UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO SOCIOECONÔMICO
DEPARTAMENTO DE CIÊNCIAS DA ADMINISTRAÇÃO

Vinicius Inácio da Silva

Análise de Metodologias Ágeis de Desenvolvimento de Software: As vantagens e desvantagens do Scrum e Extreme Programming (XP)

Florianópolis

2022

Vinicius Inácio da Silva

Análise de Metodologias Ágeis de Desenvolvimento de Software: As vantagens e desvantagens do Scrum e Extreme Programming (XP)

Trabalho Conclusão do Curso de Graduação em Administração do Centro Socioeconômico da Universidade Federal de Santa Catarina, requisito para a obtenção do título de Bacharel em Administração.
Enfoque: Monográfico - Artigo
Área de concentração: Tecnologia da Informação
Orientador: Prof. Dr. Mário de Souza Almeida

Florianópolis

2022

Catalogação na fonte elaborada pela biblioteca da Universidade Federal de Santa Catarina

Silva, Vinicius Inacio
Análise de Metodologias Ágeis de Desenvolvimento de Software : As vantagens e desvantagens do Scrum e Extreme Programming (XP) / Vinicius Inacio Silva ; orientador, Mário de Souza Almeida, 2022.
41 p.

Trabalho de Conclusão de Curso (graduação) - Universidade Federal de Santa Catarina, Centro Sócio Econômico, Graduação em Administração, Florianópolis, 2022.

Inclui referências.

1. Administração. 2. Administração. 3. Metodologias Ágeis. 4. Tecnologia. 5. Software. I. de Souza Almeida, Mário. II. Universidade Federal de Santa Catarina. Graduação em Administração. III. Título.

Vinicius Inácio da Silva

Análise de Metodologias Ágeis de Desenvolvimento de Software: As vantagens e desvantagens do Scrum e Extreme Programming (XP)

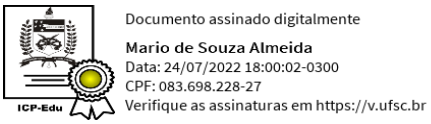
Este Trabalho de Curso foi julgado adequado e aprovado na sua forma final pela Coordenadoria. Trabalho de Curso do Departamento de Ciências da Administração da Universidade Federal de Santa Catarina.

Florianópolis, 14 de Julho de 2022.

_____

Profª. Ana Luiza Paraboni
Coordenadora de Trabalho de Curso

Avaliadores:



_____

Prof. Mário de Souza Almeida, Dr. - Orientador
Universidade Federal de Santa Catarina

_____

Prof. Marcos Baptista Lopez Dalmau, Dr. - Avaliador
Universidade Federal de Santa Catarina

_____

Prof. Pedro Antônio de Melo, Dr. - Avaliador
Universidade Federal de Santa Catarina

RESUMO

De acordo com as constantes mudanças no mercado de tecnologia, cresce a busca das organizações por formas de melhorar seus processos de desenvolvimento de software, visando resultados melhores para obter diferenciação no mercado e vantagens competitivas. Neste contexto surgem as metodologias ágeis de desenvolvimento. Este trabalho tratará das **vantagens e desvantagens das metodologias ágeis de desenvolvimento de software Scrum e Extreme Programming**. Por meio de uma pesquisa exploratória e descritiva, com abordagem qualitativa, foram realizadas entrevistas com 4 gestores de projetos de software com mais de 1 ano de experiência profissional na área. Os resultados evidenciaram as diferenças nas vantagens e desvantagens das metodologias ágeis abordadas, bem como uma relação de completividade entre elas. Para implementar processos de metodologias ágeis nos times de desenvolvimento de software, recomenda-se atenção aos processos que envolvem mudanças nos comportamentos do time.

Palavras-chave: Software, Desenvolvimento de Software, Metodologias Ágeis, Scrum, Extreme Programming

ABSTRACT

According to the constant changes in the technology market, organizations are looking for ways to improve their software development processes, aiming at better results to obtain market differentiation and competitive advantages. In this context, agile development methodologies arise. This work will address the **advantages and disadvantages of agile software development methodologies Scrum and Extreme Programming**. Through an exploratory and descriptive research, with a qualitative approach, interviews were carried out with 4 software project managers with more than 1 year of professional experience in the area. The results evidenced the differences in the advantages and disadvantages of the approached agile methodologies, as well as a relation of completeness between them. To implement agile methodologies processes in software development teams, attention is recommended to processes that involve changes in team behaviors.

Keywords: Software, Software Development, Agile Methodologies, Scrum, Extreme Programming

1 INTRODUÇÃO

É evidente que os softwares representam a tecnologia mais notável no cenário mundial. Porém, anos atrás seria impossível prever que eles se tornaram imprescindíveis para os diversos âmbitos econômicos de toda a cadeia produtiva, se espalhando pelo mundo de maneira exponencial e permitindo o compartilhamento de informações de maneira quase instantânea (PRESSMAN, 2011).

Os softwares estão incorporados nas atividades cotidianas há décadas, apesar de serem sistemas abstratos e intangíveis. Independentemente da idade, todas as pessoas do mundo industrializado impactam ou são impactadas pelo seu uso, direta ou indiretamente. Seja na visita ao médico, ao assistir televisão, ao consumir produtos de origem animal e vegetal, a sociedade moderna depende e não poderia existir sem o uso de sistemas computacionais (SOMMERVILLE, 2004).

Com o avanço da tecnologia e aumento do interesse das organizações nas inovações que estas soluções poderiam proporcionar, houve a necessidade de evolução nos métodos de desenvolvimento de software, que tem como característica sua complexidade e a dependência de pessoas para deliberar e julgar as decisões, assim como em outros processos de criação. Além disso, a função exercida pelo software sofreu importantes alterações ao longo do tempo, com evoluções consideráveis de performance, embaladas por novas descobertas que permitiram a criação de sistemas mais refinados e complexos como resultado (PRESSMAN, 2011).

Um outro aspecto a ser observado é que softwares complexos podem trazer excelentes resultados financeiros para as organizações quando são usados corretamente, entretanto, seu processo de desenvolvimento pode gerar adversidades se não mapeados os riscos corretamente (PRESSMAN, 2011).

Sendo assim, no mercado extremamente competitivo e volátil de desenvolvimento de produtos digitais, a agilidade e adaptação se tornaram aspectos imprescindíveis para empresas e produtos de sucesso. A cada dia, mais organizações percebem que, para se manter em uma posição de vantagem frente à seus concorrentes, é necessário usar abordagens de desenvolvimento que se enquadram ao comportamento do mercado (TAKEUCHI e NONAKA, 1986).

Por operarem neste ambiente com mudanças rápidas, as organizações precisam responder a novas oportunidades, mercados, a alterações nas condições econômicas e ao surgimento de produtos e serviços concorrentes. Softwares fazem parte de quase todas as

operações de negócios, assim, novos softwares são desenvolvidos rapidamente para obterem proveito de novas oportunidades e responder às pressões competitivas. O desenvolvimento e entrega rápidos são, portanto, os requisitos mais críticos para o desenvolvimento de sistemas de software. Desta forma, muitas empresas estão dispostas a trocar a qualidade e o compromisso com requisitos do software por uma implantação mais rápida do software que necessitam (SOMMERVILLE, 2004).

De acordo com Sutherland (2010), um processo de desenvolvimento engessado e relutante a mudanças resulta em sistemas medíocres, pois, ao definir seus requisitos e padrões de funcionamento como imutáveis, o sistema é fadado a ser tão bom quanto a ideia inicial, sem a possibilidade melhorias geradas pelos aprendizados e descobertas desse processo.

Estes fatores criaram a necessidade da adaptação dos processos de desenvolvimento de software, dando origem a metodologias que focam mais na interatividade, criatividade e liberdade no desenvolvimento, na busca de melhores resultados e performance, as denominadas metodologias ágeis. Estas metodologias têm como principais características a entrega de valor e mudanças rápidas no processo de desenvolvimento. Dentre as diversas metodologias ágeis de desenvolvimento existentes, duas das mais difundidas são: Scrum e Extreme Programming (XP).

No sentido dos aspectos abordados anteriormente, se apresenta a pergunta de pesquisa: “Quais as vantagens e desvantagens das metodologias ágeis de desenvolvimento de software Scrum e Extreme Programming?”.

1.1 Objetivos

O objetivo geral deste estudo é avaliar as vantagens e desvantagens das metodologias ágeis de desenvolvimento de software Scrum e Extreme Programming.

Para alcançar o objetivo geral foram escolhidos os seguintes objetivos específicos:

a) Apresentar as vantagens e desvantagens da metodologia Scrum, de acordo com os gestores de projetos de software.
b) Apresentar as vantagens e desvantagens, da metodologia Extreme Programming, de acordo com os gestores de projetos de software.
c) Fazer uma análise comparativa das metodologias apresentadas.

1.2 Justificativas

Ao observar a dificuldade das equipes de desenvolvimento de software para se adaptarem ao uso de metodologias ágeis e o crescimento do mercado de software nos últimos anos, notou-se a oportunidade de criar um projeto de pesquisa que contribuísse para a disseminação destes métodos, visando aumentar a sua adoção nas organizações para obtenção de melhores resultados nos processos de desenvolvimento.

Além da oportunidade observada, existe a motivação do autor, que atua em uma equipe de desenvolvimento de software e tem contato e admiração pela história, uso e impacto das metodologias ágeis de desenvolvimento nas organizações tecnológicas.

2 FUNDAMENTAÇÃO TEÓRICA

Ao longo deste capítulo são apresentados os assuntos norteadores a respeito de desenvolvimento de software e das metodologias ágeis, permitindo entender o que outros autores relatam sobre os seguintes temas: Software, Metodologias de desenvolvimento de software, Metodologias ágeis, Scrum e Extreme Programming.

