UNIVERSIDADE ABERTA **INSTITUTO SUPERIOR TÉCNICO**

REVISÃO SISTEMÁTICA DE ESTUDOS SOBRE A APLICAÇÃO DO SCRUM NO GERENCIAMENTO DE EQUIPES DE DESENVOLVIMENTO DE SOFTWARE GEOGRAFICAMENTE DISTRIBUÍDAS

Dérick Hogan Pimenta

Mestrado em Informação e Sistemas Empresariais
(mestrado em associação)

2021

UNIVERSIDADE ABERTA **INSTITUTO SUPERIOR TÉCNICO**

REVISÃO SISTEMÁTICA DE ESTUDOS SOBRE A APLICAÇÃO DO SCRUM NO GERENCIAMENTO DE EQUIPES DE DESENVOLVIMENTO DE SOFTWARE GEOGRAFICAMENTE DISTRIBUÍDAS

Dérick Hogan Pimenta
1801270

Mestrado em Informação e Sistemas Empresariais
(mestrado em associação)

Dissertação orientada pelo Professor Doutor Miguel Mira da Silva

2021

RESUMO

O avanço da tecnologia é diretamente ligado ao nível de desenvolvimento da sociedade. Com investimentos em metodologias de desenvolvimento de software e o atual contexto socioeconômico mundial, principalmente após o primeiro trimestre de 2020 impactados pela pandemia de COVID-19, as tecnologias de informação foram forçadas a acompanhar as novas realidades, tanto no âmbito de proporcionar facilidades para quem as consome, quanto para quem as desenvolvem. Mesmo antes deste cenário, muitos desenvolvedores de softwares já desempenhavam suas atividades fora dos espaços corporativos tradicionais, onde adaptavam os processos de gestão a este contexto. Por se tratar de uma metodologia adaptativa e com rápidas respostas a mudanças, o *Scrum* é um *framework* de gestão amplamente utilizado por equipes de desenvolvimento de *software,* criado levando em consideração equipes locais de desenvolvimento. Para analisar a aplicação efetiva do *framework* nos novos formatos de trabalho, essa pesquisa teve como objetivo apresentar uma revisão sistemática de literatura tendo como base a aplicação do *Scrum* em contextos de equipes distribuídas. A metodologia utilizada foi a própria revisão sistemática de literatura, dividida em planejamento, condução e análise dos resultados, sendo utilizados critérios de qualidade, de inclusão e de exclusão para avaliar os estudos existentes sobre o tema. Foi possível perceber que existem muitas publicações sobre Scrum e métodos ágeis, porém, no contexto distribuído, foram encontradas poucas literaturas referentes. Apesar disso, foi possível identificar inibidores que precisam de atenção durante a implementação e, também, boas práticas que podem ser úteis nesse cenário. Destacam-se três pilares fundamentais para a boa e efetiva implementação: Cultura, pessoas e gestão. Pode-se concluir que o *Scrum* se baseia em uma cultura de trabalho e processos de aprendizados empíricos. Para uma aplicação efetiva e bons resultados, faz-se necessário um trabalho de educação e conscientização dos times sobre a motivação e a importância da utilização dos artefatos e eventos, uma vez que é necessário que a equipe enxergue valor na correta execução dos ritos, caso contrário, a implementação por si só não será bem aproveitada, não trará os resultados esperados e a sua proposta de valor de entregar soluções adaptativas para problemas complexos não será alcançada.

Palavras-chave: *Scrum*; Equipes distribuídas; Desenvolvimento de *software*.

ABSTRACT

Technology's advancement is directly linked to the level of society's development. Investments in software development methodologies and the current global socioeconomic context, especially after the first quarter of 2020 impacted by COVID-19 pandemic. Information technologies were widely impacted and forced to follow the new reality, providing facilities for those who consume these technologies, as well as for those who develop them. Even before this scenario, many software developers already performed their activities outside the traditional corporate spaces, where they adapted the management processes to this context. As it is an adaptive methodology and with quick responses to changes, Scrum is a management framework widely used by software development teams, as it was created taking into account local development teams. To analyse the effective application of the framework in new work formats, this research aimed to present a systematic review of the literature based on the application of Scrum in contexts of distributed teams. The methodology used was the systematic literature review, divided into planning, conducting and analysing results, using quality, inclusion and exclusion criteria to assess existing studies on the topic. It was possible to notice that there are many publications on Scrum and agile methods, however, in the distributed context, few referring literatures were found. Despite this, it was possible to identify inhibitors that need attention during implementation and also good practices that can be useful in this scenario. Three pillars stand out that are fundamental for good and effective implementation: culture, people and management. It can be concluded that Scrum is based on a work culture and empirical learning processes. For an effective application and good results, it is necessary to educate and raise awareness among the teams about the motivation and importance of using artefacts and events, since it is necessary for the team to see value in the correct execution of the rites, otherwise, the implementation alone will not be well used, it will not bring the expected results and its value proposition of delivering adaptive solutions to complex problems will not be achieved.

Keywords: Scrum; Distributed teams; Software development.

DEDICATÓRIA

Dedico este trabalho, primeiramente, a Deus, por me conceder a graça da inquietude e da busca por conhecimento sem medir dificuldades.

À minha esposa Ariadne, por ser minha principal motivadora, amiga e conselheira, e por não me permitir desistir até mesmo nos momentos em que eu senti que isso era evidente.

Aos meus filhos Gabriel e Rafael, por serem a materialização da pureza, do amor e a canalização de tudo o que há de bom neste mundo.

À minha mãe, por sempre ter me permitido e incentivado a almejar sonhos que, outrora, eram tão distantes.

Ao meu amigo e orientador Professor Doutor Miguel Mira da Silva, pela magnífica orientação e auxílio no desenvolvimento deste trabalho.

A todos os meus colegas de curso, principalmente Francisco e Nuno, por termos superado juntos, muitos desafios e percalços no decorrer deste.

Aos meus sócios e colegas de trabalho na Congresse.me, por sempre me motivarem a crescer profissionalmente e alçar voos cada vez maiores.

Ao meu amigo e sócio Luiz Gustavo, por ser tão presente e leal e me ajudar a ser uma pessoa melhor em todos os âmbitos de minha vida.

Ao meu amigo Paulo Sergio, por ter compartilhado um pouco de sua experiência com o Scrum e ajudado no decorrer deste trabalho.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE QUADROS

ÍNDICE DE GRÁFICOS

1. INTRODUÇÃO

No presente capítulo serão abordados a motivação e justificativa do tema de pesquisa, além dos objetivos de estudo e da metodologia utilizada para o desenvolvimento do trabalho.

1.1. Justificativa

A corrente tendência dos chamados “nômades digitais”, que fazem dos mais diversos e improváveis locais seus escritórios, possibilita, por exemplo, conhecer diversos países, locais, costumes e culturas ao mesmo tempo em que pode executar suas rotinas corriqueiras de trabalho, traz uma série de desafios que não se via, normalmente, acontecer até pouco tempo atrás. Além destes, diversos outros tipos de profissionais e classes passaram a executar suas tarefas sem a necessidade de ter de se locomover de suas residências.

Este fenômeno pode ser classificado sob o substantivo de teletrabalho, (Cavalcante & Jorge, 2013), haja vista poder ser executado em domicílio, em *coworkings*, entre países etc., bastando uma conexão com a internet.

Manter a produtividade e entrega do time podem ser desafios nesse contexto, sendo necessário a utilização de bons sistemas e ferramentas de gestão. Nesse cenário, as metodologias de gestão de projetos são de suma importância para manter o engajamento da equipe, bem como garantir uma saudável evolução dos projetos em andamento.

Segundo CollabNet (2019), o *Scrum* é o *framework* mais utilizado no mundo, sendo utilizado por 54% das empresas que aplicam algum tipo de metodologia de gestão em seus projetos e equipes.

Desta forma, este trabalho traz uma abordagem sobre os principais desafios presentes nas rotinas dos profissionais e de seus gestores, e estuda como o *framework* ágil *Scrum* pode ser aplicado em um ambiente de equipes distribuídas, seguindo como embasamento literaturas correlatas.

1.2. Problema de pesquisa

Diante da pandemia do COVID-19, que teve início no final do ano de 2019, no Brasil nos primeiros meses do ano de 2020, muitas empresas que não adotavam o modelo distribuído, passaram a utilizá-lo devido as situações impostas pelas medidas sanitárias cabíveis no momento (Brant, 2020). Assim, para essas empresas, algumas questões surgiram como por exemplo, a produtividade e entrega dos times não estando no escritório e, como adaptar as ferramentas, ritos de gestão e *frameworks* utilizados dentro do ambiente presencial no ambiente distribuído, de forma com que o trabalho acontecesse de forma sustentável.

Atualmente, existem materiais publicados referente a área de estudos desta dissertação. Porém, há uma escassez de documentações acerca dos inibidores e boas práticas da implementação desta metodologia dentro deste contexto distribuído, que está cada vez mais difundido (Brant, Raquel, and Helena Cardoso Mourão, 2020). Portanto, este estudo tem como foco apresentar uma revisão sistemática de literatura acerca da aplicação do uso do *Scrum* em ambientes distribuídos.

Apesar de poucas experiências reportadas quanto ao uso do *Scrum* neste cenário, o número de empresas que se inclinam a iniciar testes de uso, e até mesmo que já iniciaram sua utilização tem crescido ano após ano (Yap, 2005).

1.3. Objetivos

Este trabalho tem como objetivo geral desenvolver uma revisão sistemática de literatura referente a aplicação do *framework* ágil *Scrum* em equipes distribuídas.

Para atingir o objetivo geral, são necessárias as seguintes etapas no estudo consideradas como objetivos específicos:

- Apresentar a ótica de diferentes autores acerca deste tema e do estado da arte sobre a utilização do referido *framework* no contexto de desenvolvimento remoto de *software;*
- Entender as estruturas e adaptações necessárias para a boa aplicação e, consequentemente, coleta de bons resultados.

1.4. Estruturação do trabalho

Esta dissertação está estruturada da seguinte forma:

Este capítulo introdutório buscou apresentar a justificativa, motivação, objetivos e uma breve descrição sobre a metodologia que será utilizada.

No capítulo 2 será apresentado os principais conceitos dentro da temática estudada através da revisão sistemática de literatura, vendo o capítulo 3, apresentar os trabalhos relacionados ao tema de estudo.

No capítulo 4, está o cerne do trabalho, com a revisão sistemática de literatura, metodologia utilizada para elaboração da presente dissertação.

Seguindo-se para o capítulo 5, é apresentado o planejamento para execução da pesquisa, seguido do capítulo 6 com a condução da mesma.

No capítulo 7 é possível verificar os principais resultados do estudo, sendo estes discutidos no capítulo 8, que apresenta o desdobramento da temática, segundo os objetivos de pesquisa e conforme a trajetória do capítulo 4.

Por fim, a conclusão é apresentada no capítulo 9, consolidando o trabalho.

Ainda, na sequência, podem ser consultadas as referências utilizadas para elaboração do presente trabalho.

2. ENQUADRAMENTO TEÓRICO

Neste capítulo serão abordadas pesquisas anteriores, sendo referenciados seus autores, que contribuíram com o embasamento para essa dissertação, bem como serão apresentados pontos importantes da temática e áreas relacionadas ao desenvolvimento de software e metodologias de gerenciamento de projetos.

2.1. Processo de desenvolvimento de software

Os processos de desenvolvimento de *software* podem ser definidos como um conjunto de atividades inter-relacionadas que culminam em um produto de *software*. Lima (2012), elenca os processos pertinentes a estas atividades, são eles: especificação, projeto e implementação, validação e evolução.

De acordo com Sommerville (2011) estes processos são passíveis de viés a depender do contexto de onde são aplicados, ou seja, podem ser adaptados de acordo com as necessidades do negócio.

No tocante à engenharia de *software*, existem dois modelos de desenvolvimento já consagrados: Modelo Prescritivo e o Método ágil. Este último descrito por Pressman (2006), no Manifesto para o Desenvolvimento Ágil de *Software*.

2.1.1. Levantamento de Requisitos

A especificação, também chamada de Levantamento de Requisitos (Rezende, 2013), pode ser definida como a fase em que os atores participantes do processo de Engenharia de *Software* buscam compreender os problemas referentes ao *software* em pauta, bem como elencar e priorizar as necessidades dos futuros usuários deste.

Sommervile (2011) pontua que os requisitos descrevem funcionalidades e restrições de que o sistema disporá. Também é nesta etapa onde são definidos os

atributos, características e os problemas a serem resolvidos pelo sistema que está sendo esboçado.

De acordo com Sommervile (2007), os requisitos podem ser de usuários ou sistemas.

- **Requisitos de Usuários**: São as definições em linguagem humana (alto nível), especificando verbalmente apenas os objetivos finais de cada uma das *features* disponíveis no sistema. Segundo Bezerra (2015), uma das técnicas que podem ser aplicadas neste contexto é a Análise Textual de Abbott, que identifica, através de uma descrição proveniente do ponto de vista dos usuários, quais serão as ações (métodos), características (atributos) e entidades (classes) necessárias ao desenvolvimento.
  É importante ressaltar que, neste contexto, devem ser evitadas expressões técnicas ou qualquer tipo de linguagem que não seja de fácil compreensão.

- **Requisitos de Sistema**: Estes tipos de requisitos, ainda segundo Sommervile (2007), são aqueles pertinentes aos detalhes de implementação do sistema. Estes são abstraídos pelos engenheiros de software a partir do levantamento dos requisitos de usuários. Um cenário ideal, no tocante às equipes, é que estas sejam compostas pelos engenheiros responsáveis pelo desenvolvimento, assim como pelos usuários que farão uso deste (Pressman, 2011). O autor defende que uma equipe balanceada, fomenta a produção de um produto de *software* mais assertivo, ao passo que, constantemente, todos os atores estão alinhados para com os objetivos do projeto.

2.1.2. Projeto e Implementação

Na fase do projeto, são definidos e considerados os aspectos de funcionamento interno do sistema, a fim de certificar que os requisitos dos usuários sejam plenamente atendidos. Dentre os pontos considerados nesta etapa, cita-se:

Arquitetura do sistema, linguagem de programação a ser utilizada, SGBD (Sistema Gerenciador de Banco de Dados), identidade visual etc. (Girardi, 2004).

Ainda segundo o autor, nessa fase é produzido um projeto arquitetural e um projeto detalhado do *software*. Sendo:

- **Projeto arquitetural:** Engloba os diferentes componentes da arquitetura e como estes se orquestram para atingir os objetivos que embasam e justificam o desenvolvimento do sistema.
- **Projeto detalhado:** Descreve o comportamento e as características de dados de cada um dos componentes da arquitetura.

Nesta etapa também é produzido o código fonte do projeto, baseando-se nos produtos gerados nas etapas anteriores e obedecendo às regras de negócio referentes aos problemas a serem resolvidos. Neste contexto, Von Mayrhauser (1990), explana que enquanto as fases anteriores concentram-se no “o quê”, esta fase descreve “como” este artefato de software será implementado.

2.1.3. Validação e evolução

Com o intuito de se confirmar que o projeto atendeu a todos os objetivos para os quais foi proposto, são executadas baterias de testes. Esta prática, caracterizada como teste de *software* por Costa (2001), visa garantir que o *software* está em conformidade com o proposto nas etapas anteriores e que o mesmo esteja livre de erros de implementação. Von Mayrhauser (1990) elenca alguns tipos de estratégias de teste a serem executadas. São elas: Teste de unidade, teste de integração, teste de sistema, teste de instalação e teste de aceitação.

Quanto aos tipos de testes mais utilizados, pode-se citar os *alpha-tests* que são aqueles executados em ambientes controlados, com um número restrito de usuários e com a presença dos engenheiros de software que participaram dos processos da implementação; e os *beta-tests*, caracterizados por serem

executados por um número maior de usuários, onde eles se comprometem a fornecer *feedbacks* quanto a sugestões, melhorias e erros encontrados.

