SERVIÇO PÚBLICO FEDERAL
UNIVERSIDADE FEDERAL DO PARÁ
CAMPUS UNIVERSITÁRIO DE CASTANHAL.
FACULDADE DE COMPUTAÇÃO
CURSO DE BACHARELADO EM SISTEMAS DE INFORMAÇÃO

MARIA DAYANE MACARIO PEREIRA

ANÁLISE DE MÉTODOS DE PROCESSO DE DESENVOLVIMENTO DE SOFTWARE: UMA METODOLOGIA DE PESQUISA COMPARATIVA ENTRE MÉTODOS ÁGEIS E TRADICIONAIS

CASTANHAL – PA

2018

MARIA DAYANE MACARIO PEREIRA

ANÁLISE DE MÉTODOS DE PROCESSO DE DESENVOLVIMENTO DE SOFTWARE: UMA METODOLOGIA DE PESQUISA COMPARATIVA ENTRE MÉTODOS ÁGEIS E TRADICIONAIS

Trabalho de conclusão de curso submetido a análise da banca examinadora do curso de Bacharelado em Sistemas de Informação, como requisito para a obtenção do grau de Bacharelado em Sistemas de Informação.

Orientador: Prof. Dr. Tássio Costa de Carvalho

CASTANHAL – PAL

2018

MARIA DAYANE MACARIO PEREIRA

ANÁLISE DE MÉTODOS DE PROCESSO DE DESENVOLVIMENTO DE SOFTWARE: UMA METODOLOGIA DE PESQUISA COMPARATIVA ENTRE MÉTODOS ÁGEIS E TRADICIONAIS

APROVADA EM:_____/_____/ _____

BANCA EXAMINADORA

_____

Prof. Dr. Tássio Costa de Carvalho
(Orientador – FACOMP/UFPA)

_____

Prof. Dr. José Jailton Henrique Ferreira Júnior
(Avaliador Interno – FACOMP/UFPA)

_____

Prof. Dr. Igor Ruiz Gomes
(Avaliador Interno – FACOMP/UFPA)

_____

Prof. Me. Jorge Amaro de Sarges Cardoso
(Avaliador Interno – FACOMP/UFPA)

Dedico esse trabalho primeiramente a Deus, que foi minha maior força nos momentos difíceis; À minha querida família, por sua capacidade de acreditar em mim e me amar sem medidas, e em especial a minha amada avó.

In memoriam

AGRADECIMENTOS

Desejo em primeiro momento agradecer a Deus que permitiu que tudo isso acontecesse, e não somente nestes anos como universitária, mas em todos os momentos de minha vida, não me deixou desistir e nem abater.

A minha amada avó que ainda em vida, me incentivou e me deu todo apoio e amor do mundo. Aos meus queridos pais, Ailton Pereira e Maria Macario, que nunca desistiram de sonhar comigo, me incentivaram desde de sempre e nunca mediram esforços para a realização deste sonho e que também é de toda a família. Aos meus Irmãos e familiares, pela compreensão, incentivo, amor e apoio incondicional em todos os dias de minha vida.

Ao meu namorado Wallace Maia, que esteve comigo em todas as horas, por toda a paciência, estímulo, afeto e por não me deixar desistir quando tudo parecia ir mal.

Ao Prof. Dr. Tassio Carvalho, meu querido orientador, pela oportunidade, apoio, confiança e empenho na elaboração deste trabalho. Por todo o conhecimento que me foi passado, tornando a realização deste possível.

Agradeço a todos os professores que tive a oportunidade de conhecer, por terem me proporcionado o conhecimento que obtive, não somente por terem me ensinado, mas por terem me feito aprender, e também a Universidade Federal do Pará, pela oportunidade de fazer o curso. Sou grata também, aos membros da banca, por aceitarem o convite de participar da apresentação deste trabalho de conclusão de curso.

Meus agradecimentos aos meus queridos amigos, que a faculdade me presenteou, amizades que conquistei ao longo dessa jornada, em especial ao Gildson de Oliveira que foi meu parceiro incansável desde o início do curso, esteve ao meu lado em todos os momentos, foi meu braço direito, ao Breno Santos que veio trazer alegria ao nosso grupo, amigo de viagens de volta para casa e de boas conversas, obrigado pelo aprendizado e ao Rogério Pereira companheiro de viagens, estudos, parceiro para todas as horas. Agradeço por todo carinho e paciência, principalmente quando eu não estava nos meus melhores dias, e claro pelos bons lanches pagos.

E por último e não menos especial, agradeço ao Ranilson Pereira, que não fazia parte da minha turma, mas cultivamos uma boa amizade, foi em alguns momentos como um parceiro de trabalho, quando precisei nunca colocou dificuldades para atender aos meus chamados. Obrigada pelas conversas e por toda a assistência nessa caminhada

E a todos que não mencionei, e que direta ou indiretamente fizeram parte da minha caminhada e me motivaram a nunca desistir, o meu muito obrigado.

Não há ações sem motivos. Todo passo é um passo rumo a alguma direção. “A única coisa que importa é colocar em prática, com sinceridade e seriedade, aquilo em que se acredita.” (Dalai Lama)

SUMÁRIO

LISTA DE ABREVIATURAS E SIGLAS

ABNT	ASSOCIAÇÃO BRASILEIRA DE NORMAS E TECNICAS
CASE	COMPUTER- AIDED SOFTWARE ENGINEERING
CBSE	COMPONENT BASED SOFTWARE ENGINEERING
DSDM	DYNAMIC SYSTEMS DEVELOPMENT METHOD
FDD	FEATURE DRIVEN DEVELOPMENT
IBM	INTERNATIONAL BUSINESS MACHINES
IEEE	INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS
RAD	RAPID APPLICATION DEVELOPMENT
RUP	RATIONAL UNIFIED PROCESS
TI	TECNOLOGIA DA INFORMAÇÃO
UML	UNIFIED MODELING LANGUAGE
UFPA	UNIVERSIDADE FEDERAL DO PARÁ
XP	EXTREME PROGRAMMING

LISTA DE FIGURAS

LISTA DE TABELAS

RESUMO

A habilidade de criar e entregar mais rapidamente os produtos de *software*, com mais qualidade e que melhor reflita nas reais necessidades dos clientes, tem se tornado o desejo de muitas organizações, assim como possuir o seu produto todo documentado, planejado desde o início, com prazo estipulado e valores fixos. A escolha entre o uso de metodologias de desenvolvimento de *software*, continua sendo bastante discutida em diversos fatores. Portanto tendo como principal razão de desenvolvimento, a presente pesquisa realizou uma análise comparativa entre as metodologias ágeis e tradicionais, a partir de aspectos encontrados na literatura especializada, que serviram como objeto para a comparação, bem como realizar uma análise discursiva a partir dos resultados obtidos, além de oferecer fatores para auxílio na escolha das metodologias. Para tanto, desenvolveu-se uma pesquisa qualitativa descritiva, cuja finalidade foi observar, registrar e analisar às informações necessárias dos métodos. Com o desenvolvimento deste trabalho, espera-se que empresas, desenvolvedores e pesquisadores sejam capazes de fazerem uma boa escolha em relação aos modelos de processo de desenvolvimento de software. Todo o levantamento e análise de informações necessárias para esta pesquisa, foram realizados através de investigações bibliográficas em acervos de livros, artigos e monografias já existentes.

Palavras-chaves: Modelos de processos, desenvolvimento de *software*, metodologias ágeis, metodologias tradicionais e métodos.