2.1 Software

De acordo com Pressman (2011), um software pode ser constituído principalmente de 3 características básicas, sendo elas:

a) Instruções (programas de computador) que, quando executadas, fornecem características, funções e desempenho desejados;
b) Estruturas de dados que possibilitam aos programas manipular informações adequadamente; e
c) Informação descritiva, tanto na forma impressa como na virtual, descrevendo a operação e o uso dos programas.

Por outro lado, Sommerville (2004, p. 2) descreve os sistemas de softwares como “abstratos e intangíveis”, afirmando que estes sistemas não podem ser limitados pelas características dos materiais e nem geridos pelas propriedades físicas ou procedimentos de produção. Este fato simplifica o processo de criação de softwares, porque elimina as barreiras para as oportunidades de desenvolvimento.

Um outro aspecto a ser abordado é que a falta de fronteiras para balizar esse desenvolvimento pode ter o efeito contrário, tornando os sistemas labirínticos e, consequentemente, onerosos em caso de alterações (SOMMERVILLE, 2004).

De acordo com Pressman (2011), os softwares representam o resultado do empenho das equipes de desenvolvimento em algum processo, o qual as mesmas equipes darão apoio ao longo do tempo para manter seu funcionamento. Desta forma, eles englobam sistemas executados em dispositivos de qualquer tamanho ou estrutura, que exibem assuntos de maneira virtual ou impressa.

Além disso, Sommerville (2004) afirma que quando criados de maneira profissional os softwares são, frequentemente, mais que apenas um programa, representando na verdade um conjunto deles, que atuam separadamente para atender ao objetivo principal. Estes programas incluem: As informações relacionadas ao sistema, que descrevem sua organização; as informações direcionadas ao cliente final, que ensinam o usuário como utilizar o sistema e endereços na web, para que os usuários possam se atualizar com informações recentes relacionadas ao produto.

Sendo assim, quando se trata de softwares e sua construção, não se deve abordar apenas o sistema em si, mas sim toda a documentação que o integra e as informações usadas para sua manutenção, garantindo que ele opere da maneira esperada (SOMMERVILLE, 2004).

Para Pressman (2011), os softwares compartilham o item mais relevante desta época - a informação. Estes sistemas são responsáveis por modificar dados pessoais e torná-los mais proveitosos em circunstâncias diferentes, como os sistemas que fazem cruzamento de dados e conseguem identificar comportamentos relacionados ao crescimento de determinadas doenças, por exemplo. Ademais, estes sistemas de informação contribuem para que as organizações alcancem vantagens competitivas e tenham acesso ao maior portal de informação existente, que é a internet.

De acordo com Pressman (2011), os softwares são divididos em 7 grandes categorias, sendo elas:

a) **Software de sistema** - Sua principal característica é a alta interatividade com o hardware, que representa os componentes físicos do computador. Além disso, os softwares de sistema possuem diversas interfaces externas.
b) **Software de aplicação** - Estes representam os programas que são fabricados de acordo com a demanda determinada pelo cliente, para atender alguma exigência observada. Os sistemas de aplicação são utilizados na maioria das

vezes para administração em tempo real de setores das organizações, permitindo a observação de informações de maneira confiável e desburocratizada.

c) **Software científico/de engenharia** - Este tipo de software tem como atributo o tratamento massivo de dados numéricos, sendo usados em todos os ramos que envolvem a pesquisa e produção de conteúdo científico.
d) **Software embutido** - Os softwares embutidos estão presentes em diversos produtos do dia a dia, estes são configurados para executar e administrar comportamentos pré-definidos para os usuários ou permitir que o usuário tenha controle sobre determinados aspectos do produto. Nesse caso, um exemplo são os botões da televisão que proporcionam ao usuário escolhas pré-determinadas de canais, volume etc.
e) **Software para linha de produtos** - Neste caso, o objetivo principal é equipar os usuários com uma ferramenta que possibilita uma capacidade específica, seja ela de processamento, controle ou qualquer outra finalidade.
f) **Aplicações para a Web** - Esta categoria se caracteriza por informações provenientes de arquivos, que, ao se conectarem, mostram informações de texto e gráficos para os usuários, em páginas da web. Estas aplicações permitem que o usuário consiga utilizar suas funções sem instalar um software em sua máquina, processando em servidores terceiros.
g) **Software de inteligência artificial** - Nesta categoria se enquadram os sistemas que utilizam algoritmos que não são provenientes de números para abordar situações difíceis de serem processados analisando diretamente. Estas tecnologias buscam também compreender, imitar e superar a inteligência humana em alguns aspectos.

Gudwinet (2018) também aponta a existência de diferentes tipos de software, acarretando a necessidade de diferentes metodologias de desenvolvimento para sua criação.

2.2 Metodologias de Desenvolvimento de Software

Um dos mais relevantes aprendizados que a área de pesquisa em metodologias de desenvolvimento de software teve foi o aumento na conscientização de que a concepção

de software é um processo complicado. Desta forma, este processo não engloba apenas o uso de códigos e ferramentas que funcionam e sim outros fatores externos que tornam este empenho coletivo, complexo e criativo. Como resultado, a qualidade do sistema desenvolvido varia de acordo com as pessoas, organizações e processos que foram utilizados para criá-lo (FUGGETTA, 2000).

Sendo classificado como complicado e complexo, é exigido o controle apropriado dos fatores que o influenciam, por isso organizações que são expostas às mudanças no ambiente e buscam se adaptar a elas são mais favorecidas pela evolução (SCHWABER, 1997).

Sommerville (2004) defende que, apesar dos diferentes tipos de sistemas de informação, 4 atividades estão envolvidas e são imprescindíveis para todos eles, sendo elas:

a) **Especificação de software**: Atividade em que os usuários finais e o time responsável pela produção determinam as características do sistema que será produzido, bem como as limitações de seu funcionamento;
b) **Desenvolvimento de software**: Atividade em que a equipe atua na criação efetiva do software e torna reais as características definidas anteriormente;
c) **Validação de software**: Atividade em que o software é examinado pelo usuário, para constatar se ele atende às expectativas do cliente;
d) **Evolução de software**: Atividade em que o software é alterado para reproduzir as variações do cliente e do mercado.

Além das 4 atividades abordadas, Sommerville (2004, p.19) afirma que existem fundamentos do processo de construção de sistemas que se enquadram a todos os tipos abordados:

> a. Eles devem ser desenvolvidos em um processo gerenciado e compreendido. A organização que desenvolve o software deve planejar o processo de desenvolvimento e ter ideias claras do que será produzido e quando estará finalizado.
> b. Confiança e desempenho são importantes para todos os tipos de sistemas. O software deve se comportar conforme o esperado, sem falhas, e deve estar disponível para uso quando requerido. Deve ser seguro em sua operação e deve ser, tanto quanto possível, protegido contra ataques externos.
> c. É importante entender e gerenciar a especificação e os requisitos de software (o que o software deve fazer). Você deve saber o que clientes e usuários esperam dele e deve gerenciar suas expectativas para que um sistema útil possa ser entregue dentro do orçamento e do cronograma.
> d. Você deve fazer o melhor uso possível dos recursos existentes. Isso significa que, quando apropriado, você deve usar o software já

> desenvolvido, em vez de escrever um novo. Essas noções básicas de processo, confiança, requisitos, gerenciamento e reuso são temas importantes desta obra.

Os processos tradicionais consideram que o desenvolvimento pode ser feito com atividades fáceis de caracterizar, recorrentes e previsíveis. Porém, diferente de outros meios de produção, a criação de software exige criatividade e compreende um elevado grau de incerteza (AMBLER e LINES, 2012).

Sendo assim, metodologias que focam nesta abordagem sequencial, em que os requisitos são completamente especificados, projetados, construídos e testados, não estão de acordo com o desenvolvimento acelerado de software pois, ao observarem a necessidade de alterações nos requisitos, todo o processo precisaria ser feito ou testado novamente. Estas características de gestão resultam em um retardamento na finalização do projeto de software (SOMMERVILLE, 2004).

2.3 Metodologias Ágeis

Por um longo período, os processos de desenvolvimento de software buscaram inspiração nos processos de manufatura para a definição de suas metodologias. Com sua origem na segunda metade do século XX, é evidente que os setores que estavam em crescimento na indústria da época influenciaram diretamente a organização das metodologias de desenvolvimento. Motivados pelo modelo de produção em série de Henry Ford, as áreas de desenvolvimento de software tinham como principal objetivo a uniformização dos componentes e procedimentos, sob a influência do setor automobilístico, em alto crescimento na época (GOMES, WILLI, & REHEM, 2014).

Gomes, Willi, & Rehem (2014) afirmam que, por volta dos anos 90, iniciaram-se as discussões sobre processos diferentes de desenvolvimento de softwares, com o objetivo de confrontar as metodologias utilizadas na época. Estas novas metodologias, que foram denominadas metodologias “leves”, alegavam ser o contrário das tradicionais, consideradas demasiadamente metódicas, morosas e com muitas etapas que existiam sem necessidade, não se adequando à natureza da atividade. Como o oposto das novas metodologias, as tradicionais foram apelidadas de metodologias "pesadas".

Provocados pelo descontentamento com o uso destas maneiras antiquadas de criar software, tornaram-se mais frequentes as discussões sobre novos ‘métodos ágeis’, que

possibilitam às equipes de desenvolvimento concentração no software e não em sua concepção ou registros em documentos (SOMMERVILLE, 2004).

