PONTIFÍCIA UNIVERSIDADE CATÓLICA DE GOIÁS
ESCOLA POLITÉCNICA
GRADUAÇÃO EM ENGENHARIA DE COMPUTAÇÃO

O Estado da Arte em Metodologias e Práticas Ágeis de Desenvolvimento de Software

LAIS PEREIRA FELIPE

GOIÂNIA
2021

Lais Pereira Felipe

O Estado da Arte em Metodologias e Práticas Ágeis de Desenvolvimento de Software

Trabalho de Conclusão de Curso II apresentado à Escola Politécnica, da Pontifícia Universidade Católica de Goiás, como parte dos requisitos para a obtenção do título de Bacharel em Engenharia de Computação.

Orientadora: Ma. Adriana Silveira De Souza
Coorientador: Dr. Juliano Lopes de Oliveira

GOIÂNIA
2021

AGRADECIMENTOS

Agradeço aos meus companheiros de pesquisa, a professora Adriana Silveira de Souza, o professor Juliano Lopes de Oliveira e a Amanda Kelly Rodrigues Cândido, sendo os dois primeiros pesquisadores seniores e a última pesquisadora iniciante, que foram partes fundamentais na elaboração e construção deste trabalho. Agradeço também por toda a dedicação e orientação dos pesquisadores seniores desta revisão.

“Decidi amar o destino que me escolheu.”
O Rei Eterno

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

Resumo

Contexto: Metodologias Ágeis de desenvolvimento de software têm sido discutidas há mais de duas décadas e propõem diferentes abordagens para lidar com desafios de entrega de *software* de qualidade, em tempos cada vez mais curtos e com mudanças constantes de requisitos. **Objetivo**: compreender as principais propostas apresentadas na literatura científica atual sobre metodologias e práticas ágeis de desenvolvimento de *software*. **Método**: foi realizado um mapeamento sistemático da literatura, considerando publicações realizadas a partir do ano de 2019. **Resultados**: foram identificadas as metodologias ágeis mais referenciadas na atualidade e as práticas ágeis mais discutidas no contexto de desenvolvimento de *software*.

Abstract.

***Context**: Agile software development methodologies have been dis- cussed for over two decades and propose different approaches to deal with chal-lenges related to delivering quality software, in increasingly shorter times and with constant changes in requirements. **Objective**: to understand the main proposals presented in the current scientific literature on agile software development methodologies and practices. **Method**: a systematic mapping of the literature was performed, considering publications made since 2019. **Results**: the identification of the current most referenced agile methodologies and the most discussed agile practices in the context of software development.*

1. Introdução

Apesar de existirem há mais de duas décadas [8], ainda ocorre uma significativa falta de conhecimento e de compreensão sobre metodologias ágeis e sobre a forma de utilização de práticas ágeis de desenvolvimento de *software* [25, 30].

Um dos fatores que contribuem para dificultar a compreensão de metodologias ágeis de desenvolvimento de *software* é a diversidade de propostas metodológicas e de práticas de desenvolvimento de *software*, tanto técnicas quanto gerenciais, propostas por estas metodologias. Embora lidem com desafios comuns, relacionados com a necessidadede entrega de software de qualidade em prazos curtos e dentro de cenários com mudanças frequentes de requisitos, as chamadas metodologias ágeis de desenvolvimento de *software* apresentam mais diferenças do que semelhanças.

Este trabalho apresenta os resultados de um mapeamento sistemático da literatura conduzido com o objetivo de contribuir para uma melhor compreensão dessas metodologias ágeis e das práticas de engenharia de *software* por elas adotadas.

O restante deste trabalho descreve o processo de mapeamento da literatura realizado e está organizado da seguinte forma. A Seção 2 apresenta o protocolo utilizado na condução do trabalho de mapeamento. A Seção 3 descreve a forma como o protocolo foi seguido para realização do mapeamento da literatura. A Seção 4 discute os resultados obtidos do trabalho de mapeamento. Finalmente, a Seção 5 traz conclusões sobre o trabalho e indica direções para futuras pesquisas.

2. Protocolo de Mapeamento Sistemático da Literatura

O trabalho de mapeamento da literatura foi feito usando como principal ferramenta de apoio o *software* Parsifal [45]. O estudo foi conduzido com três motores (máquinas) de busca de artigos: *IEEE Xplore* [17], *ACM Digital Library* [1] e *Scopus* [11].

2.1. Questões de Pesquisa

A primeira etapa para definição do protocolo de mapeamento destinou-se a determinar os parâmetros PICOC (*Population, Intervention, Comparison, Outcome, Context*). Com o objetivo de identificar um conjunto de práticas comumente citadas por metodologias de desenvolvimento de software que se autodenominam como ágeis, esses parâmetros foram definidos da seguinte forma:

- População: Metodologias de desenvolvimento de software que se autodenominam como ágeis.
- Intervenção: Práticas citadas pelas metodologias.
- Comparação: Comparar a frequência de citação de práticas pelas metodologias.
- Resultado: Um conjunto de práticas comumente citadas por metodologias de desenvolvimento de software que se autodenominam como ágeis.
- Contexto: Avaliar a população tanto da indústria como da academia para identificar similaridades.

Em seguida foram estabelecidas as questões de pesquisa que orientaram a metodologia deste estudo. São elas:

Q1: Quais metodologias se autodenominam, ou são referenciadas como, metodologias ágeis de desenvolvimento de *software*?

Q2: Quais práticas de engenharia de *software* são referenciadas ou recomendadas como práticas ágeis de desenvolvimento de *software*?

Q3: Quais das práticas ágeis de desenvolvimento de *software* são referenciadas ou recomendadas por metodologias ágeis de desenvolvimento de *software*?

2.2. Palavras-chave para Pesquisa

A partir dos parâmetros PICOC a ferramenta de apoio ao mapeamento (Parsifal) gerou um conjunto de palavras-chave:

"Metodologias de desenvolvimento de *software* que se autodenominam como ágeis"; "Práticas citadas pelas metodologias"; "Comparar a frequência de citação de práticas pelas metodologias"; "Conjunto de práticas comumente citadas por metodologias de desenvolvimento de software que se autodenominam como ágeis".

A ferramenta também gerou uma sentença lógica (*string* de busca) para ser usada como chave de pesquisa nas máquinas de busca:

> (("Metodologias de desenvolvimento de *software* que se autodenominam como ágeis") *AND* ("Práticas citadas pelas metodologias") *AND* ("Comparar a frequência de citação de práticas pelas metodologias") *AND* ("Um conjunto de práticas comumente citadas por metodologias de desenvolvimento de software que se autodenominam como ágeis")).

No entanto, ao traduzir e aplicar esta sentença nos motores de busca, foi observado que os resultados traziam uma quantidade excessiva de trabalhos não relacionados ao tema de interesse da presente revisão de literatura e não abrangiam certos documentos que os pesquisadores esperavam que se enquadrassem no resultado da pesquisa.