2.2. Métodos ágeis

As metodologias ágeis vieram em resposta ao modelo de desenvolvimento tradicional. Mesmo com as evoluções tecnológicas e com as ferramentas de desenvolvimento, em alguns cenários ainda são complexos a previsão de escopo, os prazos e os custos acerca do desenvolvimento do produto. Assim, viu-se a necessidade de um novo modelo de trabalho que trouxesse uma melhor resposta a essas mudanças sendo mais adaptável (dos Santos Soares, 2004).

2.2.1. Lean manufacturing

O *Lean manufacturing* conhecido também como Sistema Toyota de Produção foi um grande norteador para a base da agilidade, criado após o fim da segunda guerra mundial, no Japão, na fábrica de automóveis da Toyota (Poppendieck & Poppendieck, 2003). Na época, os recursos eram escassos no país e as empresas tinham dificuldade em adotar um meio de produção em massa e elevar a produtividade.

Esta abordagem foca na redução de desperdícios e melhoria contínua, gerando entregas para o cliente, tais como, o que ele quer, onde e quando ele quiser, sem desperdícios. No trabalho do referido autor, são apresentados sete princípios do *lean* adaptados para o desenvolvimento de *software*.

1- Amplificar o aprendizado: Segundo Bassi (2008), as lições durante o processo de desenvolvimento devem servir como fonte de aprendizado e conhecimento e incorporados ao processo, trazendo amadurecimento para a equipe e para o processo.

2- Tornar a equipe responsável: A equipe do projeto precisa estar comprometida com o resultado final do trabalho que está sendo desenvolvido. Para isso, envolver essas pessoas no processo de tomada de decisão acerca dos diversos detalhes que compõem o processo de desenvolvimento é fundamental para garantir o sucesso da implementação do artefato de *software*.

3- Adiar comprometimentos e manter a flexibilidade: Adiar a tomada de decisão possibilita uma maior bagagem de experiência adquirida durante o processo. Para isso, é importante que a equipe seja capaz de absorver mudanças, focando no objetivo final e não no planejamento em si. Dessa forma, os planejamentos são vistos como estratégia para o objetivo final e não como compromissos.

4- Entregar rápido: Bassi (2008) apresenta que ciclos interativos com o cliente refinando suas necessidades e vendo os resultados em curto prazo trazem mais valor para o cliente, experiência e embasamento para a equipe tomar decisões. A entrega rápida de software além de anteceder erros, atende as necessidades atuais do cliente e adianta a tomada de decisão (Franco, 2007).

5- Construir integridade: De acordo com Bassi (2008), a equipe de desenvolvimento deve trabalhar em um ambiente em que os assegurem que estão trabalhando em um produto de qualidade. Para isso, é importante a utilização de arquitetura adequada bem como a utilização de testes automatizados e de alta cobertura prevenindo erros futuros e preservando a flexibilidade para mudanças. Assim, ao contrário de alocar recursos para encontrar e corrigir erros e defeitos, o tempo deve ser investido em preveni-los através dos testes. Franco (2007) apresenta que o *software* precisa manter sua integridade e funcionamento ao longo do tempo, e para isso deve possuir uma arquitetura coerente, boa usabilidade e atendimento ao objetivo para o qual foi proposto.

6- Visualizar o todo: Franco (2007), defende que é preciso conhecimento profundo em diversas áreas para que o software seja íntegro. Bassi

(2008) coloca que em grandes *softwares*, as integrações devem se apresentar de forma uniforme e com bons resultados quando trabalhando de forma conjunta. A visão de usuário traz a perspectiva de visualização do todo em alto nível e de acordo com o *Lean manufacturing*, são recomendadas métricas que representam indicativos de evolução e que englobam satisfação do cliente e qualidade.

7- Eliminar desperdício: Dentro do processo de desenvolvimento de *software* existem diversos fatores que podem ser fontes de desperdício, como tempo, recursos, dinheiro, esforço e espaço. Além disso, cada etapa do desenvolvimento precisa contribuir para que o produto seja construído com tempo e qualidades desejáveis.

Bassi (2008) apresenta os cenários a seguir onde os desperdícios são evidenciados:

- **Funcionalidades incompletas:** geram esforço e não trazem valor ao desenvolvimento, uma vez que podem vir a se tornarem obsoletas no decorrer do desenvolvimento.

- **Excesso de processos e processos complexos:** demandam recursos e fazem com que as tarefas demorem mais a serem concluídas, gerando assim desperdício de tempo. A criação de documentações torna o processo inflado e consome tempo, além disso, os documentos podem ficar desatualizados, não serem lidos e tornarem a comunicação mais lenta e menos fluida.

- **Antecipar funcionalidades:** gera desperdícios por aumentar a complexidade e criar código com mais trabalho de testes e integrações. Atrelado a isso, a troca de tarefas excessivas e mudança de contexto reduz a produtividade.

- **Espera por requisitos**, **testes ou aprovações:** trazem ociosidade para o desenvolvimento.

- **Defeitos:** geram custos conforme o projeto evolui e a complexidade do código aumenta, dificultando a resolução dos mesmos

Assim, a comunicação e o gerenciamento devem ser simples e objetivos, envolvendo o menor número de pessoas possíveis e com processos com menos etapas para uma conclusão mais rápida dos ciclos de desenvolvimento.

As metodologias tradicionais de gerenciamento de projetos, também classificadas por Royce (1970) como pesadas ou orientadas à documentação, surgiram em um contexto bastante defasado se comparadas com as atuais tecnologias. Em meados dos anos 70, todo o desenvolvimento de *software* era baseado apenas em *mainframes* e “terminais burros”, como o próprio autor classifica.

Por apresentar custos elevadíssimos, restrições de acesso aos computadores e falta de ferramentas que apoiassem o desenvolvimento, todo o *software* era planejado e documentado antes mesmo de ser desenvolvido, e a principal metodologia de gerenciamento utilizada era o modelo Clássico.

Este modelo foi o primeiro protocolo de gestão de *software* publicado e, até o fim da década de 90, ainda era bastante difundido entre as empresas mais tradicionalistas (Pressman, 2001). Esta metodologia consiste em etapas, onde cada avanço está atrelado à elaboração de uma documentação que deve ser debatida, se necessário reescrita e, posteriormente aprovada, a fim de se avançar para a próxima etapa.

Apesar de amplamente utilizada até o fim da referida década, alguns pesquisadores, como Brooks & Bullet (1987) e Gilb & Finzi (1988) afirmam que é impossível descrever com exatidão a implementação de um *software* antes que esta seja iniciada.

Na figura 2.1 a seguir, pode-se analisar os dados do *Chaos Report* (2015), divulgado pelo Standish Group, contendo métricas de entregas de mais de 25.000 projetos desde 2011. Observa-se que, em 2015, 36% dos projetos catalogados foram entregues respeitando os prazos e orçamentos iniciais; 19% foram cancelados durante o desenvolvimento e 45% foram concluídos, porém, com graves problemas relacionados a prazos e orçamento e, por vezes, com um escopo menor do que o inicial, ou seja, é inegável que este tipo de metodologia de gestão fez com que muitos possíveis bons projetos nem chegassem a ser, de fato, executados, porém, foi necessário para que novas propostas de gestão pudessem ser estruturadas.

Figura 2.1 - Tradicional Resolution for all projects

TRADITIONAL RESOLUTION FOR ALL PROJECTS

	2011	2012	2013	2014	2015
SUCCESSFUL	39%	37%	41%	36%	36%
CHALLENGED	39%	46%	40%	47%	45%
FAILED	22%	17%	19%	17%	19%

The Traditional resolution of all software projects from FY2011–2015 within the new CHAOS database.

Fonte: Chaos Report (2015)

2.2.2. Manifesto Ágil

No início dos anos 2000, depois de todos os aprendizados extraídos de todos aqueles insucessos na gestão de projetos de desenvolvimentos de *software* nos anos 90, um grupo de 17 especialistas, propuseram o que ficou conhecido como “Manifesto Ágil[1]”, que basicamente, consiste em uma compilação de regras

[1] Manifesto Ágil https://agilemanifesto.org. (13 de março de 2020)

e princípios compartilhados pelas principais metodologias de gestão que já existiam na época, tais como XP, *Scrum* e *Crystal*. Os conceitos-chave computados neste manifesto são, nomeadamente, segundo Prikladnicki, Willi & Milani (2014):

- **Indivíduos e interações** são mais importantes do que processos e ferramentas;
- ***Softwares* que funcionam** são mais importantes do que uma vasta documentação sobre este;
- **Foco na interação com o cliente** é mais importante do que negociação de contratos;
- **Responder às mudanças** é mais importante do que seguir um plano pré-definido.

É importante salientar que este conjunto de definições não exclui nenhum processo, documento, contrato, planos ou ferramentas. Apenas estabelece quais deveriam ser as prioridades a serem seguidas por todos os atores participantes deste mecanismo. Além destas regras, os supracitados pesquisadores redigiram 12 princípios denominados do denominado “Manifesto Ágil” a serem seguidos. São eles:

- As necessidades dos clientes devem, sempre, ser priorizadas com entregas contínuas e com o desenvolvimento do projeto com valor agregado;
- Mudanças nos requisitos são bem aproveitadas, uma vez que podem trazer benefícios para o projeto e para o cliente, fazendo com que o produto final tenha mais competitividade;
- Sempre entregar versões de teste em funcionamento. É mais importante apresentar versões alpha ou beta em funcionamento em menos tempo do que aguardar uma versão finalizada que pode demorar mais para ser apreciada;
- Profissionais de *business*, de tecnologia e de gestão devem trabalhar, diariamente, em conjunto;
- Manter a equipe motivada;

- Estimular a troca de experiências e de conhecimentos através de conversas face a face;
- Garantir o funcionamento do *software*, pois este é a medida de evolução do projeto;
- Manter um ritmo constante de trabalho tende a promover um desenvolvimento sustentável e salutar, tanto para o projeto quanto para os atores envolvidos;
- Excelência técnica em todos os aspectos pertinentes faz com que a agilidade no processo seja aumentada;
- Manter a simplicidade em tudo aquilo que é passível desta, é essencial;
- Equipes auto organizáveis e auto gerenciáveis tendem a propor e executar as melhores arquiteturas de software;
- Periodicamente, a equipe analisa o trabalho e identifica como podem melhorar o processo, sendo mais eficazes e assim, efetuar os ajustes necessários.

Com base nestas regras e recomendações, todas as metodologias ágeis que são utilizadas hoje em dia tendem a possibilitar uma gestão otimizada de suas equipes e projetos, porém, devido a determinadas situações, cada uma delas pode ou não ser a indicada.

2.3. Scrum

Scrum é um *framework* que ajuda pessoas, times e organizações a gerarem valor por meio de soluções adaptativas para problemas complexos.

2.3.1. Teoria do *Scrum*

O *Scrum* é um *framework* que tem como base o empirismo, ou seja, o aprendizado pela prática e o *Lean thinking*, que direciona os processos do *Scrum* rumo a redução de desperdícios e o foco no que é essencial para o desenvolvimento no momento.

Dentro do *framework*, existem quatro eventos formais que auxiliam a adaptação e inspeção no processo: *Sprints*, *Daily*, Reunião de Retrospectiva e *Sprint Planning*. Os mesmos são descritos no item 2.3.4 apresentado na sequência.

Além disso, o método possui três pilares que, atrelados a esses eventos, contribuem para o bom funcionamento e o empirismo do *Scrum*, que são transparência, inspeção e adaptabilidade.

- **Transparência:** No *Scrum* o fluxo de trabalho deve ser visível por todo o time e pelo contratante do serviço. A baixa transparência pode diminuir o valor e aumentar o risco das decisões.
- **Inspeção:** Ao longo do processo de desenvolvimento, deve haver a inspeção para garantir que as atividades estão indo a caminho do objetivo do produto. A inspeção possibilita a adaptação e, caso esta não aconteça, a adaptação perde seu objetivo.
- **Adaptação:** Caso algum processo ou tarefa desvie dos limites aceitáveis ou esteja incondizente com os objetivos, o mesmo deve ser adaptado o mais rápido possível para diminuir os impactos e os riscos.

2.3.2. Valores do *Scrum*

O sucesso do *Scrum* está atrelado aos valores do *framework* e depende da vivência e aplicação dos mesmos no dia a dia, que são: Compromisso, Foco, Abertura, Respeito e Coragem. O time precisa se comprometer com o trabalho e as entregas, com os objetivos e uns com os outros. O trabalho deve ser focado nas atividades da *sprint*, entregando o melhor possível dentro do tempo cabível para a realização das tarefas.

O time é aberto para discutir sobre o trabalho, com respeito e coragem para superar os empecilhos, as oportunidades de melhoria e os objetivos do projeto. A prática desses valores pelo time, faz com que os pilares empíricos do *framework*

sejam aplicados, trazendo transparência, inspeção, adaptação e construção de confiança.

2.3.3. Time *Scrum*

O time é um pequeno grupo multidisciplinar, sem hierarquia. Uma unidade coesa com profissionais focados em um objetivo de cada vez. Fazem parte do time o *Scrum* master, *product owner* e desenvolvedores. Os membros, têm a autonomia para se auto gerenciarem e decidirem quem faz o quê, quando e como. Além de terem todas as habilidades técnicas necessárias para desenvolver e gerar valor na *sprint*.

Segundo o *Scrum Guide*[2], times pequenos se comunicam e integram melhor. Desta forma, no *framework* normalmente o time é composto por até 10 pessoas. Em caso de projetos grandes e com muitas pessoas envolvidas, é indicado a divisão em times menores que compartilham a mesma meta de produto.

Esse papel é responsável por todas as atividades e entregas do produto, desde o relacionamento com os *stakeholders* até as entregas de valor. Eles são estruturados e empoderados para auto gerenciarem seu trabalho, trabalhando com foco nos objetivos da *sprint* e do produto. Todo o time é responsável por uma entrega valiosa.

Existem alguns atores fundamentais na execução desta metodologia, que têm definições e responsabilidades específicas (Hogan, 2006). São eles:

- ***Product Owner***: Este ator é responsável por definir os requisitos.
- ***Scrum Master***: Tem a responsabilidade de garantir o correto andamento do projeto, ratificando-se que as práticas, valores e regras estão sendo

[2] Scrum Guide 2020 - https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-PortugueseBR-2.0.pdf (13 de março de 2020).

respeitadas, e que o projeto está evoluindo de acordo com o que foi preestabelecido pelos clientes. Além disso, este é responsável por, também, remover quaisquer obstáculos que possam vir a impedir o correto andamento do projeto. Neste caso, o ator possui autonomia para interagir tanto com o *Scrum Team* quanto com os clientes.

- ***Scrum Team***: Trata-se da equipe participante do desenvolvimento do projeto. Esta, por sua vez, tem autonomia para estimar o esforço necessário para a confecção de uma tarefa ou funcionalidade, sugerir os obstáculos que precisam ser superados etc.
- ***Client***: Ator responsável por definir as regras de negócio do produto, fornece a lista de funcionalidades necessárias e elabora os requisitos do mesmo.
- ***Manager***: Este interage diretamente com os clientes e é responsável por fazer as devidas leituras nos gráficos desenvolvidos e tomar quaisquer decisões finais necessárias.

2.3.4. Eventos do *Scrum*

Como exposto, dentro do *framework*, existem quatro eventos formais que auxiliam a adaptação e inspeção no processo:

- ***Sprints*:** Tempo predeterminado no qual os trabalhos são divididos. Geralmente, duram entre duas a quatro semanas.
- ***Daily Meeting*:** Também chamadas de *stand-up meetings*, são as reuniões diárias, que duram cerca de 10 minutos e servem para que a equipe faça uma breve interação entre si a fim de relatarem se o projeto está evoluindo de acordo com o esperado.
- **Reunião de Retrospectiva:** Momento em que a equipe se reúne para expor os problemas e desafios enfrentados, a fim de que os próprios membros possam auxiliar na busca pelas melhores maneiras de superar tais empecilhos.

