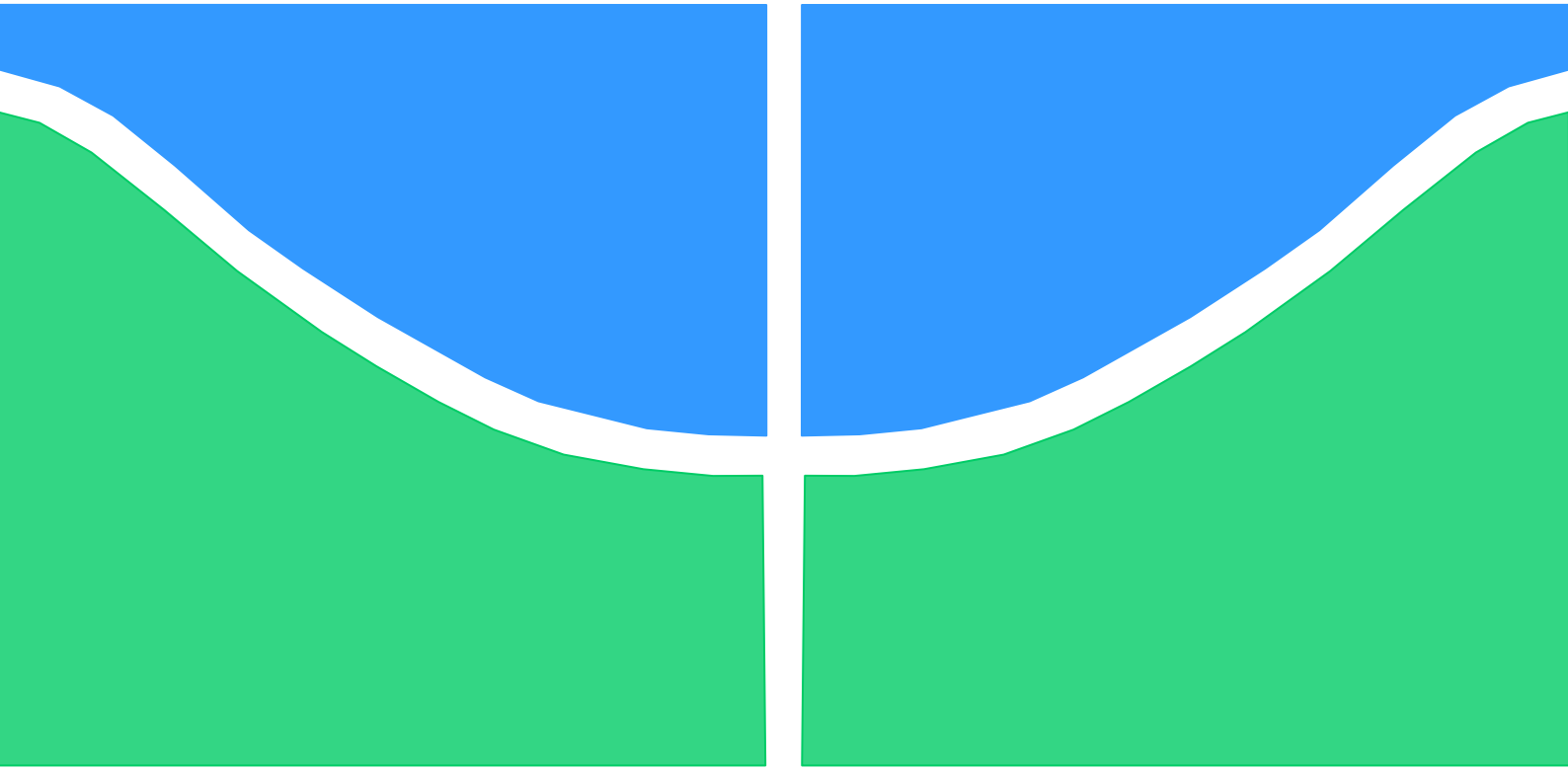


**PROJETO DE GRADUAÇÃO**

# DESENVOLVIMENTO ÁGIL DE SOFTWARE: UMA REVISÃO SISTEMÁTICA DA LITERATURA


Por,
**Rafaela Sano Machado**


**Brasília, 05 de julho de 2018**


**UNIVERSIDADE DE BRASILIA**

FACULDADE DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA DE PRODUÇÃO



UNIVERSIDADE DE BRASILIA

Faculdade de Tecnologia
Departamento de Engenharia de Produção

PROJETO DE GRADUAÇÃO

# DESENVOLVIMENTO ÁGIL DE SOFTWARE: UMA REVISÃO SISTEMÁTICA DA LITERATURA

POR,

**Rafaela Sano Machado**

Relatório submetido como requisito parcial para obtenção
do grau de Engenheiro de Produção

**Banca Examinadora**

Prof.ª Dra. Simone Borges Simão Monteiro, UnB/ EPR (Orientadora) _____

Prof. Marcito Ribeiro Madeira Campos, UnB/EPR (Coorientador) _____

Prof.ª. Dra. Adriana Regina Martin, UnB/EPR _____

Brasília, 05 de julho de 2018

*"Inteligência é a capacidade de se adaptar à mudança."*

***Stephen Hawking***

**Agradecimentos**

*Agradeço aos meus pais por todo apoio e incentivo não só ao longo de minha jornada na Universidade, mas durante toda a minha vida. Obrigada por serem minhas fontes de inspiração e por terem me passado os melhores valores para me tornar quem sou hoje.*

*Ao meu irmão, por sempre estar ao meu lado, ser companheiro e nunca medir esforços para me ajudar.*

*À minha família, por todo carinho, pela torcida de sempre e por cada sorriso nas alegrias que compartilhamos.*

*Aos meus amigos, que sempre estiveram presentes e que com certeza comemoraremos ainda muitas conquistas profissionais e pessoais de cada um de nós.*

*À minha orientadora Simone Borges, por todos os ensinamentos durante o curso e pela orientação na execução deste trabalho.*

*Aos meus coorientadores Ari Melo Mariano e Marcito Campos, por toda disponibilidade e apoio na realização deste trabalho.*

*A todos que direta ou indiretamente fizeram parte da minha formação, o meu muito obrigada.*

**RESUMO**

As necessidades de as empresas produzirem softwares de qualidade, com prazo curto e conforme as necessidades dos clientes fizeram com que aumentasse o interesse pelos métodos ágeis de desenvolvimento de software. Este trabalho tem como propósito realizar uma revisão sistemática da literatura sobre desenvolvimento ágil de software, verificando os temas de estudos mais presentes nos principais trabalhos científicos e verificando as linhas de pesquisa atuais sobre este tema. O estudo foi realizado por meio da Teoria do Enfoque Meta-Analítico Consolidado (TEMAC), de onde foi possível extrair resultados relevantes para a pesquisa. Softwares de apoio foram utilizados para obter dados mais aprofundados e leis da bibliometria foram aplicadas a fim de assegurar a consistência dos dados apresentados. Os resultados mostram que os principais temas de interesse dos autores são relativos ao conhecimento científico, aos métodos ágeis de desenvolvimento de *software* e ao envolvimento do cliente durante o desenvolvimento da aplicação. São apresentados também os métodos ágeis mais presentes nos artigos e as linhas de pesquisa atuais.

Palavras-chave: Desenvolvimento Ágil de Software, Métodos Ágeis, TEMAC.

**ABSTRACT**

The need for companies to produce quality, short-term, customer-driven software has increased interest in agile software development methods. The purpose of this work is to carry out a systematic review of the literature on agile software development, verifying the most present study themes in the main scientific works and checking current research lines on this topic. The study was carried out through the Theory of the Consolidated Meta-Analytic Approach (TEMAC), from which it was possible to extract relevant results for the research. Supporting software was used to obtain more in-depth data and laws of bibliometrics were applied in order to ensure the consistency of the presented data. The results show that the main topics of interest of the authors are related to scientific knowledge, agile methods of software development and customer involvement during application development. Also presented are the agile methods most present in the articles and the current lines of research.

Key words: Software *Agile* Development, Agile Methods, TEMAC.

# SUMÁRIO

# LISTA DE FIGURAS

# LISTA DE TABELAS

# LISTA DE SIGLAS

| | |
|---|---|
| AFC | Análise Fatorial Combinatória |
| ASD | Adaptative Software Development |
| CHD | Classificação Hierárquica Descendente |
| DCU | Design Centrado no Usuário |
| DSD | Distributed Software Development |
| FDD | Feature-Driven Development |
| JCR | Journal Citation Reports |
| TEMAC | Teoria do Enfoque Meta-Analítico Consolidado |
| XP | Extreme Programming |

# 1 INTRODUÇÃO

O desenvolvimento de software é um processo que consome tempo e recursos. No mercado altamente competitivo, as empresas buscam meios de tornarem seus processos mais eficientes e com menores custos possíveis, e isso engloba, principalmente, a indústria de desenvolvimento de software. Houve aumento da automação das atividades de desenvolvimento de software, porém os recursos são escassos. Portanto, é preciso reunir métodos e procedimentos que auxiliem a tomada de decisões, o planejamento e o agendamento das atividades do processo, além de ser necessário alocar recursos para as diferentes atividades que acontecem durante o seu desenvolvimento. As métricas para desenvolver um software são importantes para identificar em quais partes os recursos são necessários, pois eles são importantes fontes de tomadas de decisões (HARRISON; DUKE, 1995)

Os softwares, independentemente de serem complexos ou não, estão sendo cada vez mais utilizados nas organizações em todos os países, sendo que muitas dependem de sistemas complexos e automatizados. Indústrias, organizações financeiras, entre outros, estão completamente automatizadas, e muitas empresas estão caminhando para essa tendência. De acordo com Sommerville (2007), é essencial produzir e manter o software com custos adequados e com qualidade a fim de garantir o funcionamento da economia nacional e internacional.

Diversos tipos de métodos foram criados com a finalidade de auxiliar o desenvolvimento de um software, e podem ser divididos entre métodos tradicionais e métodos ágeis. Entretanto, devido a necessidade de muitas empresas desenvolvedoras de software entregarem seus produtos de forma mais rápida em relação aos modelos prescritivos, ou tradicionais, os métodos ágeis vêm sendo muito utilizados a fim de suprir essa demanda (PRESSMAN, 2011).

Contudo, em cada caso de desenvolvimento de software deve-se analisar a natureza do projeto e da aplicação, os métodos e ferramentas a serem utilizados, e os controles e produtos que devem ser entregues. Analisando esses pontos, fica mais claro verificar qual método de desenvolvimento seria melhor aplicado.

De acordo com Pressman (2011), a aplicação dos princípios de gestão de projetos auxilia uma gestão efetiva de projetos de desenvolvimento. Porém, há evidências de que os métodos tradicionais de Gestão de Projetos não são suficientes quando se avaliam os resultados dos projetos de desenvolvimento de softwares.

Encontram-se na literatura referências e estudos sobre utilização de metodologias híbridas ou mistas, que são misturas entre metodologias tradicionais e ágeis no processo de desenvolvimento de uma aplicação. Entretanto, neste trabalho não será realizada a revisão sistemática em relação a este tema, pois o foco será em métodos ágeis de desenvolvimento de software.

## 1.1 JUSTIFICATIVA

Em meados da década de 90, devido ao aumento do interesse em *hardware* e em engenharia de software por parte das indústrias, muitas empresas procuraram adotar métodos de gerenciamento de projetos efetivos para conseguirem atingir os seus objetivos organizacionais. A escolha adequada de uma metodologia de gerenciamento de projeto faz com que a organização seja mais eficiente em termos de planejamento, de definição de marcos no projeto e de melhoria da qualidade dos seus produtos. Como em diversas áreas de aplicação os produtos e serviços são desenvolvidos a partir de projetos, muitas empresas estão buscando adotar uma metodologia para gerencia-los (KERZNER; KERZNER, 2017).

Sendo assim, as indústrias que desenvolvem software têm adotado de forma crescente práticas ágeis e o foco de pesquisa nessa área vem aumentando ao longo dos anos (DINGSØYR et al., 2012).

Há autores, tais como Tore Dybå, Nils Brede Moe, Torgeir Dingsøyr, entre outros, que são grandes referências na área de desenvolvimento ágil e que estudam e propõem muitos trabalhos sobre melhorias de processos de desenvolvimento de software. Os estudos sobre como o desenvolvimento de software deve ser realizado para que a organização possa realizar as suas entregas de forma mais rápida, com maior qualidade e com menor custo estão sendo discutidas por especialistas há décadas (DYBÅ; DINGSØYR, 2008).

O aumento de estudos empíricos sobre o tema também foi evidente, havendo muitos trabalhos sobre a mudança de aplicação de métodos tradicionais para ágeis nas organizações, assim como do uso de ambas metodologias em um mesmo projeto.

O desenvolvimento ágil de software tem se tornado popular ao longo dos anos. Por meio dos estudos relacionados à melhoria do desenvolvimento de software, muitos métodos ágeis foram criados e introduzidos nas organizações. Apesar de terem diferenças, todos os métodos ágeis criados têm como princípio comum aumentar a satisfação do cliente, rápida resposta à

mudança de requisitos, entregas frequentes de resultados, colaboração e envolvimento entre a equipe e o cliente (PAETSCH; EBERLEIN; MAURER, 2003).

Dessa forma, o estudo sobre as novas frentes de pesquisas, abordagens e autores principais sobre o tema é fundamental. Este trabalho tem a importância de reunir os resultados obtidos na revisão sistemática por meio da Teoria do Enfoque Meta-Analítico Consolidado (TEMAC), de Mariano e Rocha (2017), com o intuito de identificar quais são os principais autores, artigos, países, revistas, entre outros elementos, para que possa construir um Modelo Integrador contendo os principais temas de interesse de estudo sobre desenvolvimento ágil de software. Os resultados apresentados consolidam significativos conhecimentos sobre as pesquisas de autores que são referência na área de desenvolvimento de software e suas linhas de pesquisa, proporcionando conhecimento não só aos profissionais da área de desenvolvimento de software, como também das diversas áreas da engenharia.

## 1.2 PROBLEMA

Com base no que foi apresentado como tema, determina-se como problema desse trabalho: identificar quais são os principais temas estudados sobre desenvolvimento ágil de software nos principais documentos presentes na literatura científica.

## 1.3 OBJETIVOS

### 1.3.1 Objetivo Geral

Este trabalho tem como objetivo geral identificar as principais frentes de pesquisa sobre desenvolvimento ágil de software a fim de elaborar um modelo integrador com os principais temas estudados sobre esta área.

### 1.3.2 Objetivos Específicos

Para alcançar o objetivo geral, este trabalho contempla os seguintes objetivos específicos:

- Realizar levantamento bibliográfico sobre desenvolvimento ágil de software para aprofundar o conhecimento sobre o tema;
- Aplicar a Teoria do Enfoque Meta-Analítico Consolidado (TEMAC) a fim de reunir os artigos mais citados sobre desenvolvimento ágil de software;

- Analisar os resultados obtidos na revisão sistemática para identificar as principais frentes de pesquisa sobre desenvolvimento ágil de software;
- Apresentar os principais temas estudados sobre desenvolvimento ágil de software por meio de um modelo integrador que classifica as principais temáticas sobre esta área.

## 1.4 ESTRUTURA DO TRABALHO

O presente trabalho está estruturado em cinco capítulos. O Capítulo 1 contempla a Introdução do trabalho, na qual serão apresentados o contexto, a justificativa, o problema de pesquisa, o objetivo geral e os objetivos específicos, e a estrutura do trabalho.

No Capítulo 2 é apresentado o Referencial Teórico, contendo conceitos de software, Engenharia de Software, processo de software, métodos de desenvolvimento de software, sendo que haverá breve conceituação sobre o método tradicional e uma análise mais aprofundada sobre os principais métodos ágeis, os conceitos sobre o TEMAC e, por fim, as Leis da Bibliometria;

No Capítulo 3 é apresentada a Metodologia de Pesquisa deste estudo, a qual está dividida em duas partes: na primeira é abordada a classificação do trabalho quanto a natureza, objetivos, procedimentos técnicos e abordagem utilizada. Na segunda parte são apresentadas as etapas do método utilizado para a realização desta pesquisa e a explicação das atividades realizadas em cada uma delas.

No capítulo 4 serão apresentados os resultados da revisão sistemática por meio do TEMAC sobre o desenvolvimento ágil de software, abrangendo o Modelo Integrador com os principais temas estudados nos artigos mais citados.

Por fim, no capítulo 5 será apresentada a conclusão desta pesquisa.

# 2 REVISÃO BIBLIOGRÁFICA

## 2.1 DEFINIÇÃO DE SOFTWARE

Pressman (2011) propõe que software consiste em: "(1) instruções (programas de computador) que, quando executadas fornecem características, funções e desempenhos desejados; (2) estruturas de dados que possibilitam aos programas a manipulação de informações de forma adequada; e (3) informação descritiva, tanto na forma impressa como na virtual, descrevendo a operação e o uso dos programas". De acordo com o autor, a qualidade do software é obtida por meio de bom projeto, pois cada defeito do software indica um erro no projeto ou no processo. As manutenções e solicitações de mudanças que por ventura forem necessárias ao software são mais complexas. Pressman (2011) também defende que o seu custo depende do processo de engenharia. É comum que componentes de software sejam projetados e produzidos para serem reutilizados em outros programas. Um exemplo disso são as interfaces interativas com o usuário, uma vez que nelas são reutilizados componentes de software que possibilitem criar janelas gráficas, além de outros mecanismos de interação. Isso reduz os custos do software.

Dessa forma, Pressman (2011) propõe que "o software, em todas as suas formas e em todos os seus campos de aplicação, deve passar pelos processos de engenharia".

O software tornou-se um elemento crítico em todos os aspectos da vida moderna, apoiando a realização de atividades de valor das organizações. As demandas colocadas no software, como produtividade, flexibilidade, robustez e qualidade, aumentaram a um nível exponencial, levando a novos paradigmas de desenvolvimento, formalismos e métodos de trabalho, cujo sucesso tem sido verdadeiramente notável (BENNETT et al., 2000).

O objetivo principal de um software é dar suporte aos usuários; logo, para o software ter sucesso, ele deverá atender às expectativas e necessidades dos usuários que o utilizarão. Neste contexto, a palavra "usuário" não se refere apenas a pessoas, ela pode designar também outros sistemas que tenham que interagir com o software em questão (SCHMIDT; SANTOS; MARTINS, 2006).

Os aspectos gerenciais do desenvolvimento de software recebem grande ênfase na disciplina de Engenharia de Software.

## 2.2 ENGENHARIA DE SOFTWARE

O termo engenharia de software surgiu em 1968 em uma conferência organizada para discutir problemas no seu desenvolvimento, tais como a recorrência de muitos atrasos para entrega e o alto custo de produção. Novas técnicas e métodos se mostraram necessários para controlar as etapas de desenvolvimento de software, principalmente em sistemas grandes e complexos (SOMMERVILLE, 2007).

A abordagem inicial para elaboração desses métodos tinha como foco a produção de software de qualidade, dentro do prazo e com custos previsíveis. Foram desenvolvidos, então, métodos que são utilizados até os dias atuais devido a sua vasta abordagem na condução dos levantamentos de requisitos, controles de projetos, controle sobre o desenvolvimento, controle sobre a manutenção e finalização dos projetos (SBROCCO; MACEDO, 2012).

Em contrapartida, como o desenvolvimento de software não se assemelha ao desenvolvimento de um produto físico, por exemplo, *hardware*, onde a matéria-prima é palpável e a dificuldade é em reproduzi-lo, com o software, o problema principal é no desenvolvimento, já que sua replicação tem praticamente custo zero. Pensando nisso, na década de 90, iniciaram-se estudos para definição de metodologias que focaram nos fatores humanos e no atendimento das expectativas dos clientes e não meramente em atender a burocracia do processo, cuja técnica ficou conhecida como metodologia leve. Como a aplicação dessa técnica teve retorno positivo, em 2001 seus idealizadores promoveram um manifesto reunindo todas essas práticas que são conhecidas atualmente como métodos ágeis (BASSI FILHO, 2008).

A Engenharia de Software é uma disciplina que pode ser definida por meio de três diferentes perspectivas: Ciência Aplicada, Engenharia Mecânica e Engenharia Industrial. Na visão da Ciência Aplicada, a Engenharia de Software pode ser vista como a aplicação de abordagem formal e matemática. Na Engenharia Mecânica, a Engenharia de Software pode ser vista como a aplicação de conceitos clássicos da área, tais como design e montagem de produtos baseados em peças e componentes padrões. Já na perspectiva da Engenharia Industrial, a Engenharia de Software pode ser definida como aplicação sistemática de planejamento, gerenciamento e métodos de controle na produção de artefatos de software (MAHONEY, 2004).

De acordo com Pressman (2006), a Engenharia de Software é formada por quatro camadas: Ferramentas, Métodos, Processo e Foco na Qualidade, e essas devem se apoiar em um compromisso organizacional. A Figura 1 apresenta as quatro camadas da Engenharia de Software propostas por Pressman (2006).


**Figura 1 - Camadas da Engenharia de Software**
**Fonte:** Pressman (2006)

Os processos de software constituem a base para o controle gerencial de projetos de software e é conforme o seu contexto que se define quais serão os métodos técnicos a serem utilizados, quais serão os produtos de trabalho a serem produzidos, quais serão os marcos do projeto estabelecidos, sendo que todos esses fatores asseguram a qualidade e as modificações podem ser adequadamente geridas (PRESSMAN, 2006).

Os métodos consistem na técnica de "como fazer" para desenvolver softwares, e abrangem amplo conjunto de tarefas que englobam comunicação, análise de requisitos, modelagem de projeto, construção de programas, testes e manutenção.

As ferramentas servem de apoio, que pode ser tanto automatizado quanto semiautomatizado para o processo de desenvolvimento de software e para os métodos (PRESSMAN, 2006).

O autor observa que embora existam muitos processos de software diferentes, todos possuem algumas atividades em comum, como:

- Especificação: onde os clientes e engenheiros definem o software que deve ser produzido e as restrições sobre o seu funcionamento, ou seja, nessa atividade, a funcionalidade do software e as restrições sobre sua operação devem ser definidas;
- Projeto e implementação: o software é projetado e programado;
- Validação: o software deve ser validado para garantir o atendimento às necessidades do cliente;
- Evolução: O software é modificado para refletir as mudanças de requisitos do cliente e do mercado.

De acordo com Hirama (2011), em busca de atingir os objetivos do software, com o tempo foram propostos alguns processos para conduzir o seu desenvolvimento, que são os processos de software, e são detalhados no item 2.3.

## 2.3 PROCESSO DE SOFTWARE

Segundo Sommerville (2007), processo de software se trata de um conjunto estruturado de atividades que são necessárias para ocorrer o desenvolvimento de um sistema de software, e esses processos se diferenciam de acordo com o grau de maturidade das empresas, dos tipos de produtos e do tamanho da empresa. Existem quatro classes de processos de software, segundo Sommerville (2007):

- Processos informais: não existe processo definido, sendo que fica a critério da equipe de desenvolvimento escolher qual processo será utilizado. Após a equipe definir o processo, os procedimentos e suas relações também serão por ela escolhidos.
- Processos gerenciados: existe um modelo de processo definido, a programação e os relacionamentos dos procedimentos, e estes são seguidos.
- Processos metódicos: utilizam-se um ou mais métodos para realizá-los. Normalmente são utilizados em sistemas que passaram por reengenharia.
- Processos de aprimoramento: visam a melhoria do processo de forma customizada, adequando-o às características da organização.

Assim, na Engenharia de Software o processo é uma abordagem adaptável e que, de acordo com o seu contexto, a equipe desenvolvedora de software define qual será o trabalho a ser realizado e quais serão as ações e as tarefas. O foco do processo definido é sempre busca em cumprir o prazo estimado e garantir a qualidade suficiente para satisfazer aos clientes e àqueles que de fato usarão o sistema.

Um processo de software pode ser definido como um conjunto coerente de políticas, estruturas organizacionais, tecnologias, procedimentos e artefatos necessários para conceber, desenvolver, implantar e manter um produto de software (FUGGETTA, 2000). Um processo de software bem definido deve indicar as atividades a serem executadas, os recursos requeridos, os artefatos a serem utilizados e produzidos, e os procedimentos a serem adotados, nos quais são especificados os métodos, as técnicas, entre outros.

Apesar da crescente demanda por software em praticamente todas as áreas do conhecimento, o processo de produção continua sendo um esforço coletivo, criativo e complexo, por isso, precisa ser disciplinado, acompanhado e controlado de forma a se tornar efetivo e eficiente para a organização. O foco no processo permite que um grupo de indivíduos alinhe o comportamento

e as atividades de cada membro no sentido de alcançar o objetivo comum. Assim, acredita-se que a qualidade do produto fina1 está fortemente relacionada à qualidade do processo utilizado para o seu desenvolvimento e manutenção (FUGGETTA, 2000).

A definição de processos de desenvolvimento de software tem a importância de auxiliar o aumento da produtividade e a redução de riscos de um projeto (VIEIRA, 2013). Isso pelo fato de, com o crescimento da demanda de produção de software, os prazos estão cada vez mais curtos e a cobrança com a melhor qualidade estão sendo mais exigidas. A qualidade de um produto de software está relacionada com o atendimento das exigências do cliente. De acordo com Tavares (2008), esse é um processo dinâmico e, como a natureza do produto também é dinâmica, o gerenciamento do desenvolvimento torna-se mais complexo devido, por exemplo, à essência volátil dos requisitos do cliente (TAVARES, 2008).

Considerando que para cada contexto de desenvolvimento um processo pode ser melhor aplicado, foram analisados e comparados métodos de desenvolvimento de software.

## 2.4 MÉTODOS DE DESENVOLVIMENTO DE SOFTWARE

### 2.4.1 Métodos Tradicionais de Desenvolvimento de Software

Os Métodos Tradicionais de desenvolvimento de software, também conhecidos como metodologias pesadas ou orientadas à documentação, surgiram em um contexto em que havia maior foco no desenvolvimento de softwares para ambiente de *mainframe* (ROYCE, 1970).

Durante este período, realizar alterações e correções no software gerava um custo muito alto, uma vez que o acesso a computadores era muito restrito e limitado. Outro fator limitante era a não existência de ferramentas modernas, como depuradores e analisadores de código para auxiliarem o desenvolvimento do software. Assim, a fase de planejamento era significativamente extensa e a documentação era detalhada antes de iniciar a fase de implementação do software.

Foram criados modelos de desenvolvimento de software, sendo que o primeiro foi o Modelo Sequencial Linear, também conhecido como Modelo Clássico ou em Cascata.

#### 2.4.1.1 Modelo Sequencial Linear (Clássico)

O Modelo Sequencial Linear, também é conhecido como Modelo Clássico ou Modelo em Cascata, foi descrito pela primeira vez por Winston W. Royce em 1970. É o modelo tradicional

mais antigo e, de acordo com Sbrocco e Macedo (2012), dentre todos os modelos tradicionais, este foi o mais amplamente utilizado, e até a década de 90 foi o modelo que dominou a forma de desenvolver softwares.

O Modelo Sequencial é caracterizado por ser preditivo, descritivo, sequencial, burocrático, rigoroso, orientado a processos e dados, formal e controlado. Ao implementar o Modelo Sequencial para desenvolver um software, sabe-se que o sucesso é alcançado desde que o produto final esteja em conformidade com o que foi planejado (FILHO, 2003; PRESSMAN, 2006).

Nesse modelo, a ideia é desenvolver as atividades de acordo com uma sequência, em que uma etapa só é iniciada após a conclusão da etapa imediatamente anterior. De acordo com Royce (1970), as etapas são definidas pelas atividades de análise, projeto, codificação, teste e manutenção. Pressman adaptou o modelo de Royce (1970) e a Figura 2 apresenta a proposta:

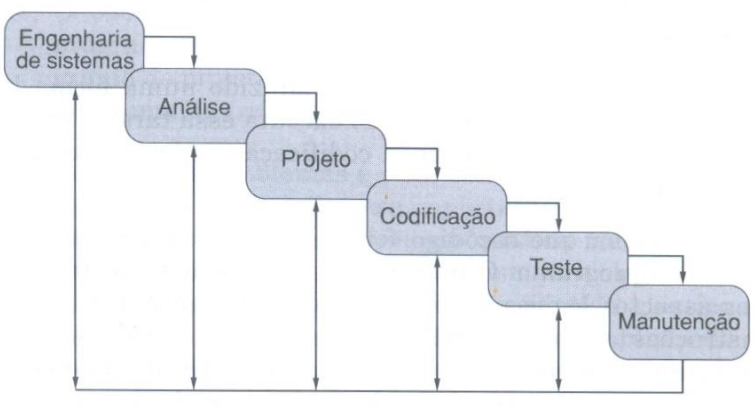



**Figura 2 - Modelo Sequencial proposto por Pressman (2006)**

**Fonte:** Pressman (2006)

O Modelo Sequencial proposto por Pressman (2006) contempla as atividades de Engenharia de Sistemas, Análise, Projeto, Codificação, Teste e Manutenção.