Ainda de acordo com Sommerville (2004), estas metodologias, em geral, se caracterizam por um desenvolvimento incremental para o detalhamento, criação e entrega do software, sendo mais apropriados para o desenvolvimento de sistemas em que os requisitos se alteram de maneira frequente durante seu processo de criação. Seu principal objetivo é oferecer o software funcionando para os clientes o mais rápido possível, para que estes possam, então, sugerir melhorias e alterações, que podem ser abordadas em momentos futuros pelo time de desenvolvimento. Estes processos também visam mitigar a burocracia na criação de softwares, concentrando os esforços apenas no que agrega valor ao produto, no curto e longo prazo.

Estas metodologias foram denominadas ágeis somente após o ano de 2001, quando um conjunto de especialistas se reuniu com o objetivo de debater sobre formas mais leves e ágeis de desenvolver softwares, com o foco nas pessoas. Desta forma, foram criados os termos “Desenvolvimento Ágil de Software” e “Métodos Ágeis”. Além destes termos, este grupo de especialistas também deu origem ao Manifesto Ágil, caracterizado pelos valores e princípios que regem estas metodologias (GOMES, WILLI, & REHEM, 2014).

Gomes (2013, p. 4) define o Manifesto Ágil como o “embasamento filosófico de todos os métodos ágeis”, alegando que inúmeras metodologias de desenvolvimento de software estão relacionadas a ele. Desta forma, a maioria é caracterizada por ciclos pequenos, denominados iterações e duram poucas semanas, com o objetivo de assegurar as trocas frequentes de informações e a reação rápida a mudanças.

Cada método ágil pode ser definido por suas respectivas características e práticas, mas, em algum momento, estes compartilham dos valores e princípios do Manifesto Ágil (GOMES, WILLI, & REHEM, 2014).

De acordo com Beck et al. (2001, p.1), o manifesto foi fruto de um uma busca por maneiras melhores de desenvolver softwares, permitindo que esta busca auxilie outras equipes, passando a valorizar:

a. Indivíduos e interação mais que processos e ferramentas;
b. Software funcionando mais que documentação abrangente;
c. Colaboração com o cliente mais que negociação de contratos;
d. Responder a mudanças mais que seguir um plano

Sendo assim, apesar dos itens à direita (processos, ferramentas, documentação, contratos e planos) serem valorizados, os itens à esquerda (indivíduos e interações,

software funcionando, colaboração com o cliente e resposta a mudanças) são considerados mais importantes e por isso devem ser priorizados (BECK et al. 2001).

Além dos 4 valores citados, o Manifesto Ágil é complementado por 12 princípios, que servem como pilares sobre os quais são construídos os chamados métodos ágeis, são eles:

> a. Nossa maior prioridade é satisfazer ao cliente com entregas contínua e adiantada de software com valor agregado;
> b. Mudanças nos requisitos são bem-vindas, mesmo tardiamente no desenvolvimento. Os processos ágeis tiram vantagem das mudanças, visando à vantagem competitiva para o cliente;
> c. Entregar frequentemente software funcionando, de poucas semanas a poucos meses, com preferência à menor escala de tempo;
> d. Pessoas de negócio e desenvolvedores devem trabalhar diariamente em conjunto por todo o projeto;
> e. Construa projetos em torno de indivíduos motivados. Dê a eles o ambiente e o suporte necessários e confie neles para realizar o trabalho;
> f. O método mais eficiente e eficaz de transmitir informação para a equipe e entre a equipe de desenvolvimento é a conversa frente a frente;
> g. Software funcional é a medida primária de progresso;
> h. Os processos ágeis promovem desenvolvimento sustentável. Os patrocinadores, desenvolvedores e usuários devem ser capazes de manter um ritmo constante sempre;
> i. Contínua atenção à excelência técnica e bom design aumenta a agilidade;
> j. Simplicidade – a arte de maximizar a quantidade de trabalho não realizado – é essencial;
> k. As melhores arquiteturas, requisitos e design emergem de times auto-organizáveis;
> l. Em intervalos regulares, o time reflete sobre como se tornar mais eficaz e então refina e ajusta seu comportamento de acordo (BECK et al., 2001, p.2).

Segundo Pressman (2011), não são todos os métodos ágeis que aplicam estes doze princípios considerando-os com o mesmo peso e importância, alguns optam por ignorar um ou mais deles. Entretanto, estes princípios definem um pensamento ágil que é mantido em todos os métodos.

De acordo com Apello (2011), o Manifesto Ágil não foi apenas uma reação à burocracia das abordagens de desenvolvimento consideradas mais formais, foi também um posicionamento contra os programadores indisciplinados, processos caóticos e produtos de baixa qualidade, que dominavam o mundo dos softwares que eram desenvolvidos para demandas pontuais e específicas. Os autores buscavam difundir a ideia de que existia um meio-termo entre a estrutura e a falta dela, um ponto mediano entre a ordem e o caos.

Para Gomes (2013), as metodologias ágeis partem do pressuposto de que o desenvolvimento de software é um processo imprevisível, dadas as suas particularidades. Desta forma, julga-se que o cliente também está em processo de aprendizado em relação

às suas necessidades e cada nova informação pode modificar sua visão e, consequentemente, o escopo do projeto.

Estas metodologias também consideram que os valores estimados de tempo e esforço não são exatos e estão passíveis a erros. Ao assumir que mudanças podem e devem ocorrer durante o processo de desenvolvimento, é possibilitado ao cliente e a equipe se modificarem rapidamente em resposta às alterações, elevando a probabilidade de sucesso do projeto (GOMES, 2013).

Sendo assim, em contraste às metodologias tradicionais, Gomes (2013, p. 4) afirma que, nas metodologias ágeis, uma iteração "contém todas as etapas necessárias para que se realize um incremento no produto, ou seja, no software: planejamento, análise, design, codificação, testes e documentação".

2.4 Scrum

Dadas as mudanças no mercado de desenvolvimento de produtos digitais, as empresas notaram que, para se manter em constante evolução e com espaço no mercado, seria necessário mais que o básico (alta qualidade, baixo custo e diferenciação), as organizações precisariam de velocidade e flexibilidade (TAKEUCHI e NONAKA, 1986).

Para Sutherland (2010), a metodologia Scrum pode ser definida como uma estrutura de trabalho iterativa e incremental para o desenvolvimento de projetos, produtos e softwares, que visa aprimorar a previsibilidade e controlar o risco, envolvendo um conjunto de pessoas que, ao se juntarem, possuem todas as capacidades essenciais para a conclusão do trabalho ou condições de alcançar estas capacidades, caso necessário.

De acordo com Takeuchi e Nonaka (1986), esse foco na velocidade e na flexibilidade cria a necessidade de uma nova abordagem no desenvolvimento de software, já que os métodos tradicionais, em que o processo de desenvolvimento era tratado de maneira sequencial, ou seja, um processo se inicia somente após o fim do anterior, parecido com uma corrida de revezamento, não iam de encontro com as metas.

Desta forma, foi sugerida uma atitude mais holística, para se enquadrar aos requisitos competitivos do mercado. Em analogia ao Rugby, esporte em que o time tenta percorrer a distância como uma unidade, passando a bola para frente e para trás, surgiu o termo Scrum. No esporte, o termo se refere à formação em que os atacantes de um mesmo time ficam muito próximos, em posições específicas, para proteger a bola (TAKEUCHI e NONAKA, 1986; GARTNER, 1996).

2.4.1 O time Scrum

A parte essencial do Scrum é uma equipe formada por um Scrum Master, um Product Owner e os Developers (Desenvolvedores). Dentro do time Scrum não existem subdivisões ou processos hierárquicos, todos os membros atuam juntos para alcançar um objetivo comum ao time (SCHWABER e SUTHERLAND, 2020)**.**

Ainda de acordo com Schwaber e Sutherland (2020), os times Scrum, que possuem uma gestão independente, determinando internamente a divisão de tarefas, são versáteis e possuem todas as habilidades e conhecimentos essenciais para adicionar valor ao produto a cada ciclo. Com capacidades autogerenciais, o time começa a criar sua própria identidade e maneira de trabalho (TAKEUCHI e NONAKA, 1986).

Sutherland (2010) afirma que cada integrante do time possui responsabilidades e tarefas diferentes, que se complementam e auxiliam o time na busca pelo alcance do objetivo escolhido. Os papéis do time Scrum são definidos como:

a) **Scrum Master -** Este profissional não é considerado o gestor do time ou do projeto, mas sim encarregado de garantir que o Scrum seja praticado corretamente pelo time, da mesma maneira como é definido pelo Guia do Scrum. Com isso, o Scrum Master se assegura de que os membros estão focados no projeto sem sofrerem interferências externas (SCHWABER e SUTHERLAND, 2020);
b) **Product Owner -** As principais funções do Product Owner envolvem a responsabilidade de identificar funcionalidades e incrementos que podem ser incluídos no produto para maximizar o seu valor, transformando os itens identificados em uma lista priorizada para que seja possível deliberar o que será incluído em cada ciclo de desenvolvimento. Esta lista priorizada recebe o nome de Product Backlog (SUTHERLAND, 2010);
c) **Developers** - Os Developers, também chamados de desenvolvedores, são os membros que irão construir o produto a ser utilizado pelo cliente final. Além disso, suas responsabilidades também envolvem a criação de um plano para a sprint e discussões técnicas sobre abordagens de desenvolvimento (SCHWABER e SUTHERLAND, 2020).