- ***Sprint Planning*:** É a cerimônia simples onde a equipe, junto ao *Scrum master*, decide quais serão os *backlogs* a serem contemplados para a próxima entrega.

2.3.5. Planning Poker

Antes de o desenvolvimento de um *Sprint*, de fato, ser iniciado, é necessário que o tamanho de cada item do *backlog* seja avaliado. Para isto, pode ser utilizado o *planning poker*, que consiste em cada membro da equipe atribuir valores a estes itens. Por sua vez, uma média entre os valores é calculada e este valor é atribuído, individualmente, a cada item e a média destes é atribuída ao *backlog* (Pereira, 2007).

Os valores disponíveis e passíveis de serem atribuídos, devem seguir uma sequência lógica, como por exemplo, a sequência de Fibonacci.

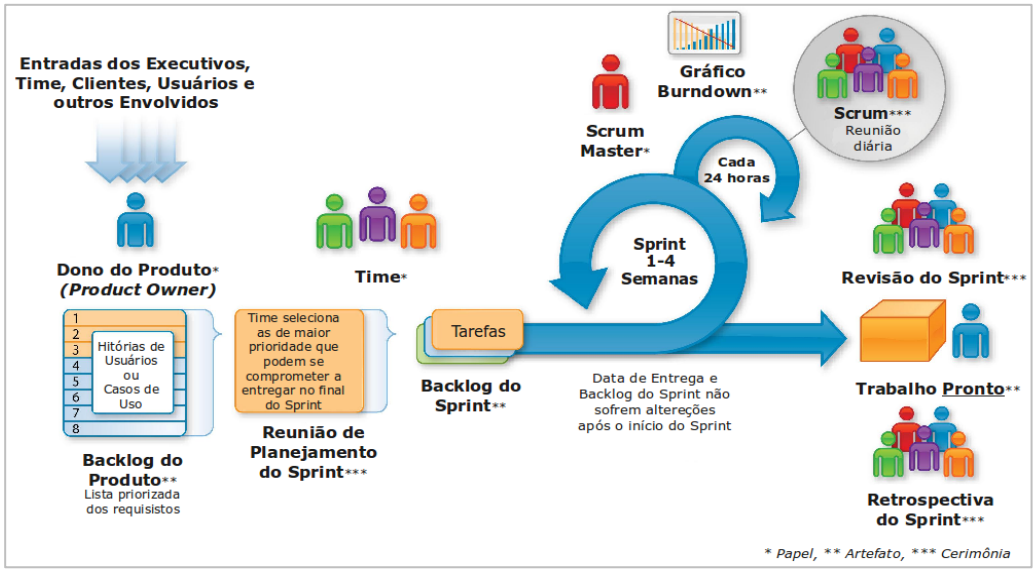
2.3.6. Entregas Contínuas

Dentre as propostas do *Scrum*, estão as entregas contínuas, que segundo Araujo (2005) são definidas como sendo um sistema composto por:

- ***Backlogs***: Requisitos funcionais do projeto.
- ***Sprints***: Tempo predeterminado no qual os trabalhos são divididos. Geralmente, duram entre duas a quatro semanas.
- **Reunião de Retrospectiva**: Conforme apresentado em 2.3.4 é o momento em que a equipe se reúne para expor os problemas e desafios enfrentados a fim de que os próprios membros possam auxiliar na busca pelas melhores maneiras de superar tais empecilhos.
- ***Sprint Planning***: Conforme apresentado em 2.3.4 é a cerimônia simples onde a equipe, junto ao *Scrum master*, decide quais serão os backlogs a serem contemplados para a próxima entrega.

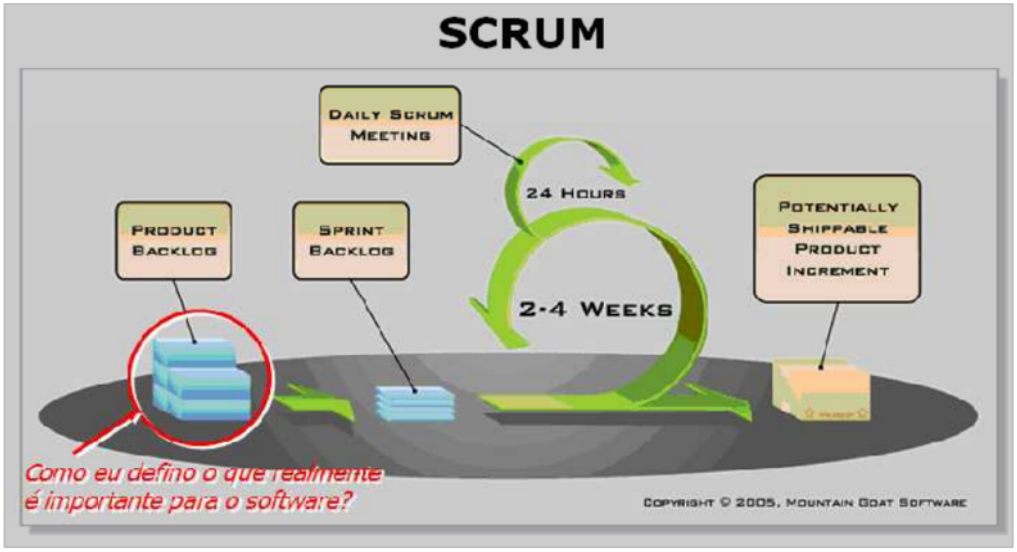
A figura 2.2 a seguir, ilustra os elementos do *Scrum*. A figura 2.3 ilustra o ciclo de entregas contínuas.

Figura 2.2 - Elementos do *Scrum*



Fonte: Oliveira (2016)

Figura 2.3 - Ciclo de entregas contínuas



Fonte: Araujo (2005)

Os referidos autores citam a seguinte lista de atividades mandatórias ao se planejar uma entrega:

- Discussão da estória por detrás do *backlog*;
- Decompor esta estória em tarefas menores;
- Os desenvolvedores devem aceitar a responsabilidade por cada tarefa;
- Os desenvolvedores, após escolherem suas tarefas, devem estimar os prazos e a complexidade para concluir cada uma delas.

Há também a necessidade de se medir o desempenho da equipe. O desempenho será definido tendo como base o histórico de entregas e finalizações de projetos (Fadel, 2010). Estas ações definirão uma média de pontos para o time, podendo alcançar um total de 20 pontos por Sprint. Tal desempenho pode ser afetado por diversos fatores, sejam externos ou internos e, por sua vez, estes devem ser colocados em pauta nas reuniões de retrospectiva e planejamento.

2.3.7. *Scrum* em equipes distribuídas

Os estudos existentes atualmente apresentam predominantemente a aplicação do *Scrum* em equipes locais. Porém, existem casos de sucesso no emprego desta metodologia na gestão remota de equipes e projetos (Paasivaara, 2009), tais como os expostos no Quadro 2.1, a seguir.

Quadro 2.1 - Casos de Uso – *Scrum* em equipes distribuídas

Case	Type of company & Type of development	Project duration (Scrum usage)	Countries involved (participating personnel)	Teams per site
PrintCo	Service company: Development of a new version of printing service software for internal use in new markets.	2 years (5 months)	Onsite: Finland (7) Offsite: Latvia, two sites (2+1) Germany (1)	People from all sites viewed as one single team
PulpCo	Industrial company: Further development and maintenance of an information management tool for internal use.	3 years (14 months)	Onsite: Finland, main site + one subcontractor consultant working close to onsite (9+1) Offsite: Russia, subcontractor (6)	One team onsite, one team offsite
Energy-Software	IT company: Further development and maintenance of a large energy software product that is in use in several companies all over the world.	~10 years (1,5 years)	Onsite: Norway (~20) Offsite: Malaysia (~20)	5-7 teams, often combined across sites, number of teams and persons in each team varies accross iterations

Fonte: Paasivaara (2009)

2.4. Equipes distribuídas

Equipes locais, que também podem ser chamadas de co-localizadas, ou seja, aquelas que compartilham do mesmo ambiente de trabalho. São aquelas que utilizam um espaço comum da empresa para a execução de suas tarefas (Frank & Ribeiro, 2012).

No tocante ao desenvolvimento de software, alguns problemas são comuns tanto às equipes locais quanto para aquelas distribuídas. São eles: prazos e orçamentos, problemas quanto à disponibilidade e versões de sistemas necessários, resistência a mudanças, falhas na comunicação (agravada em equipes distribuídas) e etc., porém, há de se observar os desafios enfrentados exclusivamente por aqueles que compartilham o local de trabalho com os demais membros, que incluem problemas de mobilidade urbana em grandes centros, eventuais desentendimentos com outros membros, problemas de cunho psicológico etc. (Ågerfalk & Fitzgerald, 2006; Jacques, 2007).

O termo teletrabalho, teve origem na palavra *telecommuting*, utilizada pela primeira vez por Jack Nilles, em uma publicação denominada "*The Telecommunications-Transportation Trade Off*", datada de 1976. Este termo, por sua vez, teve origem na expressão *commuting*, utilizado para caracterizar as viagens necessárias ao ir e vir do local de trabalho (Sakuda, 2005). Teletrabalho, portanto, pode ser definido como aceitação do uso de computadores e telecomunicações para mudar a geografia do trabalho. Esta definição condiz com a realidade, uma vez que o indivíduo não precisa estar fisicamente em seu ambiente de trabalho para se fazer presente.

Conforme já explicado, qualquer profissional que executa suas tarefas e rotinas fora do ambiente empresarial, pode ser considerado um teletrabalhador e mesmo assim continua a fazer parte de uma equipe, que compartilha os mesmos objetivos de maneira coordenada (Smith, 2002; Montoya, 2009).

Se por um lado, há um ganho na flexibilidade, na execução das tarefas já que, ao indivíduo, não cabe a necessidade de estar localmente onde a atividade principal é desenvolvida (Smith & Blanck, 2002; Song *et al*, 2007), por outro lado, a falta de interação face a face, traz o risco de haver uma diminuição das capacidades de transmissão de conhecimentos, haja vista que o número médio de pessoas contactadas semanalmente pelo indivíduo neste tipo de equipe é de 4.9, ante o número de 16 contatos, se tratando de membros de equipes co-localizadas. Além disso, o tempo investido na resolução de problemas tende a ser até 2,5 vezes maior em equipes distribuídas quando comparados às equipes tradicionalistas (Herbsleb & Moitra, 2001; Trindade, Moraes & Meira, 2008; Frank & Ribeiro, 2012).

Apesar de não ser novo o modelo de trabalho, durante o ano de 2020 com a pandemia do Coronavírus diversas empresas precisaram se adaptar para o novo contexto. A falta de informação e conhecimento sobre o modelo distribuído de trabalho fez com que muitas empresas passassem por dificuldades no processo. Além do modelo totalmente *home office*, existe o modelo híbrido, onde existe a sede física da empresa e os colaboradores podem trabalhar remotamente. Empresas com sedes em diferentes localidades como, por exemplo, *Spotify*[3] e *Netflix*[4], já utilizam esta modalidade de trabalho.

[3] Serviço de streaming digital. https://support.spotify.com/br/article/what-is-spotify/. (13 de março de 2020)

[4] A Netflix é um serviço de streaming por assinatura que permite assistir a séries e filmes sem comerciais em um aparelho conectado à internet. https://help.netflix.com/pt/node/412. (13 de março de 2020)

3. TRABALHOS RELACIONADOS

Neste capítulo, serão apresentados trabalhos correlatos a esta pesquisa, sobre a aplicação do *Scrum* em equipes distribuídas, segundo a visão de dois estudos de Berczuk (2007) e Sutherland *et al* (2007).

No trabalho desenvolvido por Berczuk (2007) é apresentada a aplicação do *Scrum* no contexto distribuído, passando por todos os ritos e eventos do *Scrum* e os obstáculos de implementação e boas práticas que podem ser aplicadas. O autor apresenta que apesar de equipes distribuídas tornarem o ágil difícil, o maior problema está no fato de a distribuição amplificar problemas já existentes, e não da aplicação do *Scrum* em si. Além disso, a autonomia da gestão com seus times, o trabalho em equipe e a cultura podem tornar o caminho mais fácil.

Berczuk (2007) apresenta que principalmente os papéis de *product owner* e *Scrum Master* precisam entender muito bem os processos do *Scrum*. Uma vez com o processo entendido, seguir a implementação como apresenta o guia do *Scrum*, é uma boa prática ao contrário de começar direto com adaptações. Após o time atingir maior maturidade e conhecimento do processo, começam-se os refinamentos e adaptações necessárias com base no que foi percebido na aplicação. Ter um conhecimento sobre a base do *framework* por parte do time, facilita colaborações e entendimento de papéis.

O *Scrum* e o ágil prezam pelo foco e comprometimento com a entrega dentro do determinado tempo, a *sprint*. Ter um *backlog* bem definido, claro e com tarefas bem descritas pode facilitar o processo. Nesse sentido, é apresentado sobre a importância do entendimento do ritmo de entrega do time, uma vez que o contexto distribuído pode afetar a produtividade. Ter uma boa estimativa, utilizando gráficos de *burndown* para acompanhar o andamento das tarefas, realizar um bom planejamento de *sprint* podem trazer sucesso nesse sentido (Berczuk, 2007).

Além disso, no ambiente co-localizado é comum as equipes possuírem painéis visuais com as histórias de usuário e andamento da *sprint*. No ambiente

distribuído, utilizar ferramentas que cumprem esse papel torna-se fundamental para promover a transparência do andamento do trabalho.

Berczuk (2007), apresenta que o *scrum master* junto do *product owner* têm um papel importante em remover os impedimentos do time, analisarem o uso e efetividade das ferramentas que auxiliam a comunicação, transparência e acompanhamento do trabalho. Por meio dessas, é possível identificar se as *sprints* estão sendo concluídas e rastrear possíveis desvios como muitas solicitações dentro desta. Nesse sentido, a utilização de *sprints* menores, de 1 a 2 semanas, podem auxiliar, principalmente no início, a identificar gargalos e permitir validação e alinhamento mais frequente.

No estudo é apresentado também sobre a importância das boas práticas de engenharia no desenvolvimento, para o bom andamento e produtividade da *sprint*. Entregas bem testadas e com qualidade, evitam interrupções por necessidade de resolução de questões urgentes, permitindo-se o foco no ciclo atual, fomentando o bom andamento deste.

O autor, por fim, apresenta que dentro dos eventos do *Scrum*, a *daily* e revisão de *sprint* podem ser uma questão de atenção dentro da aplicação do time distribuído, principalmente em times com fusos horários diferentes. Torna-se um desafio criar momentos que sejam de compartilhamento sobre o trabalho e não “relatório de tarefas”, com *report* de trabalhos. Além disso, ferramentas estáveis de comunicação e de visualização do andamento do trabalho precisam ser levadas em consideração.

Sutherland *et al* (2007), reforça o exposto no presente trabalho que falta de comunicação, comportamentos, processos, tecnologias conflitantes podem trazer problemas para o desenvolvimento. Nesse sentido, o autor apresenta a aplicação de *Scrum* distribuído e *Scrum* de *Scrums*, como duas alternativas aplicáveis ao contexto.

O primeiro diz respeito a aplicação em times com pessoas distribuídas geograficamente e os desafios presentes nesse cenário, como também apresenta Berczuk (2007). Já o modelo de *Scrum* integrado por times ágeis distribuídos traz a aplicação do *Scrum* de *Scrums* como solução aos problemas expostos.