1. Engenharia de Sistemas: Nesta etapa, os requisitos são estabelecidos para todos os elementos do sistema e são atribuídos um subconjunto desses requisitos para o software. O objetivo é obter uma visão essencial do software em relação aos aspetos de interface com outros elementos, tais como *hardware*, pessoas e banco de dados.
2. Análise de Requisitos de Software: Os requisitos são coletados e analisados a fim de obter compreensão do que será desenvolvido e, também, de verificar se são corretos para

desempenhar a função e a interface exigidos. Os requisitos do sistema e do software são documentados e revisados posteriormente com o cliente e, após a aprovação dos requisitos, inicia-se a etapa seguinte.

3. Projeto: A documentação do projeto é elaborada, contemplando a configuração do software. Na documentação são registrados aspectos de estrutura de dados, arquitetura do software, representações da interface e detalhes procedimentais.
4. Codificação: A criação do código do software é realizada. O projeto é traduzido para uma linguagem de programação, resultando em procedimentos executáveis pelo computador.
5. Testes: As instruções do software são verificadas. Os esforços nessa atividade concentram-se nos aspectos funcionais externos a fim de verificar a existência de erros e, também, para garantir que as entradas definidas produzam os resultados esperados.
6. Manutenção: Realizam-se mudanças no software devido a erros que podem ser ocasionados ou devido à necessidade de adaptar o software às mudanças em seu ambiente externo, ou devido a exigências recebidas pelo cliente, que podem ser solicitações de melhorias funcionais e de desempenho. Após a conclusão dessa etapa, cada uma das fases é novamente replicada.

Embora o Modelo Sequencial tenha sido amplamente utilizado, houve muitas críticas relatadas por autores. Pressman (2006) afirma problemas, tais como o fato de este modelo raramente ser seguido nos projetos reais, uma vez que alguma iteração sempre ocorre e isso resulta em problemas na aplicação do paradigma; outro problema está relacionado à questão de muitas vezes ser difícil para o cliente declarar todas as exigências explicitamente e, assim, o modelo tem dificuldades de aceitar a incerteza natural existente no início de muitos projetos de desenvolvimento de software; e, também, a disponibilidade tardia da apresentação do software ao cliente, podendo muitas vezes não contemplar as funcionalidades que foram solicitadas.

Entretanto, Peters e Pedrycz (2001) apresentam como vantagem do Modelo Sequencial a elaboração de um conjunto fixo de documentos a cada final de uma etapa, facilitando o gerenciamento do projeto.

### 2.4.2 Mudança para Métodos Ágeis de Desenvolvimento de Software

A mudança para os Métodos Ágeis de desenvolvimento de software teve início em torno da década de 1990, e foi proposta por muitos praticantes de diversas nacionalidades e em perspectivas de projetos diferentes (LEFFINGWELL, 2007).

Em uma edição da Revista *Computer*, no ano 2000, Boehm (2002) afirmou que era necessário mudar o desenvolvimento de software de um processo lento para um mais rápido e dinâmico. No ano 2001, os métodos ágeis tornaram-se mais conhecidos devido à publicação do Manifesto Ágil (BECK et al., 2001). Nele, houve a união de dezessete especialistas em diversas abordagens de processos de desenvolvimento de software que buscaram estabelecer princípios comuns em práticas ágeis e compartilharam todas essas perspectivas. O Manifesto Ágil propõe a filosofia de que no desenvolvimento do software os recursos são escassos e que os requisitos são altamente voláteis. Práticas ágeis podem auxiliar o cumprimento do desenvolvimento do software devido às práticas de gerenciamento, de processo e de desenvolvimento de software (LEE; YONG, 2010).

### 2.4.3 Desenvolvimento Ágil de Software

Desenvolvimento ágil de software consiste em um grupo de métodos ágeis de desenvolvimento de software, tais como *Extreme Programming (XP), Scrum, Crystal Clear, Lean* Software *Development,* entre outros, que focam em realizar entregas do produto de software em períodos curtos de iterações, em haver maior envolvimento e aproximação com os clientes e ser adaptáveis às mudanças (ABRAHAMSSON et al., 2002).

Conforme a necessidade das organizações entregarem produtos em menos tempo e com qualidade, o desenvolvimento ágil de software tem a importância de suprir essa demanda por meio de suas abordagens mais flexíveis (DYBÅ; DINGSØYR, 2008).

Em geral, o desenvolvimento ágil de software tem como características a sua natureza incremental, cooperativa, direta e adaptativa a mudanças ao longo do processo. É incremental, uma vez que visa a entrega em partes do software e em ciclos rápidos de desenvolvimento. É cooperativa, já que o Método Ágil pode ser facilmente aprendido e modificado, além de a documentação não ser pesada. É adaptativa, pois permite a habilidade de reagir a mudanças até mesmo no final do desenvolvimento. Conboy (2009) define agilidade de desenvolvimento de software como uma rápida e inerente criação de mudança, que aprende com as mudanças

enquanto contribui em agregar valor da entrega, nos âmbitos econômicos, de qualidade e simplicidade, ao cliente, além de possuir um ambiente que envolve envolvimento e maior relacionamento entre a equipe.

Sendo assim, o tópico 2.4.4 apresenta os conceitos dos Métodos Ágeis de Desenvolvimento de Software.

### 2.4.4 Métodos Ágeis de Desenvolvimento De Software

Os métodos ágeis de desenvolvimento de software têm sido amplamente utilizados pelas indústrias especializadas em tecnologia, pois permite à equipe de desenvolvimento maior capacidade de lidar com mudanças de requisitos e, também, permite o ganho de produtividade e alinhamento de negócios (CAMPANELLI; PARREIRAS, 2015).

Em relação aos métodos ágeis, Highsmith e Cockburn (2001) afirmaram que não são as práticas que foram trazidas como novidade, mas sim o reconhecimento das pessoas como impulsionadores do sucesso do projeto, pois assim é possível haver foco intenso na eficácia e na capacidade de responder a mudanças. Tais fatores formam um conjunto de valores que definem a visão de mundo "ágil" (HIGHSMITH; COCKBURN, 2001).

Os princípios ágeis enunciados no Manifesto Ágil (2001) refletem as ideias comuns sobre o desenvolvimento ágil de software que os desenvolvedores objetivaram repassar, e elas consistiam em tratar o desenvolvimento de software com técnicas de excelência e *design* simplificado, na qual haviam entregas constantes, em curto intervalo de tempo, a fim de criar maior valor de negócio à entrega. No Manifesto Ágil (2001) existem princípios que têm como propósito auxiliar os desenvolvedores a fazerem entregas com mais valor ao cliente. Um dos fundamentos dessas práticas ágeis é organizar pequenos times que trabalhem conjuntamente, proporcionando maior comunicação, estímulo à criatividade e aumento da produtividade (DINGSØYR et al., 2012).

O conceito de "agilidade" pode ser definido como uma rápida, dinâmica e agressiva resposta a mudanças, com foco em melhorias (GOLDMAN; NAGEL; PREISS, 1995). Em sua essência, agilidade consiste na habilidade de rapidamente criar e responder, de forma flexível, às mudanças no negócio e nos domínios técnicos (HIGHSMITH; COCKBURN, 2001). Outros aspectos explorados nos métodos ágeis são a leveza e simplicidade, tais como a redução da formalidade dos processos (COCKBURN, 2006). Para Henderson-Sellers e Serour (2005), a

agilidade envolve tanto a habilidade de adaptação a diversas mudanças quanto a habilidade de refinar e ajustar os processos de desenvolvimento quando for necessário.

De acordo com Cockburn (2006), a metodologia consiste em um "procedimento sistemático", semelhante a uma técnica. Assim, busca garantir a execução consistente e estruturar as atividades contempladas no desenvolvimento.

Os Métodos Ágeis de desenvolvimento de software têm como ideia principal redefinir continuamente as prioridades. Apesar de não visar alto grau de formalidade, esses métodos não descartaram o uso de processos, ferramentas, documentos, negociação de contratos, entre outros aspectos, apenas propõem utilizar essas perspectivas de forma reduzida e objetiva. Agilidade, de acordo com Qumer e Henderson-Sellers (2006), pode ser definida como a habilidade de reagir a mudanças, esperadas ou não, em um ambiente dinâmico, de forma simples, econômica, havendo um produto de qualidade e estratégia de pequenas iterações de forma a trocar informações.

Os quatro valores do Manifesto Ágil são:

1. Indivíduos e interações entre eles mais do que processos e ferramentas;
2. Software executável mais do que documentação;
3. Colaboração do cliente mais que negociação de contratos;
4. Respostas rápidas a mudanças mais que seguir planos.

O Método Ágil não propõe a exclusão de práticas como documentação, negociação de contratos, planejamento, processos e ferramentas, mas sim a sua importância secundária, sendo os indivíduos, interações, software executável, colaboração do cliente e respostas rápidas a mudanças estabelecidos como prioridade no desenvolvimento do software.

Ser ágil é estar preparado para rapidamente agir a mudanças de forma flexível (QUMER; HENDERSON-SELLERS, 2006). Essa capacidade está relacionada à natureza flexível, rápida, enxuta e de resposta às mudanças.

Técnicas ágeis têm sido adotadas em muitas organizações como forma de reduzir custos e responder a mudanças que ocorrem no mercado altamente dinâmico. De acordo com a *Version One 12h Annual State of Agile Report* (2018) há 15 técnicas ágeis mais adotadas, conforme apresentado na Tabela 1.

**Tabela 1 - 15 Técnicas ágeis mais adotadas**

| ***Ranking*** | **Técnica** | **Porcentagem de Adoção (%)** |
|---|---|---|
| 1 | Reunião Diária | 90 |
| 2 | Planejamento da *Sprint* | 88 |
| 3 | Retrospectivas | 85 |
| 4 | Revisão da *Sprint* | 80 |
| 5 | Iterações Curtas | 69 |
| 6 | Planejamento de *Releases* | 67 |
| 7 | *Planning Poker* | 65 |
| 8 | Kanban | 65 |
| 9 | Dedicação do *Product Owner* | 63 |
| 10 | Equipe Única | 52 |
| 11 | Entregas Frequentes | 51 |
| 12 | Área de Trabalho em Comum | 47 |
| 13 | *Roadmap* de Produto | 46 |
| 14 | *Story Mapping* | 44 |
| 15 | Planejamento do Portfólio Ágil | 35 |

Fonte: Version One 12th Annual State of Agile Report (2018)

A Tabela 1 mostra que as técnicas ágeis mais adotadas são as reuniões diárias, seguido por planejamento da *Sprint*, retrospectiva do projeto e testes unitários. Cada princípio ágil surgiu em determinado método ágil.

Neste trabalho serão apresentados os conceitos dos principais métodos ágeis identificados na literatura: *Extreme Programming* (XP) e Scrum.

#### 2.4.4.1 *Extreme Programming* (XP)

O *Extreme Programming* (XP) é uma das metodologias ágeis mais conhecidas. É indicada para equipes pequenas e médias que desenvolvem software baseado em requisitos vagos e que se modificam rapidamente (BECK; GAMMA, 2000).

Este método foca nas melhores práticas para o desenvolvimento, pois a utilização de cada uma delas contribui para o aumento da qualidade do software e ajuda a garantir que o produto final agregue valor e atenda às necessidades do negócio (PRIKLADNICKI; WILLI; MILANI, 2014). De acordo com Beck (1999), o XP possui quatro valores: comunicação, simplicidade, *feedback* e coragem.

A comunicação é muito importante para haver melhor relacionamento possível entre clientes e desenvolvedores, preferindo conversas pessoais a outros meios de comunicação.

A simplicidade está relacionada a criar um código simples, sem conter funções desnecessárias, e que atendam aos requisitos atuais, não necessitando acrescentar requisitos que atendam às necessidades futuras. *Feedbacks* constantes são importantes para que o desenvolvedor tenha informações constantes do código e do cliente.

O *feedback* do código é recebido durante os testes constantes do software, nos quais é possível identificar erros e falhas tanto individuais quanto no software integrado. Já o *feedback* constante do cliente é importante para verificar se até a etapa atual as funcionalidades do software foram aprovadas ou não, e o cliente pode sugerir melhorias ou alterações no software.

Assim, erros, falhas e sugestões do cliente são corrigidos e realizados para as próximas versões e o produto final tende a estar de acordo com as expectativas reais do cliente. A coragem é importante para a implantação dos três valores anteriores.

A metodologia XP possui 12 práticas estabelecidas, segundo Beck (1999), sendo as seguintes:

- Planejamento: é definido o que é necessário ser feito e o que pode ser adiado no projeto. São levados em consideração os requisitos atuais para desenvolvimento de software, e não os futuros. Além disso, a XP procura evitar os problemas de relacionamento entre a área de negócios (clientes) e a área de desenvolvimento. Essas áreas devem cooperar para o sucesso do projeto, cada uma focando em partes específicas. Desta forma, enquanto a área de negócios deve decidir sobre o escopo, a composição das versões e as datas de entrega, os desenvolvedores devem decidir sobre as estimativas de prazo, o processo de desenvolvimento e o cronograma detalhado para que o software seja entregue nas datas especificadas com o cliente.
- Entregas frequentes: cada versão entregue deve ter o menor tamanho possível, contendo os requisitos de maior valor para o negócio. Propõe-se que as entregas devem ser realizadas a cada mês, ou no máximo a cada dois meses, aumentando a frequência de

*feedback* rápido do cliente. Essa prática evita surpresas de quando o software é entregue após muito tempo, melhora as avaliações do cliente e aumenta a probabilidade do software final estar de acordo com os seus requisitos.

- Metáfora: são as descrições de um software sem a utilização de termos técnicos, com o intuito de guiar o desenvolvimento do software.
- Projeto simples: o programa desenvolvido deve ser o mais simples possível e deve satisfazer os requisitos atuais, sem haver a preocupação com os requisitos futuros, pois estes serão adicionados assim que realmente existirem.
- Testes: a validação do projeto durante todo o processo de desenvolvimento é realizada frequentemente a fim de identificar falhas e melhorias. Antes de desenvolver o software, os programadores criam primeiramente os testes.
- Programação em pares (*Pair Programming)*: a implementação do código é feita em dupla e trabalhando em um único computador. Um dos desenvolvedores da dupla fica com o controle do teclado e implementa o código, enquanto que o outro observa continuamente o código que está sendo elaborado, procurando identificar erros sintáticos e semânticos e contribuindo estrategicamente em como melhorar o código. Esses papéis podem e devem ser alterados continuamente. Essa prática ágil possibilita que os desenvolvedores aprendam um com o outro.
- Refatoração: é realizada para aperfeiçoar o software. Visa preencher atalhos, eliminar a duplicação e tornar o *design* e a lógica do software mais claros, e após realiza-la, é preciso aplicar testes contínuos para validação.
- Propriedade coletiva: o código do projeto pertence a todos os membros da equipe. Assim, qualquer pessoa pode adicionar valor a um código, mesmo que não tenha sido quem o desenvolveu. Entretanto, quando isso é realizado, muitos testes devem ser realizados. No método XP, todos da equipe são responsáveis pelo software inteiro. Dessa forma, caso um membro da equipe deixe o projeto antes do fim, o seu desenvolvimento pode ser levado adiante com poucas dificuldades, pois todos conhecem as partes do software, mesmo que não seja de forma detalhada.
- Integração contínua: é a prática de interagir e construir o sistema de software diversas vezes por dia, mantendo os programadores em sintonia e nivelados, além de possibilitar

ocorrer processos mais rapidamente. Esta prática faz o uso de apenas uma máquina de integração, que deve ter livre acesso a todos os membros da equipe.

- 40 horas de trabalho semanal: assume-se que não se deve fazer horas extras constantemente e, caso seja necessário realizá-las, recomenda-se melhor planejamento do projeto. Esta prática busca manter o foco nas pessoas e não em processos. Caso seja necessário, os planos devem ser alterados para não sobrecarregar os membros da equipe.
- Cliente presente: é fundamental a participação do cliente durante todo o desenvolvimento do projeto para, assim, a equipe de desenvolvimento poder sanar quaisquer dúvidas de requisitos, evitando atrasos e até mesmo construções erradas. No XP é comum manter o cliente como parte integrante da equipe de desenvolvimento.
- Código padrão: padronização na arquitetura do código, para que este possa ser compartilhado entre todos os programadores.

Outro método ágil muito utilizado pela indústria de desenvolvimento de software é o Scrum, e será apresentado no item seguinte.

#### 2.4.4.2 Scrum

A primeira constatação da palavra Scrum surgiu no artigo *"The New Product Development Game*", ou "O Novo Jogo de Desenvolvimento do Produto", de Takeuchi e Nonaka (1986), em que foi notado que os projetos que possuíam equipes menores e multidisciplinares obtinham melhor performance e resultado. O nome Scrum tem a sua origem baseada no jogo de *Rugby*, e denota à ideia de "colocar uma bola fora de jogo de volta ao jogo" por meio do trabalho em equipe (SCHWABER; BEEDLE, 2002).

Posteriormente, em 1995, Ken Schwaber formalizou a definição de Scrum e suas práticas foram globalmente aplicadas no desenvolvimento de software. Schwaber e Beedle (2002) definem Scrum como uma abordagem para gerenciar sistemas de desenvolvimento de processos, e para desenvolvê-la, foram aplicadas teorias de conceitos de processos industriais para resultar em uma abordagem mais flexível, adaptativa e produtiva. É previsível que muitas mudanças possam ocorrer durante o desenvolvimento de software, e a flexibilidade do Scrum visa adaptar a essa realidade e acompanhar essas mudanças (PRIKLADNICKI; WILLI; MILANI, 2014).

Schwaber e Sutherland (2013) afirmam que o Scrum consiste em um processo empírico de controle e que é uma abordagem incremental e iterativa para otimizar e controlar riscos. Há três

pilares para implementar e controlar qualquer processo empírico: transparência, inspeção e adaptação (SCHWABER; SUTHERLAND, 2013). Transparência seria possibilitar aspectos visíveis sobre o resultado. A inspeção deve ser realizada frequentemente nos artefatos do Scrum de forma a verificar se o progresso está realmente em direção ao objetivo da Sprint, e podem ocorrer detecções de variâncias indesejáveis. Já adaptação se refere à reação, ou ajuste, de quando for verificado que aspectos do processo ou do produto estão se desviando do limite aceitável e, nesse caso, o ajuste deve ser realizado o mais rápido possível para evitar maiores desvios.

No Scrum, de acordo com Schwaber e Sutherland (2013), há quatro eventos formais para realizar a inspeção e a adaptação: Planejamento de Sprint (*Sprint Planning),* Scrum Diário (*Daily Scrum),* Revisão da Sprint (*Sprint Review*) e Retrospectiva da Sprint (*Sprint Retrospective*). As *sprints* são as iterações que ocorrem a cada ciclo.

O Planejamento da Sprint é uma reunião que ocorre antes da execução da *sprint.* Nela, o *Product Owner*, ou o dono do negócio, irá apresentar os requisitos de maior prioridade para o sistema. Há grande troca de informações entre o *Product Owner* e o time de projeto para que possa assegurar o entendimento do que será desenvolvido.

O Scrum Diário, ou Reunião Diária, é uma reunião de, no máximo 15 minutos e em pé, em que ocorre o nivelamento entre a equipe. Cada membro expõe as atividades que desenvolveram, as que estão desenvolvendo e se estão tendo dificuldade em algum ponto.

A Revisão da Sprint é uma reunião que ocorre ao final de cada *sprint* para apresentar o que foi realizado ao *Product Owner* e aos *stakeholders*.

A Retrospectiva da Sprint é uma reunião que ocorre apenas com os integrantes da equipe e que tem a finalidade de todos avaliarem como a última *sprint* foi executada e identificar pontos a serem melhorados no processo ou na execução.

Filho (2003) descreve os elementos de apoio do Scrum, sendo que os únicos elementos que a equipe produz para seguir a práticas de Scrum são cartões com as funcionalidades e gráficos de acompanhamento. Os cartões agrupados formam o *Backlog* do Produto e outros *backlogs*. Os gráficos são atualizados frequentemente e devem refletir o estado do projeto.

Os eventos realizados ao longo do ciclo do Scrum geram artefatos, tais como: *Product Backlog (backlog do produto)* e *Sprint Backlog*. O *Backlog* do Produto é uma lista que contém os requisitos mais prioritários e a sua respectiva ordem de relevância. A priorização é realizada juntamente com o *Product Owner*. Já na *Sprint Backlog*, os requisitos que foram definidos no

*Backlog* do Produto são divididos em tarefas menores e alguns são selecionados para serem realizados durante a *sprint.* O tempo necessário para concluir cada tarefa é estimado.

Schwaber e Beedle (2002) afirmam que o Scrum pode ser utilizado tanto em projetos novos, quanto existentes, uma vez que não estabelece práticas de engenharia.

O item 2.4.5 apresenta a Teoria do Enfoque Meta-Analítico Consolidado (TEMAC), que foi utilizada como base para o desenvolvimento dessa pesquisa.

### 2.4.5 Teoria Do Enfoque Meta-Analítico Consolidado (TEMAC)

Ao realizar uma revisão sistemática da literatura é importante utilizar um método que garanta a veracidade das informações que estão sendo analisadas. Segundo Dollaghan (2007), nas revisões sistemáticas buscam-se identificar as evidências externas obtidas em múltiplos estudos que foram selecionados com base em critérios adequados e procedimentos explícitos e transparentes.

Linde e Willich (2003) propõem que as revisões sistemáticas têm a importância de integrar as informações obtidas em um conjunto de estudos realizados separadamente sobre determinada intervenção, que podem apresentar tanto resultados conflitantes quanto coincidentes, bem como possibilitar a identificação de temas que necessitam de evidência, auxiliando na orientação para investigações futuras.

Segundo Mariano e Rocha (2017), por meio da Teoria do Enfoque Meta-Analítico Consolidado (TEMAC) é possível integrar as exigências presentes na literatura atual em relação aos trabalhos científicos de forma precisa, robusta, com validade, funcionalidade, tempo e custos, conforme estabelecido por Abramo e D'Angelo (2011). Dessa forma, realizando os procedimentos da pesquisa TEMAC é possível verificar quais são os melhores autores, artigos e revistas que tratam sobre um determinado tema, além de também tornar possível realizar uma análise das técnicas estatísticas das linhas mais pesquisadas e das abordagens mais utilizadas (MARIANO; GARCÍA CRUZ; ARENAS GAITÁN, 2011).

O Enfoque Meta-Analítico é uma abordagem que permite selecionar e filtrar os dados de forma sistemática a fim de realizar um estado da arte que compete a um determinado assunto (MARIANO; GARCÍA CRUZ; ARENAS GAITÁN, 2011).

Após observar as abordagens do Enfoque Meta-Analítico, Mariano e Rocha (2017) elaboraram o modelo unificado denominado Teoria do Enfoque Meta Analítico Consolidado (TEMAC), o qual unifica os aportes do uso do Enfoque Meta-Analítico e também garante que as características necessárias para uma avaliação de qualidade de artigo sejam respeitadas. Por meio do TEMAC é possível obter amplo número de possibilidades de interrelações e de inferências sobre o tema de interesse (MARIANO; ROCHA, 2017).

De acordo com Mariano e Rocha (2017), o TEMAC pode ser aplicado em três etapas: preparação da pesquisa, apresentação e interrelação dos dados, e detalhamento, elaboração do modelo integrador e validação por evidências. Há utilização de teorias bibliométricas em seus princípios, além de fazer o uso de softwares gratuitos, que têm a importância de apoiar as análises do pesquisador durante a realização das etapas. A Figura 3 apresenta as três etapas da pesquisa TEMAC.

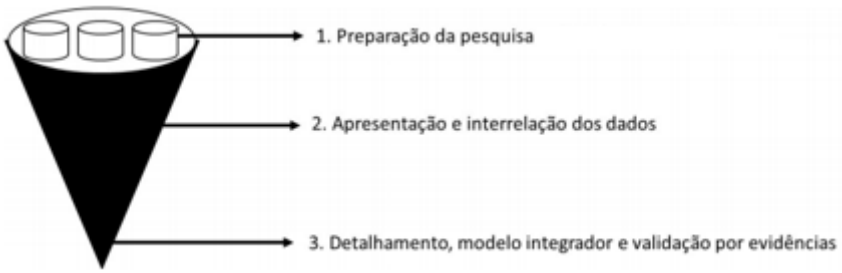



**Figura 3 - Etapas da Teoria do Enfoque Meta Analítico Consolidado (TEMAC)**

**Fonte:** Mariano e Rocha (2017)

A seguir serão apresentadas as três etapas:

1. Etapa 1: Preparação da pesquisa.

   Esta etapa consiste em identificar elementos necessários para desenvolver a revisão sistemática, tais como: palavra-chave da pesquisa ou *string*, que consiste no conjunto de palavras-chaves, o campo espaço-tempo da pesquisa, as bases de dados que serão utilizadas e as áreas de conhecimento que serão utilizadas. A escolha adequada de cada um desses elementos é fundamental, uma vez que para cada escolha determinada, resultados diferentes podem ser obtidos (MARIANO; ROCHA, 2017). As bases de dados possuem as opções de relação “OR”, que em português significa “OU”), e “AND”, que em português significa “E”, facilitando as combinações das *strings* de pesquisa. Quando

a *string* consistir em uma palavra composta, deve ser escrita entre aspas. Para escolher as palavras-chave, Mariano e Rocha (2017) recomendam observar nos artigos científicos que são relacionados ao tema de interesse do pesquisador, quais foram as palavras-chave que seus autores mais utilizaram nos trabalhos científicos. Dessa forma, serão obtidos mais trabalhos que possuem conteúdo de interesse para a realização da pesquisa. A determinação do campo espaço-tempo da pesquisa é outro fator fundamental, uma vez que bases de dados distintas podem fornecer diferentes trabalhos por período, influenciando a limitação do campo espaço-tempo. A base de dados é a fonte de obtenção dos trabalhos científicos e é importante realizar a pesquisa em bases de dados confiáveis. Assim, deve ser definida para buscar os trabalhos mais adequados para a pesquisa. Por fim, as áreas de conhecimento devem ser definidas a fim de obter mais trabalhos relacionados às áreas de interesse para a pesquisa.

2. Etapa 2: Apresentação e interrelação dos dados.

   Após definir os elementos citados na Etapa 1, a pesquisa é realizada e os resultados são apresentados e inter-relacionados nessa etapa. De acordo com Mariano e Rocha (2017), fica a critério do pesquisador realizar esta fase, porém há resultados em comum que devem ser contemplados na apresentação das pesquisas realizadas por meio do enfoque meta-analítico, tais como: análise das revistas mais relevantes, análise das revistas que mais publicam sobre o tema, evolução do tema ano a ano, apresentação dos documentos mais citados, apresentação dos autores que mais publicaram e os autores que mais foram citados sobre o tema, países que mais publicaram, agências que mais financiam a pesquisa, áreas que mais publicam e a frequência das palavras chaves. A análise dos resultados foi embasada com princípios ou leis da bibliometria.

3. Etapa 3: Detalhamento, modelo integrador e validação por evidências.

   Esta fase tem como objetivo aprofundar e detalhar as análises e as interrelações realizadas na Etapa 2. O detalhamento é realizado com o auxílio de softwares a fim de identificar e definir os autores que são essenciais para serem contemplados no referencial teórico sobre o tema, as principais abordagens, as linhas de pesquisa sobre o tema. Para obter essas informações é necessário realizar as buscas por meio de índices bibliométricos de co-citação, *coupling* (acoplamento bibliográfico) e co-autoria. Obtendo os dados desses índices bibliométricos é possível relacionar os autores, países e referências. Todas essas informações servirão de insumo para compor o Modelo Integrador, o qual contempla a

comparação das informações obtidas no detalhamento. Por fim, será realizada a validação por evidências.

As leis da bibliometria são utilizadas na pesquisa TEMAC a fim de auxiliar a apresentação dos dados obtidos.

### 2.4.6 Leis da Bibliometria

De acordo com Araújo (2006), a bibliometria se trata de uma técnica quantitativa e estatística que mede os índices de produção e de disseminação do conhecimento científico originado em decorrência da necessidade do estudo e da avaliação das atividades de produção e comunicação científica. O desenvolvimento da bibliometria se dá a partir da elaboração de leis empíricas que tratam sobre o comportamento da literatura, e exemplos são a Lei de Lotka (1926), a lei de dispersão do conhecimento científico de Bradford (1934) e a Lei de Zipf (1949) (ARAÚJO, 2006). Esses são exemplos de Leis da Bibliometria que são utilizadas para instituir modelos de comportamento e padrões de análises de dados.