2.4.2 Cerimônias do Scrum

Schwaber e Sutherland (2020) abordam a existência de 4 cerimônias no Scrum, estas cerimônias formam as Sprints, que são ciclos que possuem uma duração fixa de um mês ou menos. Estas cerimônias são também uma ocasião para verificar o avanço do time em direção aos objetivos. As cerimônias do Scrum são:

a) **Sprint Planning -** Esta é a primeira cerimônia e marca o início da Sprint tendo como objetivo definir e planejar o trabalho a ser realizado. O plano é resultado da colaboração de todo o time Scrum. Nesta cerimônia, os Developers e o Product Owner definem o objetivo da sprint, que representa o motivo que justifica o valor desta sprint para a empresa e as pessoas envolvidas. Estas cerimônias podem durar no máximo 8 horas para uma sprint de 1 mês.

b) **Daily Scrum -** O objetivo desta cerimônia é acompanhar o avanço do time rumo ao objetivo da Sprint. Esta cerimônia é caracterizada por encontros diários de 15 minutos de duração entre os Developers do time Scrum, que compartilham brevemente seu avanço e dificuldades enfrentadas. Estas cerimônias promovem a comunicação entre os membros do time e permitem uma ação rápida em caso de problemas.

c) **Sprint Review -** A Sprint Review tem como finalidade a verificação do resultado alcançado na Sprint e a definição de adaptações que serão necessárias no futuro. Nesta cerimônia o time Scrum e os stakeholders examinam os itens que foram concluídos e discutem os avanços em relação à meta traçada anteriormente. Esta cerimônia tem a duração máxima de 4 horas para uma sprint de 1 mês.

d) Sprint Retrospective - Esta cerimônia marca o fim da Sprint e tem como foco principal o planejamento de maneiras de intensificar a qualidade e a eficácia do time, buscando a sua constante evolução. Durante este evento, é feita uma análise do time, de suas interações, ferramentas e processos, para que os pontos de melhoria, quando identificados, sejam abordados rapidamente. Sua duração máxima é de 3 horas para um Sprint de um mês.

2.5 Extreme Programming

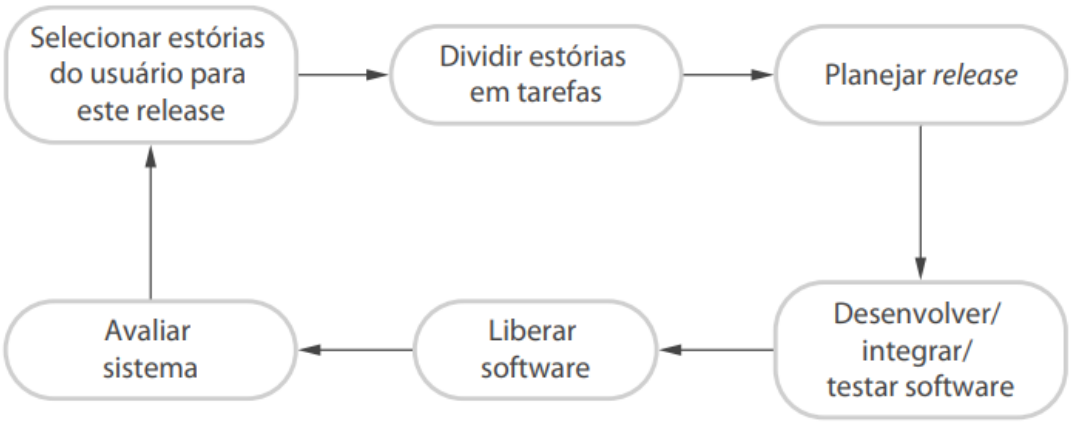
De acordo com Beck (2004), Extreme Programming é uma metodologia ágil que serve principalmente para times pequenos e médios que estão desenvolvendo software em projetos com requisitos vagos ou em constante mudança. Seu objetivo é elevar ao máximo (extremo) o uso de práticas já consolidadas no desenvolvimento de software, aumentando o impacto positivo dessas ações no resultado.

Beck (2004) também afirma que esta metodologia atende aos objetivos de todos os grupos envolvidos na criação de um software: Para os programadores, o XP busca permitir que concentrem seu trabalho nas coisas mais importantes, todos os dias, sem enfrentar situações desafiadoras sozinhos e tomando apenas as decisões que eles são os membros mais qualificados para tomar. Para os clientes e gestores, o XP visa possibilitar que os produtos tenham o máximo de incrementos possível a cada semana de programação, permitindo que estes vejam o progresso das metas mais relevantes em curtos períodos. Além disso, esta abordagem também permite que o direcionamento do projeto seja alterado sem acarretar custos elevados.

Na metodologia Extreme Programming, os requisitos do software são relatados como situações, denominadas histórias de usuário, que descrevem o comportamento esperado de cada incremento a ser desenvolvido. Estas histórias são divididas em tarefas e os programadores trabalham em pares na criação de testes para estas tarefas e, posteriormente, começam o desenvolvimento. Ao adicionar um novo incremento ao sistema, todos os testes criados anteriormente devem ser executados com êxito (SOMMERVILLE, 2011).

Ainda de acordo com Sommerville (2011), uma das características da metodologia XP é a proximidade com os clientes, que estão profundamente envolvidos no processo de desenvolvimento e atuam diretamente na especificação e priorização dos requisitos do software. Nesta metodologia, o cliente é um dos membros do time e argumenta sobre possíveis cenários (histórias de usuário) com os outros integrantes, para que estes cenários sejam implementados continuamente de acordo com a prioridade definida. Cada *release* representa uma nova versão do software, após receber incrementos ou sofrer alterações.

Figura 1 - O ciclo de um *release* em Extreme Programming



Fonte: Sommerville (2011, p. 44).

A metodologia XP busca mitigar o risco nos projetos de software, aumentar a velocidade das reações a mudanças no ambiente organizacional, melhorar a eficiência durante o ciclo de vida de um produto e tornar seu processo de construção mais divertido (BECK, 2004).

Além disso, Beck (2004) afirma que o XP é composto por 5 valores, que se equilibram e dão suporte uns aos outros: Comunicação, simplicidade, *feedback*, coragem e respeito.

a) **Comunicação**: Considerado um dos valores mais importantes para os times de desenvolvimento de software, a comunicação ajuda na resolução de conflitos, problemas e no compartilhamento de conhecimento entre os membros de um time (BECK, 2004).
b) **Simplicidade**: Este valor representa a constante busca pela construção de sistemas simples o bastante para resolver os problemas imediatos, sem considerar as necessidades que podem surgir futuramente (PRESSMAN, 2011).
c) ***Feedback***: São raros os direcionamentos que se mantêm válidos por longos períodos quando se trata de detalhes do desenvolvimento de software, requerimentos de um sistema ou arquitetura de um sistema. Quando as decisões são tomadas antes da experiência, é inevitável que surja a necessidade de alterações em um curto período. O *feedback* é fornecido pelo software, pelo cliente e por outros membros da equipe de software por meio dos testes (BECK, 2004; PRESSMAN, 2011).

d) **Coragem**: De acordo com Pressman (2011), este valor é importante para que os membros do time consigam aplicar as práticas do XP, dada a pressão constante para que os projetos atendam tanto requisitos atuais quanto futuros. É necessário que a equipe tenha coragem para focar em atender as necessidades atuais, visto que as necessidades futuras podem se alterar e causar um retrabalho considerável no projeto já implementado. Por outro lado, Beck (2004) afirma que a coragem como valor principal sem um equilíbrio dos outros valores podem ser perigoso, já que agir sem pensar nas consequências não é a melhor forma de trabalhar em equipe.

e) **Respeito**: O respeito é a consequência dos 4 valores anteriores. Para que um projeto de desenvolvimento de software funcione, é necessário que os membros da equipe se respeitem e se importem tanto uns com os outros quanto com o sistema que está sendo produzido (BECK, 2004).

2.5.1 Práticas do Extreme Programming

Segundo Beck (2004), as práticas são irrelevantes se não forem pautadas a partir dos valores definidos anteriormente. Estas práticas, apesar de funcionarem melhor quando aplicadas em conjunto, podem ser adotadas de acordo com a situação e necessidade do time, ainda que os valores continuem os mesmos.

Para Teles (2014), o XP se baseia nas seguintes práticas:

a) **Cliente presente:** Esta prática foi criada a partir da premissa de que o cliente deve ser um dos condutores do desenvolvimento, a partir da troca de feedbacks entre cliente e sistema. Para que o cliente obtenha o maior valor possível como resultado, é essencial que ele participe da equipe e facilite diversos processos, como o de comunicação com o time, por exemplo.

b) **Jogo do planejamento:** Para garantir que a equipe trabalhe no que é mais relevante no momento, o XP usa algumas formas de planejamento que dividem o projeto em *releases* e iterações. Os *releases* são partes do sistema que geram um valor definido para o cliente. Já as iterações, representam períodos (média de duas semanas) em que a equipe executa um grupo de incrementos acordados com o cliente. No começo de cada iteração ocorre o jogo do planejamento, que é onde o cliente seleciona e avalia as

funcionalidades a serem desenvolvidas, elegendo as que estarão no próximo release ou iteração.

c) **Stand up meeting:** Esta prática é uma reunião diária que acontece com a equipe de desenvolvimento toda manhã, com o objetivo de analisar o trabalho feito no dia anterior e definir o que será priorizado para desenvolvimento no dia da reunião. O nome, traduzido do inglês, significa reunião em pé.

d) **Programação em par:** Para que o código seja avaliado enquanto ele é criado, os desenvolvedores que trabalham na metodologia XP costumam criar as funcionalidades em pares. Com dois desenvolvedores diante de cada computador, é possível tornar o processo mais eficaz, pois os desenvolvedores podem discutir entre si, se complementar e o resultado pode ser uma solução inovadora.

e) **Desenvolvimento guiado por testes:** Como um dos principais objetivos do XP é a criação de sistemas com qualidade, mecanismos de validação se tornam necessários para garantir que o software esteja funcionando corretamente. Além da programação em par, o desenvolvimento guiado por testes é outro mecanismo que permite avaliar a performance do software desenvolvido. Antes de iniciar o processo de desenvolvimento, os testes de cada funcionalidade são escritos, permitindo que as necessidades do cliente sejam entendidas previamente e os desafios sejam abordados.

