p. 61–72, 1988.

BOLLATI, V. A. et al. The state of agile development adoption in argentine software companies. In: **2017 XLIII Latin American Computer Conference (CLEI)**. [S.l.: s.n.], 2017. p. 1–10.

BOOTLA, P.; ROJANAPORNPUN, O.; MONGKOLNAM, P. Necessary Skills and Attitudes for Development Team Members in Scrum:. p. 184–189, 2015.

COHN, M. **Product Backlog Refinement (Grooming)**. 2015.

DAVIS, J.; DANIELS, R. **Effective DevOps: Building a Culture of Collaboration, Affinity, and Tooling at Scale**. O'Reilly Media, 2016. ISBN 9781491926420. Disponível em: <https://books.google.com.br/books?id=nO1FDAAAQBAJ>.

DIEL, E. et al. What is agile, which practices are used, and which skills are necessary according to brazilian professionals: Findings of an initial survey. In: **2015 6th Brazilian Workshop on Agile Methods (WBMA)**. [S.l.: s.n.], 2015. p. 18–24.

ELORANTA, V.-P. et al. Scrum anti-patterns – an empirical study. In: **Asia-Pacific Software Engineering Conference (APSEC)**. [S.l.: s.n.], 2013. p. 503–510.

FELDT, R.; MAGAZINIUS, A. Validity threats in empirical software engineering research-an initial survey. In: **Seke**. [S.l.: s.n.], 2010. p. 374–379.

FERREIRA, R. H. M. et al. ARRANJO PRODUTIVO LOCAL - APL DE TECNOLOGIA DE INFORMAção - TI NO SUDOESTE DO PARANá: MUDANçAS PARADIGMáTICAS DA INOVAção à DIMENSão SOCIAL. **Redes**, APESC - Associacao Pro-Ensino em Santa Cruz do Sul, v. 20, n. 3, p. 241, dez. 2015. Disponível em: <https://doi.org/10.17058/redes.v20i3.3938>.

FERREIRA, R. H. M.; PICININ, C. T. Panorama das empresas de ti no sudoeste paranaense. **Congresso Internacional de Administração**, 2017.

GANDOMANI, J. et al. How Human Aspects Impress Agile Software Development Transition and Adoption. **International Journal of Software Engineering and Its Applications**, v. 8, n. 1, p. 129–148, 2014.

HODA, R.; SALLEH, N.; GRUNDY, J. The rise and evolution of agile software development. **IEEE Software**, v. 35, n. 5, p. 58–63, Sep. 2018. ISSN 0740-7459.

HTTERMANN, M. **DevOps for Developers**. Apress, 2012. (Expert's voice in Web development). ISBN 9781430245704. Disponível em: <https://books.google.com.br/books?id=JfUAkB8AA7EC>.

HUGHES, P. et al. **Agile Alliance**. 2018. Disponível em: <https://www.agilealliance.org/>.

IBBA, S. et al. Survey: How much the academic startups know and use agile software and lean startup methodologies? In: **Proceedings of the 19th International Conference on Agile Software Development: Companion**. New York, NY, USA: ACM, 2018. (XP '18), p. 2:1–2:3. ISBN 978-1-4503-6422-5. Disponível em: <http://doi.acm.org/10.1145/3234152.3234198>.

IHME, T. Scrum adoption and architectural extensions in developing new service applications of large financial IT systems. **Journal of the Brazilian Computer Society**, v. 19, n. 3, p. 257–274, 2013.

IPARDES, I. P. d. D. E. e. S. Arranjo produtivo local de software de pato branco, dois vizinhos e região sudoeste: estudo de caso. 2006.

IQBAL, J.; OMAR, M.; YASIN, A. An empirical analysis of the effect of agile teams on software productivity. In: **2019 2nd International Conference on Computing, Mathematics and Engineering Technologies (iCoMET)**. [S.l.: s.n.], 2019. p. 1–8.