Neste tópico serão contextualizadas as principais Leis da Bibliometria, que serão utilizadas nas análises dos dados obtidos por meio da revisão sistemática através da pesquisa TEMAC.

#### 2.4.6.1 Lei de Lotka

A Lei de Lotka foi formulada em 1926 e sua ideia advém de um estudo que aborda a produtividade de cientistas, que foi realizado a partir da contagem de autores presentes no *Chemical Abstracts* no período dos anos 1909 e 1916 (ARAÚJO, 2006). Lotka constatou que grande proporção da literatura científica é realizada por pequeno número de autores, enquanto que grande número de pequenos produtores se iguala, em produção, ao reduzido número de grandes produtores. A partir dessa abordagem, a Lei de Lotka, também denominada Lei dos Quadrados Inversos, foi formulada, conforme Equação 1.

$$yx = \frac{6}{p^2}\ xa$$

**Equação 1 - Fórmula da Lei de Lotka**

Em que “yx” é a frequência de autores que publicaram determinado número “x” de trabalhos e “a” é um valor constante denominado para cada campo científico.

#### 2.4.6.2 Lei de Bradford

A Lei de Bradford, também denominada Lei da Dispersão, surgiu a partir de pesquisas médicas conduzidas por Hill Bradford e outros médicos do conselho de pesquisas médicas americano. O objetivo das pesquisas era identificar a extensão da publicação de artigos científicos sobre um assunto específico, publicados em revistas especializadas daquele tema. Os resultados do estudo mostraram a existência de um pequeno núcleo de periódicos que trata sobre o tema de maneira mais extensiva, em uma vasta região periférica dividida em zonas. Observou-se nestas zonas o aumento do número de periódicos que reduzem a produtividade de publicação de artigos do respectivo assunto.

Dessa forma, a Lei de Bradford tem o propósito de estimar o grau de relevância de periódicos que atuam em certas áreas de conhecimento específicas. Periódicos que possuem maior quantidade de publicação de artigos sobre determinado tema tendem a estabelecer um núcleo de qualidade superior e maior relevância nesta área do conhecimento. Segundo esse princípio, os artigos iniciais de um determinado assunto são submetidos a um número restrito de periódicos, e a aceitação e publicação destes artigos incentivam outros autores deste assunto a encaminhar seus artigos para estes periódicos. Ao mesmo tempo, outros periódicos observam essa tendência de crescimento de publicações sobre o assunto e iniciam a publicação de artigos sobre a mesma temática (SANTOS, 2009).

Brookes (1969) afirma que, se os periódicos científicos forem apresentados em ordem decrescente de produtividade de artigos sobre determinado assunto, será possível distinguir um núcleo de periódicos mais particularmente dedicado a este tema e vários grupos ou zonas que incluem o mesmo número de artigos que o núcleo, sempre que o número de periódicos existentes no núcleo e nas zonas sucessivas seja da ordem de 1 : n: n2 (LIMA, 1984).

Dessa forma, o aumento de interesse sobre o assunto e as crescentes publicações tornam possível o estabelecimento de um núcleo de periódicos mais produtivos nessa área.

#### 2.4.6.3 Lei de Zipf

A Lei de Zipf, ou Lei do Mínimo Esforço, foi formulada em 1949 e consiste em medir a frequência do aparecimento das palavras em diversos textos, gerando uma lista ordenada de termos de uma determinada disciplina ou assunto (GUEDES; BORSCHIVER, 2005).

Segundo Araújo (2006), a Lei de Zipf descreve a relação que existe entre palavras de um determinado texto suficientemente grande e a ordem de série destas palavras. A proposta dessa

Lei é de que, ao listar as palavras que ocorrem em um texto em ordem decrescente de frequência, a posição de uma palavra na lista multiplicada por sua frequência é igual a uma constante.

Assim, George Zipf formulou o princípio do menor esforço, o qual defende que existe uma economia do uso de palavras, e se a tendência é usar o mínimo significa que uma mesma palavra vai ser usada muitas vezes, e as palavras mais usadas indicam o assunto do documento (SANTOS, 2009).

#### 2.4.6.4 Lei de 80/20

A Lei de 80/20 se trata de um fenômeno que foi inicialmente observado no comércio e na indústria, que propõe que, em sistemas de informação, 80% da demanda de informação se satisfaz com 20% do conjunto de fontes de informação (TRUESWELL, 1969). Em sistemas de informação, esta lei pode ser usada nas tomadas de decisão relacionadas à composição e redução de acervos.

De acordo com Guedes (2012), a Lei de 80/20 pode ser utilizada nas tomadas de decisão relacionadas à composição e redução de acervos.

#### 2.4.6.5 Fator De Impacto

O Fator de Impacto, ou Fator de Imediatismo, pode ser calculado com base na análise de citações, em que é investigada a concentração de citações a um determinado artigo, em documentos publicados nos últimos quinze anos (PRICE, 1965).

A hipótese dessa lei bibliométrica é de que, em uma dada área científica específica, artigos de periódicos citados com maior frequência são mais relevantes do que artigos menos citados (GUEDES, 2012). De acordo com Jones (2003), o Fator de Impacto de um título de periódico específico é calculado por meio da divisão do número de citações no ano corrente aos artigos do periódico publicados nos últimos dois anos.

#### 2.4.6.6 Obsolescência da Literatura e Vida-Média

A Obsolescência da Literatura é estimada pela análise do declínio do uso da literatura, em um período de tempo, e a Vida-Média é calculada a partir da razão de obsolescência e da razão de crescimento, de um determinado corpus da literatura (LINE, 1970).

No que se refere ao uso da literatura, a Vida-Média tem sido determinada pela análise do número de citações feitas a um determinado item (GUEDES, 2012).

A vida-média é um indicativo relativo à razão de obsolescência da literatura científica e permite determinar a idade e o declínio da literatura de uma determinada área de conhecimento, e esses indicadores são informações que os pesquisadores utilizam para poder distinguir os trabalhos que são clássicos e aqueles que são atuais em sua área de conhecimento, permitindo se atualizarem sobre determinado conhecimento produzido no campo de conhecimento em que atuam (MAROLDI; LIMA; HAYASHI, 2018).

#### 2.4.6.7 Lei do Elitismo

A Lei do Elitismo foi elaborada por Price (1965) quando, ao investigar as citações em uma área de assunto, verificou que uma parte da literatura existente é estreitamente entrelaçada e constatou que a frente de pesquisa de uma determinada área do conhecimento advém de um pequeno grupo de artigos. A Lei do Elitismo enuncia que a população de produtores da elite pode ser evidenciada por meio do cálculo da raiz quadrada do número total de autores (JUNIOR et al., 2016).

#### 2.4.6.8 Teoria Epidêmica de Goffman

A Teoria Epidêmica de Goffman foi elaborada em 1967 por Goffman e Newill seguindo o raciocínio de que as ideias científicas são materiais infecciosos e transmitidas, por exemplo, por meio de comunicações diretas ou exposta por um autor em forma de artigos de periódicos para um determinado público (ARAÚJO, 2006). Segundo Goffman (1966), sua teoria possibilita analisar a importância de linhas de pesquisa sobre um determinado tema e estimar o comportamento dessas linhas de pesquisa.

#### 2.4.6.9 Ponto de Transição (T) De Goffman

O Ponto T de Goffman foi elaborado após Goffman analisar a segunda Lei de Zipf, que foi modificada por Booth, e determina graficamente a localização onde ocorre a transição das palavras de baixa frequência para as de alta frequência (GUEDES, 2012). Assim, propõe que existe uma determinada região, ao redor de um ponto, com probabilidade de concentrar as palavras mais significativas e, portanto, essas seriam usadas para indexação de um texto em questão (GUEDES; BORSCHIVER, 2005).

#### 2.4.6.10 Frente de Pesquisa e Colégios Visíveis

A Frente de Pesquisa de uma determinada área científica é identificada a partir da análise de citações de um conjunto de autores que se citam na literatura recente, revelando um padrão estreito de relações múltiplas sobre o tema, e analisando esse pequeno conjunto de artigos entrelaçados, o trabalho de centenas de colaboradores formam os Colégios Visíveis (GUEDES; BORSCHIVER, 2005).

Assim, esses autores acabam exercendo maior influência sobre determinado assunto e recebem maior número de citações.

#### 2.4.6.11 Acoplamento Bibliográfico e Co-Citação

O Acoplamento Bibliográfico e a Co-citação foram desenvolvidos por Kessler em 1963 e também estão relacionados à análise de citações, porém possuem diferenças no contexto científico (GRÁCIO, 2016). Esses métodos bibliométricos baseiam-se nas ligações das citações e revelam informações referentes ao domínio da comunicação científica refletidos na literatura e nas ligações das citações dos pesquisadores em suas publicações (BÖRNER; CHEN; BOYACK, 2003).

Quando dois artigos abordam um item de referência em comum, eles estão bibliograficamente acoplados (EGGHE; ROUSSEAU, 2002). Já a co-citação identifica, via frequência de citações, a ligação e a semelhança entre dois documentos citados (GRÁCIO, 2016).

Definidos os temas tratados neste estudo, o Capítulo seguinte apresenta a metodologia de pesquisa utilizada no desenvolvimento deste trabalho.

# 3 METODOLOGIA

## 3.1 MÉTODO DA PESQUISA

O Método da Pesquisa pode ser classificado, segundo Silva e Menezes (2005), quanto à sua natureza, à sua forma de abordagem, aos seus procedimentos técnicos e aos seus objetivos.

Em relação à natureza, este trabalho se trata de uma pesquisa básica, uma vez que foi realizado com o objetivo de gerar conhecimentos úteis sobre desenvolvimento ágil de software que podem agregar e auxiliar profissionais que atuam na indústria de desenvolvimento de software e, de acordo com Silva e Menezes (2005), a pesquisa básica tem como o objetivo gerar conhecimentos novos e úteis para o avanço da ciência sem aplicação prática prevista, e envolve verdades e interesses universais.

Quanto à forma de abordagem, este trabalho pode ser enquadrado em uma pesquisa tanto qualitativa quanto quantitativa, uma vez que será aplicada a Teoria do Enfoque Meta Analítico Consolidado (TEMAC), de Mariano e Rocha (2017), por onde serão obtidos resultados quantitativos, sendo que a pesquisa quantitativa requer o uso de técnicas estatísticas para reunir informações que serão analisadas (SILVA; MENEZES, 2005). E é qualitativa por haver a compreensão e o aprofundamento de um grupo social, de uma organização, entre outros aspectos (GERHARDT; SILVEIRA, 2009).

Já em relação ao objetivo de pesquisa, este trabalho consiste em uma pesquisa exploratória, pois tem como propósito proporcionar maior familiaridade quanto ao tema de desenvolvimento ágil de software. Nessa perspectiva, Gil (2002) defende que pesquisas exploratórias têm como objetivo principal o levantamento de ideias ou a descoberta de intuições.

Quanto à estratégia da pesquisa, este trabalho consiste em uma pesquisa bibliográfica, uma vez que será efetuado o estado da arte por meio de uma revisão sistemática sobre desenvolvimento ágil de software. Gil (2002) defende que pesquisas bibliográficas são elaboradas com base em informações obtidas por materiais já publicados, tais como artigos de periódicos, livros, entre outros materiais disponíveis, assim como foi desenvolvido este trabalho por meio da utilização do TEMAC, de Mariano e Rocha (2017).

Assim, de acordo com os autores Silva e Menezes (2005) e Gil (2002), este trabalho possui as seguintes classificações, conforme apresentado na Tabela 2:

**Tabela 2 - Classificações da Pesquisa**

| Dimensões de Classificação | Alternativas de Classificações | Classificação para este trabalho |
|---|---|---|
| Natureza | Pesquisa Básica<br>Pesquisa Aplicada | Pesquisa Básica |
| Abordagem do Problema | Pesquisa Quantitativa<br>Pesquisa Qualitativa | Pesquisa Quantitativa e Qualitativa |
| Estratégia de Pesquisa | Pesquisa Bibliográfica<br>Pesquisa Documental<br>Pesquisa Experimental<br>Levantamento<br>Estudo de Caso<br>Pesquisa Expost-Facto<br>Pesquisa Ação<br>Pesquisa Participante | Pesquisa Bibliográfica |
| Objetivos | Pesquisa Exploratória<br>Pesquisa Descritiva<br>Pesquisa Explicativa | Pesquisa Exploratória |

Fonte: da própria autora

A Estruturação da Pesquisa do presente trabalho será apresentada no item 3.2.

## 3.2 ESTRUTURAÇÃO DA PESQUISA

Conforme citado na Seção 3.1, este trabalho é classificado em uma pesquisa básica, quantitativa e qualitativa, bibliográfica e exploratória, em que será aplicada a Teoria do Enfoque Meta Analítico Consolidado (TEMAC) para identificar as principais informações necessárias para cumprir o objetivo geral deste trabalho.

A realização da pesquisa foi estruturada com base nas três etapas do TEMAC (MARIANO; ROCHA, 2017).

1. Etapa 1: Preparação da pesquisa.

   Esta etapa apresenta a base de dados escolhida para realizar a revisão sistemática e a sua justificativa. Também foi descrito como foram definidos a *string*, as áreas de conhecimento e o campo espaço-tempo utilizados para a pesquisa. Todos esses elementos foram aplicados na pesquisa a fim de realizar a Etapa 2 do TEMAC.

2. Etapa 2: Apresentação e interrelação dos dados.

   A pesquisa ocorreu por meio da utilização dos elementos definidos na Etapa 1 e nessa etapa são apresentados a análise das revistas mais relevantes na área de Engenharia de Produção, a análise das revistas que mais publicam sobre desenvolvimento ágil de software, a evolução do tema ano a ano, a apresentação dos documentos mais citados, a apresentação dos autores que mais publicaram e os autores que mais foram citados sobre desenvolvimento ágil de software, os países que mais publicaram, as agências que mais financiam a pesquisa sobre o tema, as áreas que mais publicam e a frequência das palavras chaves sobre desenvolvimento ágil de software.

   Para identificar as revistas mais relevantes na área de Engenharia de Produção foi utilizada a plataforma *Journal Citations Reports (JCR),* pertencente à base de dados *Web of Science,* a fim de identificar as revistas que apresentam maior Fator de Impacto.

3. Etapa 3: Detalhamento, modelo integrador e validação por evidências.

   As análises dos índices bibliométricos de co-citação, *coupling* (acoplamento bibliográfico) e co-autoria foram realizadas, e posteriormente foram apresentados e analisados o Modelo Integrador, e a validação por evidências do modelo foi realizada. Em seguida, foram apresentadas a Análise Fatorial Confirmatória e a Análise de Similitude. O Modelo Integrador, a Análise Fatorial Confirmatória e a Análise de Similitude foram elaborados com base nos 70 artigos mais citados sobre desenvolvimento ágil de *software.*

## 3.3 INSTRUMENTO DE COLETA DE DADOS

Na primeira etapa do TEMAC, para definir a *string* foi utilizada a base de dados *Web of Science.*

Na segunda etapa do TEMAC, para realizar análises mais aprofundadas dos dados obtidos pela *Web of Science*, foi utilizado o software bibliométrico *VOSViewer* 1.6.5 para verificar a co-ocorrência de palavras-chave, *coupling* (acoplamento bibliográfico), co-citação e co-autor. A co-ocorrência de palavras-chave evidencia a frequência de palavras-chaves presentes nas principais linhas de pesquisa. A análise de *Coupling* revela as frentes de pesquisa, enquanto que a co-citação traz uma perspectiva das abordagens mais utilizadas. A análise de co-autoria revela os autores que mais publicam em parceria (MARIANO; ROCHA, 2017). A visualização do

tipo *Density Visualization* foi utilizada para obter os mapas de calor de co-ocorrência de palavras-chave, co-citação, *coupling* e co-autoria.

O *VOSViewer* 1.6.5 é um software bibliométrico gratuito utilizado para a criação, visualização e exploração de mapas baseados em redes de dados, e através de algoritmos de clusterização, realiza a separação dos autores em grupos, de acordo com as suas linhas de estudo.

Para identificar as palavras-chaves mais frequentes nos documentos científicos utilizou-se também o *site TagCrowd,* que é uma ferramenta gratuita e *online* que, a partir de um texto, cria uma nuvem de palavras, apresentando os termos mais utilizados e suas respectivas frequências. Assim, foi possível verificar quais foram as palavras mais citadas nos 500 artigos mais citados sobre desenvolvimento ágil de software, obtidos no *Web of Science*.

Na terceira etapa do TEMAC, a fim de construir o Modelo Integrador e, também, a fim de obter outras análises mais aprofundadas e com embasamento estatístico sobre desenvolvimento ágil de software, foram extraídos, do *Web of Science*, os resumos dos 70 artigos mais citados sobre desenvolvimento ágil de software e esses foram inseridos no software Interface de R *pour les Analyses Multidimensionnelles de Textes et de Questionnaires* (Iramuteq), por meio do qual foi possível obter também a Análise Fatorial Confirmatória e a Análise de Similitude.

De acordo com Camargo e Justo (2013), o software Iramuteq realiza análises estatísticas por meio da qual identifica a quantidade de palavras, a frequência, a média e o número de hapax, que são as palavras citadas apenas uma vez, pesquisa o vocabulário e reduz as palavras com base em suas raízes.

O Modelo Integrador foi elaborado com base nos resultados obtidos na Classificação Hierárquica Descendente (CHD), obtida pelo Iramuteq. O CHD foi proposto por Reinert (1990) e classifica os segmentos de texto de acordo com os seus respectivos vocabulários, e o seu conjunto é repartido de acordo com as frequências das formas reduzidas (CAMARGO, 2005). De acordo com Camargo (2005), analisando a CHD é possível verificar os segmentos de textos que possuem tanto vocabulário semelhante entre si quanto vocabulário diferente de outros segmentos de texto, calculando as distâncias e as proximidades a partir de testes do Qui-Quadrado.

A Análise Fatorial Confirmatória (AFC) é uma das técnicas estatísticas que estão sendo mais utilizadas em pesquisa aplicada e permite visualizar, sob a forma de um plano fatorial, os resultados contemplados na CHD.

A Análise de Similitude apresenta as co-ocorrências entre as palavras e, também, indica as conexões entre as palavras, auxiliando na identificação da estrutura de um corpus textual, distinguindo também os elementos comuns e as especificidades em função das variáveis ilustrativas (MARCHAND; RATINAUD, 2012).

# 4 RESULTADOS

## 4.1 APRESENTAÇÃO DOS RESULTADOS OBTIDOS NA APLICAÇÃO DO TEMAC

Após seguir os procedimentos técnicos apontados na Estruturação da Pesquisa, contemplada na Seção 3.2, os resultados das três etapas da revisão sistemática por meio da Teoria do Enfoque Meta-Analítico Consolidado (TEMAC) sobre métodos ágeis de desenvolvimento de software serão apresentados. Para cada etapa, os resultados serão apresentados e analisados com base em trabalhos científicos sobre o tema e utilizando leis da bibliometria a fim de assegurar a consistência dos dados obtidos no estudo.

### 4.1.1 Resultados da preparação da pesquisa

Segundo a primeira etapa do TEMAC, de Mariano e Rocha (2017), antes de apresentar os resultados da preparação da pesquisa é fundamental explicar a escolha de cada base de dados, e as suas respectivas definições, a escolha das *strings* selecionadas para serem utilizadas para o desenvolvimento do presente trabalho, o porquê da escolha do intervalo de tempo de interesse da pesquisa e, também, a data de coleta dos dados.

Como foi constatado que alguns artigos realizaram a revisão sistemática de desenvolvimento ágil de software a partir do ano 2000, um ano antes da difusão dos métodos ágeis devido a publicação do Manifesto Ágil (BECK et al., 2001), o campo espacial do presente trabalho foi definido entre os anos 2000 a 2018.

As buscas para atingir os objetivos de cada fase da Revisão Sistemática por Meio do Enfoque Meta-Analítico ocorreram do dia 31 de maio até o dia 24 de junho de 2018. A base de dados escolhida para realizar a revisão sistemática através do TEMAC foi a *ISI Web of Science*, pois trata-se de uma base de dados conhecida internacionalmente como uma das melhores e mais completas por possibilitar e dispor de diversas ferramentas para a exportação, manipulação e estudo dos dados por ela fornecidos (GARCÍA; RAMIREZ, 2004). As buscas foram definidas com base nas palavras-chaves que mais apareciam em artigos sobre o tema referente a métodos ágeis de desenvolvimento de software. Neste trabalho foi utilizado o conector AND para termos alternativos de cada palavra-chave a fim de encontrar mais pesquisas que são relevantes para o presente trabalho. Observando nos artigos relativos a desenvolvimento ágil de software foram

observadas as palavras-chaves que eram mais utilizadas e essas foram aplicadas no *ISI Web of Science* a fim de verificar qual seria a *string* que traria mais artigos relevantes para a pesquisa. Após analisar as palavras-chave que os autores utilizaram em suas pesquisas, foram definidas as *strings* a serem testadas: software *development,* software *engineering, agile approach, agile methods e agile* software *development.*

As áreas de interesse foram definidas em seguida para que pudesse obter trabalhos científicos que tivessem conteúdo da área de interesse, que é a Engenharia de Produção. Assim, foram selecionados os seguintes filtros de área: *Engineering Industrial, Engineering Manufacturing, Management* e *Business.* A Tabela 3 apresenta a quantidade encontrada de resultados encontrados para cada *string* das pesquisas realizadas para o desenvolvimento ágil de software e, também, apresenta a quantidade de resultados encontrados após o refinamento por áreas de pesquisa.

**Tabela 3 - *Strings* da Pesquisa sobre desenvolvimento ágil de software**

| ***Strings* utilizadas** | **Resultados sem filtro** | **Resultados com filtros *(Business, Computer Science Software Engineering, Computer Science Theory Methods, Computer Science Information Systems, Computer Science Interdisciplinary Applications, Engineering Industrial, Engineering Manufacturing, Management,)*** |
|---|---|---|
| *"Agile* Software *Development"* | 1.224 | 1.045 |
| *"*Software *Development" AND agile* | 2.440 | 2.083 |
| *"*Software *Development" AND "agile approach"* | 111 | 90 |
| *"*Software *Development" AND "agile methods"* | 548 | 488 |
| *"*Software *Engineering" AND "agile approach"* | 33 | 25 |
| *"*Software *Engineering" AND "agile methods"* | 195 | 172 |
| *Software AND agile* | 4.205 | 3.363 |
| *"Requirements Engineering" AND "Agile Development"* | 33 | 31 (Não havia "*Business", "Management*" e "*Engineering Manufacturing*") |

Fonte: da própria autora

Como a *string "*Software *Development" AND agile"* resultou em mais artigos com temáticas distintas e com maior resultado relevante para a revisão sistemática, definiu-se que

essas seriam as *strings* escolhidas para desenvolver a revisão sistemática por meio da Teoria do Enfoque Meta-Analítico Consolidado (TEMAC), de Mariano e Rocha (2017).

Assim, a Tabela 4 apresenta as escolhas de cada elemento que era necessário definir na primeira etapa do TEMAC, que são as *strings* de pesquisa, as palavras chaves a serem utilizadas, o campo espaço-tempo, a base de dados que será utilizada e as áreas de conhecimento que serão utilizadas.

**Tabela 4 - Escolhas de cada elemento da Etapa 1 do TEMAC**

| Fator | Escolha |
|---|---|
| Base de Dados | *ISI Web of Science* |
| Campo Espacial (Anos) | 2000 a 2018 |
| *String* selecionada para Método Ágil | *"*Software *Development" AND "agile"* |
| Áreas de Conhecimento | *Engineering Industrial, Engineering Manufacturing, Business, Management, Computer Science* Software *Engineering, Computer Science Theory Methods, Computer Science Information Systems, Computer Science Interdisciplinary Applications.* |

Fonte: da própria autora

### 4.1.2 Apresentação e interrelação dos dados

A segunda etapa do TEMAC consiste em apresentar e inter-relacionar os dados. Os resultados comuns a serem apresentados contemplarão neste tópico, e são os seguintes: análises das revistas mais relevantes sobre desenvolvimento ágil de software, análises das revistas que mais publicam, a evolução de cada tema ano a ano, documentos mais citados, autores que mais publicaram *versus* autores mais citados, países que mais publicaram, conferências que mais contribuíram, universidades que mais publicaram, agências que mais financiam a pesquisa, áreas que mais publicam e a frequência de palavras chaves (MARIANO; ROCHA, 2017).

#### 4.1.2.1 Análise das revistas mais relevantes sobre desenvolvimento ágil de software

O primeiro resultado a ser apresentado é referente às revistas mais relevantes sobre desenvolvimento ágil de software. Essas revistas foram selecionadas segundo o fator de

impacto ISI (*Institute for Scientific Information*). O Fator de impacto de determinado periódico é definido como a razão entre o número de citações feitas no corrente ano a itens publicados no periódico nos últimos dois anos e o número de artigos publicados nos mesmos dois anos pelo mesmo periódico *(INSTITUTE FOR SCIENTIFIC INFORMATION*, 1993*).*

Como a área de Engenharia de Produção é de grande interesse para este trabalho, foram identificadas as revistas mais conceituadas nessa área. Para identifica-las, foi necessário acessar a plataforma *Journal Citation Reports (JCR).* Feito isso, foram selecionadas as categorias *Engineering Industrial* e *Engineering Manufacturing* a fim de obter as revistas de maior referência na área de Engenharia de Produção. O ano 2016 foi adotado como base na seleção das revistas. Assim, a Tabela 5 apresenta as cinco revistas mais relevantes na área de Engenharia de Produção de acordo com o fator de impacto segundo *ISI Journal Citation Reports Edition (JCR)* do ano 2016. Foram encontradas 79 revistas, sendo que 33 revistas possuíam fator de impacto maior que 2.

**Tabela 5 - Revistas mais relevantes na área de Engenharia de Produção**

| ***Rank*** | **Nome da Revista** | **Fator de Impacto** |
|---|---|---|
| 1 | IEEE Transactions on Industrial Informatics | 6,764 |
| 2 | IEEE – ASME Transactions on Mechatronics | 4,357 |
| 3 | Composites Part A – Applied Science and Manufacturing | 4,075 |
| 4 | International Journal of Machine Tools & Manufacture | 3,995 |
| 5 | Journal of Product Innovation Management | 3,759 |

Fonte: Journal Citation Reports (JCR)

Após identificar as principais revistas da área de Engenharia de Produção, a próxima etapa consistiu em analisar se as revistas que mais publicaram trabalhos científicos sobre desenvolvimento ágil de desenvolvimento de software, identificadas no item 4.1.2.2 correspondiam a alguma revista presente entre as principais da área de Engenharia de Produção.

Dentre as 1098 revistas que mais publicaram sobre desenvolvimento ágil software, sete (7) faziam parte das revistas mais relevantes da área de Engenharia de Produção, conforme mostrado na Tabela 6. Dessas revistas, três apresentam fator de impacto maior que dois (2). Sendo assim, essas são as principais revistas caso seja de interesse do pesquisador publicar artigo sobre desenvolvimento ágil de software na área de Engenharia de Produção. Segundo

Barros (2015), para saber se uma revista possui significativo fator de impacto, basta compará-la com outras revistas dentro de sua categoria de assunto.

**Tabela 6 - Revistas que mais publicam sobre Métodos Ágeis e que possuem relevância na Engenharia de Produção**

| Ordem de Relevância | Revista | FI |
|---|---|---|
| 1 | JOURNAL OF PRODUCT INNOVATION MANAGEMENT | 3,759 |
| 2 | TECHNOVATION | 3,265 |
| 3 | JOURNAL OF ENGINEERING AND TECHNOLOGY MANAGEMENT | 2,419 |
| 4 | JOURNAL OF COMPUTING AND INFORMATION SCIENCE IN ENGINEERING | 1,357 |
| 5 | IEEE TRANSACTIONS ON ENGINEERING MANAGEMENT | 1,188 |
| 6 | SOUTH AFRICAN JOURNAL OF INDUSTRIAL ENGINEERING | 0,576 |
| 7 | ENGINEERING MANAGEMENT JOURNAL | 0,548 |

Fonte: da própria autora

Este mesmo procedimento foi realizado com as principais revistas da área de Ciência da Computação segundo fator de impacto na *ISI Journal Citation Report Edition (JCR)* do ano 2016, uma vez que foi verificado, no item 4.1.2.11, que essa é a área que apresenta maior interesse sobre desenvolvimento ágil de software. Para identificá-las, foram selecionadas as categorias *"Computer Science, Information Systems", "Computer Science,* Software *Engineering"*, *"Computer Science, Interdisciplinary Applications"* e *"Computer Science, Theory & Methods"*. Foram obtidas 387 revistas, sendo que 144 apresentaram fator de impacto maior que dois (2).