ABSTRACT

The ability to create and deliver software products faster, with better quality and better reflected in the real needs of customers, has become the desire of many organizations, as well as having their entire documented product planned from the beginning with term and fixed amounts. The choice between the use of software development methodologies, is still discussed in many factors. Therefore, having as main reason for development, this research carried out a comparative analysis between agile and traditional methodologies, based on aspects found in the specialized literature, that served as object for the comparison, as well as perform a discursive analysis from the results obtained, besides offering factors to aid in the choice of methodologies. For that, a descriptive qualitative research was developed, whose purpose was to observe, record and analyze the necessary information of the methods. With the development of this work, it is expected that companies, developers and researchers will be able to make a good choice in relation to the software development process models. All the information required for this research was collected and analyzed through bibliographic research in existing books, articles and monographs.

Key-words: Process models, software development, agile methodologies, traditional methodologies and methods.

1. INTRODUÇÃO

O presente capítulo apresenta a introdução da pesquisa proposta, contextualizando os seguintes tópicos respectivamente: Contextualização onde está abordado como era o cenário de desenvolvimento de software; Problemática ressaltando os problemas existentes nos processos de desenvolvimento; Motivação ou justificativa pela qual o presente trabalho foi elaborado; Objetivos gerais e específicos; Metodologia de pesquisa a qual trata que metodologia foi utilizada e em fim a estrutura do trabalho que apresenta como estará divido o presente documento.

1.1 CONTEXTUALIZAÇÃO

O processo de desenvolvimento de *software*, tem avançado muito nos últimos anos, mas muitas barreiras ainda devem ser quebradas e desafios deveram ser vencidos. Nas primeiras décadas, o processo de desenvolvimento de software podia ser caracterizado como direcionado a pratica, sem uma base conceitual ou de alguma metodologia que fosse coerente. Naquele momento era um processo quase que individual. E certamente, as questões financeiras deveria encabeçar a lista de maiores desafios neste tempo, no desenvolvimento de software, afinal de contas, planejar e produzir um bom software envolve gastos com recursos que nem sempre estão disponíveis na empresa.

O principal desafio na área de engenharia de software nas últimas décadas, “tem sido o estudo, a melhoria da qualidade e redução de custo de software, que vinham sendo produzidos” (PRESSMAM, 2016). De uma maneira geral, pode-se entender engenharia de software, como sendo uma disciplina que agrupa metodologias, métodos e boas práticas a serem utilizadas.

Ao longo do tempo o processo de desenvolvimento de software, passou por uma grande evolução, uma mudança radical, em termos de metodologia e abordagens de desenvolvimento de software, que é uma prática que exige muita organização e planejamento. Se o objetivo é oferecer um produto de qualidade, sem ter gastos exorbitantes em sua criação, é fundamental definir estratégias para otimizar o processo do início ao fim. Entretanto, na prática isso pode ser um grande desafio.

Como solução para amenizar problemas e vencer desafios, surgiram propostas de maior rigor e modelos e métodos no processo de desenvolvimento. Os modelos de processos de desenvolvimento de software, surgiram pela necessidade de dar respostas as

situações a analisar, porque só na altura em que se enfrenta o problema, é que pode-se escolher o modelo.

Dependendo de sua aplicação, ambiente e objetivo, a utilização de um processo ou modelo especifico, pode ser vantajoso ou não. Cabe a cada organização avaliar o seu problema com cuidado, e usar os modelos apresentados, como um guia para o desenvolvimento do seu próprio processo de desenvolvimento.

Conforme Sommerville (2011, p.), "um modelo de processo é uma visão geral acerca de todas as atividades e tarefas a serem desempenhadas, para se alcançar o objetivo final, em que esse conjunto de tarefas trata-se da metodologia a ser aplicada".

Em dado momento da história, os métodos tradicionais foram os responsáveis por ordenarem, de modo sistemático o processo de desenvolvimento de software, para tentar terminar com a desordem, que por sua vez, foi crescendo conforme o tempo foi passando e ainda existe. (PRESSMAN, 2016).

Logo depois, com o intuito de diminuir lacunas não preenchidas pelas metodologias tradicionais, e ainda com novas características que abordam áreas diferentes do contexto tradicional de desenvolvimento, surgem novas metodologias, os "métodos ágeis, apresentando quantidade reduzida de documentação e novos enfoques dentre suas fases" (PRESSMAN, 2016), enquanto os métodos tradicionais possuem documentação em todas as suas fases de desenvolvimento, gerando grandes quantidades de artefatos.

No atual cenário dos ambientes organizacionais, o número de empresas que estão optando por soluções, com uso de modelos de processo, como metodologias ágeis e tradicionais, vem crescendo significativamente, buscam com isso novos direcionamentos para aumentar lucros em seus negócios, diminuir riscos e ainda obter resultados que superem suas expectativas.

Com o avanço desenfreado da TI (Tecnologia da Informação), a existência de empresas no mercado, têm se tornado complexa, pois a medida em que o mercado de tecnologia cresce, torna-se mais exigente em suas entregas e com mais frequência deseja produtos com qualidade, por conta disso, veio se tornando necessário adequar e organizar muitos processos dentro dos ambientes organizacionais, especialmente na área de desenvolvimento de software.

Diante disso, é recomendável analisar como anda o desenvolvimento de software e de novas tecnologias, o que vem ocorrendo, e de que forma estão sendo realizados, pois, a maioria das organizações não usam uma metodologia padrão para o desenvolvimento,

vão se organizando de acordo com o que vai surgindo, é inegável que algumas vezes pode dar certo e surgir um bom produto, porém é indispensável para um desenvolvimento eficaz e que resulte em um produto de qualidade, o uso de um modelo de processo, como uma metodologia para que se melhor chegue aos objetivos desejados.

Todavia, além das metodologias, faz-se necessário um estudo na íntegra para identificar qual possui uma melhor adaptabilidade com o projeto escolhido, surge então a pergunta: qual metodologia corresponde com o cenário da minha empresa e assim pode atender as necessidades determinadas? Vale ressaltar que o modelo deve ser escolhido de acordo com as necessidades da área de aplicação da empresa, pois cada um atende uma necessidade diferente.

Em continuidade, é necessário também focar nos clientes, visando: qualidade, atendimento e prazo. Já os processos devem ser ágeis, produtivos e de boa qualidade. Em relação à equipe de desenvolvimento de software, é preciso considerar: competências, autonomia e colaboração. Por fim, no que se refere aos resultados financeiros, o software deve assegurar: boas vendas, bons lucros e baixos custos.

Por fim, um dos fatores determinantes para o sucesso do desenvolvimento de software, é sem dúvidas em uma empresa, a capacidade de implantar um modelo de processo para desenvolver seus projetos com qualidade.

1.2 PROBLEMATICA E JUSTIFICATIVA

Atualmente o mercado de desenvolvimento de software, é um dos mais influentes difundido pelo mundo todo e com elevada demanda, entretanto, desenvolver *software* ainda é um grande desafio, mesmo com a engenharia de *software* com anos de existência, ainda assim, é possível encontrar sérios problemas, os quais vão desde mudanças dos requisitos ao longo do desenvolvimento, prazos em atrasos, além de desenvolvedores e organizações que ainda não possuem uma metodologia padrão, que guie suas atividades, permitindo assim um desenvolvimento de qualidade.