JACOBSON, I.; BOOCH, G.; RUMBAUGH, J. **El proceso unificado de desarrollo de software/The unified software development process.** [S.l.]: Pearson Educación, 2000.

KAPITSAKI, G. M.; CHRISTOU, M. Where is scrum in the current agile world? In: **Proceedings of the 9th International Conference on Evaluation of Novel Approaches to Software Engineering**. [S.l.: s.n.], 2014. p. 101–108.

KHRAMTCHENKO, S. Comparing extreme programming and feature driven development in academic and regulated environments. **Feature Driven Development**, 2004.

KIM, G. et al. **The DevOps Handbook:: How to Create World-Class Agility, Reliability, and Security in Technology Organizations**. IT Revolution Press, 2016. (ITpro collection). ISBN 9781942788072. Disponível em: <https://books.google.com.br/books?id=ui8hDgAAQBAJ>.

LARMAN, C.; BASILI, V. R. Iterative and incremental developments. a brief history. **Computer**, v. 36, n. 6, p. 47–56, June 2003. ISSN 0018-9162.

LORBER, A. A.; MISH, K. D. How we successfully adapted agile for a research-heavy engineering software team. In: **2013 Agile Conference**. [S.l.: s.n.], 2013. p. 156–163.

LUCA, J. D.; CODE, P. **Feature Driven Development**. 2002. Disponível em: <http://www.featuredrivendevelopment.com/>.

LUSTENBERGER, F. On the fading of organizational interfaces in the era of internet-of-things. **IEEE Engineering Management Review**, IEEE, v. 44, n. 1, p. 14–15, 2016.

LóPEZ-MARTíNEZ, J. et al. Problems in the adoption of agile-scrum methodologies: A systematic literature review. In: **4th International Conference in Software Engineering Research and Innovation**. [S.l.: s.n.], 2016.

MELO, C. de O. et al. The evolution of agile software development in Brazil. **Journal of the Brazilian Computer Society**, v. 19, n. 4, p. 523–552, 2013.

MIDDLETON, P.; JOYCE, D. Lean software management: Bbc worldwide case study. **IEEE Transactions on Engineering Management**, IEEE, v. 59, n. 1, p. 20–32, 2012.

NAZIR, N.; HASTEER, N.; BANSAL, A. A survey on agile practices in the indian it industry. In: **2016 6th International Conference - Cloud System and Big Data Engineering (Confluence)**. [S.l.: s.n.], 2016. p. 635–640.

OUNSRIMUANG, P.; NOOTYASKOOL, S. Introducing scrum process optimization. In: **2017 International Conference on Machine Learning and Cybernetics (ICMLC)**. [S.l.: s.n.], 2017. v. 1, p. 175–181.

PETERSEN, K.; WOHLIN, C. A comparison of issues and advantages in agile and incremental development between state of the art and an industrial case. **Journal of systems and software**, Elsevier, v. 82, n. 9, p. 1479–1490, 2009.

PETERSEN, K.; WOHLIN, C. Measuring the flow in lean software development. **Software: Practice and experience**, Wiley Online Library, v. 41, n. 9, p. 975–996, 2011.

POPPENDIECK, M.; CUSUMANO, M. A. Lean software development: A tutorial. **IEEE software**, IEEE, v. 29, n. 5, p. 26–32, 2012.

PRESSMAN, R. S. **Engenharia de Software. Uma Abordagem Profissional**. Porto Alegre: AMGH, Edição: 7ª, 2011.

QUELAL, R. E.; VILLAVICENCIO, M.; MENDOZA, L. E. A survey of agile software development methodologies in ecuador. In: **2018 13th Iberian Conference on Information Systems and Technologies (CISTI)**. [S.l.: s.n.], 2018. p. 1–6.

RAMIREZ-MORA, S. L.; OKTABA, H. Productivity in agile software development: A systematic mapping study. In: **2017 5th International Conference in Software Engineering Research and Innovation (CONISOFT)**. [S.l.: s.n.], 2017. p. 44–53.