f) **Refatoração**: A refatoração significa modificar o código sem alterar suas características de funcionamento. Este processo visa fazer com que os objetivos do sistema sejam expressos de maneira fácil e clara. Esta prática possibilita que o sistema evolua de maneira incremental e seja modificado facilmente, já que os sistemas podem sofrer alterações a qualquer momento devido às solicitações do cliente.

g) **Código coletivo**: Com o objetivo de fornecer mais agilidade nos processos e adicionar mais uma forma de revisão e verificação, o código na metodologia XP não é segmentado em partes diferentes. Sendo assim, os desenvolvedores do time podem acessar qualquer parte do código e fazer alterações onde acharem necessário. Nesse caso não é necessária a permissão de outras pessoas, pois o código é de todos os membros do time.

h) **Código padronizado**: A equipe de desenvolvimento deve estabelecer padrões de codificação, para permitir que os desenvolvedores do time façam alterações em qualquer parte do código e facilitar manutenções futuras.

i) **Design simples**: Como os desenvolvedores têm como base a premissa de que será possível desenvolver funcionalidades para adaptar o sistema caso apareçam novas necessidades, a equipe sempre busca desenvolver produtos com um design mais simples e suficientes para atender às necessidades do momento, sem criar generalizações para suprir demandas futuras.

j) **Metáfora**: Esta prática serve para simplificar as explicações de programas, ideias ou etapas do desenvolvimento. As metáforas ajudam na comunicação entre os membros do time e favorecem a disseminação de conhecimento, permitindo que os membros vejam as situações com um olhar diferente.

k) **Ritmo sustentável**: Dada a quantidade de problemas que o cansaço pode causar (desatenção, falta de produtividade, erros nos códigos), a metodologia XP indica que não se faça o uso de horas-extras para desenvolvimento de projetos de software. Esta prática visa aumentar a agilidade do projeto e demonstrar mais respeito com a individualidade e integridade da equipe, permitindo que eles trabalhem apenas 8 horas por dia.

l) **Integração contínua**: Para evitar problemas na integração de sistemas feitos por membros da equipe quando eles estão trabalhando em áreas diferentes, que podem ser causados pela falta de entendimento do processo como um todo, a metodologia XP sugere que sejam feitas integrações no menor intervalo de tempo possível.

m) ***Releases* curtos**: O objetivo principal desta prática é criar um fluxo constante de valor para o cliente, para isso, a equipe desenvolve um grupo pequeno de funcionalidades e permite que o cliente possa utilizar, testar e dar feedback em curtos períodos, tendo contato com o benefício da sua criação.

2.5.2 O time em Extreme Programming

Ainda de acordo com Teles (2004), a composição de uma equipe que utiliza XP deve ser feita pelos seguintes papéis:

a) **Gerente de projeto**: Este tem como maior responsabilidade o relacionamento entre a equipe de desenvolvimento e cliente, informando o cliente sobre o

progresso do time, os problemas enfrentados e os riscos encontrados ao longo do projeto. Além disso, o gestor de projeto também atua como um filtro para a equipe de desenvolvimento, discutindo os assuntos burocráticos com o cliente e garantindo que a equipe foque apenas no que é necessário para aquele momento de trabalho.

b) **Coach**: Ao coach é atribuída a responsabilidade de garantir que a equipe respeite e use todos os dias os valores e práticas do Extreme Programming. Uma das características principais dos responsáveis por esse papel é o conhecimento profundo do processo de desenvolvimento de software. Como o XP é uma metodologia mais voltada para a conduta da equipe, seu papel é fundamental para verificar o andamento dos membros diariamente e mostrar eventuais erros que são cometidos.

c) **Analista de teste**: O papel do analista é testar e assegurar a qualidade do sistema desenvolvido. Sua atuação consiste em auxiliar o cliente a escrever os testes de aceitação e fazer com que o sistema atenda a todos os testes estipulados. Desta forma, o processo de validação pode ser feito pelo cliente, seguindo os roteiros escritos previamente. A importância deste papel é evidente, pois garante que o sistema não possua erros graves detectados apenas quando o sistema for publicado.

d) **Redator técnico**: O redator tem como principal responsabilidade zelar pela atualização frequente da documentação do sistema. Desta forma, ele impede que os desenvolvedores gastem muito tempo no trabalho de documentação, assegurando que a principal tarefa dos desenvolvedores seja, de fato, desenvolver código para os softwares.

e) **Desenvolvedor**: Papel essencial nos times que utilizam Extreme Programming, o desenvolvedor deve analisar, projetar e codificar o software. Apesar de não existirem hierarquias nas equipes que utilizam Extreme Programming, existem pessoas com diferentes níveis de habilidade. Por isso, mecanismos como a programação em par buscam uniformizar estas habilidades e aumentar a capacidade da equipe de atuar no desenvolvimento dos sistemas.

3 PROCEDIMENTOS METODOLÓGICOS

Este capítulo aborda os procedimentos metodológicos que viabilizaram a pesquisa científica do tema, descrevendo seus objetivos, abordagens e métodos de coleta, investigação e análise de dados para atender ao objetivo de pesquisa.

Quanto a seus objetivos, o presente artigo pode ser classificado como uma pesquisa exploratória e descritiva. De acordo com Gil (2002), as pesquisas exploratórias visam aumentar o entendimento relacionado a um determinado problema, para torná-lo mais evidente ou permitir a criação de hipóteses. Além disso, Gil (2002) também afirma que as pesquisas descritivas objetivam principalmente a descrição das características de algum grupo de pessoas ou fenômenos, ou a determinação de relação entre variáveis. Desta forma, classifica-se esta pesquisa como descritiva por ter como objetivo avaliar as vantagens e desvantagens das metodologias ágeis de desenvolvimento de software Scrum e Extreme Programming e exploratória por obter um maior entendimento sobre as metodologias ágeis em questão.

Além disso, a pesquisa se classifica como bibliográfica, definida por ser baseada em material existente, predominantemente livros e artigos científicos (GIL, 2002).

Em relação à abordagem, foi realizada uma pesquisa qualitativa, com coleta de dados, por meio de entrevistas semiestruturadas, com gestores de projeto de software que trabalham em empresas que utilizam metodologias ágeis de desenvolvimento. Gil (2002, p. 117) define as entrevistas semiestruturadas como “guiadas por relação de pontos de interesse que o entrevistador vai explorando ao longo de seu curso. Para a aplicação da entrevista, foram selecionados 2 gestores com mais de 1 ano de experiência para cada uma das metodologias ágeis (Scrum e Extreme Programming) totalizando assim 4 entrevistados, referidos neste artigo como: E1, E2, E3 e E4. A seleção dos entrevistados se deu por acessibilidade do autor por meio de sua rede de contatos.

Nas entrevistas, buscou-se entender qual a visão dos gestores em relação às metodologias ágeis adotadas em seus times de desenvolvimento de software, abordando as vantagens e desvantagens observadas em suas experiências para posterior comparação. O instrumento de coleta de dados utilizado para a entrevista semiestruturada foi um roteiro de entrevista, tendo como base a pesquisa bibliográfica (ALMEIDA, 2014). O roteiro se encontra no apêndice A. Estas entrevistas foram realizadas entre os 24/06/2022 e 01/07/2022 por meio da plataforma videoconferência *Google Meet*, que possibilitou a gravação, transcrição e análise de conteúdo.

4 RESULTADOS

Nesta seção são apresentados por meio de tópicos os resultados das entrevistas com os gestores de projetos de software, evidenciando seus pensamentos em relação às metodologias ágeis de maneira geral e sobre o Scrum e Extreme Programming.

4.1 Impacto das metodologias ágeis nos processos de desenvolvimento

Ao questionar os entrevistados se acreditam que as metodologias ágeis possuem um impacto positivo no desenvolvimento de software, o E3 afirma que: “Acredito que precisamos das metodologias ágeis de desenvolvimento de software”

E2, diz que:

> O desenvolvimento "cascata" funciona historicamente e estatisticamente, mas é muito mais dolorido. À medida que o desenvolvimento do software vai acontecendo, os times não vão aprendendo durante esse processo. Por exemplo: Um time fez um planejamento com o cliente para lançar um produto para daqui a 2 anos, é um tempo muito grande, às vezes as coisas mudaram, apareceu outro competidor do mercado, por isso o planejamento deve ser adaptável.

Desta forma, o observou-se também que o ganho das equipes com o uso de metodologias ágeis está na possibilidade de alterar o foco do processo de desenvolvimento e se adequar de acordo com os acontecimentos. Esta característica pode ser notada na narrativa de E2, que afirma:

> No meu contexto, o desenvolvimento ágil não significa desenvolver rápido, mas sim entregar valor e mudar de direção rapidamente. Entender a necessidade do cliente, entregar valor de maneira rápida e ir testando aquilo que está sendo feito, este é o principal valor do ágil.

Pode-se perceber que o impacto positivo das metodologias ágeis no processo de desenvolvimento de software foi apontado como inquestionável por todos os entrevistados, que defendem o uso desta metodologia como forma de obter respostas mais rápidas para problemas e antecipar erros, garantindo um melhor desempenho no produto.

Dentre os principais motivos citados para este sucesso das metodologias ágeis está a sua diferenciação quando comparado com os métodos tradicionais, por trazer uma visão mais realista para o mundo do desenvolvimento de software, que tem como característica a dinamicidade. Um dos métodos tradicionais citados é o desenvolvimento em cascata, que, de acordo com Sommerville (2004), é quando o software é primeiramente planejado em todas as etapas, de maneira minuciosa, e só depois desenvolvido seguindo o

planejamento feito anteriormente, considerando etapas separadas e não um projeto conjunto.