Dentre as revistas que mais publicaram sobre desenvolvimento ágil de software, 112 fazem parte da amostra das revistas que possuem maior relevância na área de Ciência da Computação. A Tabela 7 apresenta as 15 revistas que correspondem a essa amostra. O fator de impacto é maior em relação à comparação das revistas que mais publicam sobre desenvolvimento de software e que estão presentes dentro do conjunto das que possuem maior fator de impacto na área de Engenharia de Produção.

**Tabela 7 - Revistas que mais publicam sobre desenvolvimento ágil de software e que possuem relevância na Ciência da Computação**

| Ordem de Relevância | Revista | FI |
|---|---|---|
| 1 | MIS QUARTERLY | 7,268 |
| 2 | JOURNAL OF INFORMATION TECHNOLOGY | 6,953 |
| 3 | ACM COMPUTING SURVEYS | 6,748 |
| 4 | COMMUNICATIONS OF THE ACM | 4,027 |
| 5 | COMPUTER PHYSICS COMMUNICATIONS | 3,936 |

Fonte: da própria autora

Assim, ainda pode ser mais explorada a publicação de artigos sobre desenvolvimento ágil de software em revistas com maior de impacto na área de Engenharia de Produção. Entretanto, este número pode aumentar, uma vez que foi analisado que as áreas de interesse *Management* (Gerenciamento) e *Business* (Negócios) são, respectivamente, a sétima e a nona maiores áreas que possuem mais interesse sobre desenvolvimento ágil de software dentre as áreas de Engenharia, conforme apresenta o gráfico da Figura 12, no item 4.1.2.11.

As revistas que mais publicaram sobre desenvolvimento ágil de software não estão presentes nas revistas que possuem maior fator de impacto na área de Engenharia de Produção. Entretanto, pode ser que este quadro mude com o aumento de publicações. A Lei de Bradford, segundo Guedes e Borschiver (2005), considera que à medida em que os primeiros artigos sobre um novo assunto são escritos, eles são submetidos a uma pequena seleção, por periódicos específicos, e, se publicados, esses periódicos atraem maior quantidade de artigos no decorrer do desenvolvimento da área de assunto. Assim, pode-se inferir que, com o aumento de artigos sobre o tema nessas revistas, pode ser que emerja eventualmente um núcleo de periódicos, correspondente aos periódicos mais produtivos em termos de artigos, sobre o tal assunto.

Em contrapartida, um resultado foi positivo ao constatar que a terceira e a quarta revista que mais publicam sobre desenvolvimento ágil de software fazem parte das revistas de maior fator de impacto na área de Ciência da Computação, possuindo, respectivamente, fatores de impacto igual a 2,444 e 2,694.

#### 4.1.2.2 Análise das revistas que mais publicam sobre desenvolvimento de software

Após verificar as revistas mais conceituadas na área de Engenharia de Produção, foram identificadas as revistas que mais publicam sobre desenvolvimento ágil de software a fim de verificar quais são as revistas que mais possuem interesse neste tema. No total, foram obtidas

1.098 revistas que publicaram sobre desenvolvimento ágil de software. A revista que fez mais publicações foi a *Lectures Notes in Computer Science*, contemplando 9,43% dos registros, ou 197 artigos, e em seguida, foi a revista *Lecture Notes in Business Information Processing*, totalizando 8,29% trabalhos científicos, com 173 artigos publicados. A Figura 4 apresenta um Gráfico de Pareto das 21 revistas que mais publicaram sobre desenvolvimento ágil de software e a sua respectiva porcentagem em relação ao total de revistas identificadas nos resultados.

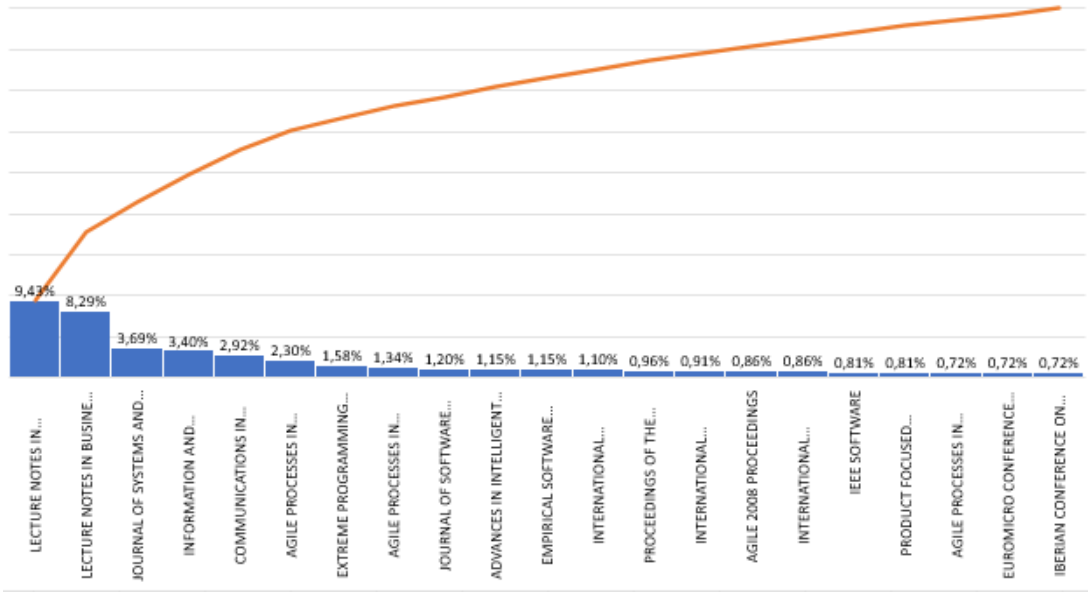



**Figura 4 - Gráfico de Pareto das revistas que publicaram sobre desenvolvimento ágil de software**

Fonte: *Web of Science*

Estima-se que 21,41% das publicações sobre desenvolvimento ágil de software são realizados pelas revistas *Lectures Notes in Computer Science*, *Lecture Notes in Business Information Processing* e *Journal of Systems and* Software. De acordo com a Lei de 80/20 pode-se inferir que, como correspondem a aproximadamente 20% das publicações, essas revistas são responsáveis por gerar 80% do conteúdo sobre desenvolvimento ágil de software. Entretanto, são as duas revistas que mais publicam que possuem maior peso nas publicações, uma vez que correspondem a 17,72%.

Os artigos mais citados sobre desenvolvimento ágil de software da *Lectures Notes in Computer Science* mostram muitos trabalhos com perspectivas diversas sobre o tema, sendo alguns somente sobre a perspectiva de desenvolvimento ágil e outras com a integração de princípios ágeis com outras metodologias, como por exemplo com a *User-Centered Design* (UCD), ou *Design* Centrado ao Usuário, que visa colocar as pessoas no centro da atenção ao

desenvolver um produto ou serviço por meio de sua participação integral no processo de desenvolvimento, havendo atividades de entrevistas, observações e testes, reduzindo o trabalho de adivinhar o comportamento do usuário (KALBACH, 2009).

Entretanto, a maioria dos artigos mais citados dessa revista aborda somente sobre desenvolvimento ágil de software, indicando o grande interesse sobre essa temática.

#### 4.1.2.3 Evolução do tema ano a ano

A evolução do tema ano a ano indica a constatação de novos métodos, abordagens e práticas que foram criadas para desenvolver softwares de forma ágil. Ao observar o gráfico, representado pela Figura 5, percebe-se que entre os anos 2000 a 2009 as publicações sobre desenvolvimento ágil de software foram crescentes, com queda em 2010 e aumento entre 2012 e 2016.

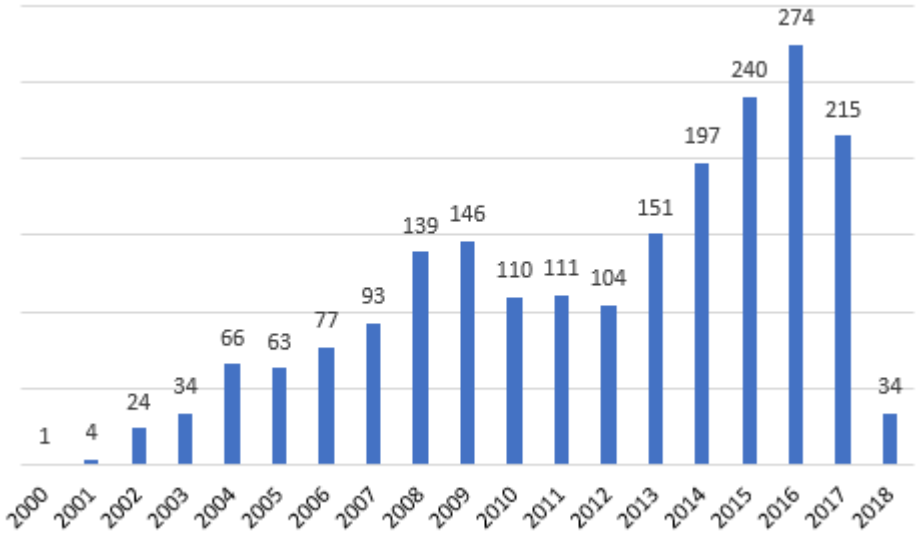



**Figura 5 - Publicação de trabalhos sobre desenvolvimento ágil de software (2000 a 2018)**

Fonte: *Web of Science*

No ano 2000 foi constatada a publicação de apenas um artigo sobre métodos ágeis de software. Esse artigo é denominado "*Service-Based* Software*: The Future for Flexible* Software*",* e já abordava a necessidade de mudança de um método mais focado em dados e processos para um método mais ágil e flexível de desenvolvimento de software nas organizações. O estudo formulou cinco hipóteses sobre como seria o futuro das práticas ágeis

até o ano 2005. Na 1ª hipótese foi proposto que haverá uma mudança no controle do serviço de centros de desenvolvimento de software para clientes e *sites* de usuários; na 2ª hipótese foi proposto que haverá uma mudança de práticas ágeis entre o trabalho de desenvolvedores e de clientes exigindo maior envolvimento entre o time e maior envolvimento do cliente na entrega do sistema; na 3ª hipótese supôs que haverá mudança na visão da qualidade em diversas perspectivas, todas contendo uma abordagem diferente de avaliação; na 4ª hipótese foi afirmado que haverá uma mudança de atitude em relação ao desenvolvimento do software e em relação às práticas de negócios, o que irá proporcionar maior aceitação de novas tecnologias; por fim, a 5ª hipótese propõe que haverá uma mudança da incapacidade de prever um comportamento de serviço para gerenciar a sua complexidade.

No ano 2001, a publicação do Manifesto Ágil trouxe muitas transformações na área de desenvolvimento de Software, pois foram introduzidos novos métodos, técnicas, ferramentas e boas práticas para desenvolver softwares. Nesse ano também há indícios na literatura sobre a evolução de métodos ágeis no Brasil, em que foi colocado que em 2001 professores e praticantes começaram a abordar sobre *Extreme Programming* nas universidades, em organizações de governança em TI e em empresas de indústria de software (MELO et al., 2013).

No artigo de Bennett e Rajlich (2001), o tema foi sobre o contexto da evolução do desenvolvimento de software, em que foi defendida a necessidade das organizações aderirem a métodos mais reativos e a utilizarem novas técnicas e ferramentas para poderem resolver melhor os problemas gerados por mudanças inesperadas. Foi proposto que os Engenheiros de Software precisam pesquisar e estudar sobre desenvolvimento de software a fim de propor melhorias em sua evolução, sendo que um dos critérios a refletir é relacionado à classificação de mudanças, pois a abordagem incremental pode resultar em acréscimos de funcionalidade, mudanças na refatoração que preservam a funcionalidade, mas que melhoram a estrutura, alterações que retraem as funcionalidades, entre outras. Foi citado como exemplo a técnica de localização conceitual, a qual auxilia a localizar em qual parte do programa a mudança será aplicada. Outro exemplo é a de propagação de mudança, na qual quando há mudança em uma parte do software é necessário realizar ajustes nos outros componentes que são interligados. Assim, os autores defendem que antes de realizar qualquer ajuste no software é necessário antes conhecer amplamente os processos e técnicas para implementar as mudanças. Outra perspectiva que os autores defendem é a necessidade de alteração da velocidade de desenvolvimento de software, pois o desenvolvimento lento pode ocasionar perdas de oportunidades de negócios, enquanto que antes o software demandava mais tempo para ser entregue.

Em 2002 já houve um salto de publicações, totalizando 24 trabalhos científicos. Grande maioria buscou informar como são os métodos ágeis, os seus processos e técnicas. Muitos citaram o *Extreme Programming* (XP). Nesse ano também houve uma conferência no Brasil, a *Extreme Programming Brazil 2002*. Alguns anos depois, indústrias, cursos acadêmicos e conferências já criaram a comunidade ágil no Brasil (MELO et al., 2013).

No ano 2005 alguns autores propuseram o equilíbrio de métodos ágeis e tradicionais em projetos. Boehm e Turner (2005) afirmaram que era adequado verificar em que circunstâncias seria mais adequado utilizar cada método para que, em seguida, implementasse o ágil ou o tradicional quando fosse mais apropriado. No artigo *Management challanges to implementing Agile Processes in traditional development organization,* Boehm e Turner (2005) estudaram a implementação de processos ágeis em organizações que utilizam predominantemente a abordagem tradicional. Os resultados do estudo mostraram que há três áreas críticas de gerenciamento conflitantes para introduzir princípios ágeis em organizações que utilizam estritamente princípios tradicionais, e essas são: Desenvolvimento de Software, Processos de Negócios e Pessoas. A primeira é considerada mais desafiadora, pois os processos ágeis e tradicionais possuem ciclos de vida diferentes, sendo que no primeiro há foco em apresentar funcionalidades a cada entrega, enquanto que no tradicional o foco é em otimizar o processo de desenvolvimento, que é realizado em um longo período. Será necessário ajustar o longo processo de desenvolvimento tradicional para o ágil, alterando, por exemplo, a atividade de documentação. Outra diferença na área de desenvolvimento de software é na forma que cada abordagem utiliza na elicitação dos requisitos do software e nas suas validações. Ao implementar princípios ágeis, será necessário a equipe manter uma relação mais informal com o cliente e serem mais funcionais ao levantar os requisitos do software, além de realizarem validações e testes constantes para detectar falhas e para verificar, a cada entrega, se os requisitos estão conforme o esperado pelo cliente. Em relação à barreira de Processos de Negócios, esta existe na forma que cada método utiliza para conduzir os processos. A forma de calcular estimativas, distribuição de recursos e folgas durante o processo variam entre as abordagens ágil e tradicional. Na ágil é preciso lidar com incertezas e ambiguidades que aparecem ao longo do processo de desenvolvimento do software. Em alguns casos foi constatado o sucesso de aplicar uma forma de mensuração ágil, a *Burndown,* nos processos tradicionais. A terceira área conflitante é a de Pessoas, pois haverá uma alteração da forma de comunicação entre a equipe, tornando-a mais interativa, com foco em equipe, e com mais

contato e trocas de informações com o cliente. Assim, haverá maior número de *feedbacks*, empoderamento individual e flexibilidade.

Em um estudo realizado por Mazuco (2017), foi identificada redução da utilização do método XP ao realizar um levantamento sobre a preferência de métodos ágeis nos anos 2008, 2012 e 2017. Em 2008, o método XP ocupava 76% da preferência, enquanto que Scrum ocupava 3%. Os 21% restantes eram compostos por outras metodologias ágeis. Em 2012, Scrum e outros métodos ultrapassaram significativamente a preferência por XP, sendo que a preferência por XP, Scrum e outros métodos foram, respectivamente, 7,3%, 51,1% e 41,6% (MELO et al., 2012). Em 2017, Mazuco (2017) identificou maior balanceamento de preferência em métodos, sendo que XP era preferência de 16,7%, Scrum era de 44,4% e outros métodos eram 38,9%, sendo que "outros métodos" eram majoritariamente compostos por métodos tradicionais, seguidos por Kanban, *Feature-Driven Development* (FDD) e *Lean* Software *Development*. Sendo assim, esses resultados apresentam que houve aumento da adoção do Scrum nos últimos anos e, também, na adoção de outros métodos ágeis que não o XP e o Scrum (MAZUCO, 2017).

#### 4.1.2.4 Autores que mais publicaram vs Autores mais citados

Os autores que mais publicaram sobre desenvolvimento ágil de software e os que foram mais citados foram identificados. A Figura 6 apresenta os 31 autores que mais publicaram e os seus respectivos quantitativos de publicação. Segundo Guedes (2012), analisar os autores que foram mais citados permite identificar as possíveis frentes de pesquisa de uma determinada área científica.

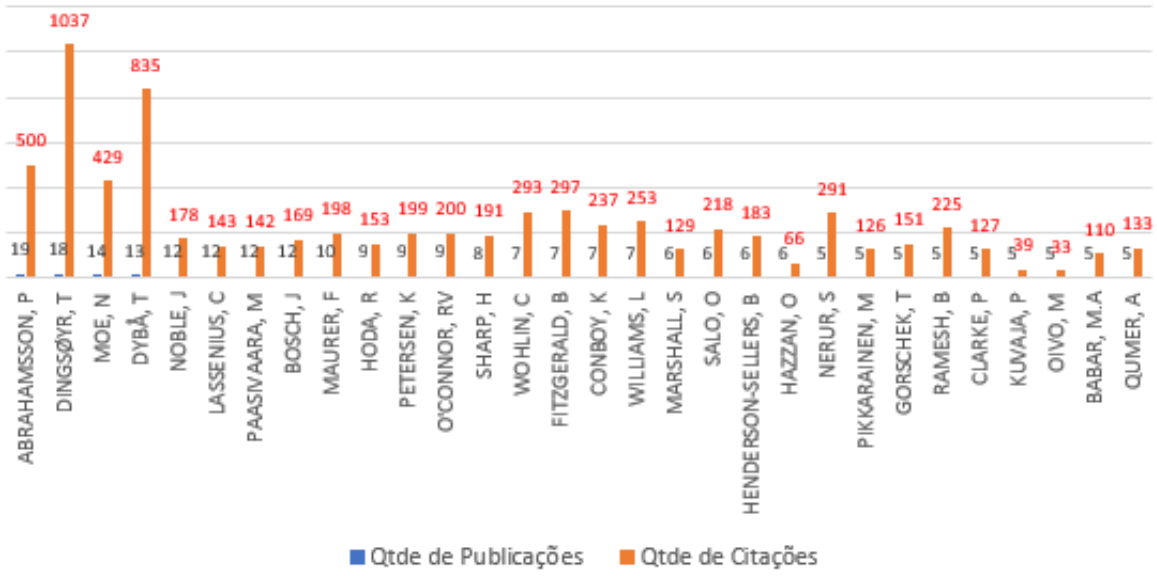



**Figura 6 – Autores mais citados sobre desenvolvimento ágil de software**

Fonte: da própria autora

Abrahamsson foi o autor que mais publicou sobre o tema e foi o sétimo autor mais citado. Os seus trabalhos abordam muito sobre comparações entre os métodos ágeis de desenvolvimento de software existentes e, também, aborda estudos sobre aplicação de métodos ágeis em uma organização que utiliza métodos tradicionais, como por exemplo no estudo de caso na Nokia, em que foi aplicado questionário para verificar a satisfação na mudança de adoção de método tradicional para ágil. O autor também realizou estudos com a temática sobre comunicação em projetos ágeis, contendo publicação de artigo sobre o impacto positivo das práticas ágeis na comunicação do desenvolvimento de software, e outro artigo que aborda o impacto da comunicação com o cliente no desenvolvimento ágil de software. O método ágil mais presente em seus estudos é o XP, porém publicou um trabalho sobre como estava sendo o uso dessa metodologia e do Scrum em 2008. Outra publicação foi sobre o impacto de Programação em Pares, prática do método XP, na qualidade do produto.

Torgeir Dingsøyr foi o segundo autor que mais publicou e, dentre todos, foi o mais citado. Publicou o artigo mais citado sobre desenvolvimento ágil de software, conforme abordado na Tabela 8, item 4.1.2.5. Em seus artigos ele apresenta a importância dos fatores sociais, tais como o trabalho em equipe, no desempenho do projeto de desenvolvimento do software. Muitos de seus artigos foram publicados juntamente com Nils Brede Moe e Tore Dybå, que são, respectivamente, o terceiro e o quarto autores mais citados, e trabalham com linhas similares de pesquisa, conforme apresentado no item 4.1.3.3.

Nils Brede Moe foi o terceiro autor que mais publicou sobre o tema desenvolvimento ágil de software. Este autor também possui muito interesse na temática sobre o impacto do trabalho em equipe no desempenho positivo software produzido. Dentre os métodos ágeis que mais aborda em seus artigos, destacam-se o Scrum e o XP.

#### 4.1.2.5 Documentos mais citados

Os artigos mais citados sobre desenvolvimento ágil de software foram identificados por meio da utilização das *strings "*Software *Development"*, ou Desenvolvimento de Software, *AND "Agile"*, ou Ágil. O número de citações atribui importância aos documentos à medida que são citados por outros autores.

A Tabela 8 apresenta o título dos cinco artigos mais citados, com o ano e o total de citações de cada um.

**Tabela 8 - Artigos mais citados sobre desenvolvimento ágil de software**

| Ordem de Citação | Título | Autores | Ano de Publicação | Total de citações |
|---|---|---|---|---|
| 1 | Empirical studies of agile software development: A systematic review | Dybå, T.; *Dingsøyr*, T. | 2008 | 542 |
| 2 | Agile software development: The business of innovation | Cockburn, A; Highsmith, J | 2001 | 253 |
| 3 | Agile software development: The people factor | Cockburn, A; Highsmith, J | 2001 | 222 |
| 4 | A survey study of critical success factors in agile software projects | Chow, T.; Cao, D. | 2008 | 190 |
| 5 | A decade of agile methodologies: Towards explaining agile software development | Dingsøyr, T.; Nerur, S.; Balijepally, V.; Moe, N. | 2012 | 183 |

Fonte: *Web of Science*

O artigo mais citado se trata de uma revisão sistemática realizada com trabalhos publicados entre os anos 1996 a 2005 e citou os benefícios e as limitações encontradas no desenvolvimento ágil de software, buscando agregar conhecimento para a indústria de software e para a comunidade de pesquisa. Foram selecionados e analisados 36 estudos científicos e foi constatado que o método ágil mais abordado foi o XP, seguido por *Scrum* e *Lean* Software *Development*, ambos encontrados em um artigo. Foi constatado que grande parte dos artigos tiveram como assuntos recorrentes a questão dos fatores sociais e humanos, sobretudo em como eles exercem influência nos métodos ágeis de desenvolvimento, e como são afetados por esses. Habilidades interpessoais e confiança foram apontados como importantes características de

sucesso para um time que adota a metodologia XP. Em relação aos benefícios, foram reportadas melhorias nas perspectivas de colaboração com o cliente, de processos de trabalho para gerenciamento de defeitos, de aprendizado em programação em pares, de visão sobre gerenciamento, de foco nos trabalhos atuais para engenheiros de software e de estimativas do processo. Além disso, alterou a estrutura das organizações que eram hierárquicas e passaram a ser menos centralizadas. Muitos estudos sobre a aplicação de método XP indicaram melhorias, pois constatou-se que houve sucesso nos resultados ao alterar a autonomia individual para a conjunta, pois fez com que aumentasse a responsabilidade corporativa e a crença nas habilidades e na qualidade do trabalho de cada um. O envolvimento do cliente e os fatores humanos trouxeram resultados de sucessos nos projetos dos times que adotaram os métodos XP, e essa prática é possível de ser utilizada por diversas organizações. Outro ponto interessante apontado nos resultados foi a questão dos clientes se mostrarem satisfeitos pela oportunidade de darem *feedbacks* e melhor resposta a mudanças. Além disso, os desenvolvedores de software se mostraram mais satisfeitos com as práticas XP.

Em relação às limitações dos métodos ágeis, o artigo apontou que as técnicas no desenvolvimento *Lean* Software *Development* não funcionaram de forma esperada para um dos times que tentou aplica-las, e também que a programação em pares se mostrou ineficiente. Outro estudo analisado no artigo afirmou que a programação em pares se mostrou exaustiva por requerer concentração pesada e, também, apontou que é difícil de funcionar quando há grande diferença de habilidades entre os membros dos pares que trabalham juntos. Também foi encontrada a dificuldade de introduzir a metodologia XP em uma organização complexa, sendo mais fácil de ser introduzida em outros tipos de organizações. Isso traz de encontro a Mann e Maurer (2005), que sugerem que métodos ágeis de desenvolvimento são mais apropriados para times pequenos do que para projetos maiores. Entretanto, os autores afirmaram que os métodos ágeis, mesmo sendo difíceis de serem implementados em organizações complexas, trazem bons resultados, tais como aumento de produtividade, de satisfação do cliente e de satisfação da equipe com o projeto.

Nos resultados também foram apresentadas as constatações dos artigos que compararam métodos tradicionais e ágeis de desenvolvimento. Foram encontradas diferenças nas práticas de gerenciamento de projetos de ambas. Alguns estudos sugeriram que no desenvolvimento ágil as mudanças são incorporadas mais facilmente e que o valor do negócio é demonstrado de forma mais eficiente. Foi visto também que é possível combinar o gerenciamento ágil e tradicional de projetos. Karlström e Runeson (2005), ao verificarem essa questão em seu

trabalho, concluíram que foi benéfico utilizar a prática ágil de interações com o cliente, pois puderam se comunicar de forma mais eficiente do que se estivessem escrevendo documentos relativos ao software, uma vez que nessas interações era possível reportar o progresso do desenvolvimento do software e puderam ter maior controle do projeto. Como prática tradicional, foi utilizado o modelo de gerenciamento de projeto *stage-gate* para melhorar o controle de custos, da funcionalidade do produto e do cumprimento do prazo de entrega.

O segundo artigo mais citado foi publicado em 2001, ano de divulgação do Manifesto Ágil, e o conhecimento sobre práticas ágeis, portanto, estava começando a ser disseminado. Foram apresentadas as práticas comuns dos métodos ágeis de desenvolvimento de software e as suas características em alguns métodos, tais como XP, métodos *Crystal*, Desenvolvimento *Lean*, *Scrum* e *Adaptative* Software *Development* (ASD). O objetivo era auxiliar em como lidar da melhor forma com as mudanças inevitáveis que ocorrerem ao longo do ciclo de vida de desenvolvimento do software. Highsmith e Cockburn (2001) citaram que as abordagens tradicionais, quando bem trabalhadas, tornam possíveis a listagem completa dos requisitos antecipadamente e a redução de custo ocasionado pelas alterações. Contudo, para o desenvolvimento tradicional de software, essas variações são vistas como resultados de erros. Além disso, na abordagem tradicional são realizadas mensurações contínuas, identificações de erros e refinamento do processo de desenvolvimento de software. Cockburn e Highsmith (2001) afirmam que, como mudanças ao longo do desenvolvimento são inevitáveis e que não podem ser eliminadas, reduzir o custo de responder a elas é a única estratégia viável. Assim, ao invés de eliminar o retrabalho, a nova estratégia seria reduzir o seu custo. Ainda afirmam que é importante haver a preocupação em reter qualidade, pois a demanda do mercado e o aumento da inovação trazem essa exigência ao software. O artigo aponta as características comuns dos métodos ágeis, como por exemplo a estratégia de reduzir os custos ocasionados por mudanças ao longo do projeto. No método XP, por exemplo, é proposto que a equipe de desenvolvimento de software produza a primeira entrega em semanas para obter rápido *feedback,* além de também inventar pequenas soluções para os problemas que ocorrerem.

Dessa forma, haverá menos pontos a serem alterados e as mudanças podem ser realizadas mais facilmente. No XP também é proposto a melhora contínua na qualidade do design do software, reduzindo o custo de implementá-lo na próxima estória. Há também o teste contínuo, que visa identificar os erros antecipadamente e minimizar os custos para consertá-los. Os autores do artigo também informaram que cada método ágil aborda a qualidade de uma maneira No *Dynamic Systems Development Methodology (DSDM),* por exemplo, há uma série de

protótipos para atacar áreas instáveis ou desconhecidas, tais como novas tecnologias, novas regras de negócios e o design de interface com o usuário. Já no Scrum há a proposta de realizar reuniões diárias, de aproximadamente 15 minutos, e iterações de duas a quatro semanas para revisar o produto do projeto. Os autores defendem as constantes iterações com o cliente e com o patrocinador, pois assim torna mais fácil a exposição das dificuldades encontradas e o ajuste das prioridades. Cockburn e Highsmith (2001) apontaram também as práticas ágeis, tais como o planejamento de recursos e a priorização dinâmica, *feedback* e mudanças e foco no trabalho em equipe, e citaram algumas diferenças que os métodos ágeis possuem sobre elas. Em relação ao planejamento de recursos e a priorização dinâmica foi enfatizado que o desenvolvimento ágil prega que sejam realizados por meio dos ciclos de iteração, pois assim a equipe pode utilizar as reuniões para tomar decisões e priorizar tarefas e requisitos. No XP é recomendado realizar iterações em ciclos curtos de duas a três semanas, enquanto que no Scrum é recomendado realizar de duas a quatro semanas. Nos métodos *Crystal* e no ASD maior variação desse intervalo é tolerada. O planejamento de recursos tem maior prioridade que o planejamento de tarefas pelo fato de ser de maior interesse do cliente. A priorização dinâmica é realizada para que, ao final da iteração, o cliente possa remanejar os recursos desejados para o próximo ciclo de iteração, podendo descartar ou acrescentar novos elementos. Em relação às práticas ágeis, os autores citaram as mais utilizadas pelos principais métodos ágeis: no XP é a estória de usuários, ou *User Stories*, e a Programação em Pares; no *Scrum* é o *Backlog*, sendo que as mudanças só podem ser realizadas ao final de cada iteração, e não durante; e no DSDM é utilizada a regra MoSCoW (*Must have, should have, could have, want to have sometime*) para planejar os recursos e os ciclos curtos com o uso de prototipagem.