ROSENBERG, D.; STEPHENS, M. **Extreme programming refactored: the case against XP**. [S.l.]: Apress, 2008.

ROYCE, W. W. Managing the development of large software systems: concepts and techniques. In: IEEE COMPUTER SOCIETY PRESS. **Proceedings of the 9th international conference on Software Engineering**. [S.l.], 1987. p. 328–338.

SCHWABER, K. **The Scrum Framework Poster**. 2018. Disponível em: <https://www.scrum.org/resources/scrum-framework-poster>.

SCHWABER, K.; SUTHERLAND, J. **The Scrum Guide**. 2017.

SOARES, M. dos S. Comparação entre metodologias ágeis e tradicionais para o desenvolvimento de software. **INFOCOMP**, v. 3, n. 2, p. 8–13, 2004.

SOMMERVILLE, I. **Engenharia de Software**. São Paulo: Pearson, Edição: 9ª, 2011.

SRIVASTAVA, A.; BHARDWAJ, S.; SARASWAT, S. Scrum model for agile methodology. In: **2017 International Conference on Computing, Communication and Automation (ICCCA)**. [S.l.: s.n.], 2017. p. 864–869.

VIRMANI, M. Understanding devops & bridging the gap from continuous integration to continuous delivery. In: IEEE. **Fifth International Conference on the Innovative Computing Technology (INTECH 2015)**. [S.l.], 2015. p. 78–82.

WELLS, D. **XP - Unit Tests**. 1999. Disponível em: <http://www.extremeprogramming.org/rules/unittests.html>.

WOHLIN, C. et al. **Experimentation in Software Engineering**. [S.l.]: Springer Publishing Company, Incorporated, 2012. ISBN 3642290434, 9783642290435.

APÊNDICE A – QUESTIONÁRIO DE INFORMAÇÕES DE CONTATO

Informações de contato

Com o objetivo de controlar os respondentes desta pesquisa, foi elaborado um formulário independente não vinculado às respostas sobre as práticas ágeis, mantendo a confidencialidade das informações.

Juan Felipe Krassmann
krassmann@alunos.utfpr.edu.br

*Obrigatório

Nome *

Esta pergunta é obrigatória

Organização *

Esta pergunta é obrigatória

Cidade onde mora *

Esta pergunta é obrigatória

E-mail

Esta pergunta é obrigatória

Telefone

Esta pergunta é obrigatória

Deseja participar de uma entrevista online ou presencial? *

Marcar apenas uma oval.

- Sim
- Não

Esta pergunta é obrigatória

Deseja receber informações sobre os dados coletados nesta entrevista? *

Marcar apenas uma oval.

- Sim
- Não

Esta pergunta é obrigatória

Enviar

Nunca envie senhas pelo Formulários Google.

APÊNDICE B – QUESTIONÁRIO SOBRE AS PRÁTICAS ÁGEIS

Práticas ágeis no desenvolvimento de software

Essa pesquisa tem como objetivo coletar informações sobre as práticas de metodologias ágeis no desenvolvimento de software para a realização de um trabalho de conclusão do curso de Engenharia de Software da Universidade Tecnológica Federal do Paraná, campus Dois Vizinhos.

O tempo médio para a conclusão é de 10 minutos.

Responda com atenção, conforme sua experiência na organização.

Obrigado.

Juan Felipe Krassmann
krassmann@alunos.utfpr.edu.br

*Obrigatório

Quantos funcionários há em sua organização? *
Marcar apenas uma oval.
- ○ Menos de 10
- ○ 10 - 20
- ○ 21 - 50
- ○ 51 - 100
- ○ 101 - 500
- ○ Mais de 500

Esta pergunta é obrigatória
Quantas equipes de desenvolvimento há na sua organização? *
Marcar apenas uma oval.
- ○ 1
- ○ 2 - 5
- ○ 6 - 10
- ○ 11 - 20
- ○ Mais de 20