Um outro aspecto citado pelos entrevistados é o cuidado para que se faça o uso correto destas metodologias ágeis nos ambientes de desenvolvimento de software, isto é, entender que estas metodologias são práticas que não resolvem sozinhas os desafios e problemas que os times enfrentam diariamente e, caso sejam aplicadas de maneira prescritiva, podem acabar prejudicando as equipes.

Na fala de E1, pode-se verificar a sua preocupação:

> Tem empresas que vão aplicar a metodologia e vão olhar para essa metodologia como um fim e não um meio. Desta forma, estão muito focadas em seguir o que a metodologia propõe e não necessariamente em garantir que as pessoas estão tendo uma boa experiência de trabalho, ou que o que está sendo entregue de fato resolve um problema e agrega valor para o usuário.

Dessa forma, pode-se ressaltar que todos os entrevistados entendem que as metodologias ágeis possuem um impacto positivo no desenvolvimento de software, agregando valor aos projetos desenvolvidos e ampliando a performance da empresa.

4.2 O uso da metodologia ágil Scrum

De acordo com os gestores de projetos de software entrevistados, um dos principais ganhos proporcionados pela metodologia Scrum é a organização de demandas de uma maneira prática e eficiente, conforme afirma E4:

> Até existem outras formas de organização das demandas de desenvolvimento de software, mas acho que a metodologia Scrum atende bastante a gente. Temos bastante demandas aqui e conseguimos cumprir justamente pela organização que a metodologia proporciona.

Sendo assim, percebe-se que a visão mais generalista da metodologia Scrum, que aborda outras esferas organizacionais do processo de desenvolvimento de software, permite que ela seja utilizada em diversas situações. Por outro lado, esta prática pode representar alguns desafios para times com menos experiência, conforme afirma E3: “Hoje, na metodologia Scrum não vejo mais um foco tão grande no processo de desenvolvimento”.

4.3 Vantagens da metodologia Scrum

Neste capítulo são abordadas as vantagens observadas, de acordo com os relatos dos gestores de projeto de software, no uso da metodologia ágil Scrum.

4.3.1 Fácil de implementar

A simplicidade para a implementação desta metodologia ágil foi um fator apontado por ambos os entrevistados como benefício desta metodologia, que possui passos bem definidos e prescreve até a ordem das cerimônias. Esta característica pode ser evidenciada na afirmação do E4 sobre o Scrum:

> Mesmo membros que não fizeram estudo aprofundado sobre esta metodologia conseguem entender o que ela faz, para que ela serve e porque a utilizamos. Acredito que o entendimento dela é muito fácil, muito tranquilo, quando comparado às outras.

Esta característica explica também a adoção desta metodologia por equipes de desenvolvimento iniciantes, conforme aponta E3:

> Uma nova gama de desenvolvedores, que estão entrando no mercado e tem um conhecimento básico sobre metodologias ágeis, normalmente tem o conhecimento apenas sobre o Scrum, por ser a metodologia mais difundida
> Vejo que o Scrum habilita muitos times, ele pode habilitar uma empresa que está no "caos", com processos bagunçados e deixar todos "na mesma página" para um posterior processo de evolução.

Desta forma, nota-se que o Scrum possibilita que equipes que atuam com métodos mais tradicionais de desenvolvimento transformem as suas organizações, a partir de uma metodologia com processos definidos de aplicação.

4.3.2 Ciclos de feedback curtos

Com a sua atuação em pequenos ciclos de até 1 semana, as chamadas sprints, o Scrum proporciona às equipes um rápido aprendizado, possibilitando a validação de hipóteses de negócio de maneira frequente e gerando constantes evoluções no software que está em processo de desenvolvimento.

Este benefício foi apontado por E4, que afirma:

> Ao usar o Scrum, conseguimos entender ao fim de cada ciclo se algum incremento ou funcionalidade teve o resultado que esperávamos e conseguimos agir rapidamente em cima da resposta, com o uso de metodologias tradicionais este processo provavelmente seria bem mais demorado.

Sobre a mesma questão, E3 adiciona: “No tradicional há um tempo muito grande de análise e levantamento de riscos. No Scrum, identificamos os riscos mais rápido e vamos atuando de acordo com o feedback."

Ainda abordando este benefício, E4 completa:

> Já atuei em projetos em que, apesar de observarmos erros ao longo do da criação do software, a maior preocupação da organização era seguir o plano feito anteriormente, e isso culminou em altos custos de refatoração que poderiam ter sido evitados.

A partir destes relatos, é possível observar que a agilidade desta metodologia permite que os erros e falhas sejam corrigidos sempre que necessário de maneira a evitar o acúmulo de problemas no futuro, tornando-os ainda mais caros. A liberdade na alteração dos objetivos a cada ciclo facilita esse processo, em contraste às metodologias mais tradicionais, em que é seguido um plano já traçado minuciosamente.

4.3.3 Utilidade na gestão de projetos de software

Por não se limitar apenas às práticas do desenvolvimento de software, a metodologia Scrum tem como grande vantagem para as equipes a sua utilidade como ferramenta de gestão, sendo útil não só para os desenvolvedores, mas para todas as partes envolvidas (stakeholders, gestores, times de desenvolvimento, clientes).

O uso desta metodologia possibilita, a partir da descrição objetiva e detalhada de cada cerimônia e o papel de cada membro, permitir que se aplique um fluxo de trabalho que já foi validado por seus criadores.

E3 aponta este benefício ao afirmar:

> Gosto muito do fato desta metodologia ensinar como cada detalhe do fluxo de desenvolvimento deve funcionar, acredito que poupa muito tempo do time na hora de implementar, pois não é necessário pensar e refletir tanto sobre quais partes serão implementadas e funcionarão pra nós, como é o caso de outras metodologias.

A organização que a aplicação do Scrum traz para os times de desenvolvimento de software é evidenciada também na afirmação de E4:

> Em nosso time Scrum temos uma pessoa responsável para fazer as gerências (Scrum Master), seu papel é organizar o fluxo de trabalho e garantir que a metodologia Scrum seja seguida, permitindo que o pensamento do time se concentre no desenvolvimento do produto.

Sendo assim, com a definição clara da maioria dos processos desta metodologia, a avaliação do que está indo de acordo com o modelo proposto é simplificada, possibilitando uma constante análise.

4.3.4 Colaboração entre os membros

O desenvolvimento colaborativo é outro benefício trazido pela metodologia Scrum, de acordo com os gestores de software entrevistados. Ao participar das cerimônias, os membros do time estão em frequente contato uns com os outros, possibilitando a resolução de desafios e impeditivos rapidamente por meio da circulação de informações.

Reconhecendo este benefício, E3 afirma:

> Através das dailies estamos em contato constante e isto possibilita que os membros saibam o que seus colegas estão tendo dificuldade ou enfrentando desafios, assim podemos resolver e transformar estas situações em aprendizado coletivo.

A partir deste relato, é reconhecido também o benefício da colaboração entre os integrantes do time, contribuindo para o sucesso na gestão do projeto de software.

4.4 Desvantagens do Scrum

Neste capítulo são abordadas as desvantagens observadas, de acordo com os relatos dos gestores de projeto de software, no uso da metodologia ágil Scrum.

4.4.1 Necessidade de mudança no comportamento do time

Quando questionados em relação às desvantagens da metodologia Scrum, E3 afirma:

> Se eu escolher um membro que era gerente tradicional e apenas mudar o título dele, fazer virar Product Owner, por exemplo. Caso eu não faça treinamentos ou atividades para que ele desenvolva um pensamento ágil, ele vai continuar fazendo as mesmas coisas que fazia anteriormente, tendo comando e controle sobre os outros membros e vai desgastar a equipe, resultando em times com turnover muito alto, e isso é muito corriqueiro.

E4 aborda a mesma desvantagem, ao relatar:

> Acredito que o Scrum é voltado para uma equipe de inovação e criatividade, mas se eu dou liberdade para que os membros do time entendam o produto e o valor que está sendo gerado, acaba perdendo um pouco do senso do uso desta metodologia.

A dificuldade gerada pela resistência a mudanças nos times também cria a necessidade de adaptações nas metodologias, gerando alguns riscos, conforme aponta E3:

> Na minha jornada já passei por diferentes contextos na implementação do Scrum, desde contextos mais fáceis até outros contextos em que era necessário aplicar a metodologia de forma mais adaptativa. Acredito que quando aplicamos de forma adaptativa é onde começamos a trazer disfunções para dentro da metodologia.

Deste modo, conclui-se que a implementação desta metodologia ágil depende diretamente do comprometimento do time quanto à sua comunicação e autonomia nas atividades, e isto pode representar um risco para seu sucesso, caso os integrantes tenham algum tipo de resistência às alterações que a metodologia propõe ou não se comprometam a atuar de acordo com seus princípios, tratando as indicações da metodologia apenas como um conjunto de atividades a serem concluídas.

4.4.2 Não indica práticas de programação

Como o Scrum tem como principal foco a gestão do processo de desenvolvimento iterativo e não aborda técnicas específicas de engenharia de software, a metodologia não indica práticas para equipes que querem desenvolver softwares melhores no quesito de qualidade de código (SOMMERVILLE, 2004).

Sobre esta característica, E3 relata:

> Acredito que está havendo um desligamento dos desenvolvedores. Hoje vemos treinamentos com gestores e outras pessoas envolvidas nos produtos e não direcionam treinamentos para a equipe de desenvolvimento, que são as pessoas que estão ali desenvolvendo os produtos e entregando serviços todos os dias.

E4 complementa o assunto ao afirmar:

> Sentimos muito a falta de indicações de como implementar práticas de desenvolvimento dentro do Scrum como cerimônias e outros tipos de atividades. Hoje sinto que o Scrum traz benefícios de organização, mas do ponto de vista técnico não faz tanta diferença para o software.