O terceiro artigo mais citado também foi publicado por Cockburn e Highsmith (2001). Nele é abordado que os fatores humanos impulsionam o sucesso dos projetos que seguem os métodos ágeis. Os exemplos de fatores humanos são: talento, habilidades pessoais e comunicação. A melhoria das competências individuais dos membros do time ágil é considerada como um fator crítico de sucesso, pois assegura que cada pessoa conseguirá agregar mais valor nas entregas com o tempo. Além disso, os membros trabalhando conjuntamente com boa comunicação podem agregar muito mais que quando trabalham individualmente. Entretanto, foi abordado que pode ter falha na implementação dos métodos ágeis nas organizações que possuem processos centrados e também nas organizações em que não há muita colaboração. O artigo citou uma pesquisa realizada pelo *Standish Group*, na qual foi apresentada uma receita de sucesso que continham dez itens. Os três primeiros itens eram: suporte dos executivos,

envolvimento do usuário e experiência em gerenciamento de projetos. Há distinção entre comunicação e colaboração: Cockburn e Highsmith (2001) afirmam que comunicação é o envio e o recebimento de informação, enquanto que colaboração consiste em trabalhar ativamente e conjuntamente a fim de entregar um produto ou realizar uma decisão.

O quarto artigo mais citado aborda os fatores de sucesso em projetos de desenvolvimento ágil de software. Consistiu em uma pesquisa que utilizou métodos quantitativos, em que coletou dados de 109 projetos ágeis de software de diversas organizações, tamanhos, tipos de indústrias e localizações geográficas. Após realizar uma análise múltipla por regressão, os únicos fatores de sucesso encontrados foram: estratégia para realizar entregas corretas, a utilização apropriada de uma prática ágil da engenharia de software e um time altamente equilibrado. Outros três fatores que poderiam ser críticos para o sucesso em outras dimensões seriam: um bom gerenciamento ágil do processo, um ambiente ágil e amigável e envolvimento do cliente. O artigo enfatiza, assim, que para que o projeto obtenha sucesso, a equipe deve focar no gerenciamento do projeto quando a equipe adota métodos ágeis em seus projetos de desenvolvimento de software.

O quinto artigo mais citado sobre desenvolvimento ágil de software foi publicado em 2012 e buscou estudar os trabalhos científicos publicados ao longo de uma década a fim de verificar o progresso das pesquisas sobre métodos ágeis desde a publicação do Manifesto Ágil, em 2001. Foi verificado que de todos os métodos ágeis, o XP foi o mais presente nos trabalhos analisados. Foi verificado também que os métodos ágeis já estavam significativamente disseminados nas organizações de diversos países, sendo mais presente nos Estados Unidos, Canadá, Alemanha, Finlândia e Reino Unido. Dentre as perspectivas teóricas encontradas nas pesquisas analisadas sobre métodos ágeis, a mais estudada foi sobre conhecimentos em gerenciamento de projeto. Outras perspectivas teóricas mais populares foram personalidade e aprendizado organizacional. Foi citada a teoria de personalidade Big Five, que se mostra importante para explorar as dinâmicas interpessoais de programação em pares, técnica presente no método XP. Os princípios ágeis de adaptabilidade e de resposta a mudanças podem ser desenvolvidos durante a aprendizagem organizacional do desenvolvimento ágil.

#### 4.1.2.6 Países que mais publicaram sobre desenvolvimento ágil de software

A revisão sistemática da literatura permitiu examinar também os países que mais publicaram sobre desenvolvimento ágil de software. Na perspectiva de software, assim como qualquer atividade humana, a sua forma de desenvolvimento é altamente influenciada de acordo com a

cultura da região ou do local (MISRA; KUMAR; KUMAR, 2009). Apesar dos autores que criaram cada método ágil compartilharem práticas e filosofias similares, algumas características de suas metodologias são diferentes. Cockburn e Williams (2003) colocaram as regiões nas quais alguns métodos ágeis foram criados: *Dynamic Systems Development Method* foi desenvolvido na Europa, enquanto que *Feature-Driven Development* foi criado na Austrália. Já *Extreme Programming*, *Crystal*, *Adaptative* Software *Development* e *Scrum* foram criados nos Estados Unidos.

Foram obtidos 93 registros de países e o gráfico da Figura 7 apresenta os que mais publicaram sobre o tema no período dos anos 2000 a 2018. Constatou-se que os Estados Unidos (*USA)* lideraram a quantidade de publicações, totalizando 15,79% (329) publicações. Conforme apresentado, este é um país que criou muitos métodos ágeis e é natural que tenha realizado mais publicações sobre o tema.

O segundo país que mais publicou sobre Métodos Ágeis foi a Finlândia (*Finland*), totalizando 6,91% (144) trabalhos científicos, e o terceiro país que mais publicou trabalhos científicos foi a Alemanha (*Germany*), totalizando 6,82% (142) trabalhos.

Outra constatação importante foi o fato do Brasil ser o quarto país que mais publicou sobre desenvolvimento ágil de software, totalizando 138 trabalhos científicos. A aplicação de princípios ágeis no Brasil tem crescido significativamente.

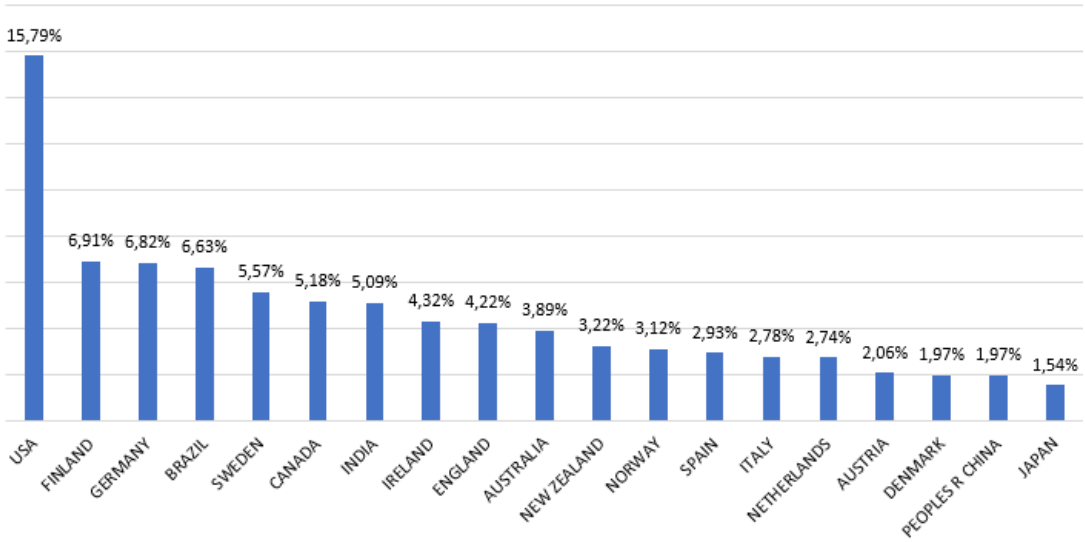



**Figura 7 - Países que mais publicaram (2000 a 2018)**

**Fonte:** *Web of Science*

Em uma pesquisa realizada em 2001 pelo *Cutter Consortium*, cerca de 200 pessoas de diversos países, tais como Austrália, Índia e outros países localizados na América do Norte e na Europa responderam a um questionário sobre o uso de métodos ágeis e tradicionais em suas organizações. Os resultados da pesquisa mostraram que a maioria dessas pessoas utilizavam pelo menos um método ágil em suas organizações, mostrando que foi rápida a adesão dos métodos ágeis em muitos países desde o início.

No artigo "*A decade of agile methodologies: Towards explaining agile* software *development",* publicado em 2012, foi verificado que o Canadá foi o segundo país que mais publicou sobre o tema, totalizando 110 trabalhos científicos, e que a Finlândia ocupava a quarta posição, com 94 trabalhos científicos. Atualmente, analisando os dados obtidos neste presente trabalho, foi verificado que a Finlândia investiu mais em pesquisas sobre o tema e passou a ocupar o segundo lugar. Outro dado verificado foi que, em 2012, o Brasil ocupava o décimo terceiro lugar no *ranking*, com 29 publicações, e atualmente ocupa o quarto lugar. A Alemanha continuou na terceira posição do *ranking*.

De acordo com a Lei do 80/20, pode-se afirmar que Estados Unidos e Finlândia representam 22,71% das publicações, abrangendo 80% do conteúdo publicado sobre desenvolvimento ágil de software.

#### 4.1.2.7 Idiomas Predominantes

O inglês é o idioma predominante, compondo 2052 trabalhos, ou 98,51% das publicações. A língua portuguesa ocupou o segundo lugar, mesmo o Brasil sendo o quarto país que mais publicou sobre desenvolvimento ágil de software. As publicações nesse idioma corresponderam 0,72%, do total contemplando 15 artigos. Esse resultado é positivo por indicar que a pesquisa e o interesse do Brasil sobre o tema estão aumentando, havendo artigos que estão sendo publicados em conferências nacionais e internacionais.

Apesar da Finlândia ter sido o segundo país que mais publicou sobre o tema, suas publicações foram majoritariamente realizadas em língua inglesa, não havendo constatação de documentos escritos na língua finlandesa.

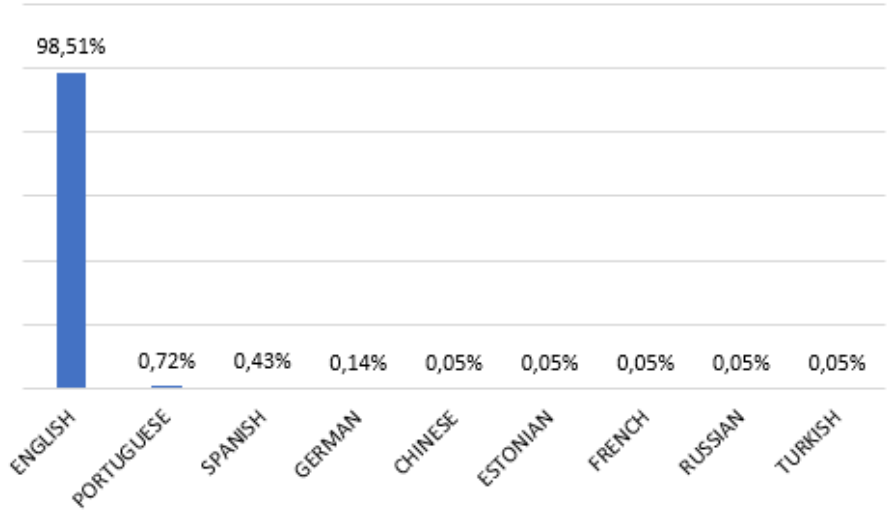



**Figura 8 - Idiomas dos artigos sobre desenvolvimento ágil de software**

Fonte: Web of Science

#### 4.1.2.8 Conferências que mais contribuíram

As principais conferências que mais contribuíram para a disseminação de conhecimento em desenvolvimento ágil de software foram identificadas. Foram obtidos 519 registros de Conferências, sendo que as 12 principais estão contempladas na Figura 9. Grande parte das conferências tem como foco a Engenharia de Software e processos de software. O método XP também é muito presente no título das conferências.

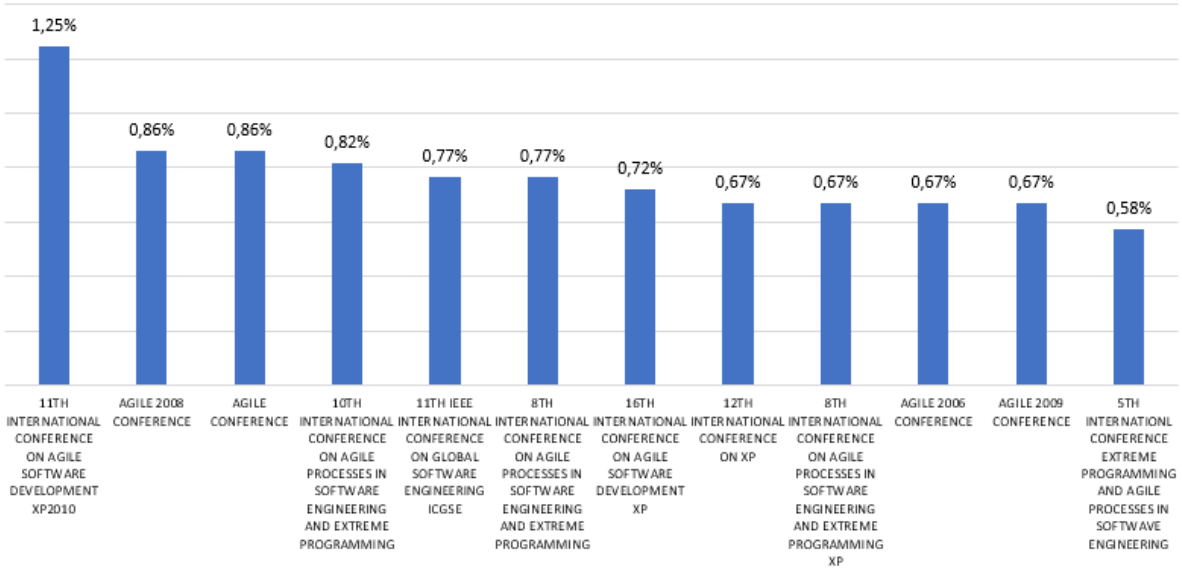



**Figura 9 - Conferências que mais contribuíram**

Fonte: Web of Science

A Lei de Bradford afirma que, se os periódicos científicos forem ordenados de forma decrescente quanto à produtividade de documentos relacionados a um tema, esses podem ser definidos como o núcleo de periódicos mais particularmente dedicados a este assunto em vários grupos ou zonas (BROOCKES, 1969).

Dessa forma, pode-se inferir que as conferências apresentadas na Figura 9 contemplam o núcleo de periódicos mais dedicado ao tema de desenvolvimento ágil de software. A principal conferência, denominada *11th International Conference on Agile* Software *Development XP 2010*, foi responsável por 1,25%, ou 26 registros, de artigos relacionados ao desenvolvimento ágil de software com enfoque em método XP, apresentando diferença significativa em relação às demais.

#### 4.1.2.9 Universidades que mais publicaram

As universidades que publicaram sobre desenvolvimento ágil de software foram identificadas, contemplando 1.615 no total. A Figura 10 apresenta as 19 universidades que mais publicaram sobre o tema.

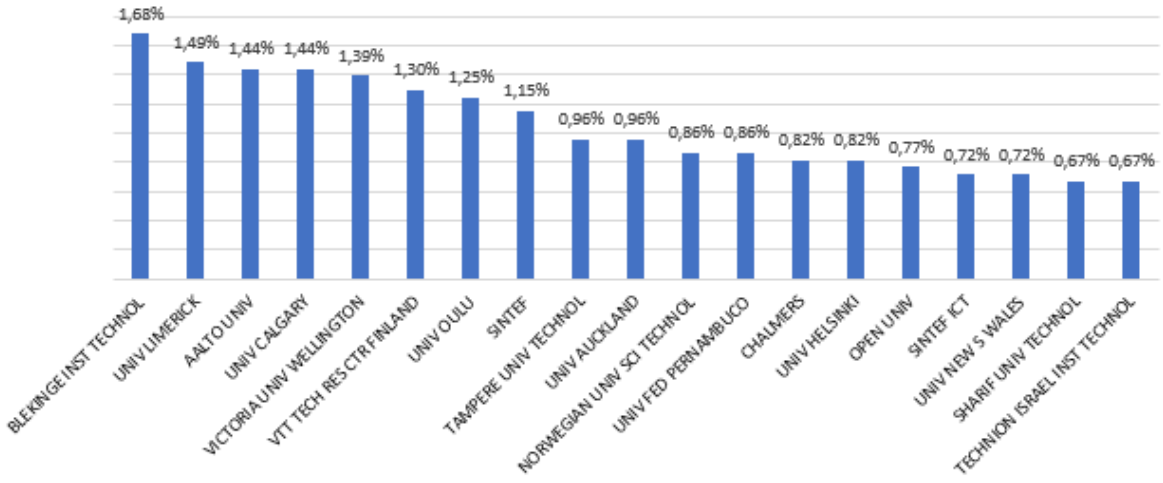



**Figura 10 – Universidades que mais publicaram**

Fonte: *Web of Science*

Somando o percentual de publicação dessas universidades é verificado que corresponde a 19,97% do total de publicações encontradas. Esse resultado representa a relevância dessas instituições no que concerne ao tema, conforme enunciado da Lei Bibliográfica dos 80/20, que afirma que em sistemas de informação, apenas 20% do conjunto de fontes de informação satisfazem 80% da demanda.

A Universidade Federal de Pernambuco ocupou a 12ª posição, com 18 registros, e a Universidade de São Paulo (USP) ocupou a 23ª posição do *ranking*, com 13 registros.

#### 4.1.2.10 Agências que mais financiam a pesquisa

As agências financiadoras que mais fomentaram a pesquisa e a divulgação científica sobre desenvolvimento ágil de software foram identificadas. Foram encontrados 1.777 registros de agências financiadoras e as que mais três que apresentaram maior índice de patrocínio foram: *Science Foundation Ireland*, *Research Council of Norway* e CNPQ.

**Tabela 9 - Três principais agências financiadoras**

Fonte: Web of Science

| ***Rank*** | **Agências financiadoras** | **Registros** | **Porcentagem** |
|---|---|---|---|
| 1 | Science Foundation Ireland | 17 | 0,82% |
| 2 | Research Council of Norway | 14 | 0,67% |
| 3 | CNPq | 12 | 0,58% |

Foram encontradas mais de 15 agências financiadoras brasileiras. O Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) ficou em terceiro lugar entre as principais agências financiadoras, com 12 estudos financiados. É interessante observar o elevado incentivo do Brasil na busca do conhecimento e da aplicação de desenvolvimento ágil de software.

#### 4.1.2.11 Áreas que mais publicam

Ao realizar a pesquisa das áreas que mais publicam sobre o tema, foram obtidas 36 áreas de pesquisa. Analisando as 24 áreas de pesquisa mais relevantes, apresentadas na Figura 11, pode-se inferir que a área de pesquisa que possui maior interesse significativo sobre o tema é a *Computer Science* (Ciência da Computação), contemplando 1.974 trabalhos publicados. Em seguida, a segunda área de pesquisa que apresentou maior interesse foi a de Engenharia (*Engineering*), com 487 trabalhos.

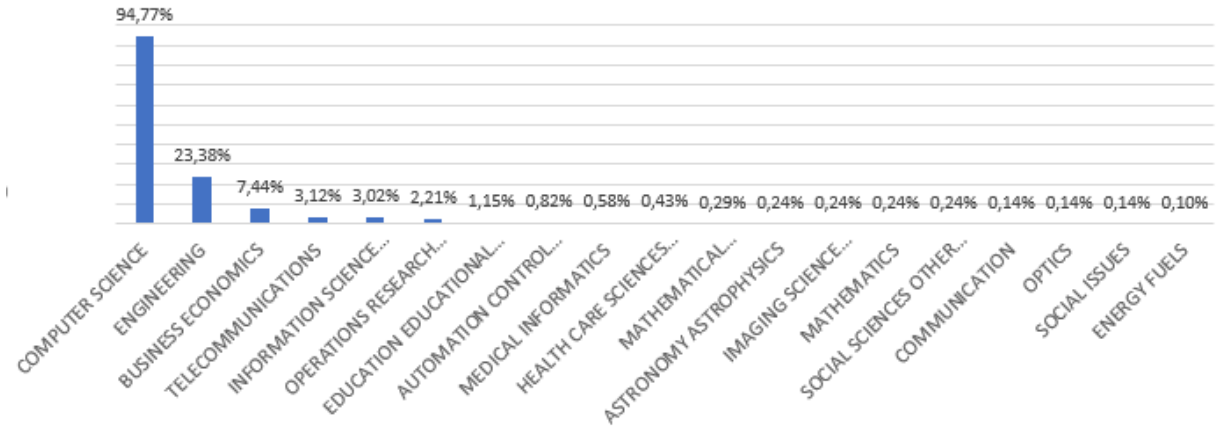



**Figura 11 – Principais áreas de pesquisa sobre desenvolvimento ágil de software**

Fonte: *Web of Science*

A área de pesquisa Engenharia foi analisada a fim de verificar quais são as maiores áreas de interesse nela contemplada sobre o tema. A Figura 12 apresenta as 13 principais, sendo que foram constatadas 56 áreas de interesse no total. A *Computer Science* Software *Engineering* foi a mais significativa, contendo 61,26%, com 1276 artigos publicados.

Analisando as outras áreas é possível constatar que a Ciência da Computação constitui a maior área de interesse sobre o tema.

As áreas de interesse *Management*, *Business* e *Engineering Industrial* estão dentro das maiores áreas de interesse na Engenharia, podendo indicar aumento futuro dos trabalhos publicados nessas áreas.

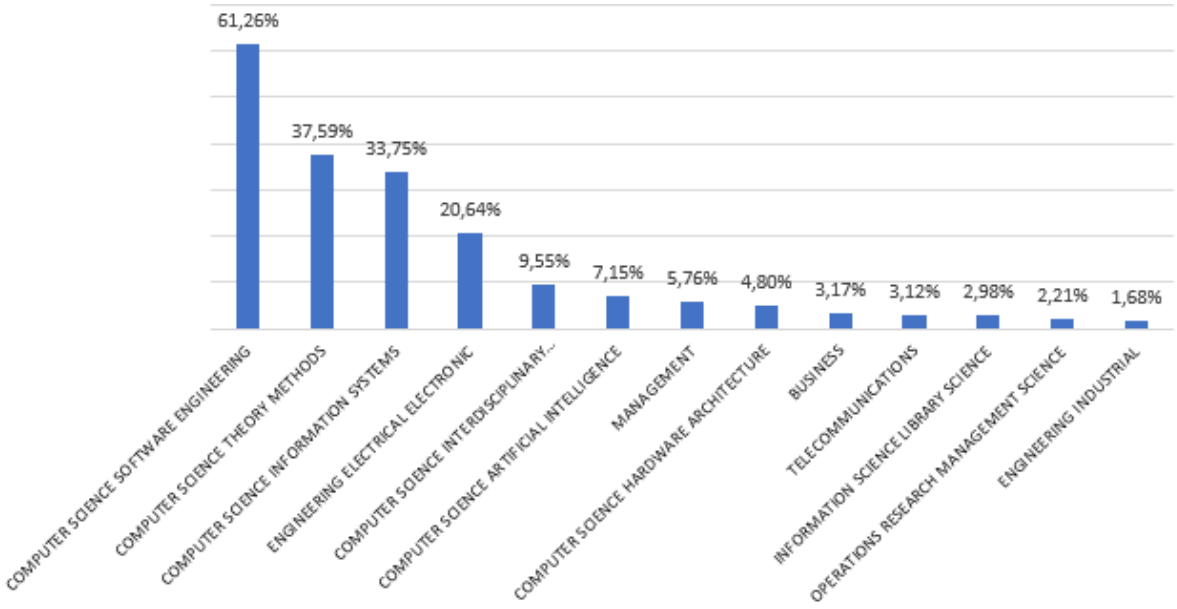



**Figura 12 - Principais áreas de interesse contempladas em Engenharias**

Fonte: *Web of Science*

#### 4.1.2.12 Frequência de palavras-chave

A análise das palavras chaves dos artigos é importante para verificar as linhas de pesquisa dentro de seu conjunto e, também, para verificar a evolução do tema em questão (MARIANO; GARCÍA CRUZ; ARENAS GAITÁN, 2011).

Assim, a fim de representar visualmente as palavras-chave mais citadas nos trabalhos relativos a desenvolvimento ágil de software, foram utilizados a ferramenta online de análise de conteúdo *TagCrowd (http://tagcrowd.com)* e o software *VOSViewer*, a fim de se obter os mapas de calor.

O resultado obtido pelo TagCrowd está apresentado na Figura 13, que apresenta as cinquenta palavras chave mais utilizadas pelos 500 registros de artigos mais citados identificados pelo *Web of Science*. As fontes das palavras exibidas são proporcionalmente maiores quando são mais citadas, permitindo a realização de diagnósticos sobre as principais linhas de pesquisa.


**Figura 13 - Frequência das palavras-chaves**

**Fonte:** TagCrowd

As palavras-chaves que são mais utilizadas nos trabalhos sobre desenvolvimento ágil de software são: *development* (desenvolvimento), com 769; software, com 705; *agile* (ágil), com 673; *Project* (Projeto), com 332; *requirements* (requisitos), com 264; *team* (time), com 252; *methods* (métodos), com 244; *process* (processo), com 242.

O mesmo procedimento foi realizado no software *VOS Viewer*. A Figura 14 apresenta as palavras-chaves mais utilizadas nos 500 artigos mais citados sobre desenvolvimento ágil de software e com os filtros de áreas. As distâncias das palavras mostram o nível de similaridade entre os temas.

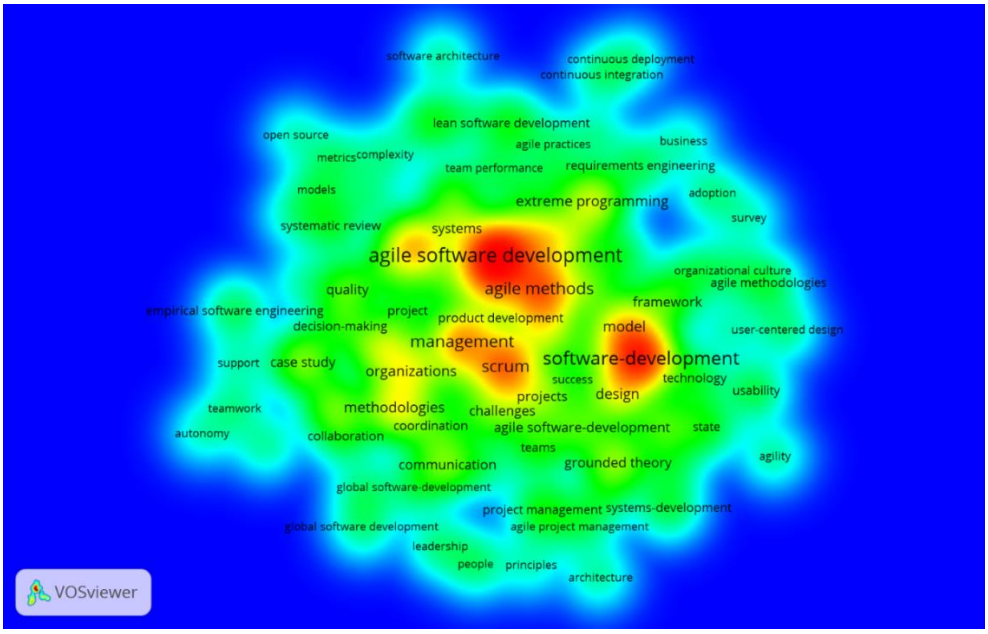



Figura 14 - Análise das palavras-chaves com filtros de área
Fonte: VOS Viewer

É possível verificar que há núcleos no mapa de calor que apresentam as palavras-chaves, indicando a similaridade de tema. As palavras-chaves mais citadas foram: "*agile* software *development* (desenvolvimento ágil de software)*", "*software *development* (desenvolvimento de software)*", "scrum", "management* (gerenciamento*)",* "*agile methods* (métodos ágeis)*", "methodologies* (metodologias)" e *"model* (modelo)".

O desenvolvimento de software, nas últimas décadas, tem sido caracterizado por duas grandes abordagens: o desenvolvimento ágil de software, o qual visa aumentar a rapidez e flexibilidade durante o processo de desenvolvimento, e a abordagem *User-Centered Design* (UCD), ou *Design* Centrado no Usuário (DCU), a qual busca dar mais valor em atingir aos objetivos e necessidades do usuário final do sistema em vez de visar a entrega com software com maior usabilidade (BRHEL et al., 2015).