Levando em conta as necessidades de *softwares*, cada vez mais funcionais e sofisticados, os desenvolvedores precisam encontrar soluções para superar os obstáculos, e atender a demanda de um público cada vez mais exigente. Além de que, o ambiente de desenvolvimento de *software* tem crescido de maneira impressionante, ocasionando maior complexidade em seus processos e o ato de realizar as atividades sem uma forma

de controlar e gerenciar os projetos, resulta na maioria das vezes em produtos de má qualidade.

Ao longo dos anos, os problemas relacionados ao processo de desenvolvimento de software ficam mais evidentes, à medida que ocorre o avanço das tecnologias, e as organizações ficam cada vez mais dependentes delas. Problemas como: alto custo, alta complexidade, dificuldade de manutenção, grande diferença entre os requisitos exigidos pelos usuários e o produto final entregue, ou seja, a discordância com o objetivo do cliente é muito presente, a falta de compreensão dos requisitos pode consequentemente alterar todo o resultado de um projeto.

De acordo com Pressman (2016), no atual cenário da indústria de *software*, os principais problemas relacionados ao setor de desenvolvimento, estão ligados à falta de documentação do *software* e comunicação. Identificam-se na literatura alguns outros problemas como: a aceitação do software pelo usuário, tempo de desenvolvimento muito longo, documentação inexistente, incompleta ou desatualizada.

Segundo o PMSurvey (PMSURVEY.ORG, 2014), 64,2% dos projetos tem problemas de comunicação, 58,5% de escopo mal definido e 54,2% mudanças de escopo constantes. A Figura 1 a seguir, demonstra o índice de causas de falhas no desenvolvimento de software, e a falta de um processo ágil possui um indicativo de 10%, ou seja, o uso de um modelo de processo possui papel fundamental na melhora pelo desenvolvimento seja ele tradicional ou ágil.

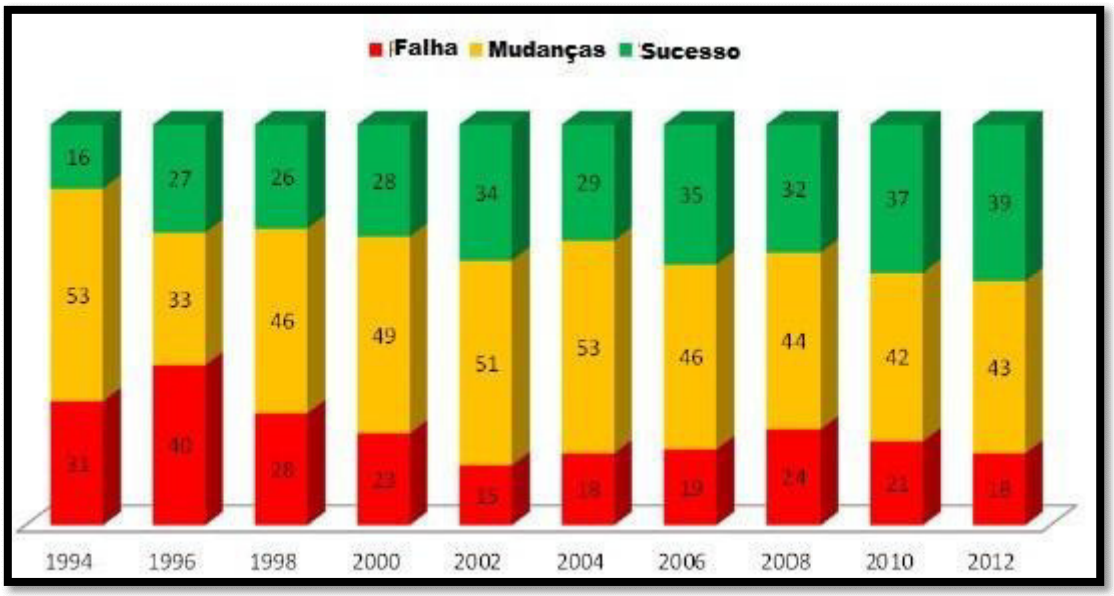
Figura 1. Principais causas de falhas em projetos

Fonte: Adaptado de Standish Group, (2013, p.05)

Em concordância, a partir de informações do *Gartner Group Research*, foi apresentado também os principais problemas do desenvolvimento de *software*, que para ele são: cumprimento dos prazos, custos elevados, manutenção de sistemas em uso, conflito entre desenvolvimento e manutenção e qualificação dos profissionais entre outros problemas existentes.

Seguidamente na Figura 2, é apresentado um gráfico que demonstra o percentual de falha, mudanças e sucesso nos projetos de desenvolvimento de software, que foi publicado no relatório gerado pelo Standish Group em março de 2013.

Figura 2. Percentual de falhas, mudanças e sucesso em projetos de software



Fonte: Adaptado de Standish Group, (2013)

A partir da figura, é possível perceber que houve uma visível melhora, desde a 1ª verificação em 1994 até a última em 2012, onde, de apenas 16% de sucesso nos projetos, alcançou-se os 39%. Isso expressa uma evolução de quase 100%. Todavia considerar que de todos os projetos, apenas 39% destes são entregues em conformidade com o planejado, torna-se um fator preocupante, segundo o relatório apresentado no *Standish Group* (2013).

Um estudo do Standish Group (2013), com 3.555 projetos de TI, revelou que somente 6% deles foram bem-sucedidos, entregues dentro do prazo e em conformidade com o objetivo desejado. Do mesmo modo, 52% dos grandes projetos foram considerados desafiadores, exemplo daqueles que utilizam um orçamento maior do que o previsto e

todos os demais projetos, por volta de 42%, fracassaram completamente ou foram cancelados em algum momento.

Partindo desse pressuposto, a presente pesquisa busca tentar esclarecer tais problemas relacionados ao desenvolvimento de software, que com base nos números apresentados anteriormente tem crescido bastante e de alguma maneira esforça-se para auxiliar organizações a compreenderem o que tem causado tantos softwares desenvolvidos com má qualidade.

1.3 MOTIVAÇÃO

O desenvolvimento de presente trabalho, está inteiramente destinado e focado em contribuir de alguma maneira com pessoas da área e estudiosos, que buscam se aprofundar no assunto.

Mediante o levantamento bibliográfico, observou-se a necessidade da pesquisa especializada e aprofundada a respeito dos métodos de desenvolvimento, visto que muitas organizações andam sofrendo com produtos deficientes ocasionando perca de clientes. Trata-se de uma abordagem presente na sociedade, pois o meio tecnológico está difundido em todos os segmentos do mercado. Contudo, é satisfatório realizar uma pesquisa, abordando um tema que foi muito expressivo ao longo da caminhada acadêmica.

A utilização de *software*, tem sido apontada como um fator primordial para o sucesso de empresas de desenvolvimento, desta maneira realizar uma pesquisa sobre, é indispensável. Portanto, a presente pesquisa visa a partir da investigação sobre informações e dados do tema proposto, permitir compreender mais sobre o desenvolvimento de software com auxílio dos modelos de processo, para que os conhecimentos adquiridos, também sejam aplicados em projetos quando necessário.

O intuito também é de auxiliar no processo de desenvolvimento caótico e desorganizado de algumas organizações e empresas, ou simplesmente no desenvolvimento de *software* na área acadêmica ou afins. Espera-se que com a implantação de uma metodologia de desenvolvimento, se consiga uma maior organização e uma maior agilidade na tarefa de produzir.