Esta pergunta é obrigatória
Qual/Quais metodologias ágeis são usadas na sua organização? *
Marque todas que se aplicam.
- ☐ Agile Unified Process (AUP)
- ☐ Dynamic Systems Development Method (DSDM)
- ☐ Extreme Programming (XP)
- ☐ Feature Driven Development (FDD)
- ☐ Lean Development
- ☐ Scrum
- ☐ Test Driven Development (TDD)
- ☐ Outro: _____

Esta pergunta é obrigatória
Há quanto tempo você possui conhecimentos sobre as práticas ágeis de desenvolvimento de software? *
Marcar apenas uma oval.
- ○ Menos de 1 ano
- ○ 1 - 3 anos
- ○ 4 - 5 anos
- ○ 6 - 10 anos
- ○ Mais de 10 anos
- ○ Não possuo conhecimento sobre as práticas ágeis

Esta pergunta é obrigatória
Há quanto tempo você desenvolve software utilizando os princípios ágeis? *
Marcar apenas uma oval.
- ○ Menos de 1 ano
- ○ 1 - 3 anos
- ○ 4 - 5 anos
- ○ 6 - 10 anos
- ○ Mais de 10 anos
- ○ Não desenvolvo software utilizando os princípios ágeis

Esta pergunta é obrigatória
Qual/Quais papéis você exerce na organização? *
Marque todas que se aplicam.
- ☐ Gerente Funcional

- ☐ Gestor de Projetos
- ☐ Líder da Equipe
- ☐ Desenvolvedor
- ☐ Testador
- ☐ Analista de Negócios
- ☐ Outro: [ ]

Esta pergunta é obrigatória

Você participa de mais de uma equipe de desenvolvimento? Se sim, quantas? *

Marcar apenas uma oval.

- ○ Não
- ○ Sim, duas
- ○ Sim, três
- ○ Sim, mais de três

Esta pergunta é obrigatória

A equipe de desenvolvimento na qual você faz parte possui quantas pessoas? *

Marcar apenas uma oval.

- ○ 2 - 5
- ○ 6 - 10
- ○ 11 - 20
- ○ Mais de 20

Esta pergunta é obrigatória

De 0 a 5, selecione o grau de prática dos seguintes princípios na sua organização.

Alterações são bem vindas, mesmo no fim do desenvolvimento. *

Marcar apenas uma oval.

	0	1	2	3	4	5	
Discordo	○	○	○	○	○	○	Concordo

Esta pergunta é obrigatória

Nós medimos e acompanhamos o progresso com base no software funcional. *

Marcar apenas uma oval.

	0	1	2	3	4	5	
Discordo	○	○	○	○	○	○	Concordo

Esta pergunta é obrigatória

Entregamos software funcionando com frequência, na escala de semanas até meses, com preferência a períodos mais curtos. *

Marcar apenas uma oval.

	0	1	2	3	4	5	
Discordo	○	○	○	○	○	○	Concordo

Esta pergunta é obrigatória

Pessoas relacionadas ao negócio e desenvolvedores trabalham em conjunto diariamente durante todo o curso do projeto. *

Marcar apenas uma oval.

	0	1	2	3	4	5	
Discordo	○	○	○	○	○	○	Concordo

Esta pergunta é obrigatória

Construímos projetos ao redor de indivíduos motivados. Damos a eles o ambiente e o suporte necessário e confiamos que eles façam seu trabalho. *

Marcar apenas uma oval.

	0	1	2	3	4	5	
Discordo	○	○	○	○	○	○	Concordo

Esta pergunta é obrigatória

Damos alta prioridade à satisfação dos clientes por meio da entrega precoce e contínua de software valioso. *

Marcar apenas uma oval.

	0	1	2	3	4	5	
Discordo	○	○	○	○	○	○	Concordo

Esta pergunta é obrigatória

O método mais eficiente para se transmitir informações em uma equipe de desenvolvimento é cara a cara. *

Marcar apenas uma oval.