Sutherland *et al* (2007) expõe que para grandes projetos, a aplicação do *Scrum* de *Scrums* pode ser útil devido a possibilidade da execução independente dos ritos por questões de horário e alinhamentos que podem ser dificultados pelo fator da distribuição. Nesse contexto, existe a responsabilidade dos *Scrum Master* realizar alinhamento entre os times para garantir o andamento do *framework* e a fluidez do processo. Uma boa prática que pode alinhar sobre o processo do Scrum e o alinhamento do *backlog* é ter a função de *Chief Scrum Master*, responsável por executar o alinhamento de *Scrum* de *Scrums* e o *Chief Product Owner* para centralizar o gerenciamento de um único *backlog e* priorização do mesmo.

Para o alinhamento dos times, é recomendado que os eventos do *Scrum* ocorram em horários próximos, levando em conta possíveis barreiras de horário e a formalização escrita dos obstáculos e desafios reportados nas *dailys* para ciência de todo time. Cada equipe local executa os ritos do *Scrum* localmente e fica a cargo do *Scrum Master* realizar o alinhamento entre os times com os demais *Scrum masters*, da mesma forma que um *Chief Product Owner* alinha os requisitos e o *backlog*.

4. METODOLOGIA DE PESQUISA

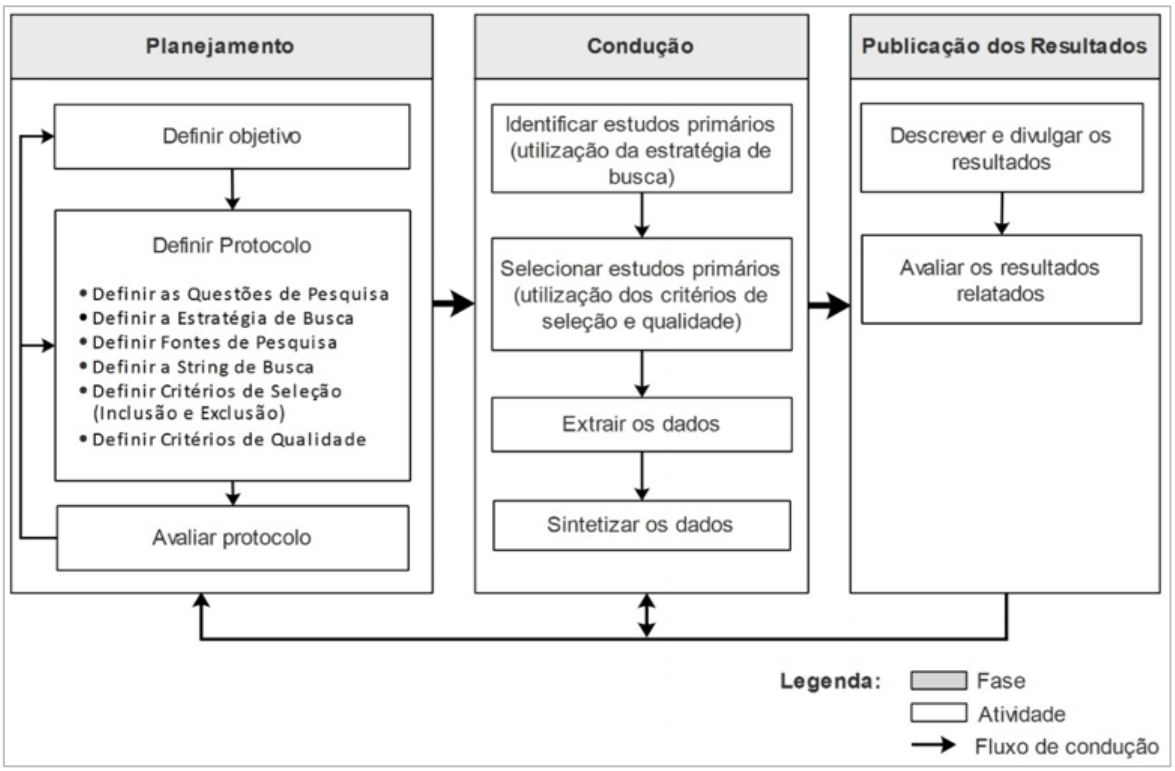
Neste capítulo será apresentada a Revisão Sistemática de Literatura, que consiste na metodologia utilizada para a elaboração deste trabalho.

4.1. Revisão Sistemática de Literatura

Uma Revisão Sistemática de Literatura é composta por um conjunto de técnicas que têm por objetivo identificar, avaliar e interpretar informações relevantes para um determinado problema e/ou questão Kitchenham (2004). Em suma, a revisão permite uma compilação entre as publicações de diversos autores acerca de um determinado tema, a fim de se encontrar pontos de convergência que serão utilizados para responder às questões de pesquisa propostas.

Segundo Kitchenham (2004), esta metodologia respeita as fases de planejamento, condução e análise dos resultados, apresentadas no fluxograma da Figura 4.1, apresentado por Cavalcante & Jorge (2013):

Figura 4.1 - Fases e Atividades de uma revisão sistemática de literatura



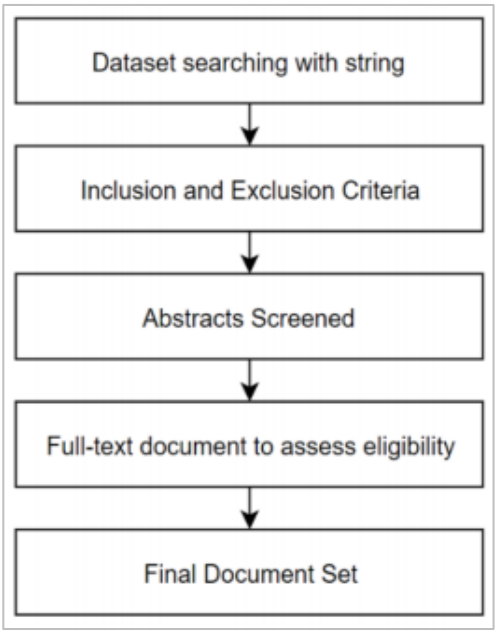
Fonte: Cavalcante & Jorge (2013)

Conforme mencionado as fases da revisão sistemática podem ser divididas em:

- **Planejamento**: Nesta fase objetivos, motivação, questões de pesquisa, *search strings* e Protocolo de Revisão devem ser definidos.
- **Condução:** Nesta fase, um banco de dados de publicações úteis é gerado após a aplicação do Protocolo de Revisão.
- **Análise dos Resultados**: Esta se configura como a fase da revisão. Neste momento, as informações são extraídas do banco de dados de publicações com o intuito de se alcançar os resultados esperados da revisão.

De acordo com Keele (2007), as revisões sistemáticas iniciam com a definição dos protocolos de revisão a serem seguidos (Figura 4.2), abordando a pesquisa a ser seguida e os métodos utilizados, fazendo assim com que a estratégia busque localizar o máximo de literatura relevante possível. A estratégia deve ser bem documentada para que seja possível auxiliar novos pesquisadores em seus estudos. Essa estratégia de busca é definida para que seja refinado o máximo possível de literaturas relevantes.

Figura 4.2 - Protocolo de Revisão



Fonte: Cavalcante & Jorge (2013)

Após a definição do protocolo, é elaborada uma estratégia de busca para que os leitores possam estudar e avaliar o processo de revisão sistêmica. Para isso, é necessário utilizar critérios claros de inclusão e exclusão para avaliar os cenários possíveis.

A estratégia de revisão sistemática de literatura utiliza critérios de qualidade para definir a relevância do estudo para a pesquisa, tendo também critérios de inclusão e exclusão para avaliar os estudos.

Segundo Keele (2007), as revisões sistemáticas são utilizadas para compilar estudos relacionados a um assunto e identificar oportunidades em pesquisas existentes bem como apresentar áreas para novas pesquisas, fornecendo base para posicionamento dos novos estudos. Além disso, é possível utilizar da abordagem para contrapor ideias e criar novas hipóteses sobre determinadas teorias.

As revisões sistemáticas visam trazer uma síntese do trabalho existente de maneira justa, seguindo um critério pré-definido de busca. Assim, as revisões sistemáticas buscam apresentar estudos favoráveis e contrários à sua pesquisa.

A utilização do modelo de forma bem definida auxilia que o estudo seja menos enviesado e tendencioso. Com isso, o método fornece informações sobre as fontes de estudos que, caso sejam consistentes, é possível evidenciar que o estudo é robusto e transferível, caso contrário, se os resultados apresentarem inconsistência, é possível analisar de forma individual as fontes de desvio.

Em contraponto, comparando com as revisões de literaturas tradicionais, as revisões sistemáticas exigem mais esforço. Além disso, pode ocorrer o aumento da meta-análise, sendo possível detectar vieses e efeitos reais no estudo.

5. PLANEJAMENTO

Tendo como base o cenário de grande aumento do trabalho distribuído nos últimos tempos, incluindo times de desenvolvimento, observou-se a necessidade do estudo da aplicação de *frameworks* de gestão de produtos em equipes fisicamente distribuídas e as adaptações necessárias para resultados efetivos.

Assim, essa revisão sistemática de literatura tem como objetivo estudar e compilar referências sobre o trabalho distribuído e a aplicação do *Scrum* nesses cenários, tendo como parâmetro os questionamentos a seguir:

- **Q1:** Quais são os inibidores de aplicar o *Scrum* em equipes distribuídas?
- **Q2**: Quais são boas práticas para o sucesso ao aplicar *Scrum* em equipes distribuídas?

Para a busca da literatura, será utilizado a *search string* a seguir como parâmetro de inclusão para a busca de literaturas relevantes para o tema.

“Scrum” AND ("shared" OR "distributed") AND ("gestor" OR "manager") AND (("equipes" OR "teams") AND ("distribuídas" OR "co-located")) AND ("desafio" OR "challenge") AND ("problemas" OR "problems") AND ("casos" OR "cases") AND "script"

Serão utilizados como fonte de pesquisa os *datasets* IEEE Digital Library, Science Direct, Academia Edu e Google Scholar. A partir do resultado obtido, serão analisadas as literaturas com intuito de filtrar as pesquisas mais relevantes de acordo com o foco do trabalho e as perguntas a serem respondidas.

O Quadro 5.1 a seguir apresenta os critérios de inclusão e exclusão dos artigos encontrados.

Quadro 5.1 - Critérios de Inclusão e Exclusão

Critérios de Inclusão	Critérios de Exclusão
SCRUM	Artigos sem citações
Gestão de equipes de co-localizadas	Arquivos incompletos
Gestão de equipes à distância	Artigos que não contenham relação com as palavras-chave presentes na *search string*
Teletrabalho	
Nômades digitais	
Gestores de projetos	
Empresas de desenvolvimento de software	
Metodologias Ágeis	

Fonte: Autor (2020)

6. CONDUÇÃO

Este capítulo apresenta a segunda fase da revisão sistemática de literatura, onde ocorre a execução e o banco de dados de publicação apurado na seção anterior é refinado.

6.1. Seleção da bibliografia

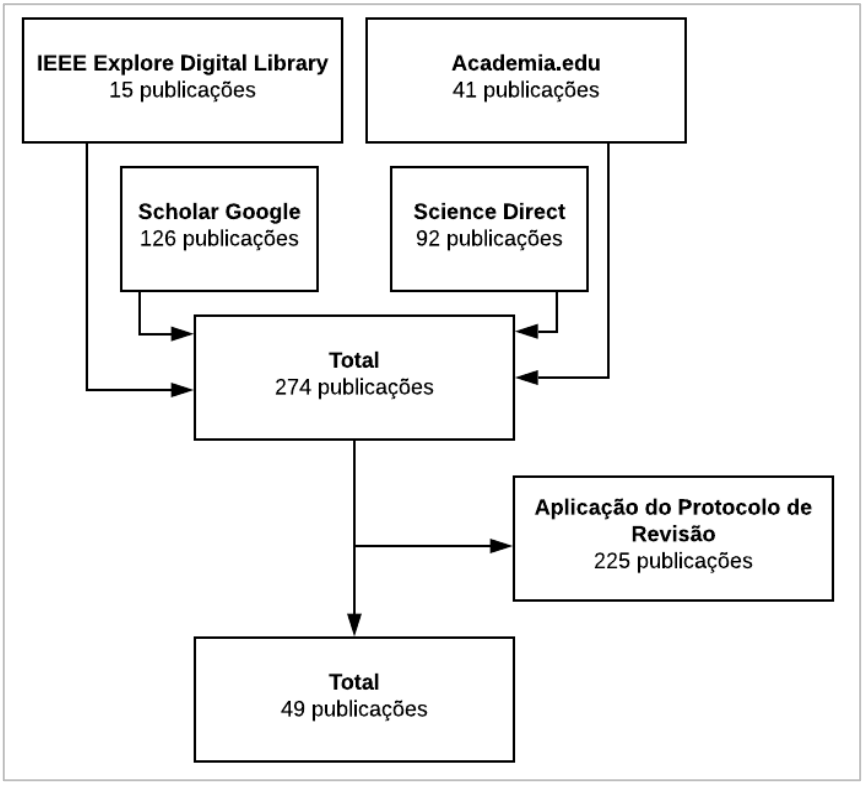
Após aplicar a *search string* nos *datasets* referenciados no protocolo de revisão, 274 publicações foram selecionadas. Dessas, uma segunda triagem foi efetuada a fim de filtrar, dentre estas, aquelas mais relevantes aos objetivos do trabalho, restando 49 publicações úteis. A Figura 6.1 exibe o fluxograma construído para ilustrar o processo de triagem das publicações por *dataset.*

6.2. Análise e extração dos dados

A análise de extração dos dados apresenta a forma sobre como os artigos foram categorizados.

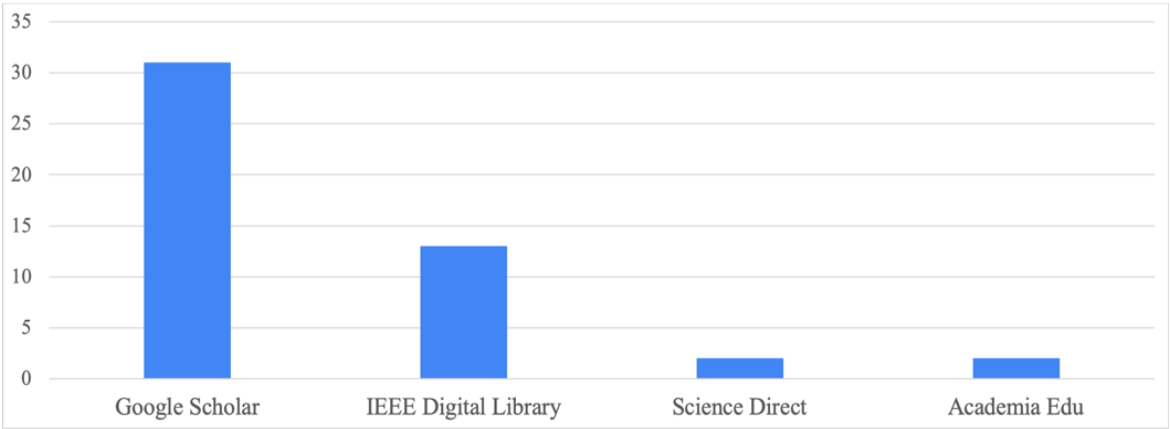
No Gráfico 6.1 é apresentado um comparativo em relação ao número de resultados encontrados pelos principais *datasets* utilizados como base para elaboração deste trabalho, destacando-se o Google Scholar com 31 trabalhos, seguido da IEEE Digital Library com 14 trabalhos e, por fim, Science Direct e Academia Edu, cada um com 2 trabalhos.