Desta forma, é possível afirmar que a falta de indicações de práticas voltadas especificamente para o processo de desenvolvimento pode resultar em uma desconexão da metodologia, caso aplicada sozinha, com times de desenvolvimento que possuem necessidades mais complexas.

4.4.3 Excesso de cerimônias

Ainda que cada cerimônia tenha um objetivo diferente, quando são realizadas em horários próximos algumas cerimônias podem se tornar repetitivas, conforme afirma E4:

> Em nossa agenda, a cerimônia de Sprint Review e Sprint Retrospective acontecem em horários próximos, isso faz com que os membros falem a mesma coisa em alguns momentos. Além disso, é complicado para o time fazer a gestão dos temas dentro de cada reunião.

Além disso, as cerimônias podem se tornar mais cansativas no contexto do trabalho remoto, já que as interações entre os membros do time passam a acontecer de maneira virtual. Sobre isso, E3 afirma:

> Tem gente que reclama muito da duração das cerimônias no home office, alguns times fazem a mesma coisa que faziam no presencial e isso pra estressar as pessoas é muito fácil. Eu nunca fui adepto de fazer sessões longas assim, sempre sugiro pausas ou cortar algumas cerimônias em 2 partes.

Deste modo, nota-se que apesar de ter como qualidade os processos bem definidos, os usuários da metodologia Scrum entendem como uma desvantagem a alta quantidade e frequência das cerimônias que acontecem dentro de cada ciclo (sprint).

4.5 O uso da metodologia ágil Extreme Programming

Em relação à aplicação da metodologia ágil Extreme Programming, os entrevistados citam a como um grande valor para as equipes de desenvolvimento sua adaptatividade para diferentes contextos, indo de encontro com o que afirma Teles (2004 p. 275), ao apresentar a metodologia XP como “um processo de desenvolvimento flexível que se adapta facilmente às mudanças no processo de desenvolvimento de software”.
Esta visão também pode ser evidenciada na resposta de E2:

> Vejo como uma metodologia muito boa, muito útil e enxergo muito valor nas práticas que ela propõe. Recomendo bastante, por ser uma metodologia bem adaptável, que me permite colher informações e aplicar para o meu contexto as práticas que fazem mais sentido.

Sendo assim, fica evidente o foco das práticas do Extreme Programming no processo de programação, tendo como principal preocupação a maneira técnica que o software será construído e se voltando menos para o processo de organização do time para que as entregas e incrementos sejam feitos.

4.6 Vantagens do Extreme Programming

Neste capítulo são abordadas as vantagens observadas, de acordo com os relatos dos gestores de projeto de software, no uso da metodologia ágil Extreme Programming.

4.6.1 Qualidade do Software

Por ter como objetivo na maioria de suas práticas (Programação em par, desenvolvimento guiado por testes, refatoração, código coletivo, código padronizado, design simples, metáfora, integração contínua e releases curtos) a qualidade no software construído, esta característica se torna também uma das principais vantagens para os times ao implementar a metodologia Extreme Programming.

Este benefício é evidenciado na resposta de E1, que afirma:

> O XP é bem focado na programação, até pelo nome, tem um foco muito grande em qualidade de software e aponta muito para a qualidade do que está sendo construído. De todas as metodologias do ágil, esta é a que mais se propõe a usar práticas para pessoas desenvolvedoras garantirem esta qualidade.

Teles (2004) afirma que ao aplicar as práticas do Extreme Programming da maneira correta é possível obter um software mais robusto e com menos erros e *bugs*.

De acordo com E1, a qualidade no código é a principal vantagem porque “quanto maior a qualidade do código/produto, mais rápido e barato será para testar e implementar novas hipóteses”.

4.6.2 Adaptabilidade do software

De acordo com o E1, a metodologia ágil Extreme Programming “aponta para o feedback rápido, propondo ciclos rápidos de entrega e mudança”, possibilitando que os times se adaptem facilmente às alterações nos ambientes competitivos das organizações e que estas modificações não tenham impactos negativos no planejamento financeiro e de recursos.

Sendo assim, a constante transformação do mercado e dos softwares pode se tornar uma vantagem competitiva para as organizações que possuem times de desenvolvimento com códigos extensíveis, ou seja, facilmente modificáveis.

E1 evidencia este fato ao afirmar:

> A aplicação de algumas práticas do XP permite que se tenha uma gestão melhor e possibilita acompanhar mais rapidamente às demandas de negócio sem precisar perder tempo corrigindo bugs e fazendo uma mudança porque o código está confuso ou difícil de mexer.

Este diferencial competitivo ocasionado pelo uso da metodologia XP é importante pois permite que as organizações adotem estratégias mais arriscadas para a criação e lançamento de produtos inovadores ao observar oportunidades emergentes em seu mercado de atuação ou até a exploração de novos mercados por meio do desenvolvimento de novas tecnologias.

4.6.3 Adaptabilidade da metodologia

De acordo com E1**, "**o XP é mais um conjunto de práticas e valores do que uma ferramenta de gestão de projetos, trazendo mais liberdade para o uso desta metodologia", isto permite que estas práticas sejam utilizadas em contextos diferentes para agregar valor aos processos de desenvolvimento de software adotados.

Além disso, E1 define a metodologia ágil XP como "ampla e aberta", por não especificar quais práticas devem ser adotadas em cada contexto e indicar aos usuários que estes devem descobrir por meio de suas próprias experiências o que trará melhores resultados.

E2 também aborda esta característica da metodologia ao afirmar:

> Ao mesmo tempo, essa liberdade na implementação também é uma das propostas do XP, que diz para que o usuário pegue o contexto e adapte, fazendo com que seja implementado com muito suor e gere aprendizado durante o processo.

A partir das entrevistas com os gestores de projeto de software também foi possível observar como esta liberdade no uso do Extreme Programming possibilita uma visão mais crítica acerca dos processos adotados nas organizações, conforme relata E1:

> Usamos tudo que entendemos como boas práticas e padrões de trabalho, mas não vemos como regras inquestionáveis. Dependendo do time e do projeto, vamos adotar certas práticas e em outros não.

4.6.4 Ampla adoção pela comunidade

Com o crescimento no uso da metodologia Extreme Programming, utilizada por diversas organizações e uma das metodologias mais conhecidas no mundo, de acordo com

Sommerville (2004), algumas de suas práticas se tornaram uma espécie de padrão para a construção de softwares de qualidade, mesmo quando adotadas sem uma conexão com a metodologia em si, esta afirmação pode ser confirmada pelo relato de E1:

> Práticas como testes unitários, programação em par e refatoração são coisas que são difíceis de não ter em um projeto, apenas em um contexto muito específico.

Sobre o mesmo tema, E2 adiciona:

> Hoje, quando se olha pro XP, vê-se as práticas como óbvias, mas deve-se pensar que foram criadas em um momento em que esta não era a realidade. Era comum ver um time de tecnologia em um país, outro time em outro e estes times não se comunicavam, até a hora de integrar as tecnologias desenvolvidas, causando muitos problemas e custos.

O crescimento no uso desta metodologia pelos times de desenvolvimento de software das organizações também possibilita que as suas práticas sejam cada vez mais adaptadas e discutidas entre seus praticantes, resultando em melhorias constantes.

4.7 Desvantagens do Extreme Programming

Neste capítulo são abordadas as desvantagens observadas, de acordo com os relatos dos gestores de projeto de software, no uso da metodologia ágil Extreme Programming.

4.7.1 Pouco útil como ferramenta de gestão de um projeto de software

Como esta metodologia é mais voltada para o trabalho dos programadores no desenvolvimento de software, apesar de representar um aumento na qualidade do produto, esta abordagem pode ser apresentada também como uma desvantagem para a organização do trabalho, conforme apontado por E1:

> Acredito que o XP foca muito na qualidade de código e acaba não trazendo muita coisa a respeito de fluxos de trabalho, ou da gestão de projeto/produto de software em si.

Evidenciando a mesma característica, E2 também cita:

> XP não vive sozinho como gestão de projeto, isto gera sobrecarga cognitiva, de estudo e tal, para que as pessoas tentem entender a sua complexidade dentro da organização que desenvolve produtos de software.

Sobre esta característica, E3 complementa:

> O XP fala muito de código, então se você não está em contexto de atuar tecnicamente no processo de desenvolvimento de software, provavelmente ele nem se aplica para você.

Sendo assim, a concentração desta metodologia no aspecto técnico da construção de um software implica em uma discrepância quando analisados outros elementos organizacionais da construção de um produto de software.

4.7.2 Difícil de implementar

Por ser uma metodologia sem processos estritamente definidos, sua implementação pode se tornar mais desafiadora para algumas equipes, que preferem seguir passos e direcionamentos ao invés de refletir sobre seus processos de desenvolvimento.

Em relação à essa característica, E2 afirma:

> As pessoas acabam optando por manuais e coisas mais simples de serem implementadas. Inclusive é por isso que o Scrum deu tão certo, porque ele é um manual simples. Não tem como ler o manual e não conseguir implementar alguma parte desta metodologia.

E2 direciona este desafio ao afirmar:

> Às vezes a pessoa que vem de outro contexto tem uma forma muito "fechada" de fazer as coisas, ou uma organização tem estruturas que não viabilizam essa dinamicidade, adotando a metodologia Extreme Programming de forma errada.

Dessa forma, é possível afirmar que abordagem desta metodologia face aos problemas e processos de aprendizado do time pode representar uma alteração intensa nas práticas adotadas por organizações com processos mais tradicionais. Tornando necessária uma adaptação para que a metodologia XP seja aplicada corretamente.