Em conclusão, o uso de metodologias, mesmo que ainda não fortemente consolidadas, no desenvolvimento de *software*, é de extrema importância, para que o sistema construído, atenda às necessidades dos interessados, com um mínimo de

qualidade, ou seja, o desenvolvimento desta pesquisa é abrangente, e pode auxiliar em diversos cenários.

1.4 OBJETIVOS

Nesta etapa, apresenta-se os objetivos do estudo, que serão utilizados para o desenvolvimento do presente trabalho e desta forma, estão listados abaixo os objetivos gerais e específicos.

1.4.1. OBJETIVO GERAL

O principal propósito deste trabalho é efetuar uma análise comparativa sobre os modelos de processos de desenvolvimento de software, metodologias ágeis e metodologias tradicionais. Com o objetivo de auxiliar no entendimento e escolha de métodos para desenvolvimento.

1.4.2. OBJETIVOS ESPECIFICOS

- Revisar e analisar a literatura a respeito das metodologias tradicionais e ágeis, que são modelos de processo de desenvolvimento de software;
- Aprofundar os conhecimentos sobre os modelos de processo;
- Identificar os principais tipos de metodologias tradicionais e ágeis, promovendo um estudo sobre suas características que em seguida servirão para objeto de comparação;
- Em seguida ao estudo da literatura, será realizado uma análise comparativa, onde os modelos de processo tradicionais e ágeis serão confrontados demonstrando suas características distintas e similares, para assim tentar auxiliar organizações na escolha;
- Criar uma análise textual discursiva qualitativa dos resultados obtidos, a partir da comparação das metodologias;
- Descrever fatores para a escolha de metodologias;

1.5 METODOLOGIA DE PESQUISA

Para a elaboração dessa pesquisa, foi realizada uma revisão da literatura especializada referente ao tema proposto. As fontes de pesquisas, foram descobertas através de investigações bibliográficas em acervos de livros, artigos e monografias já existentes.

A metodologia possui um objetivo exploratório, pois se deseja analisar e investigar informações relevantes sobre os métodos tradicionais e ágeis, além de tentar esclarecer os problemas que estão envolvidos na insatisfação quanto ao desenvolvimento de *software*.

Neste trabalho, não é o objetivo chegar a conclusões sobre a definição de termos. Portanto, será utilizado os termos metodologia, método e modelo de desenvolvimento, como sinônimos para representar formas de trabalho encontradas na literatura ou usadas por equipes de desenvolvimento de software, para tratar os modelos de processo de desenvolvimento de *software*.

A metodologia aplicada sobre o presente tema, baseou-se em quatro momentos, o primeiro consistiu na pesquisa de caráter exploratório e descritivo em bases de pesquisas, tendo como suporte, pesquisas bibliográficas dos principais autores da área, como PRESSMAN E SOMMERVILLE, dentre outros autores citados, para a melhor compreensão dos assuntos citados.

O segundo momento, consta os trabalhos que servirão de base para o desenvolvimento deste, sendo analisados e comparados com esta pesquisa.

No terceiro momento, está a parte mais importante do trabalho, com uma abordagem qualitativa, que compreende ao estudo de caso descritivo, em que será realizado a análise de características dos métodos tradicionais e ágeis para o comparativo, e a partir daí, é feita uma discussão com base nos resultados obtidos.

E o último momento, é destinado as considerações finais e trabalhos futuros. Nele será descrito o que foi observado de maior relevância, ao longo do trabalho, transcorrendo pontos como resultados encontrados, objetivos, além de dificuldades que foram encontrados ao decorrer do desenvolvimento desse trabalho e em seguida demonstrar propostas para um futuro trabalho ou até mesmo a melhoria deste.

1.6 ESTRUTURA DO TRABALHO

A monografia é apresentada em cinco capítulos, no primeiro capítulo foram apresentadas informações de caráter introdutório a respeito do tema, onde estão presentes os objetivos, a problemática, motivação entre outros.

Os próximos capítulos estão distribuídos da seguinte forma: O segundo capitulo, abordará todo o referencial teórico envolvido no presente trabalho, que serviu como base para o desenvolvimento do mesmo. O terceiro capitulo, apresentará os trabalhos

correlatos, estudos que contribuíram para a elaboração e discussão deste. O quarto capitulo, demonstrará a parte principal do trabalho, onde será realizado a pesquisa qualitativa descritiva e em seguida será tratado as considerações finais, com trabalhos futuros e dificuldades encontradas. Esta monografia é finalizada com as referências.

2. FUNDAMENTAÇÃO TEÓRICA

Neste capítulo, são abordados os conceitos e posições dos autores que serão utilizados como referência no presente trabalho. Formando um referencial teórico que será a base de todo a pesquisa.

2.1 ENGENHARIA DE SOFTWARE

A engenharia de *software*, pode ser considerada como uma das tecnologias de maior importância crítica para o futuro da humanidade. Tornou-se uma área imensa, abrangendo muitas abordagens. Sua importância é crescente, e cada vez mais pessoas de maneira em geral dependem dos sistemas de softwares econômicos e desenvolvidos da melhor forma possível.

Para Sommerville, a engenharia de *software* é importante por dois motivos:

> 1.A sociedade, cada vez mais, depende de sistemas de software avançados. 2. e a longo prazo, normalmente, é mais barato usar métodos e técnicas propostos pela engenharia de software, ao invés de somente escrever os programas como se fossem algum projeto pessoal. (SOMMERVILLE, 2011.p.5).

Ainda de acordo com Sommerville (2011), "a engenharia de *software* se torna essencial para o total funcionamento da vida em sociedade, seja ela nacional ou internacional", pois estar presente nos mais diversos segmentos.

O conceito de engenharia de *software*, surgiu por volta de 1960, exatamente e inicialmente proposto em 1968, em uma conferência organizada para tratar a "Crise do software", que era a abordagem da introdução de um novo hardware de computador baseado em circuitos integrados.

Estouro nos prazos, falta de documentação e baixa qualidade dos produtos foram alguns dos problemas encontrados durante a "Crise do *software*". "Como solução para estes problemas, surgiu a engenharia de software com suas técnicas, ferramentas e métodos" (SOUZA, 2014.p.). A partir disso seu progresso foi excepcional, e a engenharia de *software* teve e ainda tem, papel fundamental nas tecnologias.

Sommerville (2011), define engenharia de *software* da seguinte forma:

> Engenharia de *software* é uma abordagem sistemática para a produção de software, é uma disciplina de engenharia, cujo foco está em todos os aspectos da produção de software, desde os estágios iniciais de especificação do sistema até sua manutenção. (SOMMERVILLE, 2011.p.5;8).

Tanto o IEEE (1990, p.67) quanto Pressman (2016), definem engenharia de software como "uma aplicação de abordagem sistemática e disciplinada". Pressman (2016 apud IEEE, 1990.p.67) "diz ainda, que é quantificável para o desenvolvimento, operação e conservação do software".

Williams (2010), estabelece que a engenharia de software segue o conceito de disciplina na produção de software. Afirma ainda, que é fundamentado em metodologias, e que por sua vez, seguem métodos que utilizam ferramentas automáticas, para englobar as principais atividades do processo de produção.

De acordo com suas verificações, Pressman (2016), declara que: "software, em todas as suas formas e em todos seus campos de aplicação, deve passar pelos processos de engenharia".