Figura 6.1 - Processo de Seleção das Publicações



Fonte: Autor (2020)

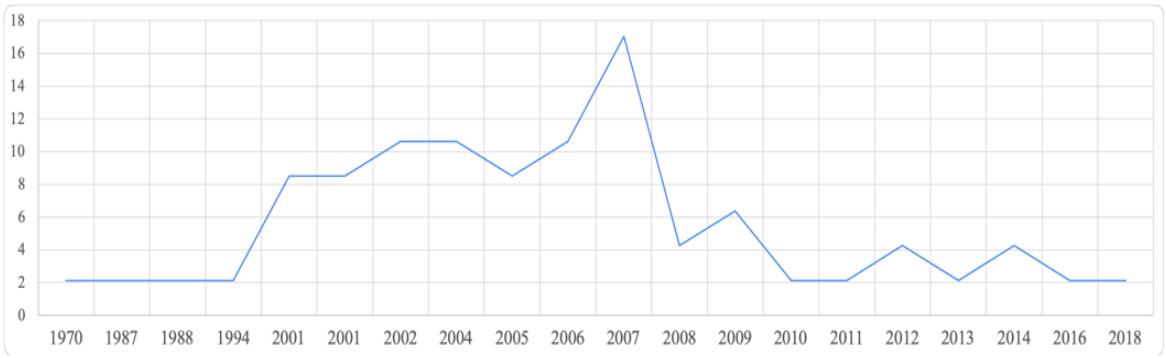
Gráfico 6.1 - Quantitativo de publicações por dataset



Fonte: Autor (2020)

Já o Gráfico 6.2 ilustra o quantitativo de publicações encontradas agrupados por ano de publicação. Foi possível identificar que o ano de 2007 apresenta uma crescente significativa em relação aos anos anteriores e os seguintes, sendo o ano com maior número de publicações relacionadas ao tema.

Gráfico 6.2 - Quantitativo de publicações por ano



Fonte: Autor (2020)

7. RESULTADOS

Este capítulo busca sintetizar os pontos principais do estudo.

7.1. Inibidores e boas práticas para aplicação do *Scrum*

Utilizando do exposto na metodologia, foi possível identificar os artigos apresentados nos Quadros 7.1 e 7.2**,** que correspondem aos parâmetros de pesquisa. Os principais artigos foram separados em dois grupos categorizados por inibidores e boas práticas para a aplicação do *Scrum* em equipes distribuídas.

Quadro 7.1 – Inibidores para a aplicação do Scrum em equipes distribuídas

INIBIDOR	ARTIGOS	REFERÊNCIA
Gestão e liderança	Problems in Agile Global Software Engineering Projects especially within Traditionally Organized Corporations: [An exploratory semi-structured interview study]	Richter, I., Raith, F., & Weber, M. (2016).
	U.S. and Indian managerial boundary spanning behaviors in globally distributed software teams	Padgett (2019).
	Challenges of Best Agile Management Practices	(Kulesovs, I., & Korkkinen, M, 2013).
	From experience: leading dispersed teams. Journal of Product Innovation Management:	Smith, P. G., & Blanck, E. L. (2002).
Cultura e Organização	Obstacles in moving to agile software development methods; at a glance	Gandomani *et al.,* (2013).
	People over process: key people challenges in agile development	Conboy *et al.,* (2011).
	The role of agile principles in success with an distributed scrum team	Berczuk, S. (2007).
Estrutura do *Scrum*	Empirical Analysis of a Distributed Software Development Project	Cichocki, P., & Maccari, A. (2007).
	Agile methods handling offshore software development issues.	Nisar, M. F. & Hameed, T. (2004).
	Distributed extreme programming. Extreme Programming and Flexible Processes in Software Engineering	Kircher, M., Jain, P., Corsaro, A., & Levine, D. (2001).
	Distributed scrum: Agile project management with outsourced development teams.	Sutherland, J., Viktorov, A., Blount, J. & Puntikov, N. (2007).
	Agile methods handling offshore software development issues.	Nisar, M. F. & Hameed, T. (2004).

Fonte: Autor (2020)

Quadro 7.2 – Boas práticas para a aplicação do *Scrum* em equipes distribuídas

BOAS PRÁTICAS	ARTIGOS	REFERÊNCIA
Gestão e liderança	Using simulation for understanding and reproducing distributed software development processes in the cloud	Lunesu, M. I., Münch, J., Marchesi, M. & Kuhrmann, M. (2018)
	Management 3.0: Leading Agile Developers, Developing Agile Leaders	Appelo (2011)
	Scaling Agile	Kniberg *et al.,* (2012)
	Methodological Support for Task Coordination in Global Software Engineering Projects at Product Software Companies	Widiyatmoko, C. B. (2017).
	Agile software development and project portfolio management in dynamic environments: A case study of an integrated solutions provider	Imbrizi, F. G. (2017)
	Global outsourcing of product software development: A knowledge management perspective	Kristjánsson (2009)
	Communication between Developers and Testers in Distributed Continuous Agile Testing	Cruzes, D. S., Moe, N. B., & Dybå, T. (2016).
Cultura e Organização	Activities in Scrum Master Teams: Process Tailoring in Large Enterprise Projects	Bass, J. M. (2014).
	SCRUM – Método Ágil: uma mudança cultural na Gestão de Projetos de Desenvolvimento de Software	Machado, M., & Medina, S. G. (2009).
	Culture and organizations. International Studies of Management & Organization	Hofstede (1980)
	Cultura organizacional na adoção de metodologias ágeis no desenvolvimento de sistemas de informação-rumo a um modelo conceitual à luz de um estudo sistemático	Amaral *et al.,* (2015)
	Aspectos humanos da transição das organizações para o modelo ágil	Pinton, M. & Junior, A. S. T. (2019).
Estrutura do *Scrum*	Applicability of agile methodologies in global software development projects: a Scrum case study	Sampaio, J. P. D. (2011).
	Requirements Engineering for Large-Scale Agile System Development: A Tooling Perspective	Gebremichael, M. G. (2019).
	Why Scrum works: A case study from an agile distributed project in Denmark and India	Pries-Heje, L. & Pries-Heje, J. (2011).
	User experience design and agile development: integration as an on-going achievement in practice	Ferreira, J. (2012).

Fonte: Autor (2020)

Acima, foram compilados e categorizados os inibidores e boas práticas mais relevantes encontradas. O quadro 7.1 apresenta o agrupamento dos estudos que apresentam questões inibidoras para aplicação do *Scrum* no contexto distribuído. Já o quadro 7.2, é composto pelo agrupamento dos artigos que apresentam as boas práticas que podem ser aplicadas para efetividade e resultado do *Scrum* nesse contexto.

Conforme abordado na seção 2.3, o *Scrum* segue um rito que deve ser respeitado, que é resultado dos 12 princípios a serem respeitados pelas metodologias ágeis (seção 2.2.2). O fato de a equipe estar distribuída pode afetar a comunicação e transparência durante a execução do projeto e acarretar a perda de oportunidades de melhoria e evolução do projeto. Além disso, os processos de adaptação organizacional como estrutura e ritos de trabalho precisam ser compreendidos e efetuados e, não somente, levados do presencial para o ambiente distribuído, por se tratar-se de contextos diferentes.

Além das adaptações no *framework Scrum* para adequação ao modelo e a empresa, a atenção para a comunicação, colaboração e confiança precisa ser redobrada. Ao ingressar no trabalho distribuído, algumas práticas de micro gerenciamento das lideranças podem ser inibidoras ao bom funcionamento do trabalho. Atrelado a isso, as más práticas de comunicação que geram ruídos e ineficiência e a falta de colaboração devido ao fato de as pessoas não estarem fisicamente juntas, podem contribuir para uma insatisfatória aplicação da metodologia no ambiente. Ao longo deste capítulo, essas perspectivas serão, novamente, abordadas.

O *Scrum* preza pela transparência, inspeção e adaptação. Esses pilares estão relacionados diretamente à comunicação das pessoas. Em um ambiente físico, torna-se mais fácil ir até a sala de alguém para sanar alguma questão, mas tentar levar essa comunicação presencial para o online pode gerar impactos no processo de comunicação, trazendo ansiedade, falta de produtividade e alta dependência.

Uma prática comum ao *Scrum* e a times ágeis em ambientes co-localizados, é o trabalho executado lado a lado, com reuniões e construções presenciais. Prática esta que pode se tornar complexa quando executada *online* por diversos motivos, como falta de ferramentas e conhecimentos necessários. Apesar disso, são atividades extremamente importantes para a colaboração do time durante o desenvolvimento do projeto utilizando *Scrum* e para tornar transparente o trabalho e a comunicação.

Por fim, no escritório físico é comum as lideranças acompanharem o trabalho dos colaboradores presencialmente e com micro gerenciamento das atividades. No ambiente online isso não é possível e pode gerar problemas de falta de confiança, criação de processos complexos e burocráticos para o acompanhamento do trabalho.

No *Scrum* existem eventos que são impactados diretamente quando não há boa comunicação, confiança e colaboração. Apesar disso, se tratando o *framework* de uma forma de organização do trabalho, o *Scrum* nos permite flexibilidade e adaptação no seu modelo, que se faz necessária para a boa execução do mesmo em um ambiente de equipes distribuídas, conforme apresentado a seguir:

- ***Daily Meeting***: Para responder às três perguntas: "O que foi feito desde o último *Daily Meeting*? Existe algum obstáculo? O que será feito antes do próximo *Daily Meeting*?", as equipes elencadas nas bibliografias faziam uso de vídeo conferências onde cada membro deveria enviar uma lista de questões a serem respondidas durante a reunião.
- ***Sprints***: As empresas em questão fizeram o uso de *sprints* com duração entre duas e quatro semanas, sempre iniciando e terminando no mesmo dia da semana. Como se trata de equipes distribuídas, alguns problemas de datas, como por exemplo, feriados nacionais em determinados países, vez ou outra ocorreram, e apenas houve a necessidade de variar o dia de finalização da entrega, sem nenhum tipo de prejuízo à evolução do projeto.

- ***Retrospective Meetings***: Em equipes co-localizadas, esta reunião ocorre no fim do sprint, porém, neste cenário remoto, ambas as companhias utilizam uma abordagem denominada *sprint* demos, que consiste em efetuar entregas menores, ou sub entregas, ao longo do *sprint*. Além disso, toda a equipe participava dessas reuniões. Isso fez com que o todo o time passasse a ser um tipo de protagonista no processo, trazendo uma sensação de comprometimento e valor para cada membro.
- ***Backlogs***: Neste quesito, foram introduzidos o uso de *softwares* de gestão tal como o Jira[5]. A responsabilidade por manter os dados sempre atualizados ficou por conta dos respectivos *product owners*. Outra determinação, foi a de disponibilizar uma versão de leitura dos backlogs para todas as equipes.

Olhando para o *framework* em si, após estudos (Simons, 2002; Nisar & Hameed, 2004), pode-se afirmar que é possível aplicar o *Scrum* com sucesso na gestão de equipes e projetos distribuídos, havendo alguns precedentes documentados, como na publicação dos autores Maria Paasivaara, Sandra Durasiewicz e Casper Lassenius, denominada "*Using Scrum in Distributed Agile Development: A Multiple Case Study*".

Como já apresentado ao longo deste trabalho, a utilização desta metodologia provoca um maior comprometimento entre os membros, faz com que a confiança do time em suas próprias capacidades sejam estimulados e permite um maior engajamento deste na resolução de problemas. Indiferentemente do tipo de equipe pelo qual o projeto está sendo desenvolvido, suas minúcias serão as mesmas, ou seja, podem ocorrer alterações de escopo, de requisitos etc. O bom andamento está diretamente atrelado às capacidades de gerenciamento da equipe, processos e ferramentas.

[5] Jira é um software comercial desenvolvido pela empresa Australiana Atlassian. https://www.atlassian.com/br/software/jira/guides/use-cases/what-is-jira-used-for. [13 de março de 2020]

Se tratando de equipes distribuídas, além desses desafios existem outros, como por exemplo, problemas na comunicação, indisponibilidade de membros, dificuldades em transferir conhecimentos, dificuldades em compartilhar opiniões, diferenças culturais etc. (Kircher *et al*, 2001; Simons, 2002; Poole, 2004; Berczuk, 2007; Sutherland *et al*, 2007). Para a solução dos desafios apresentados, o papel de uma liderança focada na gestão das pessoas torna-se fundamental.

O papel de *Scrum Master* torna-se imprescindível na remoção de obstáculos por parte dos colaboradores, sendo, o mesmo, responsável por encontrar soluções, pessoas ou processos que possam ajudar a sanar tais obstáculos que, geralmente, são identificados por meio das *daily* e *retrospective meetings*.

O *Product Owner* precisa ser um líder facilitador para conduzir as reuniões e eventos de forma que todos consigam se comunicar, com as melhores ferramentas possíveis de forma assíncrona, por meio de ferramentas para gestão de comunicação e *backlog*, e de maneira síncrona por meio de ferramentas de videoconferências. Além disso, conhecimento de ferramentas e dinâmicas de facilitação para momentos de colaboração como reuniões de *review*.

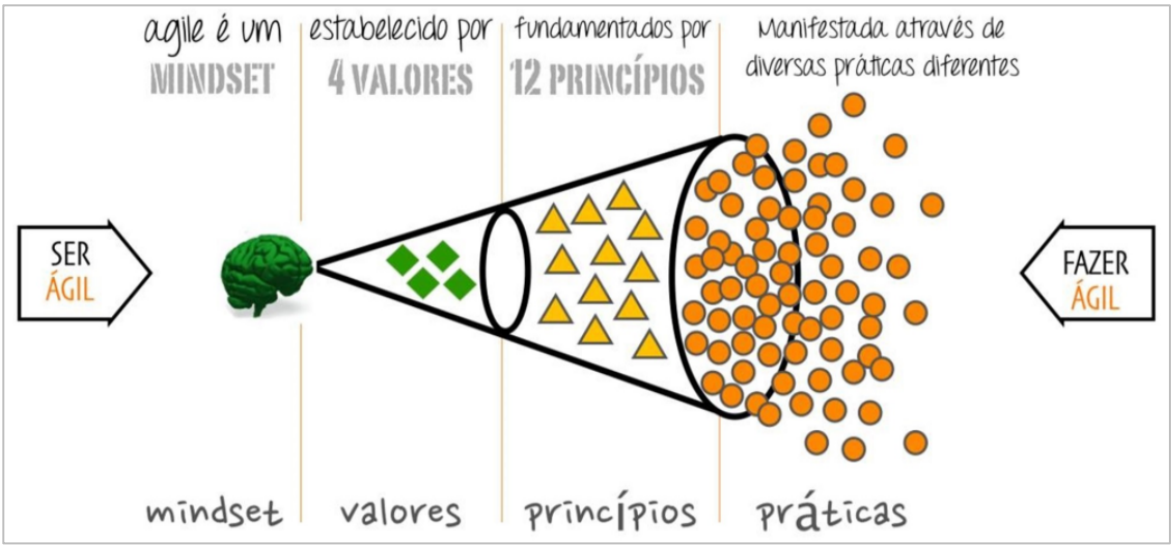
Em suma, estes profissionais têm a missão de, e não somente, conciliar agendas de vídeo conferências entre os times distribuídos, mediar possíveis conflitos oriundos, diferenças culturais, estabelecer regras na comunicação entre as equipes, manter um banco de dados sobre *lessons learned* e tudo o que mais for necessário para manter a boa saúde do time, do projeto e da aplicação da metodologia (Pries-Heje & Pries-Heje, 2011).

Tendo em vista os inibidores para a aplicação do *Scrum* necessário para a utilização nesse contexto e analisando sua implementação e boas práticas para o funcionamento, foi necessária uma visão sistêmica em relação ao processo e todos as frentes que impactam no trabalho, pessoas, processos, ferramentas e até mesmo a cultura organizacional.

O *Scrum* tem como base o manifesto ágil, seus pilares, e o foco nas pessoas. Nesse sentido, as estruturas organizacionais e dos times precisam estar preparadas e baseadas na mentalidade ágil. Segundo Howard (2010), a ausência de estruturas ou estabilidade pode levar ao caos, mas o excesso delas pode levar à rigidez. E em um contexto de trabalho distribuído, gestores de times que buscam um trabalho de controle, poderão prejudicar o desenvolvimento da equipe, dos projetos e a ineficiência do *Scrum*.