	0	1	2	3	4	5	
Discordo	○	○	○	○	○	○	Concordo

Esta pergunta é obrigatória
Promovemos um ambiente sustentável. Nossos patrocinadores, desenvolvedores e usuários mantém um ritmo constante indefinidamente. *
Marcar apenas uma oval.

	0	1	2	3	4	5	
Discordo	○	○	○	○	○	○	Concordo

Esta pergunta é obrigatória
Nossa equipe de projeto de desenvolvimento de software possui atenção contínua para excelência técnica e bom design para desenvolvimento. *
Marcar apenas uma oval.

	0	1	2	3	4	5	
Discordo	○	○	○	○	○	○	Concordo

Esta pergunta é obrigatória
Praticamos projetos, processos e abordagens simples em nossas metodologias de desenvolvimento de software. Nós implementamos recursos que são exigidos pelos clientes, nada mais. *
Marcar apenas uma oval.

	0	1	2	3	4	5	
Discordo	○	○	○	○	○	○	Concordo

Esta pergunta é obrigatória
Nossas equipes de desenvolvimento são auto gerenciáveis. Elas têm a capacidade de se organizar continuamente em diferentes configurações para atender aos requisitos em constante mudança e aos novos desafios que surgem. *
Marcar apenas uma oval.

	0	1	2	3	4	5	
Discordo	○	○	○	○	○	○	Concordo

Esta pergunta é obrigatória
Em intervalos regulares, nossa equipe reflete em como se tornar mais eficiente e realiza ajustes. *
Marcar apenas uma oval.

	0	1	2	3	4	5	
Discordo	○	○	○	○	○	○	Concordo

Esta pergunta é obrigatória

Sobre atividades realizadas na sua equipe e organização:

Selecione as atividades que são realizadas pela sua equipe. *
Marque todas que se aplicam.
- ☐ Prototipagem ágil
- ☐ Reuniões constantes com o cliente
- ☐ Programação em pares
- ☐ Reuniões diárias
- ☐ Reuniões de retrospectiva de entregas
- ☐ Reuniões de revisão
- ☐ Outro: _____

Esta pergunta é obrigatória
As atividades são feitas constantemente, com data e horas marcadas. *
Marcar apenas uma oval.

	0	1	2	3	4	5	
Discordo	○	○	○	○	○	○	Concordo

Esta pergunta é obrigatória

Sobre pessoas e papéis:

Em nossos projetos, os clientes colaboram de perto com os membros da equipe de desenvolvimento. *
Marcar apenas uma oval.

	0	1	2	3	4	5	
Discordo	○	○	○	○	○	○	Concordo

Esta pergunta é obrigatória
Em nossos projetos de desenvolvimento de software, os clientes estão comprometidos com o projeto, ou seja, são motivados, ativos e consideram-se elementos responsáveis do projeto. *
Marcar apenas uma oval.

0 1 2 3 4 5

Discordo ○ ○ ○ ○ ○ ○ Concordo

Esta pergunta é obrigatória

Os membros do nosso projeto estão geograficamente próximos. *

Marcar apenas uma oval.

0 1 2 3 4 5

Discordo ○ ○ ○ ○ ○ ○ Concordo

Esta pergunta é obrigatória

Nossa administração tem a cultura para apoiar as decisões dos desenvolvedores. *

Marcar apenas uma oval.

0 1 2 3 4 5

Discordo ○ ○ ○ ○ ○ ○ Concordo

Esta pergunta é obrigatória

Nossa cultura organizacional é centrada na equipe. *

Marcar apenas uma oval.

0 1 2 3 4 5

Discordo ○ ○ ○ ○ ○ ○ Concordo

Esta pergunta é obrigatória

Nossa cultura organizacional é centrada no cliente. *

Marcar apenas uma oval.