Pressman (2016), define software como:

> *Software* de computador é o produto que os profissionais de *software* constroem e, depois, mantêm ao longo do tempo. Abrangem programas que executem em computadores de qualquer tamanho e arquitetura. É um conjunto composto por instruções de computador, estruturas de dados e documentos.

"Quando se fala em engenharia de *software*, não se trata apenas de programas em si, mas de toda uma documentação associada, técnicas necessárias para permitir que um programa possa operar corretamente" (SOMMERVILLE, 2011.p.3). Ela tem a ver com obter resultados de qualidade, por isso a pedra fundamental que a sustenta é o foco na qualidade, sendo que os principais atributos de um produto de software, são manutenibilidade, confiança, proteção, eficiência e aceitabilidade.

Para Pressman (2016), existem pelos menos duas características que diferenciam o software do hardware, essas serão apresentadas a seguir:

- Software é desenvolvido ou passa por um processo de engenharia, não sendo fabricado no sentido clássico;

- Software não se desgasta.

A partir das características descritas por Pressman, chega-se à conclusão, de que acordo com a primeira opção, pode-se dizer que existem diversas semelhanças entre o desenvolvimento do software e a fabricação do hardware, porém as atividades executadas em ambos para o desenvolvimento, são muito diferentes. Em relação a segunda característica, o software não está sujeito aos problemas ambientais, que são fatores determinantes de desgaste do hardware.

Dando continuidade quanto a definição da engenharia de software, em conformidade com Pressman (2006), a engenharia de Software "é uma tecnologia em três camadas: processos, métodos e ferramentas". Cada ação apresentada por estas três camadas, se fazem necessárias para a construção de um *software*, logo que, fornecem as técnicas necessárias para a efetuação do mesmo e são responsáveis em apresentar um resultado com um produto final.

A engenharia, engloba a todos esses fatores especificados para o desenvolvimento. Pode verificado de acordo com a Figura 3 a seguir, que foco na qualidade é a primeira camada, assim sendo, a engenharia é fundamentada na qualidade e isso explica o fator de estar no início, ela é uma espécie de base. Em seguida e apresentado os processos, que é a abordagem que facilita a seleção de tarefas necessárias para assim tornar possível o trabalho.

Seguidamente estão presentes os métodos e as ferramentas. Os métodos fornecem técnicas para desenvolver o software, envolve diversas tarefas como comunicação, analise de requisitos entre outros, e por fim as ferramentas, que fornecem o suporte automatizado para a camada de processo e métodos.

Figura 3. Engenharia de software em camadas

Fonte: Pressman, (2006, p.17)

A abordagem sistemática usada na engenharia de software é, em algumas vezes chamada processo de software" (SOMMERVILLE, 2011.p.5). Este termo será tratado no capitulo a seguir. Para que um software seja produzido, são necessárias diversas etapas, compostas de uma série de tarefas em cada uma delas, conforme apresentado ao longo do próximo tópico.

2.2 PROCESSO DE DESENVOLVIMENTO DE SOFTWARE

Com o passar do tempo, técnicas de engenharia de *software* passaram a ser usadas para o desenvolvimento de software, objetivando definir processos para estabelecer a criação de sistemas, como base no uso de ferramentas e uma documentação detalhada de tudo o que foi realizado.

De acordo com a Associação Brasileira de Normas e Técnicas (ABNT, 1995), processo, pode ser definido como "um conjunto de atividades inter-relacionadas, que converte entradas e saídas". O processo pode ser tratado inicialmente como uma abstração do conhecimento, que deve se transformar em um produto de *software*.

Conforme Costa (2017, P.15), processos de software, tem o objeto de supervisionar e coordenar projetos de desenvolvimento verdadeiros. Impulsionam a coordenação e organização no desenvolvimento de *software*.

Com esse mesmo contexto, pode-se definir o processo de software "como uma série de passos que devem ser seguidos durante o andamento do desenvolvimento do sistema" (PRESMMAN,2016, p.18). O autor afirma ainda, que esta definição é bastante significativa, pelo motivo de que o processo proporciona estabilidade, controle e organização para a realização das atividades propostas.

Na década de 1970, a atividade de "desenvolvimento de *software*" era desempenhada de maneira desordenada, sem nenhuma estrutura e nem se quer, possuía um planejamento, tudo isso acarretava em um produto final de má qualidade, que não estava de acordo com o esperado pelo cliente, logo não atendia suas necessidades. (PRESSMAN, 2016).

As necessidades, no desenvolvimento de *software* eram visíveis, tornando-se necessário a melhoria em seu processo, para assim permanecer em um estado de organização, estruturação e planejamento, e, contudo, atender as necessidades e objetivos especificados pelo cliente.

Nenhum tipo de *software* deve ser desenvolvido de qualquer jeito, sem seguir critérios, sem que se saiba qual o próximo passo a ser dado. Por isso que o processo de software relacionado à engenharia de software deve ser utilizado.

Um processo de software auxilia na produção de sistemas, permiti deixar organizado, os passos necessários para a produção. Pode ser definido como "uma coleção de padrões que definem um conjunto de atividades, ações, tarefas de trabalho ou comportamentos relacionados e necessários ao desenvolvimento de *software* de computador" (ALVES, 2012, p).

Segundo Souza (2014), "O processo de *software*, é como uma receita a ser seguida para a execução de um projeto". Os processos são importantes porque imprimem consistência e estrutura a um conjunto de atividades, orientando desta forma as ações necessárias.

Para os processos de *software*, Sommerville (2011, p.18), aponta quatro atividades fundamentais descritas a seguir, que são comuns a todos os processos. Afirma ainda, que elas de alguma maneira devem fazer parte de todos os processos de *software*.

- Especificação de *software*: É onde o cliente e o desenvolvedor definem o *software* a ser produzido e suas restrições e limitações;
- Desenvolvimento de *software*: Atividade em que o *software* é projetado e programado;
- Validação de software: O *software* é verificado para garantir que estar sendo desenvolvido o que o cliente desejou;
- Evolução de *software*: Nesta atividade, o *software* é modificado para refletir a mudança de requisitos do cliente e do mercado se necessário.

Além das atividades essenciais citadas acima, um processo de software de qualidade, inclui algumas atividades diferenciadas. Estas, existem para dar suporte e tornar viável a execução das atividades fundamentais consideradas por Sommerville. Que possuem como objetivo, aperfeiçoar os recursos e diminuir riscos.

A seguir, de maneira resumida, serão apresentadas as atividades classificadas como auxiliares, são:

- Gerencia de alocação: Esta atividade tem como objetivo maximizar a produtividade da equipe envolvida e ainda administrar recursos como espaço e equipamentos;

- Gerencia de cronograma: É de conhecimento que geralmente um software possui um prazo determinado para desenvolvimento e um prazo esperado para entrega. Esta possui o objetivo de organizar questões de desempenho de equipe e funcionalidades desejadas;
- Gerencia de configuração: Atividade que tem como objetivo o controle de artefatos de projeto. Objetiva armazenar, revisar, auditar, e relatar as alterações realizadas ao longo do tempo;
- Documentação: O gerenciamento, a criação e atualização de documentação são consideradas essenciais para um bom processo. Ao longo do processo de desenvolvimento de software, diversas tarefas e atividades, como também funcionalidades, são descritas.

De acordo com Sommerville (2011, p.), "tipos diferentes de sistemas necessitam de diferentes processos de desenvolvimento, pois cada um tem seu proposito especifico". Para isso, atividades consequentemente terão sequência diferente para cada software desenvolvido.