O resultado dos métodos ágeis vem muito antes da aplicação dos frameworks em si. A utilização do *Scrum* sem o entendimento de contexto, estado da empresa, pode criar mais problemas ao invés de solucioná-los. Na figura 7.1, é apresentada a evolução do ágil desde a mentalidade até a prática no dia a dia.

Figura 7.1 - Ser e fazer ágil



Fonte: Adaptworks[6]

Antes de tudo, é importante entender o pensamento ágil, o contexto em que surgiu e começar a visualizar a aplicação nos processos. Entender os valores que a agilidade prega e como elas se relacionam com os valores pessoais e os

[6] Agile PMO e a transformação organizacional pela Agilidade - https://projetoseti.com.br/agile-pmo-e-a-transformacao-organizacional-pela-agilidade/ [5 de fevereiro de 2020].

princípios, vai tornando cada vez mais tangível as ações que serão feitas no dia a dia, chegando enfim nas ações ágeis.

7.2. Eficiência do modelo *Scrum*

Nesse sentido, as práticas para a boa aplicação e eficiência do modelo *Scrum* de trabalho pode ser definida em três pilares: Pessoas, Gestão e Organização (Pinton, 2019).

7.2.1. Pessoas

Um dos pilares essenciais para o bom funcionamento de qualquer método baseado na agilidade, é o foco nas pessoas. Por isso, a adoção de novos modelos de trabalho precisa ser bem comunicada e alinhada com toda a equipe envolvida, para que não haja resistência e maiores problemas (Pinton, 2019). Assim, é importante que a gestão comunique ao time os objetivos e benefícios da mudança e da transição e que a equipe seja ouvida e participe do processo de transição (Conboy *et al*., 2011).

Diferente dos métodos tradicionais, no modelo ágil é preferível o trabalho com equipes menores e multidisciplinares. Em casos de grandes projetos, o mesmo pode ser dividido em partes e novas equipes são criadas e divididas entre as frentes do projeto, cada time com seu objetivo específico, mas relacionado com o objetivo final do projeto (Gandomani, *et al*., 2013).

No ambiente distribuído, no início e nos primeiros contatos com a equipe, é importante um trabalho focado em *team building,* para conexão e vulnerabilidade das pessoas. Assim, um ambiente de confiança vai sendo criado e proporcionará melhores resultados e adequação ao modelo de trabalho ágil.

Com equipes focadas em um mesmo objetivo, o trabalho e a tomada de decisão dentro do projeto ficam mais descentralizados, assim, a tomada de decisão fica por conta de toda a equipe do projeto, onde todos participam e têm voz, e não somente nas o gestor. Ele toma-se uma figura de facilitador, diferente do modelo tradicional onde toda decisão fica a cargo do mesmo (Lee *et al*., 2016; Gandomani *et al.,* 2016).

Nesse sentido, o alinhamento de papéis e responsabilidades dentro do projeto é fundamental, além de ajustar o poder de influência dentro dele. O *product owner* dentro do *Scrum* é um líder que exerce liderança sobre influência, ou seja, não exerce autoridade ou maior poder sobre o projeto ou time. No *Scrum*, todos os papéis são importantes e tem como objetivo comum o sucesso do trabalho.

No contexto do *Scrum*, o cliente é visto como membro da equipe, diferente do modelo tradicional. Os clientes têm voz e participam das tomadas de decisão junto ao time, fornecendo insumos sobre o desenvolvimento do projeto. Ignorar a participação e o *feedback* do cliente nas tomadas de decisão durante o processo de desenvolvimento pode gerar perda de oportunidades de melhoria, retrabalho e a entrega de uma solução de pouco valor.

7.2.2. Gestão

Um ponto crucial para o funcionamento do *Scrum* e métodos ágeis em ambientes remotos é a adoção plena dos fundamentos, cultura e ritos de gestão em todos os níveis da organização e principalmente nas lideranças, que deve apresentar total apoio e aceitação do modelo ágil para que isso não torne uma barreira de implementação. O entendimento dos motivos por trás da implementação da cultura ágil é fundamental (Lee *et al*., 2016).

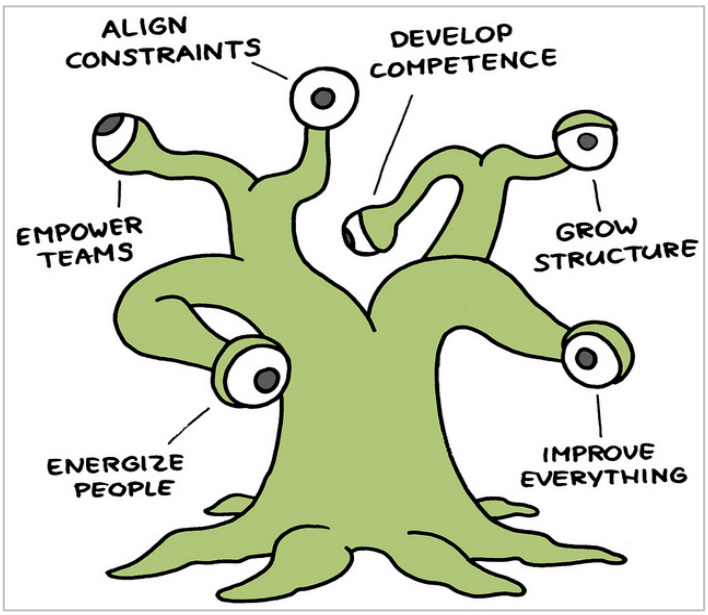
A gestão baseada no ágil traz um fator importante comparado a gestão tradicional, onde no ágil a gestão se torna descentralizada, sem micro gerenciamento é baseada na confiança (Nerur *et al*., 2005; Lee *et al*., 2016). Um

fator de alinhamento importante sobre o micro gerenciamento é que no *Scrum*, a alta gestão da empresa não tem o poder de controlar totalmente do escopo, prazo e financeiro do projeto. No ágil este acompanhamento se torna mais voltado para qualidade e valor para o negócio e o cliente do que prazos e planejamentos cumpridos. O acompanhamento leva mais em conta o *follow-up* das entregas, indicadores do projeto e o status e progresso das *sprints*.

O modelo ágil se baseia no conceito de *Management* 3.0. Diferente dos modelos 1.0 e 2.0 que tinham um grande foco no comando e controle e no gerenciamento dos processos, a gestão 3.0 entende as pessoas como o bem mais importante da empresa e que as pessoas são mutáveis. Assim, torna-se necessário um trabalho focado nos times (Kulesovs, I., & Korkkinen, M, 2013).

Appelo (2011), apresenta os principais focos da gestão 3.0: Energizar pessoas, Empoderar times, Alinhamento constante, Desenvolvimento de competências, Estrutura de crescimento e Melhoria contínua (Figura 7.2).

Figura 7.2 - Martie, the Management 3.0 model



Fonte: Appelo (2011)

Energizar pessoas: De acordo com o autor, as pessoas são as partes mais importantes de uma organização e que os gerentes devem fazer tudo o que puderem para manter as pessoas ativas, criativas e motivadas. Os energizantes podem ser definidos em conhecimento, criatividade, motivação, diversidade e personalidade. Ainda, é apresentado outros cinco que podem auxiliar na construção de um ambiente energizador como segurança, descontração, variação de rotina, visibilidade e desafio constante.

Capacitar as pessoas: Quando se fala de equipes multidisciplinares e auto organizadas é apresentado que isso exige capacitação, autorização e confiança das lideranças. Assim, além de entregar responsabilidades, é importante oferecer um espaço de suporte para assumir riscos pessoais, profissionais e de crescimento cultural.

Alinhamento constante: Devido a auto organização, é importante estabelecer objetivos, metas e propósitos claros para as pessoas e protegê-las devido às incertezas que o time pode estar cercado. Sem esse alinhamento, o direcionamento estratégico não acontece, vários focos e objetivos são criados e podem levar a um ambiente caótico.

Desenvolvimento de competências: Membros incapazes não conseguirão atingir seus objetivos. Sendo assim, é papel do líder capacitar seu time e fornecer informações e conhecimentos necessários para a realização do seu trabalho. A disciplina atrelada a outras habilidades gera competência.

Estrutura de crescimento: Segundo o autor, é importante considerar as estruturas que melhoram a comunicação. Ele apresenta que comunicação é igual a multiplicação de informações, relacionamento e *feedbacks*. Algumas características e atitudes tendem a esta zona de crescimento, tais como conectar, filtrar, ter empatia, ter compreensão, aperfeiçoamento das técnicas de desenvolvimento, gerenciamento, transmissão, influência e conversação.

Melhoria contínua: Pessoas e organizações precisam melhorar continuamente visando adiar falhas pelo maior tempo possível. Uma organização é formada por pessoas, sendo assim, um organismo vivo. Isso leva a mudanças e essas mudanças precisam buscar a melhoria contínua do trabalho. É preciso entender o “efeito borboleta”, onde qualquer mudança pode trazer uma reação no ambiente, sendo positiva ou não, mas que isso não seja causa de aversão à mudança, mas sim de combustível para que elas aconteçam. O processo de melhoria contínua está relacionado à evolução das pessoas, dos processos e até mesmo das técnicas de desenvolvimento.

7.2.3. Organização

Aqui será abordada a mudança estrutural e até mesmo hierárquica que pode acontecer para a transição para o contexto de aplicação plena e efetiva do *Scrum*.

Essa transição pode iniciar com treinamentos e capacitação de toda a equipe sobre a mentalidade e cultura ágil, abordando habilidades comportamentais e técnicas necessárias para a sua plena aplicação (Conboy *et al.,* 2011). Boas práticas nesse processo são a ajuda de *agile coach* e pessoas que trarão um olhar estratégico para o momento de transição, trazendo *feedbacks* sem vieses para o processo.

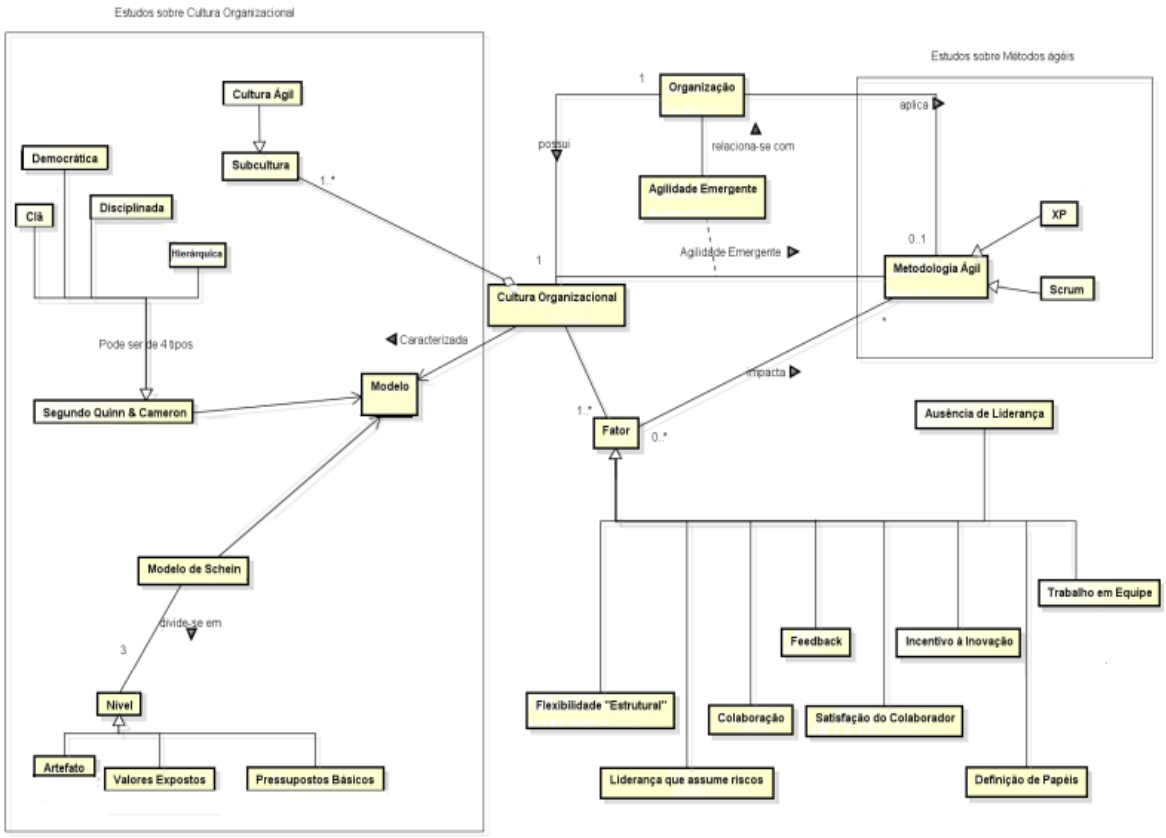
Além disso, a implementação de todos os processos de uma única vez pode trazer resistência e problemas futuros. Assim, a implementação incremental e com melhoria contínua, através de um *roadmap* com o que mais gera valor podem trazer bons benefícios. A criação de um projeto piloto para a equipe se adaptar ao novo modelo pode ser uma boa prática para identificar gargalos e impedimentos que podem ocorrer.

Com a tomada de decisão descentralizada e as equipes multidisciplinares, a alteração da estrutura pode ser necessária. A formação das equipes ocorre não

por áreas, mas por objetivos. Nesse sentido, até mesmo os recursos humanos devem se adaptar ao modelo de trabalho do *Scrum*, analisando indicadores de desempenho mais voltados para colaboração, integração e trabalho em equipe, além de novas contratações, que devem ser feitasx levando em conta a cultura do *Scrum* e *agile*.

Hofstede (1980) apresenta que a cultura organizacional corresponde a um conjunto de valores, crenças, práticas e costumes existentes em uma organização. Assim, para o bom funcionamento dos métodos ágeis nesse contexto, é necessário que a cultura organizacional esteja atrelada à mentalidade ágil.

Figura 7.3 - Um Modelo Conceitual sobre a relação entre cultura organizacional e metodologia ágil



Fonte: Amaral *et al.* (2015)

No modelo ilustrado na Figura 7.3 é possível identificar interseções entre os princípios e a base dos métodos ágeis e cultura organizacional. Dentre essas

interseções, Amaral (2015) destaca alguns pontos que podem afetar positivamente ou negativamente o processo.