4.7.3 Necessidade de mudança no comportamento do time

A serem questionados as possíveis desvantagens, E1 aponta:

> Para aplicar o XP é necessário ter um esforço maior no treinamento e na garantia de que as pessoas que estão sendo contratadas para fazer parte da equipe sejam pessoas que têm habilidades fortes de autonomia, iniciativa, auto-organização, e nem todo mundo possui esse perfil.

Além disso, este processo implementação pode ter impactos financeiros para a organização, dependendo do nível de maturidade tecnológica, conforme aponta E1:

> Por conta desta mudança, o processo de formação de times aptos a vivenciar estas práticas se torna mais caro, por possuir um recrutamento mais rigoroso e criterioso, aumentando o tempo para achar pessoas certas. Pode ser necessário também investir na formação destas pessoas, para garantir que elas tenham as habilidades necessárias.

Além do investimento em pessoas, em alguns casos é necessário também um investimento em tecnologia para que algumas práticas do Extreme Programming sejam possíveis, conforme afirma E2:

> A prática de integração contínua, por exemplo, não é uma prática de implementação simples para times de tecnologia. Até chegar ao ponto de integrar continuamente um software é preciso investimento em tecnologia e uso de boas práticas.

A partir das informações coletadas, é possível observar que como no time de Extreme Programming não existe hierarquia entre os membros (assim como no time Scrum), é exigido das pessoas uma autonomia maior para que o trabalho seja feito, já que se parte do princípio de que todos os membros conhecem todas as suas responsabilidades e estão de acordo com elas. Sendo assim, a adaptação das pessoas se torna um fator essencial (e desafiador) para lidar da melhor forma possível com as práticas do XP.

4.8 Análise comparativa

Nesta seção são analisadas as relações entre as duas metodologias ágeis abordadas, de acordo com a interpretação das vantagens e desvantagens evidenciadas pelos entrevistados.

4.8.1 Vantagens

Ao observar as vantagens e desvantagens, percebe-se que as duas metodologias, apesar de serem baseadas nos mesmos princípios de desenvolvimento ágil, funcionam de maneira diferente, e como consequência trazem resultados distintos para os times que as utilizam, conforme é possível observar no quadro 1:

Quadro 1: Vantagens do Scrum e Extreme Programming

Scrum	Extreme Programming
Fácil de implementar	Qualidade do Software
Ciclos de feedback curtos	Adaptabilidade do software
Utilidade na gestão de projetos de software	Adaptabilidade da metodologia
Colaboração entre os membros	Ampla adoção pela comunidade

Fonte: Elaborado pelo autor.

Enquanto os benefícios da metodologia Extreme Programming se concentram em sua maioria no produto do desenvolvimento de software, ou seja, no resultado final, ao visualizar os benefícios do Scrum é nota-se que seus benefícios estão mais ligados ao processo.

O E4 endossa esta comparação ao relatar:

> Vejo claramente algumas diferenças entre as metodologias mais conhecidas, que são o Scrum e Extreme Programming: O XP é mais focado em construir o software da maneira certa, enquanto o Scrum cuida da organização do time para que isto aconteça.

Ainda sobre os mesmos benefícios, o E2 argumenta:

> O XP está mais voltado para a maneira que um software será construído, e menos no processo em si de cada entrega ou incremento deste software. O XP representa mais valores do dia a dia de desenvolvimento do que a entrega em partes ou velocidade do time.

Sendo assim, é possível afirmar que as metodologias devem ser utilizadas de acordo com as necessidades e propósitos das organizações, para que seus resultados estejam de acordo com o esperado.

4.8.2 Desvantagens

Quanto às desvantagens, a necessidade de mudança no comportamento do time aparece como desafio comum às duas metodologias, já que a autonomia causa a dependência da organização a cada um dos membros do time e espera-se que eles cumpram todas as suas atividades com responsabilidade, conforme se observa no quadro 2.

Quadro 2: Desvantagens do Scrum e Extreme Programming

Scrum	Extreme Programming
Não indica práticas de programação	Pouco útil como ferramenta de gestão
Excesso de cerimônias	Difícil de implementar
Necessidade de mudança no comportamento do time	Necessidade de mudança no comportamento do time

Fonte: Elaborado pelo autor.

Um outro aspecto a ser abordado é que as desvantagens relacionadas ao XP são o contrário das vantagens relacionadas ao Scrum e vice-versa, como pode-se constatar no quadro 3:

Quadro 3: Relação entre vantagens e desvantagens do Scrum e Extreme Programming

Scrum	Extreme Programming
Não indica práticas de programação	Qualidade do Software
Excesso de cerimônias	Adaptabilidade da metodologia
Utilidade na gestão de projetos de software	Pouco útil como ferramenta de gestão
Fácil de implementar	Difícil de implementar
Vantagens | Desvantagens	

Fonte: Elaborado pelo autor.

Deste modo, fica evidente o comportamento complementar das duas metodologias ágeis, que resolvem problemas diferentes de partes específicas do processo de desenvolvimento de software.

5 CONCLUSÃO

Com o aumento da disputa entre as organizações do setor de tecnologia, cada melhoria nos processos pode fazer a diferença e se tornar uma vantagem competitiva, a partir da economia de custos ou aumento dos lucros. As metodologias ágeis visam tornar estas melhorias cada vez mais frequentes na esfera tecnológica da empresa.

O presente artigo buscou apresentar as vantagens e desvantagens das metodologias ágeis Scrum e Extreme Programming, fazendo uma análise comparativa entre as duas metodologias.

Sendo assim, conclui-se que, dentre as metodologias analisadas, não há a possibilidade de elencar uma mais vantajosa que a outra, visto que a sua aplicação depende diretamente do contexto e dos objetivos de cada time de desenvolvimento e organização, tendo uma influência direta em seus resultados. Além disso, entende-se também que estas metodologias denominadas ágeis visam agilizar o aprendizado durante o processo e não o prazo de entrega final, garantindo a entrega correta e não apenas mais rápida.

Pode-se observar também o papel da comunicação e participação das pessoas como ponto chave na implementação de novos processos, evidenciado pelos desafios citados pelos gestores de projeto.

Como lacuna entende-se a amostra pesquisada como limitada, dada a variedade de organizações que utilizam processos ágeis de desenvolvimento de software. Para oportunidades de pesquisas futuras, recomenda-se que sejam estudadas as metodologias adaptativas de desenvolvimento de software, criadas a partir da junção de duas ou mais metodologias ágeis.

REFERÊNCIAS

ALMEIDA, Mário de Souza. **Elaboração de projeto, TCC, dissertação e tese:** uma abordagem simples, prática e objetiva. São Paulo: Atlas, 2014.

AMBLER, S. W.; LINES, M. **Disciplined agile delivery:** a practitioner's guide to agile software delivery in the enterprise. Upper Saddle River, Nj: Ibm Press, 2012.

APPELO, J. **Management 3.0: leading Agile developers, developing Agile leaders**. Upper Saddle River, Nj: Addison-Wesley, 2011.

BECK, K. ET AL. **Manifesto for Agile Software Development**. Disponível em: <https://agilemanifesto.org/> Acesso em 14 de julho de 2022.

BECK, K. **Extreme programming explained:** embrace change. Boston, Mass.: Addison-Wesley, 2004.

FUGGETTA, A. Software Process: A roadmap. n. 10.1145/336512.336521, 2000.

GARTNER, L. **The Rookie Primer**. [s.l.] Radcliffe Rugby Football Club, 1996.

GIL, A. S. **Como elaborar projetos de pesquisa**. São Paulo: Atlas, 2002.

GOMES, A. F. **Agile: desenvolvimento de software com entregas frequentes e foco no valor do negócio**. [s.l.] Edição Eletrônica: Casa do Código, 2013.

GOMES, A.; WILLI, R.; REHEM, S. O Manifesto Ágil. In: **Métodos ágeis para desenvolvimento de software**. [s.l.] Bookman, 2014.

PRESSMAN, R. S. **Engenharia de software:** uma abordagem profissional. [s.l.] Porto Alegre (RS): Amgh, 2011.

SCHWABER, K. **SCRUM Development Process**. Business Object Design and Implementation, p. 117–134, 1997.

SCHWABER, K.; SUTHERLAND, J. **Scrum Guide:** The Definitive Guide to Scrum. [s.l.] Scrum Guides, 2020.

SOMMERVILLE, I. **Software engineering**. Boston: Pearson/Addison-Wesley, 2004.

SOMMERVILLE, I. **Engenharia de Software**. 9. ed. São Paulo: Pearson Prentice Hall, 2011.

SUTHERLAND, J. **Jeff Sutherland's Scrum Handbook**. [s.l.] Scrum Training Institute Press, 2010.

TAKEUCHI, H.; NONAKA, I. **The new product development game**. Journal of Product Innovation Management, v. 3, n. 3, p. 205–206, set. 1986.

TELES, V. M. **Extreme Programming:** Aprenda como encantar seus usuários desenvolvendo software com agilidade e alta qualidade. São Paulo: Novatec Editora, 2014.

APÊNDICES

APÊNDICE 1: Roteiro de Entrevista

1. Há quanto tempo você atua como gestor de projetos/produtos?
2. Você ocupa um papel de liderança dentro do time de desenvolvimento?
3. Há quanto tempo você atua em times que utilizam metodologias ágeis de desenvolvimento?
4. Você acredita que as metodologias ágeis possuem um impacto positivo no desenvolvimento de software? Por quê?
5. Qual metodologia ágil você utiliza?
6. Como foi seu primeiro contato com essa metodologia ágil?
7. Qual sua opinião sobre essa metodologia ágil?
8. Na sua opinião, qual seria a maior vantagem dessa metodologia ágil?
9. Na sua opinião, qual seria a maior desvantagem dessa metodologia ágil?
10. Na sua empresa, essa metodologia é adotada com algum tipo de adaptação ou alteração? Caso sim, qual?
11. O que você mudaria nessa metodologia?