Assim, as palavras-chave foram ajustadas para "Práticas citadas por metodologias" e "Metodologias de desenvolvimento de software que se autodenomina ágeis" e em seguida traduzidas para o inglês, resultando em: ("*Practices cited by methodologies*"; "*Software development methodologies that call themselves agile*"). Com isso, a nova string de busca utilizada neste estudo passou a ser:

> *("Software development methodologies that call themselves agile" OR "Agile software development methodologies" OR "Agile software development processes") AND ("Practices" OR "Activities" OR "Ceremonies" OR "Practices" OR "Tasks").*

A aplicação dessa nova string de busca gerou resultados satisfatórios, tanto do ponto de vista quantitativo quanto em relação ao conteúdo dos trabalhos recuperados pelas máquinas de busca, como discute a Seção 3.1.

2.3. Critérios de Inclusão e Exclusão

Por fim, foram determinados os critérios de inclusão e exclusão de trabalhos nos resultados do mapeamento sistemático, da seguinte forma:

- Critérios de inclusão de trabalhos:
    1. Trata de metodologias ágeis de desenvolvimento de *software*.
    2. Trata de práticas ágeis de desenvolvimento de *software*.
- Critérios de exclusão de trabalhos:
    1. Não aborda desenvolvimento ágil de *software*.
    2. Trata ensino e não aplicação de práticas de desenvolvimento de *software*.
    3. Texto completo do trabalho não disponível de forma livre.
    4. Pesquisa em andamento, sem apresentação de resultado final do trabalho.
    5. Artigo não escrito em inglês.
    6. Publicação original anterior a 2019.

3. Realização do Mapeamento Sistemático

A aplicação do protocolo de mapeamento sistemático definido na seção anterior ocorreu no período de junho a dezembro de 2021. A equipe que conduziu o estudo foi composta por quatro membros, sendo dois pesquisadores experientes e dois iniciantes.

3.1. Seleção e Importação de Estudos

Foram encontrados 52 estudos usando a string de busca e com filtro de data de publicação igual ou superior ao ano de 2019. A Figura 1 mostra a quantidade de artigos recuperados das máquinas de busca por ano de publicação. Em 2019 foram recuperados 23 artigos, sendo que 3 deles estavam duplicados, isto é, foram encontrados em mais de uma máquina de busca. No ano de 2020 foram obtidos 22 estudos e em 2021 foram recuperados 7 artigos.

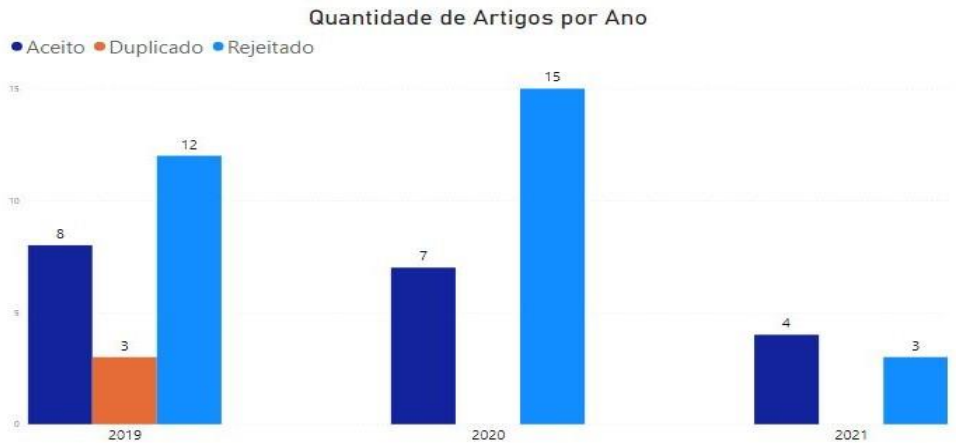

Figura 1. Quantidade de artigos aceitos, rejeitados e duplicados por ano.

Dos três trabalhos duplicados, dois deles foram encontrados na *IEEE* e na *ACM* ([32] e [31]) e o terceiro artigo duplicado ([21]) foi encontrado na Scopus e na *IEEE*.

A Tabela 1 mostra os 30 trabalhos rejeitados com base nos critérios de exclusão. Nenhum artigo foi excluído por ser um trabalho em andamento, por não ser escrito em inglês ou por ser uma publicação anterior a 2019.

Critério de Exclusão	Trabalhos Excluídos
1	[6] [10] [12] [15] [18] [20] [24] [26] [28] [37] [46] [9] [21] [32] [34] [36] [40] [38] [43] [53]
2	[13] [48] [52]
3	[5] [16] [27] [31] [47] [49]

Tabela 1. Conjunto completo de trabalhos não aceitos

A Figura 2 exibe a quantidade de estudos obtidos por motor de busca. A *Scopus* foi a fonte que trouxe a maior quantidade de publicações, sendo ao todo 25. Já a *ACM* retornou um total de 24 artigos. Por fim, a *IEEE* foi a fonte que retornou a menor quantidade de publicações, com um total de apenas 3 trabalhos.

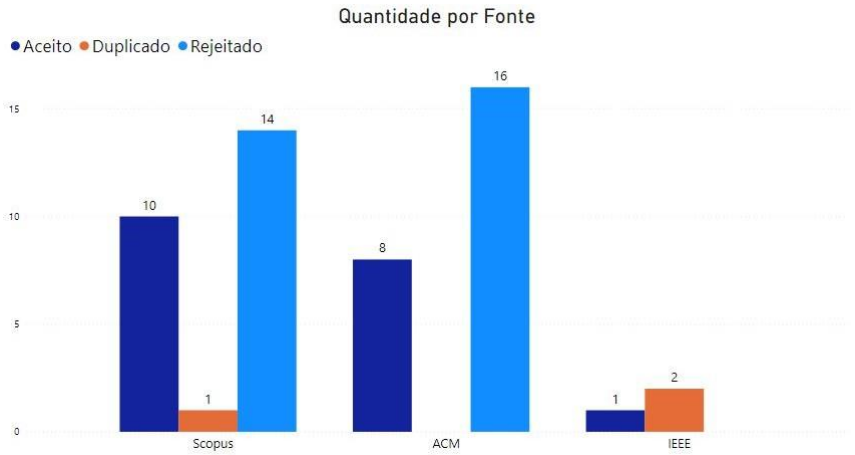

Figura 2. Quantidade de artigos aceitos, rejeitados e duplicados por fonte.

Após a remoção de duplicados e aplicação dos critérios de exclusão, foram aceitos 19 artigos para revisão neste trabalho. A Tabela 2 mostra todos esses artigos, agrupados por ano de publicação.

Ano de Publicação	Trabalhos Revisados
2019	[7] [23] [25] [29] [30] [33] [50] [51]
2020	[4] [3] [39] [22] [35] [41] [54]
2021	[14] [42] [44] [55]

Tabela 2. Conjunto completo de trabalhos revisados

3.2. Síntese dos Estudos Aceitos

Os trabalhos incluídos na presente revisão da literatura foram divididos em cinco grupos, com base no assunto específico que é trabalhado no artigo, dentro do contexto de metodologias ágeis de desenvolvimento de *software*.