Muitos modelos foram propostos na literatura para apoiar o desenvolvimento de software. Um dos artigos mais citados nos últimos três anos propôs o modelo híbrido de desenvolvimento de software nomeado "*User-Centered Agile* Software *Development* (UCASD)", ou "Modelo de Desenvolvimento de Software Ágil Centrado ao Usuário", a fim de desenvolver um software com o *design* fácil de ser utilizado e produtivo (BRHEL et al., 2015).

Outro modelo proposto por Shrivastava e Rathod (2015) aborda a combinação entre ágil e *Distributed* Software *Development (DSD),* ou Desenvolvimento Distribuído de Software

(DDS), e foi desenvolvido para produzir o software com mais qualidade e em menor tempo e custo. Essa perspectiva foi proposta para estudar como a abordagem de gerenciamento de riscos do DSD poderia controlar melhor os riscos que afetam os projetos desenvolvidos com princípios ágeis (SHRIVASTAVA; RATHOD, 2015).

Sendo assim, o constante foco em melhorias de processos de desenvolvimento de ágil de software é importante impulsor para a geração de novos modelos que auxiliam o desenvolvimento de software. Durante o movimento ágil, muitos métodos e abordagens foram propostos baseados em outras perspectivas. O Scrum é baseado em princípios do *Lean Manufacturing*, ou Manufatura Enxuta (JAMES, 2009). Uma outra metodologia que foi criada durante o movimento ágil foi o *Kanban*, que se baseou no Sistema Toyota de Produção (OHNO, 1988) e na Manufatura Enxuta (WOMACK; JONES; ROOS, 1991).

Outra palavra muito citada foi "*Scrum*", e consiste em uma metodologia de gerenciamento de projeto para desenvolvimento de software que é iterativa e incremental (LEI et al., 2017). Assim, é natural que esteja espacialmente próxima da palavra "*management* (gerenciamento)" no mapa de calor da Figura 14, uma vez que se trata de uma metodologia de desenvolvimento ágil de software e o seu interesse em pesquisas e em aplicações tem aumentado.

De acordo com Abrahamsson e Salo (2008), o *Scrum* visa fornecer uma abordagem ágil ao gerenciamento de projeto de software, aumentando a probabilidade de seu sucesso, enquanto que o XP se concentra de forma mais aprofundada nas atividades em nível de projeto nas implementações de software.

### 4.1.3 Detalhamento, modelo integrador e validação por evidências

Após identificar e apresentar as primeiras impressões sobre o tema, é necessário realizar análises mais aprofundadas a fim de identificar outros dados importantes. Sendo assim, a terceira, e última, etapa do TEMAC visa buscar outras informações mais detalhadas, tais como os autores que não podem faltar na revisão sistemática, as principais abordagens, linhas de pesquisa, validação por evidências e a entrega do modelo integrador (MARIANO; ROCHA, 2017).

Assim, serão apresentados e analisados os elementos co-citação, *coupling*, co-autoria, o modelo integrador e a sua validação por evidência a fim de assegurar a consistência dos resultados obtidos. Primeiramente será apresentada a co-citação.

#### 4.1.3.1 Co-Citação

A co-citação é uma forma de verificar quais são os pares de artigos que foram citados com certa regularidade em outros estudos (SERRA et al., 2012).

De acordo com (GRÁCIO; OLIVEIRA, 2013), por meio da análise da citação é possível identificar quais são os pesquisadores que possuem maior impacto na área e quais são os seus paradigmas e procedimentos metodológicos pertinentes.

Assim, por meio dessa análise, é possível verificar quais foram as abordagens mais citadas pelos artigos durante determinado período. Dessa forma, a análise de co-citação foi realizada a fim de levantar quais são as abordagens sobre desenvolvimento ágil de software que estão dominando a literatura no período pesquisado.

Primeiramente foram tabulados no software *VOSViewer* os 500 registros dos artigos mais citados sobre desenvolvimento ágil de software, sem filtros por área, obtidos no *Web of Science* no período entre os anos 2000 a 2018. A Figura 15 apresenta o mapa de calor dos resultados obtidos nesta análise. As distâncias das palavras mostram o nível de similaridade entre as abordagens dos artigos.

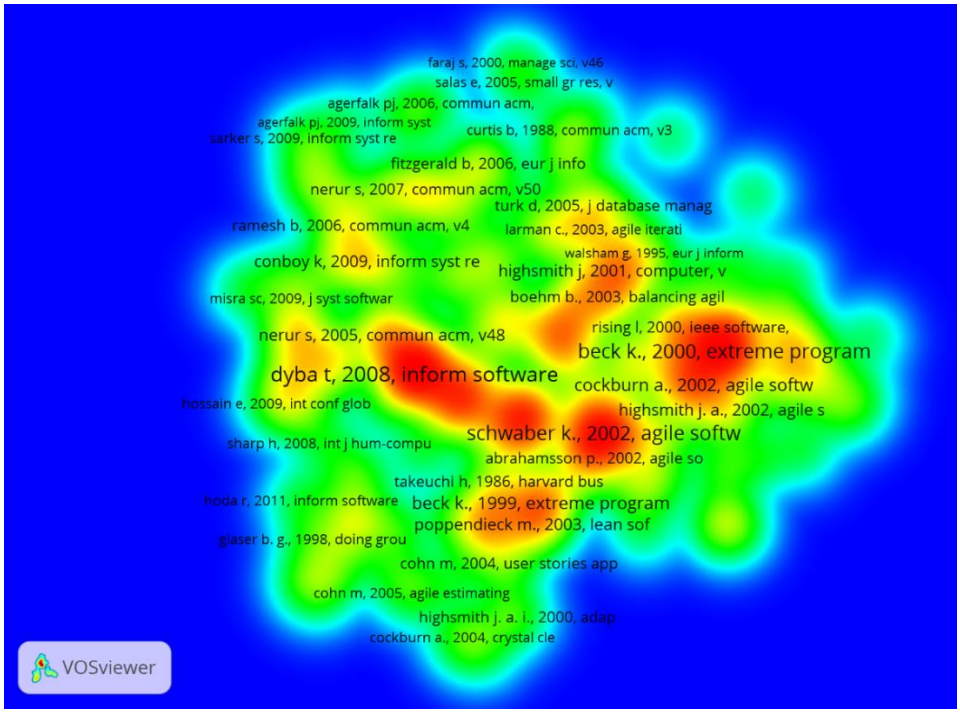



Figura 15 - Mapa de calor de co-citation (sem filtro de área)

Fonte**:** própria autora. Extraído do software *VosViewer* 1.6.5

O mapa de calor de *co-citation*, apresentado na Figura 15, mostra que os trabalhos dos autores Dybå, Schwaber, Beck e Higsmith estão se destacando entre os trabalhos citados pelos

principais autores sobre desenvolvimento ágil de software, podendo inferir que as suas linhas de pesquisa estão abrangendo conteúdo retratam e defendido por muitos outros autores.

Há outros núcleos que também apresentaram relevância e outras perspectivas, tais como os dos artigos de Nerur e Conboy. O artigo de Nerur foi publicado por Nerur, Mahapatra e Mangalaraj (2005) e aborda os desafios para migrar para métodos ágeis. Dentre as barreiras citadas, destacam-se as mudanças dos procedimentos dos processos de trabalho, ferramentas e técnicas, canais de comunicação, estratégias de solução de problemas e fatores de pessoas (NERUR; MAHAPATRA; MANGALARAJ, 2005). As práticas ágeis enfatizam muito a importância de realizar testes, solicitando aos desenvolvedores que elaborem os códigos de teste antecipadamente a fim de valida-los (HIGHSMITH, 2003). Foi citado que, de acordo com Highsmith (2003), os métodos ágeis são ideais para serem utilizados em projetos que possuam alta variabilidade das tarefas em decorrência das mudanças das exigências, que possuam pessoas com diversas habilidades e que apresentam conhecimento das tecnologias a serem utilizadas.

Em seguida, a fim de verificar as abordagens sobre desenvolvimento ágil de software que estão presentes nos artigos mais citados quando se aplicam os filtros de área no *Web of Science*, foram selecionados os 500 registros de artigos mais citados sobre o tema, entre os anos 2000 a 2018, e estes foram tabulados no software *VOSViewer*. A Figura 16 apresenta o mapa de calor de co-citation que foi encontrado após realizar esses procedimentos.

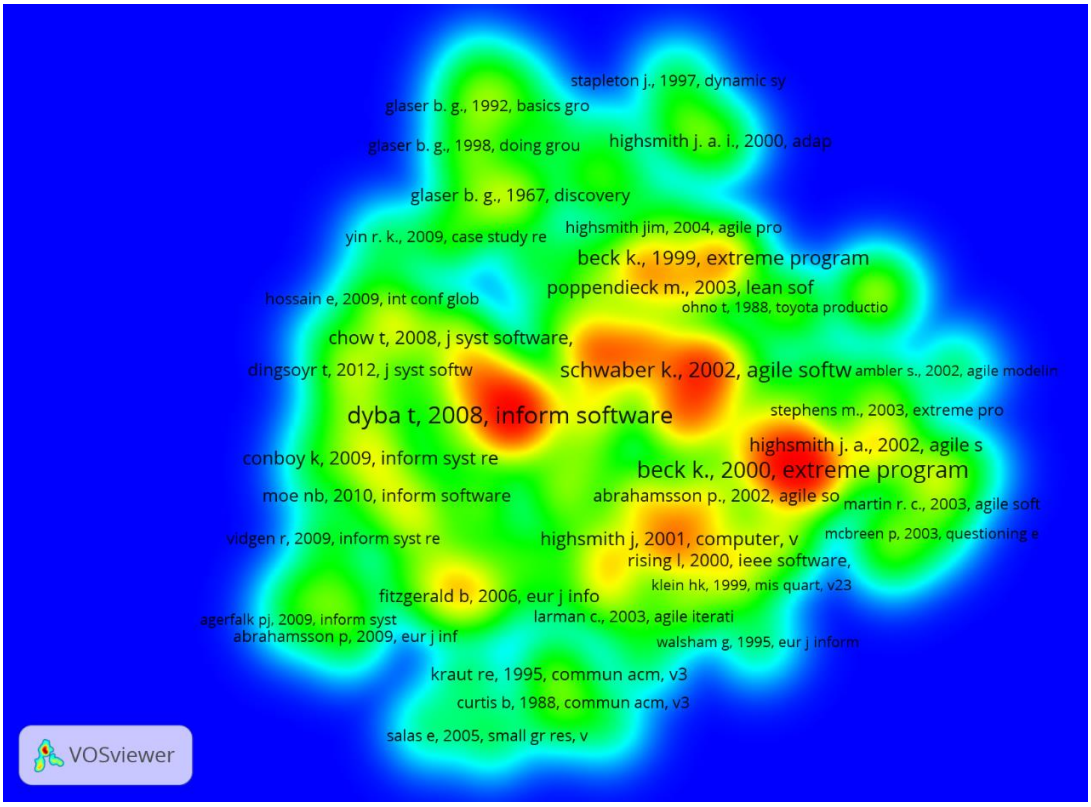



**Figura 16 - Mapa de calor de co-citation (com filtro de área)**

Fonte: VOSViewer

Pela Figura 16 é possível identificar, núcleos do mapa de calor de co-citation com os filtros de área, que se destacam os trabalhos dos autores Dybå, Schwaber, Beck e Cockburn, indicando a importância desses trabalhos para a temática de desenvolvimento ágil de software.

Tore Dybå é um engenheiro de software norueguês, doutor em Ciência da Informação pela *Norwegian University of Science and Technology*. Seus interesses de pesquisa incluem engenharia de software empírica, melhoria de processos de software e desenvolvimento ágil de software (CONRADI et al., 2006). Tore Dybå publicou, juntamente ao autor Torgeir Dingsøyr, o artigo mais citado sobre desenvolvimento ágil de software, o qual foi abordado no artigo mais citado no item 4.1.2.5. Outros trabalhos publicados pelo autor também estão na lista dos artigos mais citados sobre a temática desenvolvimento ágil de software, podendo inferir que é um autor de extrema referência científica sobre o tema. Suas últimas publicações foram no ano 2016, totalizando quatro artigos, sendo que todos abordam a temática de fator humano, tais como o trabalho em equipe e a comunicação, nos processos de desenvolvimento ágil de software. Dentre esses, o artigo mais citado é sobre as reuniões diárias, que consistem em uma das práticas ágeis mais utilizadas, e foram exploradas em seus últimos trabalhos, mesmo tendo sido raro

este tema nas pesquisas científicas. Um dos benefícios identificados na prática ágil de reuniões diárias foi a solução de problemas e a melhoria entre a comunicação interna da equipe. Uma crítica retratada no estudo empírico foi a percepção da alta frequência de reuniões com o tempo longo de duração. Assim, foram propostas sugestões para haver melhor aproveitamento dessa prática ágil.

Ken Schwaber é engenheiro de software e, juntamente com Jeff Sutherland, desenvolveu o *framework Scrum*, em 1993, com o propósito de promover à indústria de tecnologia uma forma mais rápida, confiável e eficiente para desenvolver softwares. O *Scrum* foi definido, formalizado e publicado como o primeiro método ágil de desenvolvimento de software (SUTHERLAND, 2001). Consiste em uma abordagem empírica que aplica conceitos da teoria de controle de processos industriais ao desenvolvimento de sistemas, promovendo maior adaptabilidade, flexibilidade e produtividade (SCHWABER; BEEDLE, 2002). Assim, foi uma importante figura na história do desenvolvimento ágil de software. Ken Schwaber não é um dos autores que mais publicam artigos sobre o tema, porém é possível inferir, pelo mapa de calor de co-citação, que é um autor cujas abordagens possuem extrema relevância no tema de desenvolvimento ágil de software e que seu trabalho contribui para a formulação e adaptação de novos modelos de desenvolvimento de software.

Um dos documentos mais citados foi o livro de Beck e Gamma (2000), no qual é explicado o método ágil *Extreme Programming*. Beck e Gamma (2000) colocam os 12 princípios do método XP e explica as suas raízes, filosofias, histórias e mitos, a fim de auxiliar os desenvolvedores a terem conhecimento sobre quando é adequado, ou não, aplicar o método XP. O método é melhor indicado para ser aplicado quando o time contém dois a dez programadores e que a podem realizar os testes de validação no software em algumas partes do dia. Uma das promessas do XP é acompanhar o progresso do desenvolvimento do software toda semana e poder realizar modificações no projeto sem resultar em custos altos. Com a prática de testes, o método XP visa reduzir os riscos do projeto (BECK; GAMMA, 2000).

Conforme verificado, o método XP foi o método ágil mais utilizado no início do movimento ágil e continua sendo muito aplicado nos projetos de desenvolvimento de software.

#### *4.1.3.2 Coupling*

Também foi realizada uma análise de *coupling (*acoplamento bibliográfico), cujo objetivo é identificar os principais *fronts* de pesquisa na literatura. Para isso, é preciso relacionar os autores

que publicaram artigos nos últimos três anos a fim de verificar quais foram os temas que foram mais discorridos entre eles.

Assim, foram selecionados os registros dos 500 artigos mais citados que foram publicados entre os anos 2015 a 2018, e com filtro por áreas, e estes foram tabulados no software *VOSViewer*. A Figura 17 apresenta o mapa de calor de *coupling* que foi obtido.

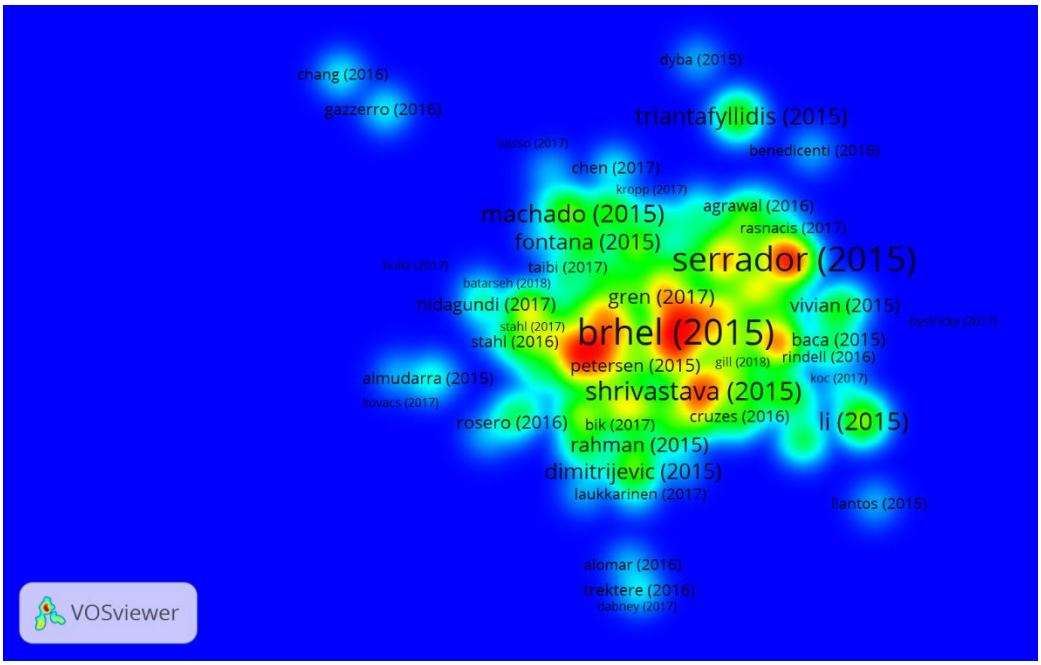



**Figura 17 - Coupling com filtro por áreas**

Fonte: VOSViewer

Os principais *fronts* de pesquisa advêm dos artigos de Brhel, Serrador e Shrivastava. Brhel et al (2015) abordam sobre a combinação entre os princípios ágeis e o *Design* Centrado ao Usuário (DCU), conforme abordado no item 4.1.2.12. O DCU tem como objetivo assegurar que as necessidades e os desejos dos usuários finais sejam prioridade, o foco, do desenvolvimento do produto (BRHEL et al., 2015). Fox, Sillito e Maurer (2008) afirmam que o DCU é desenvolvido com contínua integração com o usuário final, realizando refinamentos e avaliações interativas de conceitos de *design* e protótipos.

O segundo artigo mais evidenciado no mapa de calor de *coupling* foi o de Serrador e Pinto (2015). O artigo consiste em um estudo sobre o sucesso de projetos que utilizaram métodos ágeis, a fim de verificar a performance do método ágil. Os resultados obtidos mostraram, estatisticamente, que a utilização de métodos ágeis causou impacto positivo em três dimensões

dos projetos, tais como: eficiência, satisfação do cliente e percepção da performance do projeto como um todo. Esses resultados foram semelhantes ao trabalho de Budzier e Flyvbjerg (2013), em que ao analisar estatisticamente os dados de projetos de TI e foram encontrados que os métodos ágeis impactaram positivamente no tempo de entrega dos produtos.

O artigo de Shrivastava e Rathod (2015) consistiu em categorizar os fatores de riscos em projetos ágeis distribuídos, que consiste em uma combinação entre a abordagem ágil e a tradicional. Os resultados do estudo indicaram que houve diminuição do custo do projeto e sucesso identificados.

Assim, os *fronts* de pesquisa indicam a tendência em adaptar a abordagem ágil a outros métodos de desenvolvimento de software e a mensurar o sucesso dos projetos de desenvolvimento de software.

#### 4.1.3.3 Co-autoria

A análise de co-autoria revela as colaborações e parcerias científicas e tecnológicas, podendo obter padrões de cooperação entre indivíduos e organizações (MELIN; PERSSON, 1996; GLÄNZEL, 2002; NEWMAN, 2004).

A coautoria de um documento representa uma relação oficial de publicação entre dois ou mais autores ou organizações (GLÄNZEL; SCHUBERT, 2003). Assim, a frequência com que os pesquisadores publicam conjuntamente é verificada nessa análise.

Com a finalidade de compreender a interlocução entre os pesquisadores inseridos na temática de desenvolvimento ágil de software, foi realizada a análise de co-autoria com os registros dos 500 artigos mais citados sobre desenvolvimento ágil de software, conforme apresentado na Figura 18.

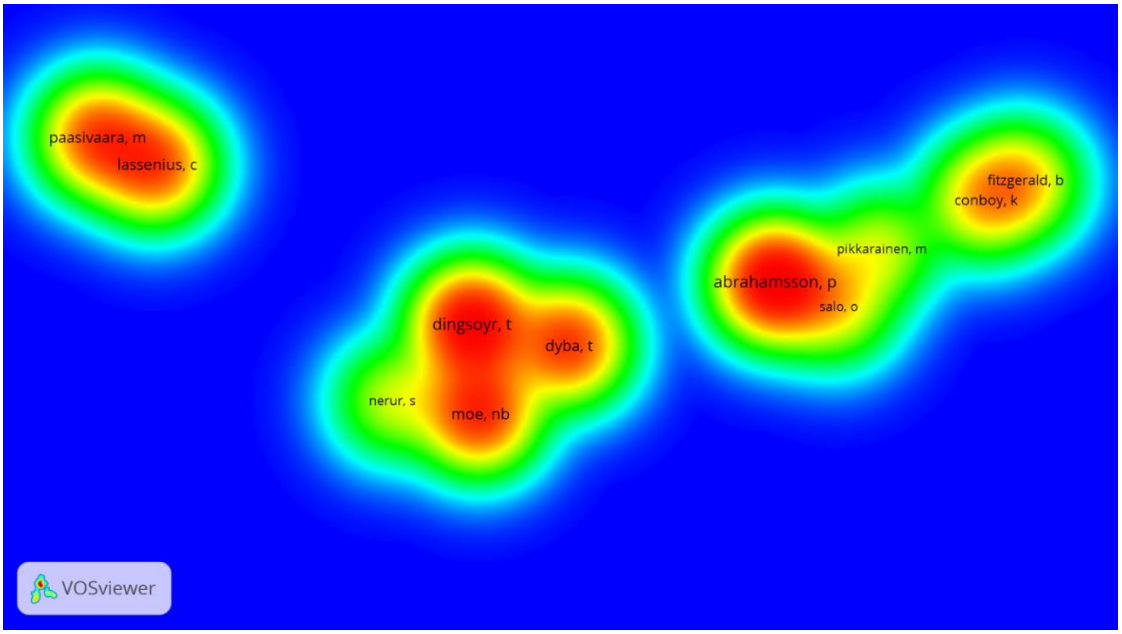



**Figura 18 - Mapa de Calor de Co-autoria**

Fonte: VOSViewer

Foram obtidos quatro *clusters*, com 11 autores que mais publicaram conjuntamente. A união de Torgeir Dingsøyr, Tore Dybå, Nils Brede Moe e Sridhar Nerur, de Pekka Abrahamsson, Minna Pikkarainen e Outi Salo, de Brian Fitzgerald e Kieran Conboy e de Maria Paasivaara e Casper Lassenius apontam que esses autores pesquisaram em grupo um grande número de vezes. Pela Figura 18 é possível verificar que Pekka Abrahamsson, Minna Pikkarainen, Outi Salo e Brian Fitzgerald e Kieran Conboy apresentam abordagens parecidas.

Com base na Figura 18 é possível verificar uma baixa concentração da produção em conjunto de autores, evidenciando a Lei do Elitismo de Price (ARAÚJO, 2006), que prevê um percentual de metade da produção para uma elite formada pela raiz quadrada do total de autores, sendo um valor bem menor do que aquele determinado para uma elite muito produtiva.

O primeiro *cluster* é formado pelos autores: Torgeir Dingsøyr, Tore Dybå, Nils Brede Moe e Sridhar Nerur. Todos esses autores trabalham com pesquisas na linha de melhorias de processo de desenvolvimento ágil de software. Os autores Torgeir Dingsøyr e Tore Dybå publicaram o artigo mais citado sobre o tema, que foi abordado no item 4.1.2.5, na Tabela 8. Torgeir Dingsøyr, Sridhar Nerur e Nils Brede Moe publicaram o quinto artigo mais citado sobre o tema. Pode-se analisar que outros artigos que esses autores publicaram em conjunto possuem a interação humana, como o trabalho em equipe dos times ágeis, como objeto de pesquisa. Um

de seus trabalhos, nomeado "*A teamwork model for understanding an agile team: a case study of a Scrum Project",* ou "Um modelo de trabalho em equipe para entender um time ágil: um caso de estudo de um Projeto de Scrum", indica que o desenvolvimento do software depende significativamente da performance do time, assim como ocorre com qualquer processo que envolva interação humana. Os autores estudaram a introdução de princípios do Scrum em uma empresa de desenvolvimento de software a fim de verificar como os mecanismos de trabalho em equipe eram compreendidos pelas pessoas envolvidas no projeto. Outro trabalho dos autores Torgeir Dingsøyr, Tore Dybå e Nils Brede Moe consistia em verificar as barreiras para um time de desenvolvimento se tornar auto-gerenciável. O trabalho em equipe proposto pelos métodos ágeis gera muitas vantagens, tais como aumento de produtividade, inovação e satisfação do empregado, porém nem sempre essa união resulta em sucesso da organização, uma vez que os times precisam aprender a trabalhar efetivamente como um time (MOE; DINGSØYR; DYBÅ, 2010).

O segundo *cluster* é formado por Pekka Abrahamsson, Minna Pikkarainen e Outi Salo. Pekka Abrahamsson é um dos autores que mais realiza publicações sobre o tema desenvolvimento ágil de software e é possível ver que muitas de suas publicações são com Minna Pikkarainen e Outi Salo. Os trabalhos desses autores abrangem a temática do efeito de práticas de métodos ágeis nas organizações e os principais métodos abordados são o XP e o Scrum. Minna Pikkarainen e Outi Salo também publicaram um artigo sobre o impacto de práticas ágeis na comunicação do desenvolvimento de software.

O terceiro *cluster* é formado por Brian Fitzgerald e Kieran Conboy. Kieran Conboy publicou artigos que se tratam sobre os métodos XP e *Lean* Software *Development.* O artigo mais citado entre os autores Brian Fitzgerald e Kieran Conboy foi "*Customising agile methods to* software *practices at Intel Shannon",* ou "Personalizando métodos ágeis para práticas de software na Intel Shannon", em que foi estudada a combinação entre os métodos XP e Scrum, concluindo que ambos os métodos se complementam positivamente, pois XP fornece suporte em aspectos técnicos e o Scrum oferece suporte no planejamento do projeto e no acompanhamento. Assim, os defeitos do código do software reduziram drasticamente, além de terem concluído mais rapidamente os projetos de software.

O quarto *cluster* é formado por Maria Paasivaara e Casper Lassenius, e foram constatadas muitas publicações sobre a abordagens do Scrum e, também, muitos trabalhos sobre o desenvolvimento ágil de software em grandes equipes.

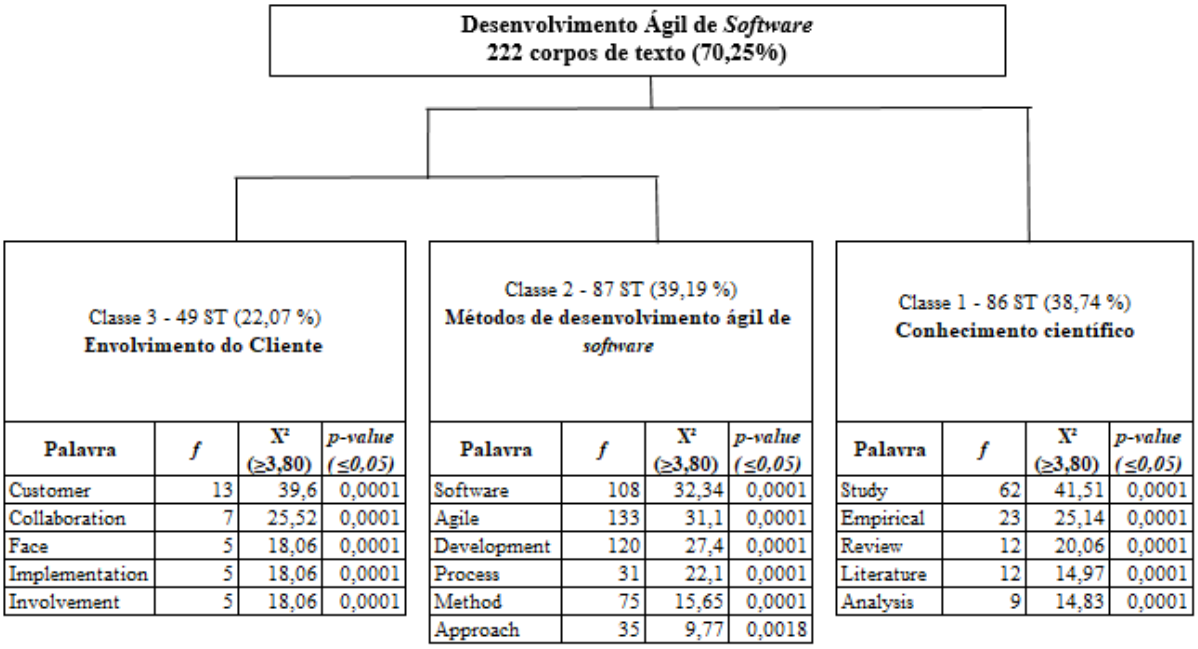
#### 4.1.3.4 Apresentação do Modelo Integrador

O Modelo Integrador apresenta os principais temas estudados no desenvolvimento ágil de software e foi elaborado com base nos dados obtidos pela Classificação Hierárquica Descendente (CHD), que foi desenvolvida pelo software Iramuteq. As análises feitas pelo software são consideradas qualitativas, uma vez que analisam as palavras que foram utilizadas em um discurso, e também são quantitativas, pois analisam as palavras do discurso pela frequência e pelo método estatístico inferencial do Qui-Quadrado.