0 1 2 3 4 5

Discordo ○ ○ ○ ○ ○ ○ Concordo

Esta pergunta é obrigatória

Nossa equipe geralmente consiste de pessoas tecnicamente competentes e experientes (que desenvolveram software semelhante no passado, têm experiência prática no domínio da tecnologia). *

Marcar apenas uma oval.

0 1 2 3 4 5

Discordo ○ ○ ○ ○ ○ ○ Concordo

Esta pergunta é obrigatória

Nossa organização possui uma estrutura de gestão burocrática. *

Marcar apenas uma oval.

0 1 2 3 4 5

Discordo ○ ○ ○ ○ ○ ○ Concordo

Esta pergunta é obrigatória

Informe os papéis definidos na sua equipe.

Esta pergunta é obrigatória

Cite alguns aspectos POSITIVOS sobre a equipe e seus papéis.

Esta pergunta é obrigatória

Cite alguns aspectos NEGATIVOS sobre a equipe e seus papéis.

Esta pergunta é obrigatória

Sobre ferramentas e frameworks:

De 0 a 5, qual importância é dada a controles QUALITATIVOS do software? *

Marcar apenas uma oval.

0 1 2 3 4 5

Discordo ○ ○ ○ ○ ○ ○ Concordo

Esta pergunta é obrigatória

De 0 a 5, qual importância é dada a controles QUANTITATIVOS do software? *

Marcar apenas uma oval.

0 1 2 3 4 5

Discordo ○ ○ ○ ○ ○ ○ Concordo

Esta pergunta é obrigatória

Nossos projetos possuem mecanismos que permitem que o pessoal se comunique e negocie com rapidez e eficiência com as áreas de desenvolvedores, operações, suporte, clientes, gerenciamento e negócios. *

Marcar apenas uma oval.

0 1 2 3 4 5

Discordo ○ ○ ○ ○ ○ ○ Concordo

Esta pergunta é obrigatória

Na maioria dos casos, a comunicação e a negociação em nossos projetos são feitas cara a cara. *

Marcar apenas uma oval.

0 1 2 3 4 5

Discordo ○ ○ ○ ○ ○ ○ Concordo

Esta pergunta é obrigatória

Cite algumas ferramentas e frameworks utilizados por sua equipe.

Esta pergunta é obrigatória

Sobre prazos e entregas:

Nossa organização incentiva feedback rápido dos clientes. *

Marcar apenas uma oval.

0 1 2 3 4 5

Discordo ○ ○ ○ ○ ○ ○ Concordo

Esta pergunta é obrigatória

Nossa organização incentiva mudanças nos requisitos. *

Marcar apenas uma oval.

0 1 2 3 4 5

Discordo ○ ○ ○ ○ ○ ○ Concordo

Esta pergunta é obrigatória

Nossos prazos para entregas são bem definidos e constantes. *

Marcar apenas uma oval.

0 1 2 3 4 5

Discordo ○ ○ ○ ○ ○ ○ Concordo

Esta pergunta é obrigatória

Quando são definidos os prazos para entregas? *

Marcar apenas uma oval.

- ○ No início do desenvolvimento
- ○ Durante o desenvolvimento
- ○ Ao término do desenvolvimento
- ○ Os prazos não são definidos
- ○ Outro: _____

Esta pergunta é obrigatória

Quem define os prazos para entregas? *

Marque todas que se aplicam.

- ☐ A equipe
- ☐ O cliente
- ☐ O responsável pela equipe
- ☐ O responsável pelo projeto
- ☐ Os prazos não são definidos

- Outro:

Esta pergunta é obrigatória
Cite alguns aspectos POSITIVOS relacionados a prazos e entregas.

Esta pergunta é obrigatória
Cite alguns aspectos NEGATIVOS relacionados a prazos e entregas.

Esta pergunta é obrigatória
Em geral, prazos e entregas são problemas na nossa organização. *
Marcar apenas uma oval.

	0	1	2	3	4	5	
Discordo							Concordo

Esta pergunta é obrigatória

Enviar

Nunca envie senhas pelo Formulários Google.