No primeiro grupo estão artigos que apresentam revisões de metodologias ágeis ou comparações entre essas metodologias e outros tipos de metodologias de desenvolvimento de *software*. A seguir aparecem trabalhos que tratam da engenharia de requisitos em metodologias ágeis. Uma classe notável de requisitos aparece no terceiro grupo de artigos,

que trata especificamente de requisitos de segurança em metodologias ágeis. O quarto grupo contempla artigos que tratam de aspectos humanos e organização de equipes ágeis. Finalmente, o quinto grupo traz trabalhos que discutem a aplicação de metodologias ágeis em domínios específicos de aplicação, tais como domínio automotivo e bancário.

3.2.1. Revisões e Comparações de Metodologias Ágeis

O trabalho [3] apresenta uma revisão sobre valores e princípios de metodologias ágeis. São discutidas as principais ideias e práticas de cinco metodologias ágeis: *Feature Driven Development (FDD); Test-Driven Development (TDD); Dynamic System Development Model (DSDM); Scrum*; e *Extreme Programming (XP)*. O artigo também cita, mas não discute, as metodologias *Crystal Methods* e *Agile Big Data Analytics (AABA)*.

[55] discorre sobre os diversos processo de desenvolvimento de software dentro da indústria chinesa. O documento faz menção às abordagens de desenvolvimento tradicionais e ágeis, no entanto, não tem foco e tampouco descreve metodologias ou práticas ágeis. Foram disponibilizadas, no artigo, duas imagens que representam as abordagens e as práticas mais utilizadas pela população pesquisada, porém a imagem que trata das práticas não deixa claro quais são consideradas ágeis. Já no caso das metodologias citadas, há uma indicação de quais são consideradas ágeis, sendo o *Kanban* a mais utilizada nas indústrias chinesas pesquisadas.

[41] propõe uma abordagem de adaptação de uso de metodologias ágeis chamada "Adaptação Heterogênea". Essa abordagem alinha práticas ágeis entre diferentes esquadrões (equipes), cada um com sua metodologia de desenvolvimento e com seus objetivos específicos, para que possam atingir um objetivo de projeto comum para todos. O estudo de caso do artigo é em cima do modelo de adaptação heterogênea do *Spotify*. O trabalho não discorre sobre uma metodologia especifica, apenas menciona *Scrum, XP, DSDM, Crystal, ASD, FDD, RUP, Kanban* e *Lean*.

[51] apresenta os modelos *MDSIC* e *MDSIC-M*, voltados para desenvolvimento rápido, cumprindo requisitos para pequenas e médias empresas. O artigo apresenta uma revisão sobre metodologias de desenvolvimento de software e as compara com *MDSIC-M*. O artigo distingue modelo de desenvolvimento de software, que é 'uma representação simplificada do processo de desenvolvimento de software a partir de uma visão específica", de metodologia de desenvolvimento de software, que é "uma abordagem

estruturada para desenvolvimento de *software* incluindo modelos de sistemas, notação, regras, sugestões de design e um guia de processo". O artigo compara com o modelo *MDSIC* a *Metodologia Crystal, DSDM (Dynamic Software Development Method), Lean SoftwareDevelopment, Scrum, AUP (Agile Process Unified)* e *Extreme Programming (XP).*

3.2.2. Requisitos em Metodologias Ágeis

[14] discute técnicas para priorização de requisitos em abordagens ágeis de desenvolvimento de software. O artigo afirma que a principal diferença entre abordagens tradicionais e ágeis é justamente o mecanismo para priorização de requisitos, pois nas metodologias tradicionais esse mecanismo é usado no início do projeto enquanto em abordagens ágeis ele é aplicado em cada iteração do projeto.

[4] propõe uma estrutura em que o modelo *Design Thinking* é integrado ao Scrum no contexto de gerenciamento de engenharia de requisitos, além de apresentar uma comparação entre ambos. Cita também *Extreme Programming (XP)* e *Kanban* como algumas das metodologias ágeis mais adotadas pela indústria, porém foca em *Scrum* como objeto de estudo. Identifica as seguintes práticas: planejar atividades a serem realizadas nas iterações; planejamento de iteração; uso de ciclos de trabalho consecutivos; *backlog* do produto constituído de definições de recursos para *software* e histórias de usuário;

Segundo [54], requisitos não funcionais usualmente são ignorados no desenvolvimento de *software* ágil. Por isso o trabalho propõe uma abordagem semiautomática para ajudar analistas e desenvolvedores na elicitação de requisitos não funcionais em metodologias ágeis, utilizando um ambiente de computação em nuvem. O artigo não se aprofunda em qualquer tipo de metodologia, porém tem como objetivo o uso da abordagem apresentado em um ambiente de desenvolvimento ágil, citando *Scrum* como exemplo. O estudo é baseado nas práticas relacionadas a histórias de usuário e a revisar e definir características necessárias ao projeto.

3.2.3. Segurança em Metodologias Ágeis

A proposta apresentada em [7] descreve mecanismos para especificar requisitos de segurança e privacidade de dados usando histórias de usuário, priorizadas em um *backlog* de proteção de dados. Os requisitos são associados a políticas de controle de acesso visando a conformidade com normas e leis, tais como a *GDPR* européia.

O artigo [29] também discute práticas ágeis de desenvolvimento focadas em requisitos de segurança e visando a incorporação destas práticas em normas e padrões de segurança de *software*, como por exemplo o *OWASP ASVS*. O artigo identifica diversas práticas ágeis relacionadas à segurança de software.

Outro trabalho que trata de questões de segurança em metodologias ágeis é [22], que propõe um modelo conceitual de segurança para facilitar a adoção de práticas de *DevOps* associadas a tecnologias de software aberto e computação em nuvem. O artigo destaca, como práticas notáveis de *DevOps*, as práticas típicas de processos tradicionais, qualificadas com o termo **contínuo**: planejamento contínuo, *design* e desenvolvimento contínuos, integração, teste, entrega e implantação contínuos, registros e monitoramento contínuos. Além disso, o artigo ressalta as práticas relacionadas a mecanismos de colaboração, comunicação e *feedback*.

3.2.4. Equipes em Metodologias Ágeis

[33] apresenta um estudo sobre crenças, atitudes sobre práticas ágeis e como essas refletem na construção de artefatos produzidos por equipes de desenvolvimento de software. Em especial, discute o emprego de abordagens ágeis de desenvolvimento e como as pessoas do time as percebem. As práticas do *Scrum* e *XP* são as mais utilizadas pelos times. O artigo também cita, como um fator crítico para o sucesso de produtos de software com qualidade, a cooperação e o empoderamento do time trazidos pelos valores e princípios ágeis. Entre as práticas utilizadas pelas equipes estão: construir orientado a teste, implementar a propriedade de código coletivo, construir histórias breves de usuário, iniciar a implementação próxima do *deadline*, efetuar verificação constante e mais cedo possível e conduzir revisão de código.