Para obter a Classificação Hierárquica Descendente (CHD), foram compilados os resumos dos 70 artigos mais citados sobre desenvolvimento ágil de software no software estatístico Iramuteq, e o corpus geral de texto dos 70 artigos foi separado em 316 segmentos de textos (ST), com aproveitamento de 222 STs, correspondendo a 70,25%. Emergiram 11418 ocorrências de palavras e vocabulários, sendo 1370 palavras distintas e 311 com uma única ocorrência.

O conteúdo foi analisado e categorizado em três classes: a Classe 1, que compreende 86STs, representando 38,74% dos segmentos de texto aproveitados, a Classe 2, que contempla 87 STs, ou 39,19% dos segmentos de texto aproveitados, e a Classe 3, que compreende 22,07% dos segmentos de texto aproveitados, constituindo 49STs.

Em seguida, após a análise de Classificação Hierárquica Descendente (CHD) obteve-se o Dendograma da CHD, que é composto pelas três Classes. Na Figura 19 são apresentadas as classes e a denominação dos conteúdos relacionados a cada uma delas. Cada classe consta das palavras com maior frequência (f) e seus respectivos Qui-Quadrados ($X^2$), que corresponde ao grau de significância das palavras que possuem mais afinidade com a classe (MUTOMBO, 2013).




| Palavra | *f* | $X^2$ (≥3,80) | *p-value* (≤0,05) |
|---|---|---|---|
| Customer | 13 | 39,6 | 0,0001 |
| Collaboration | 7 | 25,52 | 0,0001 |
| Face | 5 | 18,06 | 0,0001 |
| Implementation | 5 | 18,06 | 0,0001 |
| Involvement | 5 | 18,06 | 0,0001 |

| Palavra | *f* | $X^2$ (≥3,80) | *p-value* (≤0,05) |
|---|---|---|---|
| Software | 108 | 32,34 | 0,0001 |
| Agile | 133 | 31,1 | 0,0001 |
| Development | 120 | 27,4 | 0,0001 |
| Process | 31 | 22,1 | 0,0001 |
| Method | 75 | 15,65 | 0,0001 |
| Approach | 35 | 9,77 | 0,0018 |

| Palavra | *f* | $X^2$ (≥3,80) | *p-value* (≤0,05) |
|---|---|---|---|
| Study | 62 | 41,51 | 0,0001 |
| Empirical | 23 | 25,14 | 0,0001 |
| Review | 12 | 20,06 | 0,0001 |
| Literature | 12 | 14,97 | 0,0001 |
| Analysis | 9 | 14,83 | 0,0001 |

**Figura 19 – Dendograma da CHD sobre Desenvolvimento Ágil de Software**

**Fonte:** Iramuteq

Analisando a Figura 19 pode-se verificar que as Classes 3 e 2 apresentam relação, podendo inferir que há relação e influência do envolvimento do cliente com o desenvolvimento ágil de software. A Classe 1 não apresentou relação significativa com a Classe 3 e 2 e propõe que está sendo muito realizado estudos empíricos e revisões de literatura sobre desenvolvimento ágil de software.

A Classe 2 foi a mais presente de todas, com 39,19% dos aproveitamentos de segmentos de texto, destacando a importância dos métodos de desenvolvimento ágil de software quanto ao tema desenvolvimento ágil de software.

A seguir serão caracterizadas cada uma das três classes:

- Classe 1: Conhecimento científico sobre desenvolvimento ágil de software

Essa classe enfatiza a importância do conhecimento científico sobre desenvolvimento ágil de software advindo por meio de estudos empíricos, revisões da literatura e análise de cenários e de dados. Esses fatores apresentam forte impacto no desenvolvimento do software quando se utilizam métodos ágeis, uma vez que contemplam 38,74% dos elementos do Modelo Integrador.

Schwaber e Sutherland (2013) afirmam que o empirismo advém da experiência, enquanto que a tomada de decisões advém do conhecido.

Assim, há forte presença de discurso científico nesta classe e foi constatado que as palavras: “estudos empíricos”, “pesquisa”, “revisões”, “literatura” possuem maior frequência nos discursos, corroborando com a análise CHD conforme pode ser observado nos trechos dos resumos dos artigos a seguir:

*“...here we report our findings from **investigating empirical evidence** in **gse related research literature** by conducting a **systematic review** we observe that the **gse** field is still immature the amount of **empirical studies** is relatively small.”*

*“...in summary the **systematic review results** in several descriptive classifications of the **papers** on **empirical studies** in gse and also reports on some best practices identified from **literature**”.*

*“...and experience reports comprise the majority of the **research** in the **area** while very few **empirical research efforts** have been conducted this article **reviews** the state of **research** in xp and am and recommends areas that could benefit from further **study**.”*

*“...this **paper reports results** from four* software *development **projects** where the impact of pair programming on* software *product **quality** was **studied** our **empirical findings appear** to offer contrasting **results** regarding some of the **claimed benefits** of pair programming.”*

*“...however there has not been any effort to systematically **select review** and synthesize the **literature** on this topic we have conducted a **systematic literature review** of the **primary studies** that report using scrum practices in gsd **projects**.”*

Assim, pode ser verificado que há grande interesse em revisões na literatura sobre desenvolvimento ágil de software. Estudos experimentais têm sido realizados com o propósito de investigar a análise, o projeto, a implementação, o teste, a manutenção, a garantia da qualidade e o reuso do software.

Basili (1996) realizou uma análise sobre a importância de estudos empíricos no desenvolvimento de software. Na manufatura por exemplo, por meio de estudos empíricos é possível observar as soluções existentes, propor melhores soluções, construir e desenvolver um projeto de software, mensurar e analisar o processo e repeti-lo diversas vezes a fim de identificar mais melhorias de forma contínua. Quando há evolução do modelo de desenvolvimento de software, as mudanças refletem no processo, no produto, nas pessoas e na organização. Assim, é preciso realizar experimentos com técnicas para verificar como e quando elas realmente funcionam, para entender os seus limites e como melhorá-los. Por meio da aplicação é que se melhora o entendimento sobre o desenvolvimento do software (BASILI, 1996).

Dingsøyr et al (2012) afirmam que o desenvolvimento ágil envolve experiências pessoais e aprendizado, indicando uma evidência para os estudos empíricos sobre desenvolvimento ágil serem comumente realizados. Eles identificaram, por meio de revisão sistemática, que as perspectivas teóricas mais utilizadas nas pesquisas sobre métodos ágeis foram: conhecimento sobre gerenciamento, personalidade e aprendizado organizacional, dentre outras perspectivas teóricas. De acordo com os autores, o conhecimento sobre gerenciamento é importante para explorar a geração de conhecimento em equipes de software em geral e também em equipes ágeis desenvolvimento de software, uma vez que desenvolvimento de software envolve atividades de criação de conhecimento. Já as teorias de personalidade são importantes para explorar as dinâmicas interpessoais entre a equipe ágil, como por exemplo quando as equipes utilizam Programação em Pares. Um exemplo de teoria de personalidade que é estudada para aprimorar as relações entre o time ágil é a *Big Five personality theory*, ou Teoria dos Cinco Fatores, proposta por Mccrae e Costa (1995), que propõe um modelo geral de teorias da personalidade. O aprendizado organizacional é muito estudado pelo fato dos princípios ágeis visarem aprendizado em equipe e, também, aprendizado em como atender e adaptar melhor às mudanças (DINGSØYR et al., 2012).

Muitos estudos de caso trataram sobre a implementação de métodos ágeis em organizações que utilizam métodos tradicionais (SVENSSON; HOST, 2005). Em sua maioria é reportado o desafio da adoção de métodos ágeis de desenvolvimento de software. Em alguns estudos foi constatado que em organizações maiores e que produzem software mais complexos, apenas promover pequenas iterações não resultaram em melhoria da qualidade do software (KETTUNEN; LAANTI, 2008). Em grandes organizações é importante ter a visão sistêmica da empresa como um todo e é preciso saber em qual aspecto deve ser aplicado o conceito de agilidade. Por exemplo, a agilidade pode ser aplicada em reduzir o tempo de produção de novas características de produtos, em criar novas variações dos produtos com maior frequência, em inovações de produto de software com maior frequência, e em melhorar a eficiência e o custo de produção do software (KETTUNEN; LAANTI, 2008).

Laanti, Salo e Abrahamsson (2011) publicaram um artigo que consistia em verificar a satisfação dos colaboradores com a implementação de métodos ágeis em uma larga empresa de desenvolvimento de software que utilizada métodos tradicionais anteriormente, a Nokia. Pode-se analisar então o impacto da introdução de métodos ágeis em uma empresa altamente complexa. Antes de introduzir princípios ágeis, a empresa utilizava o Modelo Cascata. Os resultados mostraram que a maioria dos respondentes demonstraram alta satisfação, um

sentimento de efetividade, aumento da qualidade, da transparência, autonomia e detecção antecipada de falhas e defeitos no software. Além disso, 60% dos respondentes afirmaram que não têm interesse em voltar a utilizar o método de desenvolvimento de software que utilizavam antes.

- Classe 2: Métodos de desenvolvimento ágil de software.

A classe 2 aborda aspectos que compõem o desenvolvimento ágil de software. A palavra "desenvolvimento" foi frequentemente citada e há forte evidência da influência de métodos e processos em sua performance. Foram citados 75 vezes aspectos de "métodos", sendo este um dos qui-quadrados ($X^2$) maiores da classe, com 15,65, apresentando a importância de um método no desenvolvimento ágil de software. A palavra "processo" também foi muito significativa nesta classe, com qui-quadrado ($X^2$) igual a 22,1.

As constatações de citações dessas palavras podem ser verificadas nos seguintes trechos de artigos:

*"...results this has resulted in the agile requirements refinery an extension to the scrum* ***process*** *that enables product managers to cope with complex requirements in an* ***agile development*** *environment a case study is presented to illustrate how agile* ***methods*** *can be applied to* **software** *product management".*

*"...****agile*** **software** ***development approaches*** *such as Extreme Programming crystal methods lean* ***development*** *scrum adaptive* software ***development*** *asd and others view change from a perspective that mirrors today s turbulent business and technology environment".*

*"...this is illustrated in an industry study of an organization transitioning to an* ***agile*** **software** ***development*** *environment in which a minimalistic* ***agile method*** *was first constructed and then enhancements were made as the organization itself improved the* ***process****".*

*"...****process*** *flexibility and globally distributed* ***development*** *are two major current trends in* **software** *and information systems* ***development*** *isd the quest for flexibility is very much evident in the recent* ***development*** *and increasing acceptance of various* ***agile methods*** *such as Extreme Programming beck and andres 2005 and scrum schwaber and beedle 2002".*

*"...****agile approaches*** *to* **software** ***development*** *provide flexibility within the* ***method*** *but provide little or no flexibility to support* **software** ***process*** *improvement spi this second kind of flexibility is important to permit organizations to improve with time".*

Robbes, Hill e Bird 2018) afirmaram que há áreas relacionadas a software que são comumente exploradas, tais como: evolução do software, modelos de desenvolvimento de software e seus processos, caracterização dos desenvolvedores e suas atividades, futuro das qualidades de software, a previsão de defeitos de software, análises de padrões de mudança de software e análise de clones de códigos. Diante disso, o interesse por essas áreas confirma a alta frequência das palavras presentes na Classe 2.

Os métodos de desenvolvimento ágil de software tais como Scrum, *Extreme Programming*, *Lean Software Development* e Kanban são os mais utilizados atualmente pela indústria.

- Classe 3: Envolvimento do Cliente e da Equipe

Analisando os resultados apresentados na Classe 3 pode-se verificar que a colaboração do cliente e a integração entre o time de desenvolvimento ágil de software são fatores vitais para o sucesso do desenvolvimento ágil de software, uma vez que foi muito citada a importância do envolvimento do cliente durante o desenvolvimento ágil de software. Talvez o fator determinante para o sucesso do trabalho. Foram muito citadas as palavras "cliente", "colaboração" e "envolvimento", conforme apresentam os seguintes trechos de artigos:

*"...**customers** were not as involved on these agile projects as agile methods demand we describe the causes of inadequate **customer collaboration** its adverse consequences on self-organizing agile teams and agile undercover a set of strategies used by the teams to practice agile despite insufficient or ineffective **customer involvement**".*

*"...context **customer collaboration** is a vital feature of agile* software *development objective this article addresses the importance of adequate **customer involvement** on agile projects and the impact of different levels of **customer involvement** on real life agile projects".*

*"...conclusion **customer involvement** is important on agile projects inadequate **customer involvement** causes adverse problems for agile teams the agile undercover strategies we_ve identified can assist agile teams facing similar lack of **customer involvement**".*

*"...method we conducted a grounded theory study involving 30 agile practitioners from 16* software *development organizations in new zealand and india over a period of 3 years results we discovered that lack of **customer involvement** was one of the biggest challenges faced by agile teams".*

As palavras das três Classes definidas no Dendograma da CHD foram também verificadas com maior profundidade por meio da Análise Fatorial Confirmatória (AFC) e da Análise de Similitude.

A Figura 20 apresenta a Análise Fatorial Confirmatória (AFC), por onde pode-se observar espacialmente a relação entre as palavras de cada classe. As palavras que estão mais próximas simbolizam proximidade de contexto.

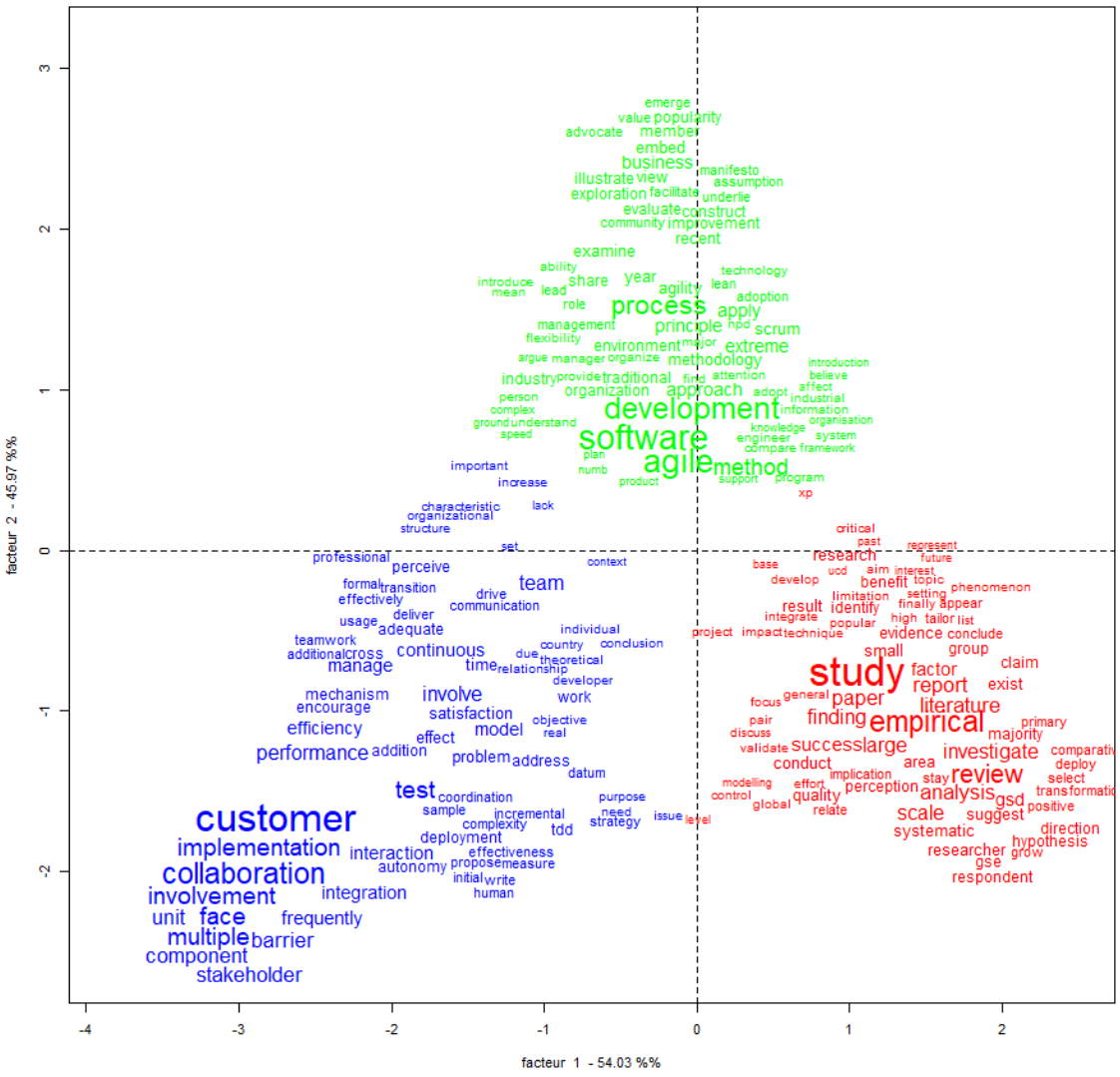



**Figura 20 - Análise Fatorial Confirmatória**

**Fonte: Iramuteq**

Assim, as palavras da Classe 3, envolvimento com o cliente, estão mais próximas da Classe 2, métodos de desenvolvimento ágil de software. Entre essas duas classes, as palavras que mais se aproximam são: “perceive (*perceber*)” e “*context (*contexto)”, podendo inferir que é necessário haver uma análise, por parte da equipe, sobre o contexto antes e durante o

desenvolvimento do software. Outras palavras são: "*team* (equipe)", que está em tamanho significativo, fortalecendo o argumento dos princípios ágeis da importância do trabalho em equipe para a qualidade do software. *Scrum* se concentra em como os membros do time devem funcionar de forma flexível para atender o ambiente de constantes mudanças (ABRAHAMSSON et al., 2002).

A classe 1, embora no CHD não tenha demonstrado significativa relação, na Análise Fatorial Confirmatória é possível verificar que há palavras que relacionam com a Classe 3, tais como "*project* (projeto)" e "*level* (nível)", "*discuss* (discussão)" e "*validate (*validação)" foram resultados interessantes. E em sua relação com a classe 2, as palavras "XP", "*critical* (crítico)", entre outras, ficaram próximas. Entretanto, pelo tamanho da fonte, é possível ver na Figura 20 que as palavras "*research (*pesquisa)", "*benefit* (benefícios)" e "*results* (resultado)" foram mais impactantes.

Em seguida, foi elaborada também no software Iramuteq a Análise de Similitude. Essa análise oferece a identificação das co-ocorrências entre as palavras e as conexões entre os termos. As ligações mais fortes e mais recorrentes no corpus textual estão representadas nas linhas mais grossas da figura, e essa espessura dos traços e o tamanho das palavras trazem indicações da conexão entre as palavras e auxilia na identificação da estrutura de um corpus textual (CAMARGO; JUSTO, 2013).

Na Figura 21 pode-se perceber que a palavra "*agile* (ágil)*"* é a aglutinadora em relação a estudos empíricos e estudos de caso, a desenvolvimento de software e a programação em pares.

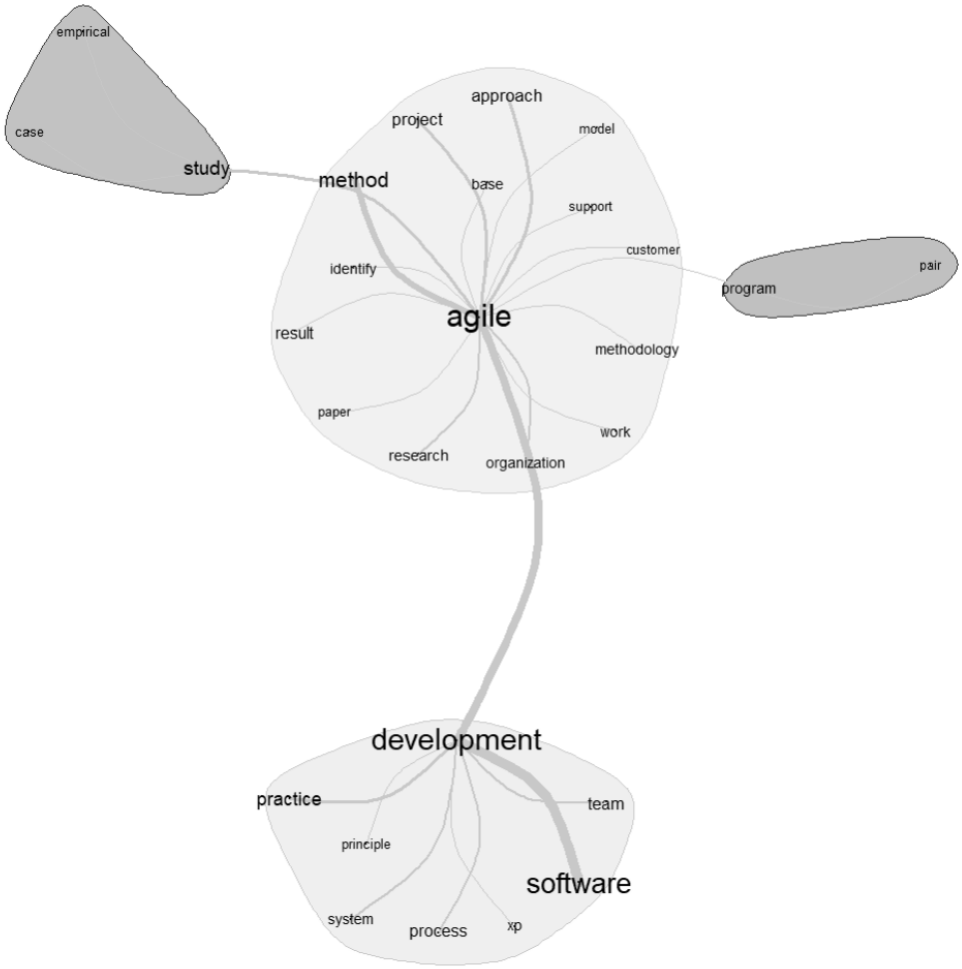



**Figura 21 - Análise de Similitude**

**Fonte:** Iramuteq

A Análise de Similitude apresenta ágil como tema central no desenvolvimento ágil de software. É um resultado esperado, uma vez que reagir de forma ágil às mudanças é o objetivo deste método. Lee e Xia (2010) definem a agilidade no desenvolvimento de software como a capacidade do time de responder, de forma eficiente e eficaz, e incorporar as mudanças de requisitos impostas pelo usuário durante o ciclo de vida do software. Nesse campo de Ágil, a linha mais grossa está ligando à palavra “método”.

No âmbito de desenvolvimento, a Análise de Similitude apresenta que a prática é um fator importante. Lyytinen e Rose (2006) sugerem que agilidade é adquirida por meio de processos de aprendizado. Assim, da forma como as palavras estão espacialmente relacionadas, pode-se inferir o uso do ágil em estudos empíricos, em Programação em Pares (*Pair Programming)*, e em desenvolvimento de software, sendo este último envolvendo práticas, princípios, processos,

time e sistema. O método XP está aparente na Análise da Similitude, indicando que é ainda um método altamente utilizado pela comunidade ágil.

A Programação em Pares é uma prática ágil utilizada no método XP e, por estar inserida nos resultados da Análise de Similitude, indica uma prática significativa dentro do campo de pesquisa e de uso no desenvolvimento ágil de software. O primeiro projeto a utilizar XP foi o C3, da Chrysler. Nesse caso citado, durante longo período foram executados projetos seguindo as metodologias tradicionais e sem os resultados esperados. Com o uso do XP foram obtidas muitas melhorias, tais como a finalização do projeto em um período menor de tempo, que é uma das propostas da metodologia.

Então, por meio da separação dos artigos identificados que constam em cada Classe da Classificação Hierárquica Descendente (CHD) foi elaborado o Modelo Integrador dos Principais Artigos Referentes às Classes da CHD. Conforme informado, este modelo contempla as principais linhas de pesquisa estudadas no desenvolvimento ágil de software, e pode ser verificado na Tabela 10.

**Tabela 10 - Modelo Integrador dos Principais Artigos Referentes às Classes da CHD**

(Continuação)

| Classe | Título do Artigo | Autor(es) e Ano |
|---|---|---|
| 1. Conhecimento Científico | Empirical evidence in global software engineering: a systematic review | Smite, D; Wohlin, C.; Gorschek, T.; Feldt, R (2010) |
| | Agile modeling, agile software development, and extreme programming: The state of research | Erickson, J; Lyytinen, K; Siau, K (2005) |
| | A multiple case study on the impact of pair programming on product quality | Hulkko, H; Abrahamsson, P (2005) |
| | Using Scrum in Global Software Development: A Systematic Literature Review | Hossain, E.; Babar, M.; Paik, H. (2009) |
| 2. Métodos de Desenvolvimento Ágil de Software | The agile requirements refinery: Applying SCRUM principles to software product management | Vlaanderen, K.; Jansen, S.; Brinkkemper, S.; Jaspers, E. (2011) |
| | Creating a dual-agility method: The value of method engineering | Henderson-Sellers, B; Serour, M.K. (2005) |
| | A framework to support the evaluation, adoption and improvement of agile methods in practice | Qumer, A.; Henderson-Sellers, B. (2008) |
| | Management challanges to implementing Agile Processes in traditional development organizations | Boehm, B; Turner, R (2005) |

(Conclusão)

| | | |
|---|---|---|
| 3. Integração entre o Time de Desenvolvimento | The impact of inadequate customer collaboration on self-organizing Agile teams | Hoda, R.; Noble, J.; Marshall, S. (2011) |
| | Coordination in co-located agile software development projects | Strode, D. E.; Huff, S. L.; Hope, B.; Link, S. (2012) |
| | Self-Organizing Roles on Agile Software Development Teams | Hoda, R.; Noble, J.; Marshall, S. (2013) |
| | The impact of agile practices on communication in software development | Pikkarainen, M.; Haikara, J.; Salo, O.; Abrahamsson, P.; Still, J. (2008) |

Fonte: da propria autora

Os artigos presentes em cada Classe foram os que mais enfatizaram o contexto abordado por ela.

Na Classe 1, os artigos apresentados mostram como os estudos empíricos e a revisão da literatura reforçam o conhecimento científico e agregam conhecimentos ao desenvolvedor de software. O artigo *"Empirical evidence in global* software *engineering: a systematic review"* teve como grande contribuição a apresentação das principais práticas ágeis identificadas na revisão sistemática, e essas foram: encontros e visitas, as quais propiciaram confiança e coesão em relação ao projeto; suporte ao gerenciamento do projeto, o qual gerou transparência em relação ao projeto; ciclos curtos de iteração no desenvolvimento, o qual auxiliou a antecipação de *feedback* recebido e capacidade de avaliação do projeto; distribuição de tarefas, a qual favoreceu o trabalho em equipe.

Na Classe 2, os artigos contemplados apresentam propostas de abordagens, métodos de desenvolvimento de software e também contribuem com o conhecimento sobre os processos. O artigo "*A framework to support the evaluation, adoption and improvement of agile methods in practice*" aborda essa temática ao propor o *Agile* Software *Solution Framework* (ASSF), ou "*Framework* de Solução Ágil de Software", que mostra como explorar os métodos ágeis, conhecimento e governança. De acordo com Qumer e Henderson-Sellers (2008), a ponte entre governança e métodos ágeis permite facilitar a implementação de princípios de desenvolvimento ágil na organização e fornece melhorias em termos de valor de negócio e de entrega.