De acordo com o levantamento bibliográfico realizado, existem quatro classes de processos de *software*:

1. Processos informais: Não existe processo definido, a equipe de desenvolvimento escolhe que processo será utilizado;
2. Processos gerenciados: Existe um modelo de processo definido, a programação e os relacionamentos dos procedimentos são seguidos;
3. Processos metódicos: É usado um ou mais métodos, esse método normalmente é utilizado em sistemas que passaram por reengenharia;
4. Processos de aprimoramento: São processos com objetivos de aprimoramento, possuem orçamentos para esse fim. Pode possuir uma avaliação quantitativa.

Para documentar e dar suporte as atividades executadas em um processo de desenvolvimento, Segundo Gudwin (2015, p.6), "um conjunto de artefatos de software são criados, desenvolvidos em uma determinada linguagem de modelagem". Exemplos de artefatos de *software* são diagramas, gráficos, textos e tabelas com finalidades explicitas.

Gudwin (2015, p.6) define linguagem de modelagem da seguinte forma:

> Linguagem de modelagem corresponde a uma linguagem padronizada na qual, artefatos de *software* possam ser desenvolvidos. Essa linguagem e geralmente uma linguagem visual, que especifica a sintaxe e a semântica de classes de diagramas, utilizados explicitamente para modelar partes de um sistema de *software* ou de planos de construção de sistemas de *software*.

De acordo com alguns estudiosos, as linguagens de modelagem começaram a aparecer na engenharia de software no final dos anos 70, com o desenvolvimento dos primeiros sistemas orientados a objeto. A partir daí, foram sucessivamente empregadas em diversos experimentos e em diferentes abordagens.

Para aprimorar os processos de software, surgiram dois modelos que auxiliaram desenvolvedores a chegarem a seus objetivos da melhor maneira possível, e a seguir serão apresentados estes modelos e os tipos que os compõe.

2.3 METODOLOGIAS DE PROCESSO DE SOFTWARE

O uso de uma metodologia para desenvolver software, é o ponto chave para o sucesso do projeto. Alistair Cockburn (2002 apud Santos, 2017, p. 34), define o termo metodologia como “uma série de métodos e técnicas”, onde método se refere a “procedimentos sistemáticos”.

Segundo Sommerville (2011), um modelo de processo de software, é uma representação abstrata, mais simples, de um processo de *software*. Os modelos de processo, incluem as atividades que fazem parte do processo. “Cada modelo representa uma perspectiva particular de um processo e, portanto, fornece informações parciais sobre ele” (SOMMERVILLE, 2011.p.19).

Conforme Oliveira (2018, p.25), “Um processo ou metodologia de desenvolvimento de software, consiste em um conjunto de atividades e resultados associados que auxiliam na produção de uma aplicação”.

Já Costa (2017, p.11), define esse mesmo processo ou metodologia, “como um conjunto de processos que contribuem na fabricação de um software, e o efeito desses conjuntos, é um produto que expõe a maneira como todo o desenvolvimento foi conduzido”.

Desde 1960, surgiram várias descrições do clássico ciclo de vida de software. Basicamente, o objetivo desses modelos, é demonstrar um esquema conceitual para auxiliar o desenvolvimento e fornecer um suporte maior ao desenvolvedor da aplicação. “A classificação mais encontrada na literatura especializada” (SANTOS, 2007.p.34), com Pressman (2016), Sommerville (2011), Cockburn (2002), Fowler (2003), é a seguinte:

- Metodologias Tradicionais, ou “pesadas”;
- Metodologias Ágeis, ou “leves”.

A seguir, os modelos de processo que serão apresentados e descritos, foram retirados de Sommerville (2011) e Pressman (2016), e de alguns outros autores que serão mencionados ao longo do estudo.

2.4 METODOLOGIA TRADICIONAL DE SOFTWARE

As metodologias tradicionais, são também chamadas de pesadas ou orientadas a documentação. “Surgiu nas circunstâncias de desenvolvimento de software bem discrepante do atual, naquela época era baseado apenas em mainframes e terminais burros” (ROYCE, 1987, p).

Os modelos de processos tradicionais, possuem uma grande importância na história da engenharia de software, por serem amplamente utilizados no auxílio de desenvolvimento pela indústria de software, e ainda muito utilizado hoje, pois ainda existem organizações que preferem continuar com o desenvolvimento de produtos, a partir do suporte desse modelo.

Surgindo com o intuito, de acabar com os insucessos no desenvolvimento de software. O objetivo principal das metodologias tradicionais, é permitir a previsão dos requisitos do sistema, que tem como proposito, tornar os projetos completamente planejados, facilitando a organização dos mesmos.

Segundo Pressman (2016), a metodologia tradicional tem como principal característica a total documentação do software antes de sua implementação e não são acessíveis a mudanças ao longo da produção. De acordo com Oliveira (2017, p.27), esta característica descrita, é a principal limitação dos modelos tradicionais, diz que seu “foco é a previsibilidade dos requisitos do sistema”.

A seguir é apresentada a Tabela 1 que contém algumas das vantagens e desvantagens a respeito dos modelos de processos tradicionais, quanto a sua utilização em desenvolvimento de software.

Tabela 1. Metodologia tradicional: Vantagens e desvantagens

Desvantagens	**Vantagens**
Possui processo burocrático, devido à divisão das atividades e que devem ser realizadas hierarquicamente.	Fundamentado em fluxos de trabalho de processo.
O planejamento é considerado rígido e permiti pouca flexibilidade de trabalho.	Quando de acordo com o planejamento, possui garantia alta de que o processo sairá do papel. É realizada a entrega do produto em sua totalidade.
Possui muita documentação.	Maior conhecimento por parte do cliente quanto ao valor total do projeto.

Fonte: Elaborada pela autora

Existem diversas metodologias tradicionais, mais aqui serão abordadas as mais comuns e utilizadas, que são: Modelo cascata (WINSTON ROYCE, 1970), Modelo incremental (SBROCCO; MACEDO, 2012), RAD (Rapid Application Development), (SOMMERVILLE,2011) e o RUP (Rational Unified Process), (KRUTCHEN, 2003) e serão descritas a seguir:

2.4.1 MODELO EM CASCATA

O modelo em cascata "sugere uma abordagem sequencial e sistemática para o desenvolvimento de software" (PRESSMAN, 2016), ou seja, possui fases bem definidas. Onde cada fase tem que ser cumprida ou finalizada para se avançar para a fase seguinte, precisa estar totalmente documentada e aprovada.

Também conhecido como modelo clássico, o modelo em cascata foi o primeiro processo de desenvolvimento publicado em 1970 por Winston Royce, quando a indústria de software teria acabado de assumir sua insuficiência para construção de *software*.

Naquele momento esse modelo foi responsável pela grande revolução, no que diz respeito ao desenvolvimento de projetos, mudando a maneira como o projeto era desenvolvido e planejado.

De acordo com Sommerville (2011, p.20), “o modelo em cascata é um exemplo de um processo dirigido a planos, em princípio, deve-se planejar e programar todas as atividades do processo antes de começar a trabalhar nelas”.