[50] trata a globalização no desenvolvimento de *software* (*GSD*) e o uso de práticas ágeis como estratégias para lidar com a distância geográfica, temporal e sóciocultural das equipes. O artigo destaca as metodologias *Scrum* e *XP*. Como práticas do *Scrum* o artigo apresenta *daily stand-ups, sprint planning, retrospectives, sprint reviews,produto backlog, self-organization* e multidisciplinaridade. Do *XP* o artigo cita as práticas *planning game, small releases, metaphor, simple design*, desenvolvimento orientado a teste, refatoração, programação em pares, propriedade coletiva do código, integração contínua, 40 horas por semana, cliente *on-site* e padrão de codificação. O artigo supõe que as abordagens ágeis focam na responsividade às necessidades do cliente e que

por isso a qualidade e a produtividade melhoram. Também assume que, com a adoção de práticas ágeis, a satisfação do cliente aumenta com entregas contínuas de software com valor. O artigo não define o que é uma abordagem ágil, mas apresenta suas características e tipos de problemas que são melhor tratados por esta abordagem.

3.2.5. Metodologias Ágeis em Domínios Específicos

O trabalho [30] discute desafios de uso combinado de paradigmas ágeis e tradicionais no planejamento de liberações (*releases*) que combinam software e hardware no domínio automotivo. Em domínios fortemente regulamentados, como o automotivo, que envolvem funções críticas, predomina o uso de paradigmas tradicionais de desenvolvimento. Todavia, ocorre com frequência cada vez maior a adoção de práticas ágeis isoladas, visando a rapidez do desenvolvimento de sistemas nesses domínios. O artigo sugere que práticas ágeis como reuniões diárias rápidas (standup meeting) e integração contínua podem contribuir para a superação de muitos desafios identificados.

[44] apresenta uma pesquisa em andamento que envolve metodologias ágeis, focando apenas no contexto de sustentabilidade de software. Neste contexto, o trabalho identifica uma metodologia (XP) e as seguintes práticas ágeis: Programar em pares (*Pair Programming*), Reunir equipe diariamente (*Daily stand-up meetings*), Planejar poriteração (*Sprint Planning*), Revisar processo da iteração (*Retrospective Meetings*) e Testar unidade (*Unit testing*). O artigo também indica como práticas ágeis: *Face-to-face communication, changes enven late in the requirements, Iterative development, Minimaldocumentation, Polymorphic design* e *Overlapping pair rotation*.

[35] apresenta um estudo sobre a efetividade da adoção de metodologias ágeis de desenvolvimento de software na criação de sistemas críticos. O trabalho também discute como seria uma possível a aplicação da estrutura apresentada em diferentes casos neste contexto. Foram analisadas algumas práticas adotadas em diferentes ramos industriais no grupo de sistemas críticos, com destaque para: entrega funcional do protótipo de software e integração do cliente no projeto; divisão de tarefas de implementação e divisão de tarefas de especificação formal a cada iteração.

[42] aborda o desenvolvimento ágil dentro de uma instituição bancária. São mencionadas duas metodologias ágeis utilizadas no ambiente de pesquisa: o Scrum e o ScrumBan. O artigo apresenta um estudo para melhorar o processo ágil dentro do banco para que se matenha ágil. Entre as práticas citadas estão: verificação constante

e mais cedo possível e conduzir revisão de código.

[25] discute como as abordagens ágeis podem ser utilizadas com a representação BPM (*Business Process Modeling*) em outros domínios. O artigo menciona que, embora a pesquisa sobre abordagens ágeis tenha mais de duas décadas, não existe uma definição padronizada do que realmente significa ágil. O artigo critica que o bem conhecido manifesto descreve agilidade de forma muito geral. No entanto, as práticas são consideradas rigorosas e operacionais. Adaptações em outros domínios também não são claras e suas representações são equivocadas. A indústria tende a implementar somente partes do manifesto ágil, não seguindo regras rígidas das metodologias reais. As empresas não conseguem avaliar o nível de agilidade nos processos de negócio. O artigo mostra diferentes práticas, mas não as associa a uma metodologia ágil específica. Entre as metodologias ágeis citadas estão *XP*, *Scrum* e *DSDM*.

3.3. Análise de Qualidade dos Estudos Aceitos

Os estudos analisados, em sua maioria, não fazem uma ligação clara entre práticas e metodologias ágeis e tampouco fornecem uma definição precisa dos conceitos de metodologia e de prática de desenvolvimento de software. Nota-se, ainda, que na maioria dos trabalhos o foco não é a metodologia ágil, mas as práticas ágeis.

Muitos trabalhos revisados visam mesclar práticas de diferentes metodologias de desenvolvimento de software, seja combinando práticas de metodologias ágeis distintas, ou aplicando práticas ágeis em metodologias tradicionais voltadas para um domínio específico de aplicação. Além disso, foram observadas muitas propostas de adaptações e combinações de metodologias ágeis com outras abordagens de desenvolvimento de software e a indicação de práticas ágeis que são apenas adaptações de práticas de metodologias tradicionais.

Também foram observados trabalhos de pesquisa sobre a indústria de software em regiões específicas, com o objetivo de reunir práticas e metodologias mais utilizadas no mercado de desenvolvimento de software dessas regiões.

4. Resultados do Mapeamento Sistemático

O vocabulário da Engenharia de *Software* [19] define uma metodologia como um sistema de práticas, técnicas, procedimentos e regras utilizados por aqueles que trabalham em uma disciplina. Uma metodologia deve especificar não apenas o processo a ser empregado e os produtos de trabalho a serem utilizados e gerados, mas também deve considerar as pessoas e ferramentas envolvidas, durante um esforço de desenvolvimento.

Logo, uma metodologia de desenvolvimento de *software* é um sistema de práticas, técnicas, procedimentos e regras utilizados por equipes de desenvolvimento de *software* para organizar o trabalho realizado durante o projeto de desenvolvimento de *software*. Essa organização do trabalho compreende o processo de desenvolvimento de *software* empregado, os produtos de trabalho utilizados como entrada em cada atividade de desenvolvimento e gerados nestas atividades, os papeis das pessoas que realizam cada atividade (tais como projetista, programador e testador) e as ferramentas empregadas para realizaras atividades durante o projeto de desenvolvimento de *software*.

Todavia, quando se adiciona o adjetivo *"ágil"* ao conceito de metodologia de desenvolvimento de *software*, definições e conceitos se tornam menos precisos e mais subjetivos. Por exemplo, a Aliança Ágil, uma das organizações mais conhecidas pela comunidade internacional de desenvolvedores de *software*, define o termo **Ágil** como *uma atitude mental baseada nos valores e princípios do Manifesto Ágil [8], que fornecem orientação sobre como responder a mudanças e lidar com incertezas* [2].

4.1. Metodologias Ágeis

Poucos dos trabalhos revisados referenciam ou recomendam metodologias específicas de desenvolvimento ágil de *software*. A maioria estudos, como por exemplo [25] [43] [56], discutem apenas práticas de desenvolvimento de *software* consideradas ágeis, sem situá-las em uma determinada metodologia, conforme discute a Seção 4.2.

As metodologias ágeis citadas por mais de três dos estudos revisados são apresentadas na Tabela 3, em ordem decrescente de número de citações.