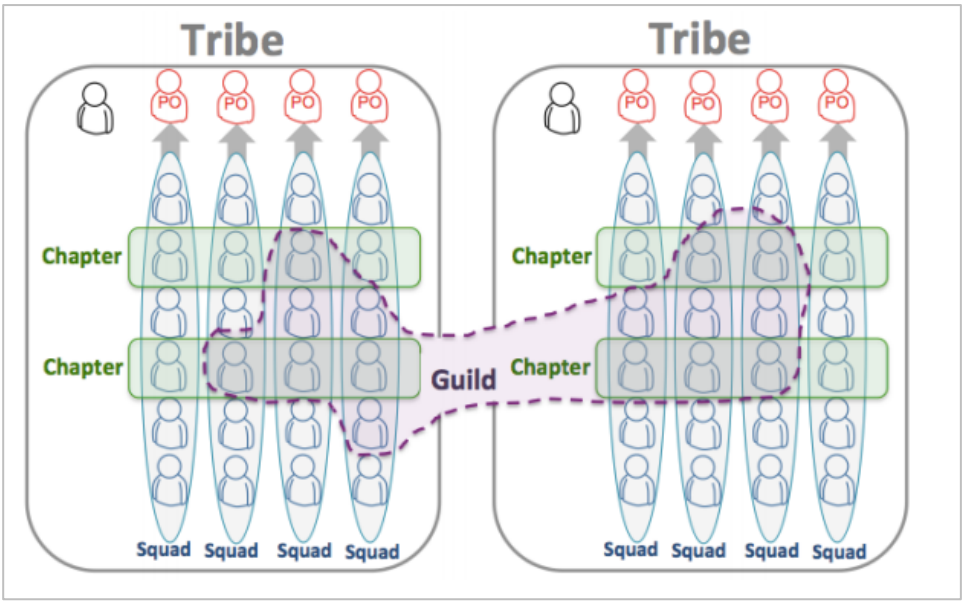
- **Flexibilidade estrutural:** Flexibilidade organizacional para receber mudanças por fatores externos ou internos em seu dia a dia.
- **Assumir riscos:** A equipe e principalmente as lideranças de equipe precisam estar preparadas para assumir riscos de eventuais imprevistos ou problemas. A Agilidade apresenta que erros são bem vindos, o que você faz com ele é o que importa.
- **Colaboração**: Todos trabalham com um foco e objetivo em comum. Entendem que o fracasso ou sucesso é fruto de todos.
- ***Feedback**:* Abertura para apresentar falhas e oportunidades de melhoria e a capacidade de aplicá-las no dia a dia da equipe e dos projetos.
- **Satisfação do time:** Oferecer aos colaboradores boas condições de trabalho, ferramentas, reconhecimento e apoio das lideranças.
- **Incentivo à inovação:** Incentivo por parte dos líderes para com os seus liderados, para que eles busquem formas diferentes de executarem seu trabalho e resolver os eventuais problemas.
- **Definição de papéis:** Todos entenderem como fazem, por que fazem e como seu trabalho impacta no todo. Uma vez que todos tenham a visão do produto final, a produtividade e motivação vem de forma natural.
- **Trabalho em equipe:** Favorece que os defeitos de uns se complementam com virtudes de outros e assim os erros e problemas se tornam mais fáceis de solucionar.
- **Falta de liderança:** Falta de uma figura de liderança que direcione o time, as tarefas e objetivos podem gerar perda de produtividade e desmotivação

Com a aplicação dessas práticas atreladas ao dia a dia do time, criando uma cultura baseada na mentalidade ágil, a aplicação do modelo de trabalho *Scrum* se torna mais fluída e consequentemente trará mais resultados e produtividade para a organização e times.

Um estudo da aplicação dessa mudança estrutural e de cultura para a aplicação do ágil é o do modelo de trabalho em *squad* desenvolvido pelo *Spotify* (Kniberg & Ivarsson, 2012). O modelo surgiu para sustentar os mais de 30 times, em 3 cidades diferentes que trabalham em conjunto. O modelo foi bem sucedido e hoje diversas empresas de diversos portes implementaram a estrutura.

Na figura 7.4 é apresentado uma representação da estrutura dos times no modelo de *squad* do *Spotify*. Dentro da estrutura, temos os *squads, chapters, tribe e guild.*

Figura 7.4 - Scaling Agile @ Spotify



Fonte: Kniberg & Ivarsson (2012)

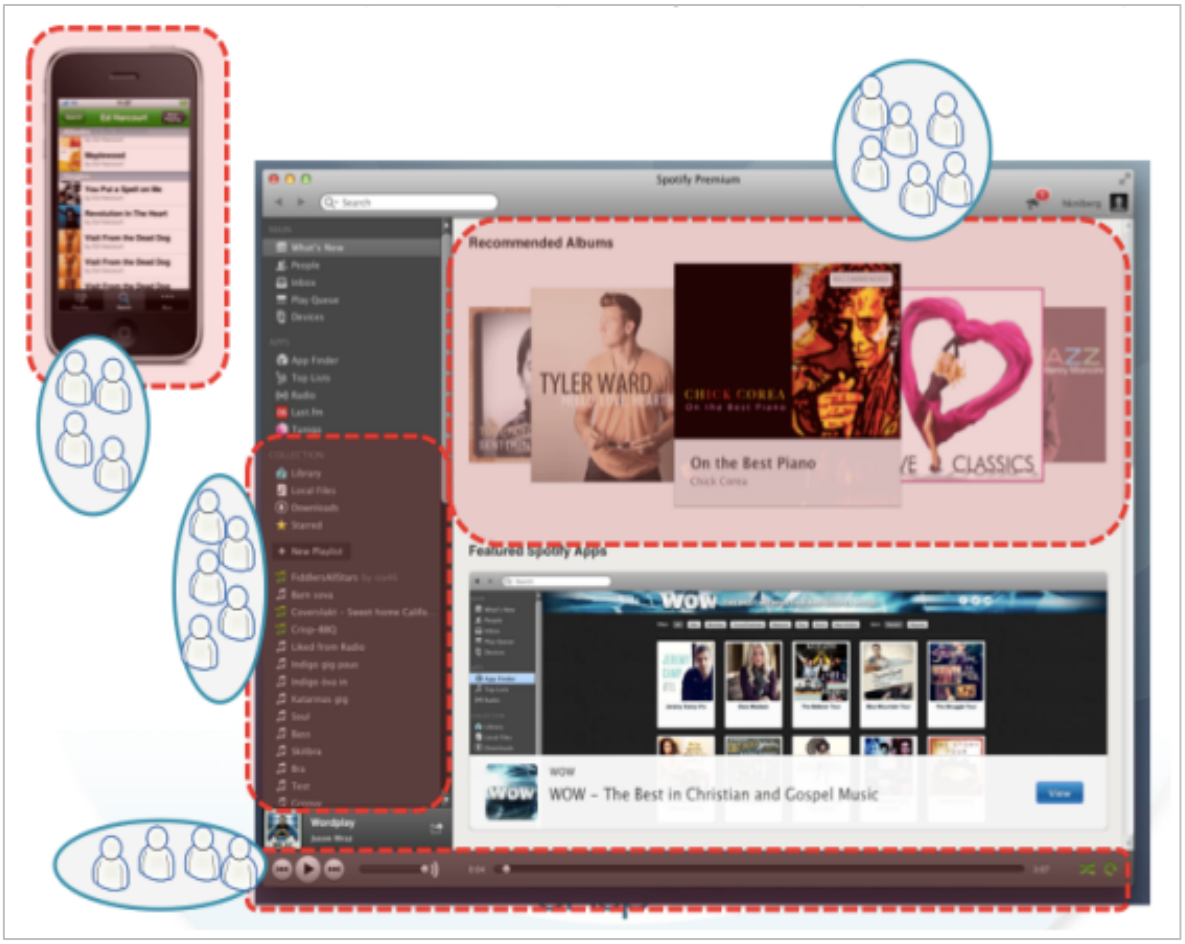
Squads é onde o trabalho de desenvolvimento acontece. Cada *squad* tem um objetivo a longo prazo para guiar o trabalho. Funcionam como um time *Scrum* e nele, existem todas as ferramentas e conhecimentos necessários para desenvolver o trabalho e testar, com autonomia para traçarem e desenvolverem o melhor caminho rumo ao objetivo a ser alcançado.

Esses objetivos estão relacionados com o objetivo maior do aplicativo de *streaming* de música, mas com o objetivo específico de trabalhar em uma parte do

aplicativo como melhorar o sistema de pagamentos, por exemplo ou experiência dentro do usuário.

Os *squads* não possuem um líder formal, mas sim o *product owner* - P.O, que auxilia no direcionamento do trabalho. Cada P.O é responsável por uma parcela do produto, que junto aos demais P.O direcionam o desenvolvimento do produto atrelado às visões do negócio. A Figura 7.5 ilustra um exemplo do trabalho dos squads.

Figura 7.5 - Exemplo do trabalho dos squads.



Fonte: Kniberg & Ivarsson (2012)

As *tribes*, ou tribos são um conjunto de *squads* que trabalham em áreas relacionadas, como por exemplo o reprodutor de música ou infraestrutura. Cada tribo possui um líder que tem como objetivo fornecer o melhor ambiente de trabalho

e desenvolvimento para os membros dos *squads*. Visando a conexão, colaboração e redução das burocracias das tribos, uma tribo normalmente não tem mais do que 100 pessoas.

Por serem da mesma área de interesse, as tribos possuem encontros regulares para compartilhamento e visão do que está sendo feito e transparência do trabalho. Quando os *squads* começam a ter muitas dependências com outras tribos, o processo e estruturas são revistos para a menor dependência possível.

Quando existem trabalhos que precisam de colaboração entre diferentes *squads*, acontece um alinhamento entre os *Scrum Masters* dos times com intuito de remover impedimentos.

Por fim, com tantos times autônomos e para que o trabalho não fique desconexo e sem trocas, existem as *chapters* (capítulos) e as *guilds* (guildas). Essas duas estruturas servem como ponto de conexão entre os *squads* e tribos.

Os *chapters* são um grupo de pessoas com habilidades semelhantes trabalhando dentro da mesma área de competência dentro da mesma tribo. Por exemplo, os *Designers* de cada *squad* da tribo de aplicativo *IOS*. Essas pessoas se reúnem frequentemente para compartilharem sobre suas dores e aprendizados. Os *chapters* possuem líderes formais que fazem o papel de gestão de recursos humanos.

Uma guilda é um conjunto de pessoas que têm interesses de conhecimento comum. As mesmas podem pegar várias tribos, *squads* e pessoas de diferentes áreas como por exemplo desenvolvimento *web*, testes e cultura ágil.

Diferente do modelo tradicional, o modelo de *squads* possui uma estrutura voltada para entrega enquanto o modelo tradicional foca em agrupar as pessoas por áreas de atuação e atribuição de projetos.

Ao utilizar corretamente a metodologia e aplicar as adaptações e configurações necessárias, tanto a gestão do desenvolvimento do projeto quanto da equipe e organização, os empecilhos tendem a diminuir e as entregas tendem a ser otimizadas.

8. DISCUSSÃO

Neste capítulo serão comentados os resultados dessa pesquisa e apresentada a relação entre as literaturas de Sutherland *et al* (2007) e Berczuk (2007) com a conclusão de seus estudos.

Durante a pesquisa, foi possível perceber que existem muitas publicações sobre *Scrum* e métodos ágeis. Porém, no contexto distribuído, o cenário é diferente, com poucas literaturas referentes. Apesar disso, com os resultados foi possível identificar inibidores que precisam de atenção durante a implementação e as boas práticas que podem ser úteis nesse cenário. Dentre os resultados encontrados, pode-se perceber que três pilares são fundamentais para a boa e efetiva implementação: cultura, pessoas e gestão.

O ágil tem o foco nas pessoas, e o alinhamento destas dentro dos métodos ágeis é fundamental para a fluidez e efetividade na aplicação. Assim, dentro da implementação destes, e do *Scrum* em questão, é importante entender o nível de conhecimento do time sobre a ferramenta, alinhar sobre as funções e ritos da empresa através de treinamentos e principalmente a implementação gradual, para evitar resistência por parte dos times. E ainda, como apresenta Berczuk (2007), até que exista uma adaptação e seja possível identificar gargalos e pontos de melhoria, é recomendado iniciar a implementação e estruturação, conforme apresentados nos guias e livros sobre *Scrum*.

Outro fato importante exposto pelo autor é que a comunicação é a base do *Scrum*, e no distribuído isso pode se tornar uma barreira e ser um dos principais inibidores. Com a má comunicação, não se tem transparência, os alinhamentos são falhos, o que pode acarretar a falta de colaboração e problemas nas entregas e incrementos da *sprint*.

Nesse sentido, a gestão e as lideranças precisam criar espaços de abertura, incentivar e orientar para que a equipe consiga encontrar o caminho do trabalho, com muito apoio e orientação. O papel da liderança, principalmente no contexto distribuído, precisa ser de confiança e autonomia. O Micro gerenciamento

pode levar a desmotivação e baixa produtividade, além de deixar o time preso e dependente, que contradiz a autonomia e a colaboratividade dos times *Scrum*.

Sutherland *et al* (2007) em seu estudo traz uma abordagem diferente do apresentado por Berczuk (2007). Enquanto Berczuk (2007) apresenta sobre a aplicação do *Scrum* com pessoas distribuídas, e a aplicação de um único modelo de trabalho, Sutherland *et al* (2007) traz sobre a aplicação de equipes *Scrum* geograficamente distribuídas, mas conectadas em um *Scrum* distribuído, escalado.

Pode-se dizer que o modelo distribuído de *Scrum* de *Scrums* é recomendado em casos em que existem diversos escritórios que trabalham em um mesmo produto, e estão dependentes entre si. Já o modelo onde as pessoas estão distribuídas, sem local físico único que possibilita esse encontro, o modelo de Berczuk (2007), onde o *Scrum* e suas cerimônias são aplicadas de forma distribuída, faz mais sentido.

Ambos apresentam vantagens e desvantagens, como o nível de alinhamento, comunicação, operação, dependência do contexto do time, da empresa e principalmente do produto.

Apesar disso, os pilares de transparência, inspeção e adaptação precisam ser levados em conta no momento da escolha entre ferramentas e alterações no modelo. Por fim, mas não menos importante, o foco nas pessoas e no entendimento dos *frameworks* e suas aplicações deve ser levado em consideração no momento da escolha e implementação dos ritos da metodologia em questão.

9. CONCLUSÃO

O desenvolvimento da revisão sistemática de literatura, veio com intuito de compilar estudos sobre a temática do *Scrum* em ambiente distribuído, visando apresentar as principais referências na literatura relacionada ao tema. Com isso, apresentar boas práticas e inibidores referentes ao tema estudado.

Apesar do *Scrum* ser um *framework* amplamente utilizado no meio de desenvolvimento de software, foi constatado que não existem muitas referências e estudos sobre a sua aplicação no contexto distribuído, sendo um fator limitante para a pesquisa, uma vez que o número de publicações e, principalmente de publicações recentes são baixas, conforme apresentado no Gráfico 6.2.

Apesar disso, o trabalho torna-se relevante pelo contexto atual da pandemia do COVID-19 e pela necessidade de adaptação das empresas para o modelo distribuído. Constituindo assim, uma fonte de referência tanto para implementação em empresas, quanto para futuros estudos relacionados ao assunto.

Com o desenvolvimento da pesquisa, percebe-se que o *Scrum* pode ser aplicado de forma efetiva em equipes distribuídas, levando em consideração a base do desenvolvimento do *framework*. Pelo fato de o *Scrum* ser um modelo de trabalho, o mesmo apresenta alguns artefatos e eventos a serem seguidos para a boa e correta aplicação, mas com abertura para adaptação conforme o ambiente em que está sendo inserido. Sendo assim, um modelo aplicável ao contexto de trabalho distribuído.

Um ponto importante percebido durante o desenvolvimento do trabalho, foi o fato de que o *Scrum* se baseia muito em uma cultura de trabalho e processos de aprendizados empíricos. Ou seja, para uma aplicação efetiva e bons resultados, faz-se necessário um trabalho de educação e conscientização dos times sobre a motivação e a importância da utilização dos artefatos e eventos, uma vez que é necessário que a equipe enxergue valor na correta execução dos ritos. Caso contrário, a implementação por si só não será bem aproveitada, não trará os

resultados esperados e a sua proposta de valor de entregar soluções adaptativas para problemas complexos não será alcançada.

Com isso, como recomendação para trabalho futuro, fica a oportunidade do desenvolvimento de um estudo de caso sobre a aplicação do *Scrum* em um contexto distribuído, podendo aplicar os levantamentos apresentados no presente estudo.

BIBLIOGRAFIA

Ågerfalk, P. & Fitzgerald, B. (2006). Old Petunias in New Bowls? Communications of the ACM, 49(10), 27-34.

Amaral, J. P., da Silva Junior, G. C., Matsubara, P. G. F. & Graciano Neto, V. V. (2015). Cultura organizacional na adoção de metodologias ágeis no desenvolvimento de sistemas de informação-rumo a um modelo conceitual à luz de um estudo sistemático. *Proc. of II WICSI*, Goiânia-GO, 21-24.

Amigoni, M. & Gurvis, S. (2009). Managing the Telecommuting Employee: Set goals, monitor progress, and maximize profit and productivity. Simon and Schuster. 320p.

Appelo, J. (2011). Management 3.0: leading Agile developers, developing Agile leaders. 1° ed. Pearson Education.