Ainda segundo Sommerville, os principais estágios do modelo em cascata refletem diretamente nas atividades fundamentais do desenvolvimento (SOMMERVILLE, 2011.p20.), e segundo ele serão apresentados abaixo:

- Análise e definição de requisitos: Os serviços, restrições e metas do sistema, são estabelecidos por meio de consulta aos usuários. E em seguida, são definidos em detalhes e funcionam como uma especificação do sistema;
- Projeto de sistema e software: O processo de projeto de sistemas aloca os requisitos tanto para sistemas de hardware como para sistemas de software, por meio da definição de uma arquitetura geral do sistema;
- Implementação e teste unitário: Durante esse estágio, o projeto do software é desenvolvido como um conjunto de programas ou unidades de programa;
- Integração e teste de sistema: As unidades individuais do programa ou programas são integradas e testadas como um sistema completo para assegurar que os requisitos do software tenham sido atendidos;
- Operação e manutenção: Normalmente, essa é a fase mais longa do ciclo de vida. O sistema é instalado e colocado em uso. A manutenção envolve a correção de erros que não foram descobertos em estágios iniciais do ciclo de vida.

Já Pressman (2016) define os seguintes estágios do modelo em cascata:

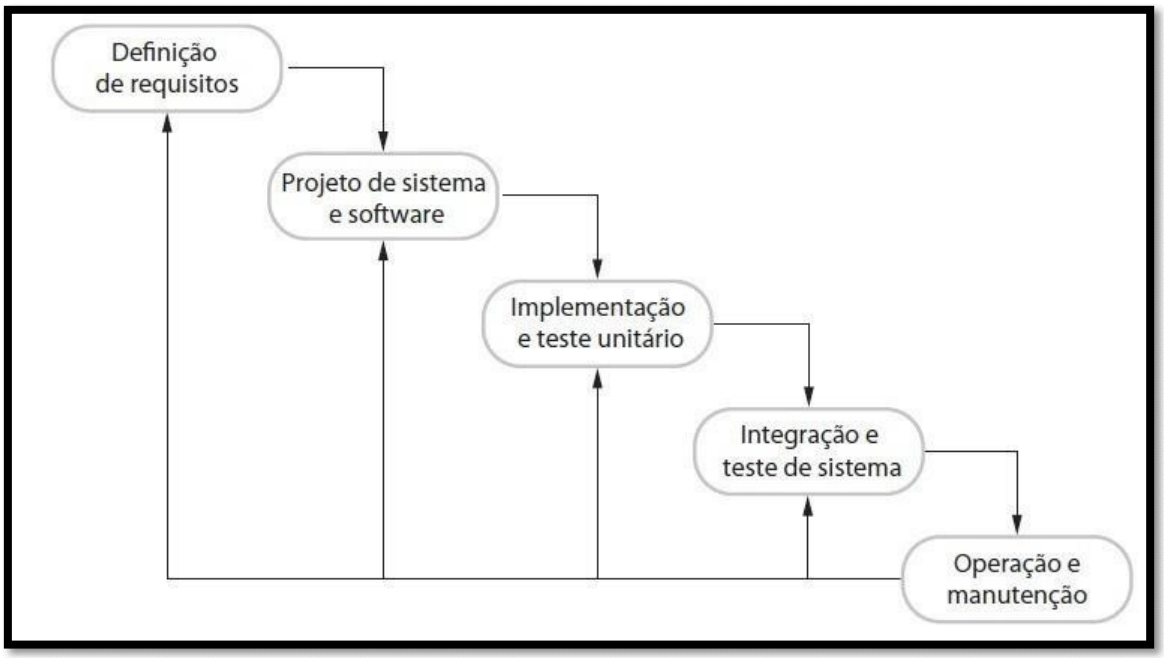
- Comunicação: início do projeto, onde é feito o levantamento de necessidades;
- Planejamento: onde são definidas estimativas e o cronograma e é feito o acompanhamento;
- Modelagem: onde é feita a análise e o projeto;
- Construção: são realizados codificação e testes.
- Emprego: é feita a entrega, o suporte e o feedback.

O modelo em cascata funciona da seguinte forma: o resultado de cada estágio é a aprovação de um ou mais documentos, ou seja, todas as vezes em que um estágio chega ao final, este é descrito em formato de documento e no fim assinado. O estágio seguinte não deve ser iniciado até que a fase anterior seja terminada. “Na prática, esses estágios se

sobrepõem e alimentam uns aos outros de informações" (SOMMERVILLE, 2011.p20.). O documento gerado no estágio anterior serve de auxilio e suporte para o próximo.

A Figura 4 em seguida, exibi a visão desses estágios descritos. Sommerville, afirma ainda que essas etapas apresentadas, são as principais desse modelo e que a origem do seu nome se deu, a partir do formato da figura abaixo (SOMERVILLE,2011, p.20).

Figura 4. Modelo em cascata



Fonte: Sommerville, (2011, p.20)

Segundo Sommerville (2011, p. 20) e Pressman (2011, p. 61), o modelo em cascata "em princípio, deve ser usado apenas quando os requisitos são bem compreendidos e pouco provavelmente venham a ser radicalmente alterados durante o desenvolvimento do sistema".

2.4.2 MODELO INCREMENTAL

No caso em que ocorre a necessidade de proporcionar um conjunto limitado de funcionalidades do software aos usuários, e depois melhorar e aumentar essa mesma funcionalidade em versões derivadas do *software*, escolhe-se o processo de desenvolvimento incremental.

O modelo incremental, tem como objetivo a entrega de um produto operacional em cada interação, são versões simplificadas do produto final que dispõem

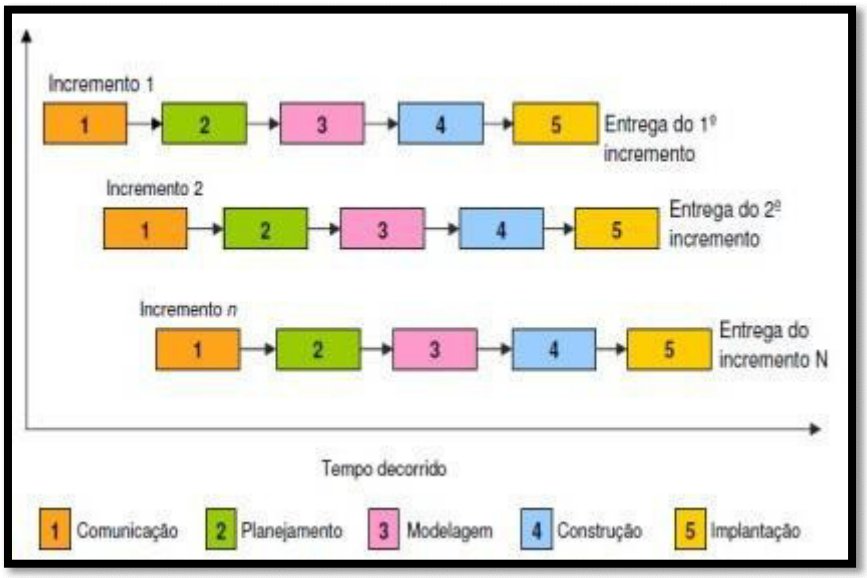
funcionalidades para os usuários. “Esse modelo pode ser útil quando não se dispõe de mão de obra suficiente para todo o projeto, permitindo adicionar mais recursos a cada incremento entregue” (SBROCCO; MACEDO, 2012).

“Este modelo é similar ao modelo em cascata” (SEMEDO, 2012, P.40.), Segundo Gonzales (2015, p.5), segue fases bem definidas assim como o modelo em cascata, no entanto, o projeto é dividido em incrementos, que também pode ser tratado como etapas, que produzem funcionalidades incrementadas ao sistema, até a entrega final.