Metodologia Ágil	Citada por
Scrum	[7] [23] [25] [29] [30] [33] [50] [51] [4] [3] [41] [54] [42] [55]
Extreme Programming (XP)	[7] [23] [25] [29] [33] [50] [51] [4] [3] [41] [44] [55]
Dynamic System Development Model (DSDM)	[25] [33] [51] [3] [41] [55]
Kanban	[29] [4] [41] [42] [55]
Crystal Methods	[51] [3] [41] [55]

Tabela 3. Metodologias ágeis mais citadas na literatura recente

A Tabela 3 fornece uma visão clara de que os autores convergem para duas metodologias específicas: *Scrum* e *XP*. Essas duas metodologias são há muito conhecidas e discutidas. As metodologias que surgiram posteriormente não mostram ter muito destaque nas pesquisas. Uma hipótese, com base nos estudos lidos, é que em geral não há grande interesse em utilizar ou estudar a metodologia em si, mas sim suas práticas a fim de adaptá-las para contextos específicos.

A Tabela 4 apresenta metodologias ágeis que foram citadas por até três dos artigos revisados.

Metodologia Ágil	Citada por
Scaled Agile Framework (SAFe)	[29] [30] [55]
Adaptative Software Development (ASD)	[23] [41]
DevOps e variantes: DevSecOps	[29] [22]
Feature Driven Development (FDD)	[3] [41]
Lean Software Development	[51] [55]
RUP e variantes: AUP (Agile Process Unified)	[51] [41]
ScrumBan	[42] [55]
Test-Driven Development (TDD)	[3] [55]
Agile Big Data Analytics (AABA)	[3]
AgileSafe	[29]
Large-Scale Scrum (LESS)	[55]
Nexus	[55]

Tabela 4. Outras metodologias ágeis citadas na literatura recente

Considerando exclusivamente o conceito de "metodologia", algumas das citadas na Tabela 4 não são compatíveis com a definição da engenharia de *software* para este conceito. Por exemplo, *DevOps* não se restringe ao ciclo de vida de desenvolvimento de *software*, logo não pode ser considerado como uma metodologia de desenvolvimento de *software*. Outras abordagens citadas são meras variantes de outras metodologias, como por exemplo, o *Large-Scale Scrum* e o *Scrumban*. Há ainda casos, como o *Test-Driven*

Development (TDD), em que a metodologia não contempla todo o ciclo de vida de desenvolvimento de *software*.

A Aliança Ágil [2] explica que apesar de usarmos o termo "Metodologia" para as abordagens de desenvolvimento ágil, uma nomenclatura mais adequada seria *"Framework"* pois são utilizadas de forma generalizada, necessitando adaptar seu uso de acordo com o domínio em que são aplicadas, já que não há uma metodologia abrangente o suficiente para se enquadrar em qualquer contexto de desenvolvimento.

A Tabela 4 mostra que existem várias combinações de *Scrum* com outras metodologias, gerando variantes como *ScrumBan* e o *Large-Scale Scrum*. Seguindo então o que a Aliança Ágil afirma a respeito de metodologias serem, de fato, *Frameworks*, pode-se observar que o *Scrum* é muito utilizado como base para atividades de gestão de projeto, que é seu forte, e complementado com outras práticas para que atenda a diferentes domínios de desenvolvimento de *software*.

O vocabulário da Engenharia de Software, definido em [19], não contém o termo específico "Metodologia Ágil" de desenvolvimento de *software*. O conceito de Metodologia Ágil é citado, juntamente com outros tipos de metodologias (preditiva, iterativa e incremental), como exemplo de *abordagem de desenvolvimento* usada para criar e desenvolver um produto durante o ciclo de vida de um projeto.

Já a Aliança Ágil define Metodologias Ágeis de Desenvolvimento de *Software* como um conjunto de convenções que uma equipe escolhe seguir de forma a ter aderência aos valores e princípios ágeis em um projeto de desenvolvimento de *software* [2]. Não há, portanto, uma definição precisa e objetiva do que caracteriza uma metodologia de desenvolvimento de software como "ágil". O único requisito para ser considerada ágil é que a metodologia siga os valores e princípios do Manifesto Ágil.

4.2. Práticas Ágeis

As práticas citadas pelos estudos revisados são apresentadas nas Tabelas 5 a 10, em ordem decrescente de número de citações, ou seja, as práticas mais citadas aparecem primeiro em cada tabela. Quando os trabalhos que citam a prática não indicam a metodologia em que ela é aplicada, a tabela contém o símbolo "–".

Grosso modo, as práticas propostas nas metodologias ágeis podem ser categorizadas de acordo com as macro-atividades do ciclo de vida genérico de

desenvolvimento de *software*: Gestão de Projeto (contemplando Planejamento e Monitoramento / Controle), Requisitos, *Design*, Construção e Validação.

As Tabelas 5, 7 e 6, possuem predominantemente práticas relacionadas ao *Scrum*. Um fato interessante éque essa éuma metodologia voltada para a gestão do projeto, e as tabelas citadas são de atividades voltadas para o planejamento, monitoramento e controle do projeto e requisitos.

Prática Ágil - Atividade de Planejamento	
Citada por	**Usada em**
Planejar *release* em iterações curtas (*Sprint Planning*)	
[23] [30] [50]	*XP*
[4] [3] [55]	*Scrum*
Planejar tarefas e estimar esforço da iteração (prerrogativa da equipe)	
[23] [4] [3]	*Scrum, XP*
[41] [55] [42]	–
Desenvolver modelo geral do projeto	
[42] [3] [41]	*FDD*
Definir equipe (pequena e sem hierarquia) para o projeto	
[42] [30] [3]	*Scrum*
Comprometer-se com o escopo da iteração (prerrogativa da equipe)	
[30] [41] [42]	*Scrum, XP*
Criar e avaliar a viabilidade uma minuta de plano de projeto	
[42] [3]	*DSDM*
Incluir engenheiro de segurança na equipe de projeto	
[29] [22]	*DevSecOps*
Planejar atividades que não incluem valor direto para o usuário (*spikes*)	
[29]	*Scrum*

Tabela 5. Práticas ágeis relacionadas ao Planejamento de Projeto de *Software*

Prática Ágil - Atividade de Monitoramento e Controle	
Citada por	**Usada em**
Reunir equipe diariamente (*Daily stand-up meetings*)	
[25] [23]	*XP*
[4] [3]	*Scrum*
[44] [55]	–
Revisar processo da iteração (*Retrospective Meetings*)	
[23] [50]	*XP*
[3] [44] [55]	*Scrum*
Comunicar-se diretamente com *stakeholders* (prerrogativa da equipe)	
[30] [35] [41]	*Scrum*
Decidir sobre o trabalho na iteração (prerrogativa da equipe)	
[30] [35]	*Scrum*
[41]	–
Monitorar status da iteração (gráfico *burndown*)	
[3]	*Scrum*