Na Classe 3, muitos artigos abordaram a importância de o time ágil de desenvolvimento conseguir trabalhar em equipe e, também, abordou a importância da colaboração do cliente no projeto de desenvolvimento de software. No artigo *"The impact of inadequate customer collaboration on self-organizing Agile teams"* foi constatado que a falta de colaboração do cliente nos projetos ágeis trouxe consequências negativas ao projeto. As principais causas da falta de envolvimento por parte do cliente foram: ceticismo, falta de tempo para se dedicar ao projeto, falta de comprometimento e distância física. Este cenário fez com que a equipe de desenvolvimento tivesse dificuldades em coletar os requisitos do software e esclarecê-los com o cliente, em priorizar os requisitos e em receber *feedbacks*. Outra consequência negativa foi a perda de produtividade da equipe. Assim, foram propostas estratégias para superar a falta de envolvimento do cliente e continuar desenvolvendo o software de forma ágil, e essas estratégias foram denominadas *Agile Undercover*, sendo que as principais consistiam em: mudar a mentalidade do cliente, fornecer opções e negociar com o cliente, avaliação inicial de riscos e colaboração à distância (HODA; NOBLE; MARSHALL, 2011).

Dessa forma, o Modelo Integrador apresentou os temas mais estudados sobre desenvolvimento ágil de software, podendo esses ser considerados as linhas de pesquisas atuais sobre o tema.

# 5 CONSIDERAÇÕES FINAIS

Este trabalho de conclusão de curso apresenta uma revisão sistemática sobre desenvolvimento ágil de software. Foram identificados os principais temas de estudos dos artigos mais citados, caracterizando os maiores temas de interesse dos autores. Os principais temas identificados são relativos ao conhecimento científico, aos métodos de desenvolvimento e ao envolvimento do cliente durante o desenvolvimento ágil de software, o qual é indicado pela literatura como fator vital para o sucesso de projetos ágeis de desenvolvimento de software. As relevantes informações contidas nessa pesquisa, sendo muitas obtidas por métodos estatísticos, fornecem importantes elementos para desenvolvimento de software com sucesso, servindo de interesse tanto para engenheiros de software quanto para outros profissionais de diversas áreas da engenharia.

Além disso, foram também identificadas as principais frentes de pesquisa atuais, indicando a forte tendência de combinação de princípios ágeis com outros métodos para garantir melhor performance e sucesso do software desenvolvido.

Portanto, o objetivo geral desta pesquisa foi cumprido com sucesso. Foi interessante verificar o quão importante é a escolha de um método e de suas práticas para as organizações no atendimento aos seus clientes, pois um ambiente altamente competitivo exige desenvolvimento de produtos que atendam aos rigorosos requisitos de custos, qualidade, prazo de entrega e que se adequem à constante volatilidade dos requisitos impostos pelos clientes. O interesse das organizações por este tema está crescendo a cada ano, uma vez que foi constatado que grande parte dos países e suas organizações utilizam pelo menos um método ágil.

Foi possível identificar grande ênfase em desenvolvimento ágil utilizando o método *Extreme Programming* (XP), uma vez que houve grande quantidade de artigos envolvendo estudos empíricos e teóricos sobre este método e sobre as suas práticas. Outros métodos ágeis muito utilizados são o *Scrum,* Kanban e o *Lean* Software *Development*. Dentre as práticas ágeis, as que mais se destacaram foram programação em pares para *feedback*, testes de validação, reuniões diárias, iterações entre o time ágil e o cliente, e realização de revisões.

Além disso, este trabalho contempla as principais constatações dos autores em relação aos seus artigos publicados, destacando os pontos mais relevantes sobre desenvolvimento ágil de software e os seus aspectos críticos.

Outra constatação foi a grande quantidade de estudos que comparam métodos ágeis com os métodos tradicionais, sendo verificado que suas maiores diferenças estão na forma de comunicação, de documentação, de reação às mudanças, de receber *feedbacks* e de definir os requisitos do software.

Foram verificados muitos artigos que exploravam e estudavam sobre a abordagem mista ou híbrida de desenvolvimento, a qual utiliza princípios tradicionais e ágeis. Outra abordagem que consta em artigos mais recentes é a mistura de princípios ágeis com o método do *Design* Centrado no Usuário. Dessa forma, deixa-se como sugestão de trabalhos futuros a realização de revisão sistemática sobre a utilização de métodos ágeis combinados com outros métodos de desenvolvimento de software.

Além disso, outras propostas de trabalhos futuros são a realização de revisão sistemática para cada uma das três classes presentes no Modelo Integrador apresentado neste trabalho, uma vez que elas correspondem aos principais temas de estudos presentes nos artigos mais citados mundialmente. Ao realizar esse estudo, seria possível verificar com maior profundidade e precisão quais são as temáticas mais estudadas em cada classe e as suas correspondentes frentes de pesquisa, complementando, assim, dados ainda mais aprofundados sobre os resultados deste presente trabalho.

# 6 REFERÊNCIAS


ABRAHAMSSON, P.; SALO, O. Agile methods in European embedded software development organisations: a survey on the actual use and usefulness of Extreme Programming and Scrum. **IET Software**, v. 1, n. 2, p. 58–64, 2008.

ABRAHAMSSON, P.; SALO, O.; RONKAINEN, J.; WARSTA, J. Agile software development methods: Review and analysis. **Espoo, Finland: Technical Research Centre of Finland, VTT Publications**, p. 478, 2002. Disponível em: <http://jeffsutherland.org/whatsnew.html>.

ABRAMO, G.; D'ANGELO, C. A. Evaluating research: From informed peer review to bibliometrics. **Scientometrics**, v. 87, n. 3, p. 499–514, 2011.

ARAÚJO, C. A. Bibliometria: evolução histórica e questões atuais. **Em Questão**, v. 12, n. 1, p. 11–32, 2006.

BARROS, M. Altmetrics: métricas alternativas de impacto científico com base em redes sociais. **Perspectivas em Ciência da Informação**, v. 20, n. 2, p. 19–37, 2015. Disponível em: <http://www.scielo.br/scielo.php?script=sci_arttext&pid=S1413-99362015000200019&lng=pt&tlng=pt>.

BASILI, V. R. The role of experimentation in software engineering: past, current, and future. **Proceedings of the 18th international conference on Software engineering**, n. IEEE Computer Society, p. 442–449, 1996.

BASSI FILHO, D. L. Experiências com desenvolvimento ágil. **Ime-Usp**, p. 170, 2008.

BECK, K. **Programação Extrema Explicada**. [s.l.] Bookman, 1999.

BECK, K.; BEEDLE, M.; VAN BENNEKUM, A.; COCKBURN, A.; CUNNINGHAM, W.; FOWLER, M.; HIGHSMITH, J.; HUNT, A.; JEFFRIES, R.; KERN, J.; MARICK, B.; MARTIN, R. C.; SCHWABER, K.; SUTHERLAND, J.; THOMAS, D. Manifesto for Agile Software Development. p. 2–3, 2001. Disponível em: <https://www.researchgate.net/file.PostFileLoader.html?id=57d055b593553b11467ddd59&assetKey=AS%3A403742915612673%401473271220194>.

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

BENNETT, K.; LAYZELL, P.; BUDGEN, D.; BRERETON, P.; MACAULAY, L.; MUNRO, M. Service-based software: The future for flexible software. **Proceedings - Asia-Pacific Software Engineering Conference, APSEC**, v. 2000–Janua, p. 214–221, 2000.

BENNETT, K.; RAJLICH, V. Software Main tenance and Evolution–A Staged Model. In: **Proc. of the Future of Software Eng.** ICSE-2000 ed. Limerick: IEEE Press, 2001. p. 73–89.

BOEHM, B.; TURNER, R. Management challenges to implementing agile processes in traditional development organizations. **IEEE Software**, v. 22, n. 5, p. 30–39, 2005.

BOEHM, B. W. Get ready for agile methods, with care. **IEEE Computer**, v. 35, p. 64–69, 2002.

BÖRNER, K.; CHEN, C.; BOYACK, K. W. Visualizing knowledge domains. **Annual review of information science and technology**, v. 37, n. 1, p. 179–255, 2003.

BRHEL, M.; METH, H.; MAEDCHE, A.; WERDER, K. Exploring Principles of User-Centered Agile Software Development: A Literature Review. **Information and Software Technology**, v. 61, p. 163–181, 2015. Disponível em: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4599453>.

BROOKES, B. C. Bradford's law and the bibliography of science. **Nature**, v. 224, n. 5223, p. 953, 1969.

BUDZIER, A.; FLYVBJERG, B. Making Sense of the Impact and Importance of Outliers in Project Management Through the Use of Power Laws. **IRNOP Conference**, p. 1–28, 2013. Disponível em: <http://papers.ssrn.com/abstract=2289549>.

CAMARGO, B. V. ALCESTE: um programa informático de análise quantitativa de dados textuais. **Perspectivas teórico-metodológicas em representações sociais**, v. 1, p. 511–539, 2005.

CAMARGO, B. V.; JUSTO, A. M. IRAMUTEQ: Um software gratuito para análise de dados textuais. **Temas em Psicologia**, v. 21, n. 2, p. 513–518, 2013. Disponível em: <http://pepsic.bvsalud.org/pdf/tp/v21n2/v21n2a16.pdf>.

CAMPANELLI, A. S.; PARREIRAS, F. S. Agile methods tailoring - A systematic literature review. **Journal of Systems and Software**, v. 110, p. 85–100, 2015. Disponível em: <http://dx.doi.org/10.1016/j.jss.2015.08.035>.

COCKBURN, A. **Agile Software Development**. [s.l.] Pearson Education, 2006.

COCKBURN, A.; HIGHSMITH, J. Agile software development: The people factor. **Computer**, v. 34, n. 11, p. 131–133, 2001.

COCKBURN, A.; WILLIAMS, L. Guest Editors' Introduction: Agile Software Development: It's about Feedback and Change. **Computer**, v. 36, n. 6, p. 39–43, 2003.

CONBOY, K. Agility from First Principles: Reconstructing the Concept of Agility in Information Systems Development. **Information Systems Research**, v. 20, n. 3, p. 329–354, 2009.

CONRADI, R.; DYBÅ, T.; SJØBERG, D. I. K.; ULSUND, T. **Software Process Improvement: Results and Experience from the Field**. New York: Springer-Verlag, 2006.

DINGSØYR, T.; NERUR, S.; BALIJEPALLY, V.; MOE, N. B. A decade of agile methodologies: Towards explaining agile software development. **Journal of Systems and Software**, v. 85, n. 6, p. 1213–1221, 2012.

DOLLAGHAN, C. A. **The handbook for evidence-based practice in communication disorders**. [s.l.] Brookes Pub., 2007.

DYBÅ, T.; DINGSØYR, T. Empirical studies of agile software development: A systematic review. **Information and Software Technology**, v. 50, n. 9–10, p. 833–859, 2008.

EGGHE, L.; ROUSSEAU, R. Co-citation, bibliographic coupling and a characterization of lattice citation networks. **Scientometrics**, v. 55, n. 3, p. 349–361, 2002.

FILHO, W. P. **Engenharia de software: fundamentos, métodos e padrões**. 2. ed. Rio de Janeiro: LTC – Livros Técnicos e Científicos S.A., 2003.

FOX, D.; SILLITO, J.; MAURER, F. Agile methods and user-centered design: How these two methodologies are being successfully integrated in industrys. **Proceedings - Agile 2008 Conference**, p. 63–72, 2008.

FUGGETTA, A. Software process. **Proceedings of the conference on The future of Software engineering - ICSE '00**, v. 97, p. 25–34, 2000. Disponível em: <http://portal.acm.org/citation.cfm?doid=336512.336521>.

GARCÍA, C. R.; RAMIREZ, C. P. El meta análisis como instrumento de investigación en la determinación y análisis del objeto del estudio: aplicado al estudio de sistema de información. **Congreso de Alicante**, p. 1–13, 2004.

GERHARDT, T. E.; SILVEIRA, D. T. **Métodos de pesquisa**. [s.l.] Plageder, 2009.

GIL, A. C. Como classificar as pesquisas? **Como elaborar projetos de pesquisa**, v. 4, n. c, p. 44–45, 2002.

GLÄNZEL, W. Coauthorship patterns and trends in the sciences (1980-1998): A bibliometric study with implications for database indexing and search strategies. **Library Trends**, v. 50, n. 3, p. 461–473, 2002. Disponível em: <http://apps.webofknowledge.com/full_record.do?product=UA&search_mode=GeneralSearch&qid=19&SID=P1btm4QggziYOhR779N&page=1&doc=1>.

GLÄNZEL, W.; SCHUBERT, A. Analyzing scientific networks through coauthorship. **Nordic Journal Of Psychiatry**, v. 57, n. 5, p. 393–394, 2003.

GOFFMAN, W. Mathematical approach to the spread of scientific ideas—the history of mast cell research. **Nature**, v. 212, n. 5061, p. 449, 1966.

GOLDMAN, S. L.; NAGEL, R. N.; PREISS, K. **Agile competitors and virtual organizations: strategies for enriching the customer**. New York: Van Nostrand Reinhold, 1995.

GRÁCIO, M. C. C. Acoplamento Bibliográfico e Análise de Cocitação: Revisão TeóricoConceitual. **Encontros Bibli: Revista Eletrônica de Biblioteconomia e Ciência da Informação**, v. 21, n. 47, p. 82–99, 2016.

GRÁCIO, M. C. C.; OLIVEIRA, E. F. T. Análise de cocitação de autores: um estudo teórico-metodológico dos indicadores de proximidade, aplicados ao GT7 da ANCIB. **LIINC em Revista**, p. 196–213, 2013.

GUEDES, V. L. da S. A bibliometria e a gestão da informação e do conhecimento científico e tecnológico: uma revisão da literatura. **PontodeAcesso**, v. 6, n. 2, p. 74–109, 2012. Disponível em: <http://www.portalseer.ufba.br/index.php/revistaici/article/view/5695/4591%5Cnhttp://www.portalseer.ufba.br/index.php/revistaici/article/view/5695>.

GUEDES, V. L. S.; BORSCHIVER, S. Bibliometria : Uma Ferramenta Estatística Para a Gestão Da Informação E Do Conhecimento , Em Sistemas De Informação , De Comunicação E De. **CINFORM - Encontro Nacional de Ciência da Informação**, p. 1–18, 2005. Disponível em: <http://dici.ibict.br/archive/00000508/01/VaniaLSGuedes.pdf>.

HARRISON, M. D.; DUKE, D. J. A Review of Formalisms for Describing Interactive Behaviour. **Proceedings of the Workshop on Software Engineering and Human-Computer Interaction**, n. February 2014, p. 49–75, 1995. Disponível em: <http://dl.acm.org/citation.cfm?id=645541.657996>.

HENDERSON-SELLERS, B.; SEROUR, M. K. Creating a Dual-Agility Method : The Value of Method Engineering. **Journal of Database Management**, v. 16, n. 4, p. 1–23, 2005.

HIGHSMITH, J. Agile project management: Principles and tools. **Cutter Consortium**, v. 4, p. 1–37, 2003.

HIGHSMITH, J.; COCKBURN, A. Agile software development: The business of innovation. **Computer**, v. 34, n. 9, p. 120–122, 2001.

HIRAMA, K. **Engenharia de software: qualidade e produtividade com tecnologia**. Rio de Janeiro: Elsevier Brasil, 2011.

HODA, R.; NOBLE, J.; MARSHALL, S. The impact of inadequate customer collaboration on self-organizing Agile teams. **Information and Software Technology**, v. 53, n. 5, p. 521–534, 2011. Disponível em: <http://dx.doi.org/10.1016/j.infsof.2010.10.009>.

JONES, A. W. Impact factors of forensic science and toxicology journals: What do the numbers really mean? **Forensic Science International**, v. 133, n. 1–2, p. 1–8, 2003.

JUNIOR, C. M.; SOUZA, M. T. S.; PARISOTTO, R. S.; PALMISANO, A. As leis da Bibliometria em diferentes Bases de dados Científicos. **Revista de Ciências da Administração**, v. 44, n. 18, p. 111, 2016. Disponível em: <http://dx.doi.org/10.5007/2175-8077.2016v18n44p111>.

KALBACH, J. **Design de navegação web: otimizando a experiência do usuário**. [s.l.] Bookman, 2009.

KARLSTRÖM, D.; RUNESON, P. Combining agile methods with stage-gate project management. **IEEE Software**, v. 22, n. 3, p. 43–49, 2005.

KERZNER, H.; KERZNER, H. R. **Project management: a systems approach to planning, scheduling, and controlling**. [s.l.] John Wiley & Sons, 2017.

KETTUNEN, P.; LAANTI, M. Combining agile software projects and large-scale organizational agility. **Software Process: Improvement and Practice**, v. 13, n. 2, p. 183–193, 2008.

LAANTI, M.; SALO, O.; ABRAHAMSSON, P. Agile methods rapidly replacing traditional methods at Nokia: A survey of opinions on agile transformation. **Information and Software Technology**, v. 53, n. 3, p. 276–290, 2011. Disponível em: <http://dx.doi.org/10.1016/j.infsof.2010.11.010>.

LEE, G.; XIA, W. Toward Agile: An Integrated Analysis of Quantitative and Qualitative Field Data on Software Development Agility. **MIS Quarterly**, v. 34, n. 1, p. 87–114, 2010.

LEE, S.; YONG, H.-S. Distributed agile: project management in a global environment. **Empirical Software Engineering**, v. 15, n. 2, p. 204–217, 2010. Disponível em: <http://link.springer.com/10.1007/s10664-009-9119-7>.

LEFFINGWELL, D. **Scaling software agility: best practices for large enterprises**. [s.l.] Pearson Education, 2007.

LEI, H.; GANJEIZADEH, F.; JAYACHANDRAN, P. K.; OZCAN, P. A statistical analysis of the effects of Scrum and Kanban on software development projects. **Robotics and Computer-Integrated Manufacturing**, v. 43, p. 59–67, 2017. Disponível em: <http://dx.doi.org/10.1016/j.rcim.2015.12.001>.

LIMA, R. C. M. Estudo Bibliométrico: Análise De Citações No Periódico "Scientometrics". **CI Inf.**, v. 13, n. 1, p. 57–66, 1984.

LINDE, K.; WILLICH, S. How objective are systematic reviews?Differences between reviews on complementary medicine. **JR Society Medical**, v. 96, p. 17–22, 2003.

LINE, M. The half life of periodical literature: apparent and real obsolescence. **Journal of Documentation**, v. 26, n. 1, p. 46–54, 1970.

LYYTINEN, K.; ROSE, G. M. Information system development agility as organizational learning. **European Journal of Information Systems**, v. 15, n. 2, p. 183–199, 2006.

MAHONEY, A. A. A constituição da pessoa: desenvolvimento e aprendizagem. In: **A constituição da pessoa na proposta de Henri Wallon**. São Paulo: Loyola, 2004. p. 13–24.

MANN, C.; MAURER, F. A case study on the impact of Scrum on overtime and customer satisfaction. Paper presented at the. **Agile Development Conference**, p. 1–10, 2005. Disponível em: <http://ebe.cpsc.ucalgary.ca/uploads/Publications/MannMaurerAU2005.pdf>.

MARCHAND, P.; RATINAUD, P. L'analyse de similitude appliquée aux corpus textuels : les primaires socialistes pour l'élection présidentielle française (septembre-octobre 2011). **Actes des 11èmes Journées Internationales d'Analyse des Données Textuelles (JADT)**, p. 687–699, 2012. Disponível em: <http://lexicometrica.univ-paris3.fr/jadt/jadt2012/Communications/Marchand, Pascal et al. - L'analyse de similitude appliquee aux corpus textuels.pdf>.

MARIANO, A. M.; GARCÍA CRUZ, R.; ARENAS GAITÁN, J. Meta Análises Como Instrumento de Pesquisa : Uma Revisão Sistemática da Bibliografia Aplicada ao Estudo das Alianças Meta Analysis as a Tool of Research : A Systematic Review of Bibliography Applied Study of International Strategic Alliances . **Revista De Congreso**, n. August 2016, p. 12, 2011.

MARIANO, A. M.; ROCHA, M. Revisão da Literatura: Apresentação de uma Abordagem Integradora. **XXVI Congresso Internacional de la Academia Europea de Dirección y Economia de la Empresa (AEDEM)**, v. 26, p. 427–443, 2017. Disponível em: <https://www.researchgate.net/profile/Ari_Mariano/publication/319547360_Revisao_da_Literatura_Apresentacao_de_uma_Abordagem_Integradora/links/59beb024aca272aff2dee36f/Revisao-da-Literatura-Apresentacao-de-uma-Abordagem-Integradora.pdf>.

MAROLDI, A. M.; LIMA, L. F. M.; HAYASHI, M. C. P. I. Vida média e obsolescência da literatura em educação indígena. **InCID: Revista de Ciência da Informação e Documentação**, v. 9, n. 1, p. 109, 2018. Disponível em: <http://www.revistas.usp.br/incid/article/view/134889>.

MAZUCO, A. S. C. Percepções de Práticas Ágeis em Desenvolvimento de Software : Benefícios e Desafios. **Mestrado Profissional em Computação Aplicada**, 2017.

MCCRAE, R. R.; COSTA, P. T. Trait explanations in personality psychology. **European Journal of Personality**, v. 9, p. 231–252, 1995.

MELIN, G.; PERSSON, O. Studying research collaboration using coauthorships. **Scientometrics**, v. 36, n. 3, p. 363–377, 1996.

MELO, C. O.; SANTOS, V.; CORBUCCI, H.; KATAYAMA, E.; GOLDMAN, A.; KON, F. Métodos ágeis no Brasil: estado da prática em times e organizações. **Relatório Técnico RT-MAC-2012-03. Departamento de Ciência da Computação. IME-USP.**, n. 2010, p. 9, 2012. Disponível em: <http://www.agilcoop.org.br/files/metodos_ageis_brasil_estado_da_pratica_em_times_e_organizacoes.pdf>.

MELO, C. O.; SANTOS, V.; KATAYAMA, E.; CORBUCCI, H.; PRIKLADNICKI, R.; GOLDMAN, A.; KON, F. The evolution of agile software development in Brazil: Education, research, and the state-of-the-practice. **Journal of the Brazilian Computer Society**, v. 19, n. 4, p. 523–552, 2013.

MISRA, S. C.; KUMAR, V.; KUMAR, U. Identifying some important success factors in adopting agile software development practices. **Journal of Systems and Software**, v. 82, n. 11, p. 1869–1890, 2009. Disponível em: <http://dx.doi.org/10.1016/j.jss.2009.05.052>.

MOE, N. B.; DINGSØYR, T.; DYBÅ, T. A teamwork model for understanding an agile team: A case

study of a Scrum project. **Information and Software Technology**, v. 52, n. 5, p. 480–491, 2010. Disponível em: <http://dx.doi.org/10.1016/j.infsof.2009.11.004>.

MUTOMBO, E. A bird's-eye view on the EC environmental policy framing: Ten years of Impact assessment at the commission. **International Conference on Public Policy**, v. 1, 2013.

NERUR, S.; MAHAPATRA, R.; MANGALARAJ, G. Challenges of migrating to agile methodologies. **Communications of the ACM**, v. 48, n. 5, p. 72–78, 2005. Disponível em: <http://portal.acm.org/citation.cfm?doid=1060710.1060712>.

NEWMAN, J. Coauthorship networks and patterns of scientific collaboration. **Proceedings of The National Academy of Sciences**, v. 101, n. 1, p. 5200–5205, 2004.

OHNO, T. **Toyota production system: beyond large-scale production**. [s.l.] Press, 1988.

PAETSCH, F.; EBERLEIN, A.; MAURER, F. Requirements engineering and agile software development. **WET ICE 2003. Proceedings. Twelfth IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, 2003.**, p. 308–313, 2003. Disponível em: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1231428>.

PETERS, J. F.; PEDRYCZ, W. Engenharia de software: teoria e prática. v. 681, n. 519.683, p. 2, 2001.

PRESSMAN, R. S. **Engenharia de Software**. 6. ed. São Paulo: Mc Graw Hill, 2006.

PRESSMAN, R. S. **Engenharia de Software – Uma Abordagem Profissional**. 7. ed. [s.l.] McGraw-Hill, 2011.

PRICE, D. Networks of scientific papers. **Science**, v. 149, n. 3683, p. 510–515, 1965.

PRIKLADNICKI, R.; WILLI, R.; MILANI, F. **Métodos Ágeis Para Desenvolvimento De Software**. Porto Alegre: Bookman, 2014.

QUMER, A.; HENDERSON-SELLERS, B. Comparative Evaluation of Xp and Scrum Using the 4D Analytical Tool ( 4-Dat ). **European and Mediterranean Conference on Information Systems (EMCIS) 2006**, n. October 2015, p. 1–8, 2006.

QUMER, A.; HENDERSON-SELLERS, B. A framework to support the evaluation, adoption and improvement of agile methods in practice. **Journal of Systems and Software**, v. 81, n. 11, p. 1899–1919, 2008.

REINERT, M. ALCESTE, une méthodologie d'analyse des données textuelles et une application: Aurélia de G. de Nerval. **Bulletin de Méthodologie Sociologique**, v. 28, p. 24–54, 1990.

ROBBES, R.; HILL, E.; BIRD, C. Guest editorial: Special section on mining software repositories. **Empirical Software Engineering**, v. 23, p. 833–834, 2018.

ROYCE, W. W. Managing the development of large software systems. **Electronics**, v. 26, n. August, p. 1–9, 1970. Disponível em: <http://www.pi.informatik.tu-darmstadt.de/fileadmin/user_upload/Group_PI/LV__SE_RE/R_01_Wasserfallmodell__Folien__Schwaiger.pdf>.

SANTOS, B. S. **A Universidade no Século XXI: para uma Reforma Democrática e Emancipatória da Universidade**. São Paulo: Cortez, 2009.

SBROCCO, J. H. T. C.; MACEDO, P. C. **Metodologias Ágeis-Engenharia de Software sob medida**.

[s.l.] São Paulo: Érica, 2012.

SCHMIDT, P.; SANTOS, J. L.; MARTINS, M. A. **Avaliação de empresas: foco na análise de desempenho para o usuário interno**. [s.l.] Atlas, 2006.

SCHWABER, K.; BEEDLE, M. **Agile software development with Scrum**. Upper Saddle River: Prentice Hall, 2002.

SCHWABER, K.; SUTHERLAND, J. The Scrum Guide. **Scrum.Org and ScrumInc**, n. July, p. 17, 2013. Disponível em: <http://www.scrumguides.org/docs/scrumguide/v1/Scrum-Guide-US.pdf>.

SERRA, F. R.; FERREIRA, M. P.; ALMEIDA, M. I. R.; VANZ, S. A. S. A pesquisa em administração estratégica nos primeiros anos do século XXI: um estudo bibliométrico de citação e co-citação no strategic management journal entre 2001 e 2007. **Revista eletrônica de estratégia e negócios**, v. 2, p. 257–274, 2012.

SERRADOR, P.; PINTO, J. K. Does Agile work? - A quantitative analysis of agile project success. **International Journal of Project Management**, v. 33, n. 5, p. 1040–1051, 2015. Disponível em: <http://dx.doi.org/10.1016/j.ijproman.2015.01.006>.

SHRIVASTAVA, S. V.; RATHOD, U. Categorization of risk factors for distributed agile projects. **Information and Software Technology**, v. 58, p. 373–387, 2015. Disponível em: <http://dx.doi.org/10.1016/j.infsof.2014.07.007>.

SILVA, E. L.; MENEZES, E. M. A pesquisa e suas classificações. **Metodologia da pesquisa e elaboração de dissertação**, v. 3, n. 2, p. 19–23, 2005.

SOMMERVILLE, I. **Software engineering**. 8. ed. [s.l.] Addison-Wesley, 2007.

SUTHERLAND, J. Agile Can Scale: Inventing and Reinventing SCRUM in Five Companies. **Cutter IT Journal**, v. 14, n. 12, p. 5–11, 2001.

SVENSSON, H.; HOST, M. Introducing an agile process in a software maintenance and evolution organization BT - Ninth European Conference on Software Maintenance and Reengineering, CSMR 2005, March 21, 2005 - March 23, 2005. p. 256–264, 2005. Disponível em: <http://dx.doi.org/10.1109/CSMR.2005.33>.

TAKEUCHI, H.; NONAKA, I. The New New Product Development Game. **Harvard business review**, v. 64, n. 1, p. 137–147, 1986.

TAVARES, A. **Gerência de Projeto com PMBOK e SCRUM-Um Estudo de Caso.** Gravataí-RS: Faculdade Cenecista Senhora dos Anjos, 2008.

TRUESWELL, R. L. Some behavioral patterns of library users: The 80/20 rule. **Wilson Libr Bull**, 1969.

VIEIRA, M. **Gerenciamento De Projetos De Tecnologia Da Informação**. 2. ed. [s.l.] Elsevier Brasil, 2013.

WOMACK, J. P.; JONES, D. T.; ROOS, D. **The machine that changed the world: the story of lean production**. New York: Harper Collins, 1991.