Araujo, R. C. & Galina, C. T. (2005). Análise de escopo e planejamento no desenvolvimento de software, sob a perspectiva ágil. Sistemas de Informação – Universidade do Vale do Rio dos Sinos (Unisinos). 12p.

Bass, J. M. (2014). Activities in Scrum Master Teams: Process Tailoring in Large Enterprise Projects. In 9ª Conferência Internacional IEEE sobre Engenharia de Software Global.

Bassi Filho, D. L. (2008). Experiências com desenvolvimento ágil. Dissertação de mestrado. São Paulo: USP. https://teses.usp.br/teses/disponiveis/45/45134/tde-06072008-203515/publico/Dissertacao_Metodos_Ageis_Dairton_Bassi.pdf. [7 de fevereiro de 2020].

Berczuk, S. (2007, August). Back to basics: The role of agile principles in success with an distributed Scrum team. In Agile 2007. 382-388. IEEE.

Braithwaite, K. & Joyce, T. (2005). XP expanded: Distributed extreme programming. In International conference on Extreme Programming and Agile Processes in Software Engineering Springer, Berlin, Heidelberg. 180-188.

Brant, R; Helena C. M. (2020). Desafios do teletrabalho na pandemia COVID-19: quando o home vira office. Caderno de Administração 28. Edição E. 71-75.

Bezerra, E. (2015). UML: princípios de análise e projeto de sistemas. 3. ed. Rio de Janeiro: Campus.

Brooks, F. P. & Bullet, N. S. (1987). Essence and accidents of software engineering. IEEE computer, 20(4), 10-19.

Cavalcante, J. D. Q. P. & Jorge, N. (2013). Francisco Ferreira. O fenômeno do teletrabalho: uma abordagem jurídica trabalhista. Justiça do Trabalho, 14-27.

Cichocki, P. & Maccari, A. (2007). Empirical analysis of a distributed software development project. In IFIP Central and East European Conference on Software Engineering Techniques. Springer, Berlin, Heidelberg. 169-181.

CollabNet VersionOne 13th Annual State of Agile Report. (2019). https://www.stateofagile.com/#ufh-i521251909-13th-annual-state-of-agilereport/473508. [1 de fevereiro de 2020].

Conboy, K., Coyle, S., Wang, X. & Pikkarainen, M. (2011). People over process: key people challenges in agile development. 28(4). https://ieeexplore.ieee.org/document/5560625. [1 de janeiro de 2021].

Costa C. G. A. (2001). Desenvolvimento e avaliação tecnológica de um sistema de prontuário eletrônico do paciente, baseado nos paradigmas da World Wide Web e da engenharia de software. Campinas, SP. Mestrado [Dissertação] - Faculdade de Engenharia Elétrica e da Computação. Universidade Estadual de Campinas.

Cruzes, D. S., Moe, N. B. & Dybå, T. (2016). Communication between developers and testers in distributed continuous agile testing. In 2016 IEEE 11th International Conference on Global Software Engineering (ICGSE). 59-68.

Danait, A. (2005). Agile offshore techniques-a case study. In Agile Development Conference (ADC'05). 214-217. IEEE.

Fadel, A. C. & Silveira, H. D. M. (2010). Metodologias ágeis no contexto de desenvolvimento de software: XP, Scrum e Lean. Monografia do Curso de Mestrado FT-027-Gestão de Projetos e Qualidade da Faculdade de Tecnologia–UNICAMP, 98, 101.

Farmer, M. (2004). Decision Space infrastructure: agile development in a large, distributed team. In Agile Development Conference. 95-99. IEEE.

Ferreira, J. (2012). User experience design and agile development: integration as an on-going achievement in practice (Doctoral dissertation, The Open University).

Fontenele, A., Oliveira, D. (2018). Desafios na potencialização de uma cultura ágil de inovação centrada no usuário: relato de experiência na tecnologia educacional do SAS Plataforma de Educação. Anais Estendidos do XVII Simpósio Brasileiro sobre Fatores Humanos em Sistemas Computacionais.

Fowler, M. (2006). Using an agile software process with offshore development. https://martinfowler.com/articles/agileOffshore.html. [2 de fevereiro de 2020].

Franco, E. F. (2007). Um modelo de gerenciamento de projetos baseado nas metodologias ágeis de desenvolvimento de software e nos princípios da produção enxuta. (Doctoral dissertation, Universidade de São Paulo).

Frank, A. G. & Ribeiro, J. L. D. (2012). Utilização da ti para transferência de conhecimentos entre equipes de desenvolvimento de produto: comparação entre equipes virtuais e distribuídas. Revista Produção Online, 12(4), 1106-1130.

Gandomani, T. J., & Nafchi, M. Z. (2016). Agile transition and adoption human-related challenges and issues: A Grounded Theory approach. *Computers in Human Behavior*, *62*, 257-266.

Gandomani, T. J., Zulzalil, H., Ghani, A. A. A., Sultan, A. B. M., & Nafchi, M. Z. (2013). Obstacles in moving to agile software development methods; at a glance. *Journal of Computer Science*, *9*(5), 620.

Gebremichael, M. G. (2019). Requirements engineering for large-scale agile system development: A tooling perspective. https://hdl.handle.net/20.500.12380/300667. [2 de fevereiro de 2021]

Gilb, T. & Finzi, S. (1988). Principles of software engineering management. Vol. 11. Reading, MA: Addison-wesley.

Girardi, R. (2004). Engenharia de Software Baseada em Agentes, IV Congresso Brasileiro de Ciência da Computação – CBCOMP. Itajaí, SC. ed. 39.

Herbsleb, J. D. & Moitra, D. (2001). Global software development. IEEE software, 18(2), 16-20.

Hofstede, G. (1980). Culture and organizations. International Studies of Management & Organization. 15-41.

Hogan, B. (2006). Lessons learned from an extremely distributed project. In AGILE 2006. 6 p. IEEE.

Howard, W. R. (2010). Agile project management: creating innovative products. Kybernetes.

Imbrizi, F. G. (2017). Agile software development and project portfolio management in dynamic environments: a case study of an integrated solutions provider (Doctoral dissertation).

Jacques, M. D. G. (2007). O nexo causal em saúde/doença mental no trabalho: uma demanda para a psicologia. Psicologia & sociedade, 19 (SPE), 112-119.

Júnior, L. (2016). A diferença entre um grupo e uma equipe. [online] Administradores.com. https://administradores.com.br/artigos/a-diferenca-entre-um-grupo-e-uma-equipe [4 de Fevereiro de 2020].

Keele, S. (2007). Guidelines for performing systematic literature reviews in software engineering. Vol. 5. Technical report, Ver. 2.3 EBSE Technical Report. EBSE.

Kircher, M., Jain, P., Corsaro, A. & Levine, D. (2001). Distributed extreme programming. Extreme Programming and Flexible Processes in Software Engineering, Italy, 66-71.

Kitchenham, B. (2004). Procedures for performing systematic reviews. Keele, UK, Keele University, 33, 1-26.

Kniberg, H. & Ivarsson, A. (2012). Scaling agile@ spotify. [online], UCVOF, ucvox. files. wordpress. com/2012/11/113617905-scaling-Agile-spotify-11.pdf.

Kristjánsson, S. B. (2009). Global outsourcing of product software development: a knowledge management perspective.

Kulesovs, I., & Korkkinen, M. (2013). Challenges of Best Agile Management Practices.

Layman, L., Williams, L., Damian, D. & Bures, H. (2006). Essential communication practices for Extreme Programming in a global software development team. Information and software technology, 48(9), 781-794.

Lee, J. C., Shiue, Y. C. & Chen, C.Y. (2016). Examining the impacts of organizational culture and top management support of knowledge sharing on the success of software process improvement. Computers in Human Behavior, 54(2016), 462–474.

Lima, M. M.; Lima, A. R.; Monteiro, A. C.; Cavalcanti Júnior, E. H.; Gomes, L. Q. L. (2012). Uma Revisão Sistemática da Literatura dos Processos de Desenvolvimento de Software Educativo. In: XXIII Simpósio Brasileiro de Informática na Educação, Rio de Janeiro, RJ.

Lunesu, M. I., Münch, J., Marchesi, M., & Kuhrmann, M. (2018). Using simulation for understanding and reproducing distributed software development processes in the cloud. Information and Software Technology, 103, 226-238.

Luz, M., Gazineu, D., & Teófilo, M. (2009). Challenges on adopting Scrum for distributed teams in home office environments. World Academy of Science, Engineering and Technology, 59, 308-311.

Machado, M. & Medina, S. G. (2009). SCRUM–Método Ágil: uma mudança cultural na Gestão de Projetos de Desenvolvimento de Software. Revista Científica Intraciência, Faculdade do Guarujá–UNIEESP, 1(1), 58-71.

Monaco, F. D. F. (2001). Criatividade no contexto das equipes de trabalho: uma avaliação nas células de gestão autônoma e círculos de controle da qualidade na

Ambev-Filial/SC. Dissertação de Mestrado. http://repositorio.ufsc.br/xmlui/handle/123456789/81847. [4 de Fevereiro de 2020].

Montoya, M. M., Massey, A. P., Hung, Y. T. C., & Crisp, C. B. (2009). Can you hear me now? Communication in virtual product development teams. Journal of Product Innovation Management, 26(2), 139-155.

Nisar, M. F., & Hameed, T. (2004, December). Agile methods handling offshore software development issues. In 8th International Multitopic Conference, 2004. Proceedings of INMIC 2004. 417-422. IEEE.

Nogueira, A. M.; Patini, A. C. (2012). Trabalho remoto e desafios dos gestores. Revista de Administração e Inovação, São Paulo, 9(4), 121-152, out/dez. 2012. www.journals.usp.br/rai/article/download/79292/83363. [1 de Fevereiro de 2020].

Oliveira, F. N. (2016). *SCRUM* como Metodologia Ágil na Produção de Jogos Digitais. https://www.fabricadejogos.net/posts/scrum-como-metodologia-agil-na-producao-de-jogos-digitais/. [1 de Fevereiro de 2020].

Padgett, M. (2019). U.S. and Indian managerial boundary spanning behaviors in globally distributed software teams (Doctoral dissertation, Temple University. Libraries). https://scholarshare.temple.edu/handle/20.500.12613/3376. [4 de Fevereiro de 2020].

Paasivaara, M., Durasiewicz, S. & Lassenius, C. (2009, July). Using Scrum in distributed agile development: A multiple case study. In 2009 Fourth IEEE International Conference on Global Software Engineering. 195-204. IEEE.

Pereira, P., Torreão, P. & Marçal, A. S. (2007). Entendendo *Scrum* para gerenciar projetos de forma ágil. Mundo PM, 1, 3-11.

Pinton, M. & Junior, A. S. T. (2019). Aspectos humanos da transição das organizações para o modelo ágil. In Congresso Transformação Digital 2019.

Poole, C. J. (2004). Distributed product development using extreme programming. In International Conference on Extreme Programming and Agile Processes in Software Engineering. Springer, Berlin, Heidelberg. 60-67.

Poppendieck, M. & Poppendieck, T. (2003). Lean Software Development: An Agile Toolkit for Software Development Managers. 1° Ed.. Boston: Addison-Wesley Professional.

Pressman, R. (2001). Engenharia de Software. McGraw-Hill.

Pressman, Roger S. (2006), Engenharia de Software. 6ª edição, McGraw-Hill, 752p.

Pressman, R. S. (2011). Engenharia de Software – Uma abordagem Profissional. Porto Alegre: AMGH.

Pries-Heje, L. & Pries-Heje, J. (2011). Why Scrum works: A case study from an agile distributed project in Denmark and India. In 2011 Agile Conference. 20-28. IEEE.

Prikladnicki, R., Willi, R. & Milani, F. (2014). Métodos ágeis para desenvolvimento de software. Bookman Editora.

Ramesh, B., Cao, L., Mohan, K. & Xu, P. (2006). Can distributed software development be agile? Communications of the ACM, 49(10), 41-46.

Rezende, L. V. R. (2013). Levantamento de requisitos para a implantação de um sistema de gerenciamento eletrônico de documentos em um software de gestão de processos. Florianópolis.

Richter, I., Raith, F., & Weber, M. (2016). Problems in Agile Global Software Engineering Projects especially within Traditionally Organised Corporations: [An exploratory semi-structured interview study]. In Proceedings of the Ninth International C* Conference on Computer Science & Software Engineering. 33-43.

Royce, W.W. (1970). Managing the development of large software systems: concepts and techniques. Proc. IEEE Westcon, Los Angeles, CA.

Sakuda, L. O. & Vasconcelos, F. D. C. (2005). Teletrabalho: desafios e perspectivas. Organizações & Sociedade, 12(33), 39-49.

Sampaio, J. P. D. (2011). Applicability of agile methodologies in global software development projects: a Scrum case study (Doctoral dissertation).

Schwaber, K. & Beedle, M. (2002). Agile software development with Scrum. Vol. 1. Upper Saddle River: Prentice Hall.

Shrivastava, S. V. & Rathod, U. (2014). Risks in distributed agile development: A review. Procedia-Social and Behavioral Sciences, 133, 417-424.

Simons, M. (2002). Internationally agile', InformIT (March 15th, 2002).

Smith, P. G. & Blanck, E. L. (2002). From experience: leading dispersed teams. Journal of Product Innovation Management: An international publication of the product development & management association, 19(4), 294-304.

Smits, H. & Pshigoda, G. (2007). Implementing Scrum in a distributed software development organization. In Agile 2007. 371-375. IEEE.

Soares, M. S. (2004). Comparação entre Metodologias Ágeis e Tradicionais para o Desenvolvimento de Software. INFOCOMP Journal of Computer Science, [S.l.], 3(2), 8-13. http://www.dcc.ufla.br/infocomp/index.php/INFOCOMP/article/view/68.

[07 de fevereiro de 2020].

Soares, M. S. (2004). Metodologias ágeis *extreme programming* e *Scrum* para o desenvolvimento de software. Revista Eletrônica de Sistemas de Informação. 3(1).
Sommerville, I. (2007). Engenharia de Software. São Paulo: Pearson Addison-Wesley.

Sommerville, I. (2011). Engenharia de Software. 9ª edição, São Paulo: Person Prentice Hall, 529 p.

Sommerville, I. (2007). Engenharia de Software. São Paulo: Pearson Addison-Wesley.

Song, M., Berends, H., Van der Bij, H. & Weggeman, M. (2007). The effect of IT and co-location on knowledge dissemination. Journal of Product Innovation Management, 24(1), 52-68.

Strode, D. E., Huff, S. L., Hope, B. & Link, S. (2012). Coordination in co-located agile software development projects. Journal of Systems and Software, 85(6), 1222-1238.

Sutherland, J., Viktorov, A., Blount, J. & Puntikov, N. (2007). Distributed Scrum: Agile project management with outsourced development teams. In 2007 40th annual Hawaii international conference on system sciences (HICSS'07) (274a-274a). IEEE.

The Chaos Report. (1994). [online] standishgroup.com. Available at: https://www.standishgroup.com/sample_research_files/chaos_report_1994.pdf.
[03 de Fevereiro de 2020].

Toyota Motor Manufacturing (2006). http://www.toyotageorgetown.com/history.asp.
[05 de Fevereiro de 2020].

Trindade, C. C., Moraes, A. K. O. & Meira, S. R. L. (2008). Comunicação em equipes distribuídas de desenvolvimento de software: Revisão sistemática. In ESELAW'08: Proceedings of the 5th Experimental Software Engineering Latin American Workshop.

Von Mayrhauser, A. (1990). Software engineering: Methods and management. Academic Press Professional, Inc.

Widiyatmoko, C. B. (2017). Methodological Support for Task Coordination in Global Software Engineering Projects at Product Software Companies (Master's thesis).

Yap, M. (2005). Follow the sun: distributed extreme programming development. In Agile Development Conference (ADC'05). 218-224. IEEE.