Tabela 6. Práticas ágeis relacionadas ao Monitoramento e Controle de Projeto

Prática Ágil - Atividade de Requisitos	
Citada por	**Usada em**
Definir e revisar características, requisitos ou histórias de usuário	
[7] [33] [4]	*FDD*
[3] [54] [14]	*Scrum*
[55] [42]	*XP*
Manter priorizada uma única lista de funcionalidades (*backlog*)	
[7] [23] [4]	*DSDM*
[3] [55] [42]	*Scrum*
Decompor histórias épicas em histórias simples, *features* ou tarefas	
[7] [33] [42]	–
Priorizar requisitos e estimar sua complexidade	
[14] [42]	–
Adicionar condições de satisfação às histórias de usuário	
[7]	–
Mudar requisito somente fora do período de iteração	
[30]	*Scrum*
Definir requisitos de segurança (histórias de abuso ou mau uso)	
[29]	–

Tabela 7. Práticas ágeis relacionadas aos Requisitos de *Software*

A Tabela 8 possui associação com metodologias menos citadas entre os estudos revisados. Também dispõe de práticas, tal como Modelar caso de Uso, que são comuns em abordagens de desenvolvimento de *software* mais rígidas. Uma hipótese de possível do motivo para não haver muitas referências a práticas de *design* na literatura ágil é que essas práticas podem necessitar de mais documentação, o que vai contra os princípios ágeis, na opinião de muitos praticantes.

Prática Ágil - Atividade de Design	
Citada por	**Usada em**
Elaborar modelo funcional (prototipagem)	
[3] [55]	*DSDM*
Modelar Caso de Uso	
[23] [55]	–
Definir Arquitetura Global do *Software*	
[23] [42]	–
Projetar classes para características necessárias para os usuários	
[3]	*FDD*

Tabela 8. Práticas ágeis relacionadas ao *Design* de *Software*

Na Tabela 9 apesar de haver uma maior similaridade entre a quantidade de metodologias associadas às praticas (TDD, XP e Scrum estão associadas à mesma quantidade de práticas), há ainda uma dominância do XP na quantidade de citações pelos estudos. Já na Tabela 10 a prática mais citada, *Sprint Review*, está associada ao *Scrum*.

Prática Ágil - Atividade de Construção	
Citada por	**Usada em**
Refatorar o código	
[50] [3] [55]	*TDD, XP*
Integrar continuamente artefatos prontos na iteração	
[30] [35] [55]	*Scrum*
Programar em pares	
[50] [3] [55]	*XP*
Implementar tarefas por iteração	
[3] [55]	*Scrum*
Construir e disponibilizar *software* para teste pelos usuários	
[3] [42]	*DSDM*
Escrever código que satisfaz os casos de teste	
[3]	*TDD*
Implementar e inspecionar característica necessária para os usuários	
[3]	*FDD*

Tabela 9. Práticas ágeis relacionadas à Construção de *Software*

Prática Ágil - Atividade de Validação	
Citada por	**Usada em**
Revisar resultado da iteração (*Sprint Review*)	
[23] [30] [33] [50] [3]	*Scrum*
[55]	–
Testar unidade (*Unit testing*)	
[33] [50] [3]	*XP*
[44] [55]	–
Executar continuamente casos de teste automatizados	
[42] [3]	*TDD*
[22]	*DevSecOps*
Revisar código	
[33] [42] [55]	–
Testar segurança e automatizar verificações de vulnerabilidade	
[22] [55]	–
Executar testes funcionais elaborados por clientes do projeto	
[3]	*XP*
Revisar segurança como critério de prontidão da história de usuário	
[29]	–
Realizar teste ponta-a-ponta	
[55]	–

Tabela 10. Práticas ágeis relacionadas à Validação de *Software*

No vocabulário da Engenharia de Software, o termo *"prática"* édefinido como um tipo específico de atividade (técnica ou de gestão) que contribui para a execução de um processo e pode empregar uma ou mais técnicas e ferramentas [19].

Verifica-se que os conceitos de práticas e de princípios ágeis misturam-se. Por exemplo, a propriedade coletiva do código é definida como uma prática do *XP.* Esta ideia está mais associada com um princípio, por sua natureza mais holística, filosófica e geral.

5. Conclusões

A questão de pesquisa "Q1: Quais metodologias se autodenominam, ou são referenciadas como, metodologias ´ageis de desenvolvimento de *software*?" é respondida nas Tabelas 3 e 4 que comportam as metodologias citadas como ágeis pelos autores dos artigos selecionados e assim respondem a primeira questão de pesquisa.

Para a questão de pesquisa "**Q2**: Quais práticas de engenharia de *software* são referenciadas ou recomendadas como práticas ágeis de desenvolvimento de *software*?" é possível encontrar a resposta nas Tabelas 5 a 10 que contêm práticas de engenharia de *software* como desenvolver modelo geral do projeto, priorizar requisitos e estimar sua complexidade, modelar casos de uso, realizar teste de unidade e etc.

Também é possível encontrar a resposta para a questão de pesquisa "**Q3**: Quais das práticas ágeis de desenvolvimento de *software* são referenciadas ou recomendadas por metodologias ágeis de desenvolvimento de software?" nas Tabelas 5 a 10 que dispõem o conjunto de práticas ágeis que são referenciadas ou recomendadas por metodologias ágeis como por exemplo *Sprint Review* recomendada pelo *Scrum e Sprint Planning* também recomendada *pelo Scrum.*

Ao analisar os dados obtidos, é possível notar que predominantemente os autores citam mais o *XP* e o *Scrum*, assim como a maioria das práticas também estão associadas a essas duas metodologias. Apesar de existirem inúmeras outras metodologias, ainda se mantém o foco em metodologias mais antigas.

A Aliança Ágil afirma que as metodologias podem ser consideradas *Frameworks* visto que são adaptadas ao contexto em que são aplicadas. Essa pode ser uma das razões para tantas combinações entre diferentes metodologias ágeis, e até entre abordagens tradicionais e ágeis. Nesse cenário, o valor estaria voltado mais para suas práticas que para as próprias metodologias, visto que podem ser utilizadas em conjunto com outras abordagens.

Por fim, não é possível encontrar definição clara, de metodologia ou prática ágil. Também não há alguma característica comum entre os estudos que possam defini-las. Parece não haver um consenso entre os autores do que são metodologias ou práticas ágeis.

Trabalhos futuros:

- Pesquisa de campo com trabalhadores da indústria de software para confirmar

as hipóteses aqui identificadas.

- Realizar nova revisão sistemática, considerando trabalhos que foram descobertos ao longo da pesquisa e que não foram recuperados pela string de busca definida no protocolo.

Limitações:

- String de busca retornando poucos documentos na fonte de busca *IEEE*, mas não houve tempo de pesquisa para avaliar os documentos deixados de fora.

- Janela de tempo da pesquisa não torna possível definir se é uma área de pesquisa crescente, decrescente ou constante.

Referências

[1] ACM. ACM Digital Library. https://dl.acm.org/, 2021.

[2] AGILE ALLIANCE. Agile 101 - What is Agile? https://www.agilealliance.org/agile101/, 2021.

[3] AL-SAQQA, S., SAWALHA, S., AND ABDELNABI, H. Agile software development: Methodologies and trends. *International Journal of Interactive Mobile Technologies (iJIM) 14*, 11 (2020), 246–270. https://doi.org/10.3991/ijim.v14i11.13269.

[4] ALHAZMI, A., AND HUANG, S. Integrating desing thinking into scrum fra- mework in the contexto of requirements engeneering management. In *Proce- edings of 2020 3rd International Conference on Computer Science and Software Engeneering (CSSE'20)* (Beijing, China, May 2020), ACM, pp. 33–45. https://doi.org/10.1145/3403746.3403902.

[5] ALIZADEH, V., OUALI, M. A., KESSENTINI, M., AND CHATER, M. Refbot: Intelligent software refactoring bot. In *Proceedings of the 34th IEEE/ACM International Conference on Automated Software Engineering* (2019), ASE '19, IEEE Press, p. 823–834.

[6] ASHOUR, O. I. A., AND PUSATLI, T. Adoption of case tools & uml: A local study. In *Proceedings of the 2020 9th International Conference on Software and Information Engineering (ICSIE)* (New York, NY, USA, 2020), ICSIE 2020, Association for Computing Machinery, p. 6–10.

[7] BARTOLINI, C., DAOUDAGH, S., LENZINI, G., AND MARCHETTI, E. Gdpr-based user stories in the access control perspective. In *Proceedings of the 12th International Conference on Quality of Information and Communications Technology (QUATIC 2019)* (Ciudad Real, Spain, September 2019), Springer, pp. 3–17.

[8] BECK, K., BEEDLE, M., VAN BENNEKUM, A., COCKBURN, A., CUNNINGHAM, W., FOWLER, M., GRENNING, J., HIGHSMITH, J., HUNT, A., JEFFRIES, R., KERN, J., MARICK, B., MARTIN, R. C., MELLOR, S., SCHWABER, K., SUTHERLAND, J., AND THOMAS, D. Manifesto para Desenvolvimento Ágil de Software. https://agilemanifesto.org/iso/ptbr/manifesto.html, 2001.

[9] BORONAT, A. Code-first model-driven engineering: On the agile adoption of mde tooling. In *Proceedings of the 34th IEEE/ACM International Conference on AutomatedSoftware Engineering* (2019), ASE '19, IEEE Press, p. 874–886.

[10] ELFAKI, A. O. Introducing script as a software design tool for agile software development methodology. *2020 International Conference on Computing and Information Technology (ICCIT-1441)* (2020), 1–5.

[11] ELSEVIER. Welcome to Scopus Preview. https://www.scopus.com/home.uri,2021.

[12] FAUSTINO, J., PEREIRA, R., ALTURAS, B., AND MIRA DA SILVA, M. Agile infor- mation technology service management with devops: an incident management casestudy. *International Journal of Agile Systems and Management 13* (01 2020), 339.

[13] FLORES, V. Improving students' learning skills for developing a software project using active learning techniques: Service-learning and agile software development. *Inter-national Journal of Engineering Education* (2020).

[14] GOVIL, N., AND SHARMA, A. Information extraction on requirement prioritization approaches in agile software development processes. In *Proceedings of the Fifth International Conference on Computing Methodologies and Communication (ICCMC2021)* (2021), IEEE, pp. 1097–1100.

[15] HANG, K., OSTRANDER, T., ORR, A., ARCHER, J., AND ULAND, S. A five-year retrospective on an applied baccalaureate program in software development. In *Pro- ceedings of the 51st ACM Technical Symposium on Computer Science Education* (New York, NY, USA, 2020), SIGCSE '20, Association for Computing Machinery,p. 894–898.

[16] HUYNH, Q.-T., PHAM, L.-T., HA, N.-H., AND NGUYEN, D.-M. An effective approach for context driven testing in practice—a case study. *International Journal of Software Engineering and Knowledge Engineering 30*, 09 (2020), 1245–1262.

[17] IEEE. IEEE Xplore. `https://ieeexplore.ieee.org/`, 2021.

[18] INGRAM, C., CHUBB, J., BOARDMAN, C., AND URSU, M. Generating real-world impact from academic research: Experience report from a university impact hub. In *Proceedings of the IEEE/ACM 42nd International Conference on Software Engine-ering Workshops* (New York, NY, USA, 2020), ICSEW'20, Association for Compu- ting Machinery, p. 611–618.

[19] ISO/IEC/IEEE. *ISO/IEC/IEEE International Standard - Systems and software enginee- ring – Vocabulary*. International Organization for Standardization, Vernier, Geneva,Switzerland, 2017.

[20] KIALBEKOV, A. Empirical study on commonly used combinations of estimation techniques in software development planning. In *Proceedings of the 2020 European Symposium on Software Engineering* (New York, NY, USA, 2020), ESSE 2020, As- sociation for Computing Machinery, p. 44–50.

[21] KOÇ, G., AYDOS, M., AND TEKEREK, M. Evaluation of trustworthy scrum employment for agile software development based on the views of software developers. In *2019 4th International Conference on Computer Science and Engineering (UBMK)*(2019), pp. 63–67.

[22] KUMAR, R., AND GOYAL, R. Modeling continuous security: A conceptual model for automated devsecops using open-source software over cloud (adoc). *Computers & Security 97*, 10 (October 2020).

[23] KUSSUNGA, F. I., RIBEIRO, P. A., AND SANTOS, N. Characterization of a visual environment to support scrum ceremonies. In *Anais da Conferência da Associação Portuguesa de Sistemas de Informação* (2019), APSI. ISSN 2183-489X.

[24] LAM, C., VAN VELTHOVEN, M. H., AND MEINERT, E. Developing a blockchain-based supply chain system for advanced therapies: Protocol for a feasibility study. *JMIR Research Protocols 9*, 12 (2020), e17005.

[25] LEDERER, M., POPOVA, O., AND SCHMID, P. Can you see the wood for the trees? In *Proceedings of the S-BPM One ACM* (2019), ACM. DOI: 10.1145/3329007.332916ISBN 978-1-4503-6250-4/19/06.

[26] LIM, Z. Y., CHUA, J. M., YANG, K., TAN, W. S., AND CHAI, Y. Web accessibilitytesting for singapore government e-services. In *Proceedings of the 17th Internatio-nal Web for All Conference* (New York, NY, USA, 2020), W4A '20, Association for Computing Machinery.

[27] LINOS, P. K., RYBARCZYK, R., AND PARTENHEIMER, N. Involving it professionals in scrum student teams: An empirical study on the impact of students' learning. In *2020 IEEE Frontiers in Education Conference (FIE)* (2020), pp. 1–9.

[28] LIU, H., EKSMO, S., RISBERG, J., AND HEBIG, R. Emerging and changing tasks in the development process for machine learning systems. In *Proceedings of the International Conference on Software and System Processes* (New York, NY, USA,2020), ICSSP '20, Association for Computing Machinery, p. 125–134.

[29] LUKASIEWICZ, K., AND CYGAŃSKA, S. Security-oriented agile approach with agilesafe and owaspasvs. In *Proceedings of the Federated Conference on Computer Science and Information Systems (ACSIS)* (2019), IEEE, pp. 875–878. DOI: 10.15439/2019F213 ISSN 2300-5963.

[30] MARNER, K., THEOBALD, S., AND WAGNER, S. Real-life challenges in automotive release planning. In *Proceedings of the Federated Conference on Computer Science and Information Systems (ACSIS)* (2019), IEEE, pp. 831–839. DOI: 10.15439/2019F326 ISSN 2300-5963.

[31] MATTHIES, C. Agile process improvement in retrospectives. In *Proceedings of the 41st International Conference on Software Engineering: Companion Proceedings* (2019), ICSE '19, IEEE Press, p. 150–152.

[32] MATTHIES, C. Feedback in scrum: Data-informed retrospectives. In *Proceedings of the 41st International Conference on Software Engineering: Companion Proceedings* (2019), ICSE '19, IEEE Press, p. 198–201.

[33] MATTHIES, C., HUEGLE, J., DURCHMID, T., AND TEUSNER, R. Attitudes, beliefs, and developement data concerning agile software development practices. In *Proceedings of the International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET)* (2019), IEEE, pp. 158–169. DOI: 10.1109/ICSE-SEET.2019.00025.

[34] MKPOJIOGU, E., HASHIM, N., AL-SAKKAF, A., AND HUSSAIN, A. Software startups: Motivations for agile adoption. *International Journal of Innovative Technology andExploring Engineering 8*, 8S (2019), 454–459.

[35] MOUNIR, M., SALAH, A., KAMEL, A., AND MOUSSA, H. Framework to measure agile software process effectiveness in critical systems development. In *Proceedings of the 2020 9th International Conference on Software and Information Engineering (ICSIE)* (Cairo, Egypt, November 2020), ACM, pp. 25–32. https://doi.org/10.1145/3436829.3436853.

[36] NAYEBI, M., CAI, Y., KAZMAN, R., RUHE, G., FENG, Q., CARLSON, C., AND CHEW, F. A longitudinal study of identifying and paying down architecture debt. In *Proceedings of the 41st International Conference on Software Engineering: Software Engineering in Practice* (2019), ICSE-SEIP '19, IEEE Press, p. 171–180.

[37] PATKAR, N., MERINO, L., AND NIERSTRASZ, O. Towards requirements engineering with immersive augmented reality. In *Conference Companion of the 4th International Conference on Art, Science, and Engineering of Programming* (New York, NY, USA, 2020), ¡programming¿ '20, Association for Computing Machinery, p. 55–60.

[38] PICO-VALENCIA, P., HOLGADO-TERRIZA, J. A., AND PADEREWSKI, P. A systematic method for building internet of agents applications based on the linked open data approach. *Future generation computer systems 94* (2019), 250–271.

[39] PINHO, D., AND AGUIAR, A. The agileco pattern language: Physical en- vironment. In *Proceedings of European Conference on Pattern Languages of Programs 2020 (EuroPLoP '20)* (Germany, July 2020), ACM, pp. 1–9. https://doi.org/10.1145/3424771.3424790.

[40] RINDELL, K., BERNSMED, K., AND JAATUN, M. G. Managing security in software: Or: How i learned to stop worrying and manage the security technical debt. In *Proceedings of the 14th International Conference on Availability, Reliability and Security* (New York, NY, USA, 2019), ARES '19, Association for Computing Machinery.

[41] SALAMEH, A., AND BASS, J. M. Heterogeneous tailoring approach using the spotify model. In *Proceedings of the Evaluation and Assessment in Software Engineering* (New York, NY, USA, 2020), EASE '20, Association for Computing Machinery, p. 293–298.

[42] SCOTT, E., MILANI, F., KILU, E., AND PFAHL, D. Enhancing agile software development in the banking sector—a comprehensive case study at LHV. *Journal of software : evolution and process 33*, 7 (July 2021).

[43] SELVARAJ, Y., AHRENDT, W., AND FABIAN, M. Verification of decision making software in an autonomous vehicle: An industrial case study. In *International Workshopon Formal Methods for Industrial Critical Systems* (2019), Springer, pp. 143–159.

[44] SHAMSHIRI, H. Supporting sustainability design through agile software development. In *Evaluation and Assessment in Software Engineering (EASE 2021)* (Trondheim, Norway, June 2021), ACM, pp. 300–304. https://doi.org/10.1145/3463274.3463347.

[45] SIMPLE COMPLEX. About Parsifal. `https://parsif.al/about/`, 2021.

[46] SINHA, R., SHAMEEM, M., AND KUMAR, C. Swot: Strength, weaknesses, opportunities, and threats for scaling agile methods in global software development. In *Proceedings of the 13th Innovations in Software Engineering Conference on Formerly Known as India Software Engineering Conference* (New York, NY, USA, 2020), ISEC 2020, Association for Computing Machinery.

[47] SRIVASTAVA, A., MEHROTRA, D., KAPUR, P., AND AGGARWAL, A. G. A literature review of critical success factors in agile testing method of software development. In *The International Conference on Recent Innovations in Computing* (2020), Springer,pp. 859–870.

[48] SULAIMAN, S. Pairing-based approach to support understanding of object-oriented concepts and programming. *International Journal on Advanced Science, Engineering and Information Technology 10* (04 2020), 599.

[49] SZABÓ, D. M., AND STEGHÖFER, J.-P. Coping strategies for temporal, geographical and sociocultural distances in agile gsd: A case study. In *Proceedings of the 41st In- ternational Conference on Software Engineering: Software Engineering in Practice*(2019), ICSE-SEIP '19, IEEE Press, p. 161–170.

[50] SZABÓ, D. M., AND STEGHÖFER, J.-P. Coping strategies for temporal, geographical and sociocultural distances in agile gsd: Case study. In *41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)* (2019), IEEE/ACM, pp. 161–170.

[51] VALDEZ, J. L. C., MEDINA, H. F., CONTRERAS, G. A. V., MORALES, G. C., AND GONZÁ LEZ, A. H. G. Collaborative interated model in agile software developemnt (mdsic/mdsic-m) - case study and practical advice. *International Journal of Advanced Computer Science and Aplications - IJACSA 10*, 7 (2019), 424–432.

[52] VERDICCHIO, M. Hurricanes and pandemics: An experience report on adapting software engineering courses to ensure continuity of instruction. *J. Comput. Sci. Coll. 36*, 5 (jan 2021), 150–159.

[53] YALCINKAYA, M., AND SINGH, V. Visualcobie for facilities management: A bim integrated, visual search and information management platform for cobie extension. *Facilities* (2019).

[54] YOUNAS, M., JAWAWI, D. N. A., SHAH, M. A., MUSTAFA, A., AWAIS, M., ISHFAQ, M. K., AND WAKIL, K. Elicitation of nonfunctional requiriments in agile development using cloud computing environment. *IEEE Access 8* (August 2020), 209153–209162. 10.1109/ACCESS.2020.3014381.

[55] ZHOU, P., ALI KHAN, A. A., LIANG, P., AND BADSHAH, S. System and software processes in practice: Insights from chinese industry. In *Evaluation and Assessment in Software Engineering* (New York, NY, USA, 2021), EASE 2021, Association forComputing Machinery, p. 394–401.