Em conformidade, Sommerville (2011, p. 21), relata que esse modelo tem como base, a noção de desenvolver uma etapa inicial, baseada em uma reunião com os envolvidos para definir os objetivos gerais do software, e desenvolver por meio de várias versões de etapas, até que um sistema adequado tenha sido desenvolvido.

Estas etapas são seguidas sequencialmente e entrega-se um incremento a cada vez que uma etapa é finalizada, como pode ser constatado na Figura 5.

Figura 5. Modelo incremental



Fonte: Pressman (2006, p.21)

De acordo com Sommerville (2011, p.22), o desenvolvimento incremental tem três vantagens importantes quando comparado ao modelo em cascata:

- O custo de acomodar as mudanças nos requisitos do cliente é reduzido. A quantidade de análise e documentação a ser refeita é muito menor do que o necessário no modelo em cascata;

- É mais fácil obter feedback dos clientes sobre o desenvolvimento que foi feito. Os clientes podem fazer comentários sobre as demonstrações do software e ver o quanto foi implementado;
- É possível obter entrega e implementação rápida de um software útil ao cliente, mesmo se toda a funcionalidade não for incluída.

Ainda de acordo com Sommerville (2011, p.22), do ponto de vista do gerenciamento, a abordagem incremental tem dois problemas:

- O processo não é visível. Os gerentes precisam de entregas regulares para mensurar o progresso;
- A estrutura do sistema tende a se corromper com a adição dos novos incrementos. A menos que tempo e dinheiro sejam dispendidos em refatoração para melhoria do software.

2.4.3 MODELO RAD

Criado por James Martins no início da década de 90, com o intuito de produzir software, com um único ciclo de desenvolvimento, de no máximo três meses. É especialmente recomendado para projetos de software que possam ser moldados com escopo claro, definido e detalhado.

A metodologia RAD, segundo Sommerville (2011), também é considerado um modelo incremental, porém adaptado para tipos de projetos mais curtos, geralmente com prazo girando em torno de três meses.

A principal característica desse modelo, é que o produto final seja desenvolvido em componentes, para que a equipe possa desenvolver um sistema de software completamente funcional.

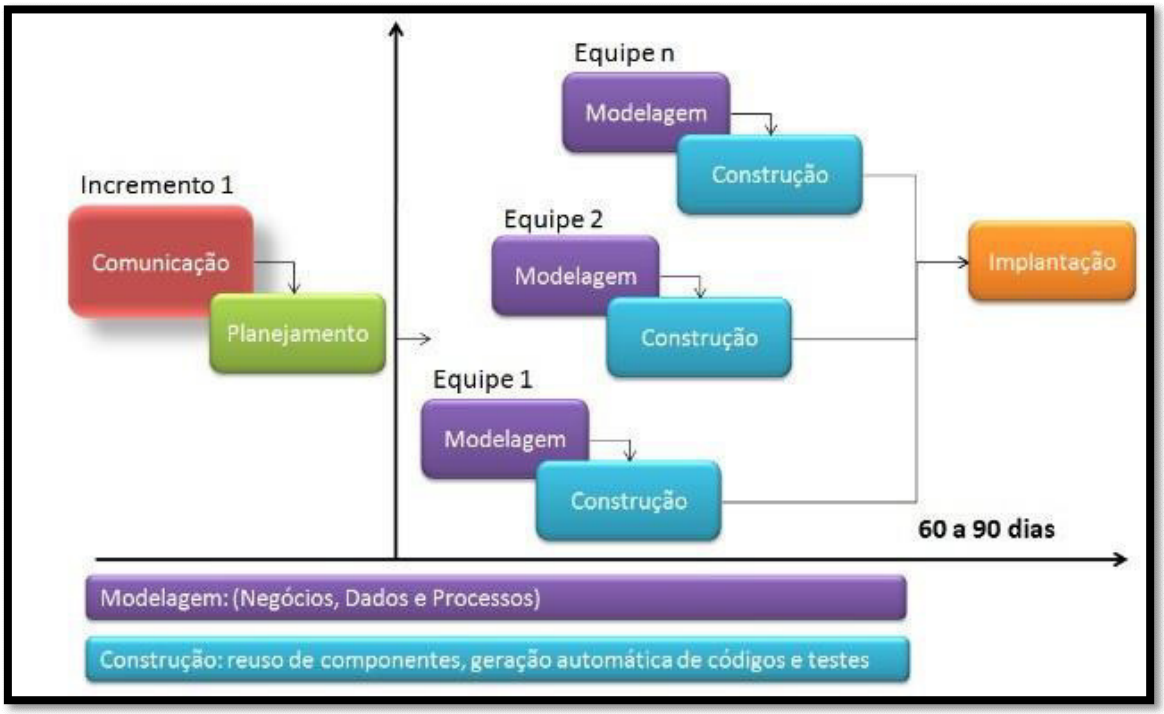
A abordagem desse modelo se baseia em quebrar o software em módulos, para que várias equipes trabalhem ao mesmo tempo.

Pode-se constatar a partir da Figura 06, que a abordagem RAD é dividida em cinco cenários de fases, conforme as definições a seguir:

1. Comunicação: Trabalha para entender os problemas do negócio e as características de informação que o software precisa guardar;
2. Planejamento: É essencial, porque várias equipes de software trabalham semelhantes em diferentes funções do sistema;

3. Modelagem: compreende a três das principais fases: modelagem do negócio, modelagem dos dados e modelagem dos processos. Também estabelece representações de projeto que servem como base para a atividade de construção do RAD;
4. Construção: Ressalta o uso de componentes de software preexistentes;
5. Implantação: Estabelece a base para iterações subsequentes, se necessárias.

Figura 6. Modelo RAD



Fonte: Adaptado de Pressman, (2006, p. 41)

Esse método é também chamado de CBSE (*Component-Based Software Engineering*), ele destaca a integração desses componentes em vez de construí-los do zero. (SOMMERVILLE, 2011).

Apesar do nome e de utilizar um ciclo de desenvolvimento considerado curto, RAD não é um método ágil, dispõe de duas principais vantagens: A de possuir uma proposta de concluir o projeto em um período pré-estabelecido e consideravelmente curto, a segunda é o incentivo à produção de componentes modularizados e reutilizáveis.

2.4.4 MODELO RUP

"O Rational Unified Process — RUP (KRUTCHEN, 2003) é um exemplo de modelo de processo moderno, derivado de trabalhos sobre a UML e o *Unified Software Development Process* associado" (SOMMERVILLE, 2011, p.34).

Considerado um processo pesado (Tradicional), por muitos especialistas, é preferencialmente aplicável a grandes equipes de desenvolvimento de projetos. Criado pela Rational Software Corporation, adquirida posteriormente pela IBM, ficando conhecida como RUP IBM ou IRUP.

Conforme Sommerville (2011, p.34), o RUP é normalmente descrito em três perspectivas:

- Uma perspectiva dinâmica, que mostra as fases do modelo ao longo do tempo;
- Uma perspectiva estática, que mostra as atividades realizadas no processo;
- Uma perspectiva prática, que sugere boas práticas a serem usadas durante o processo.

Sommerville (2011) diz ainda que "a maioria das descrições do RUP tenta combinar as perspectivas estática e dinâmica em um único diagrama" (KRUTCHEN, 2003 apud SOMERVILLE 2011, p.34). Por achar que essa tentativa torna o processo mais difícil de ser compreendido, usa descrições separadas de cada perspectiva.

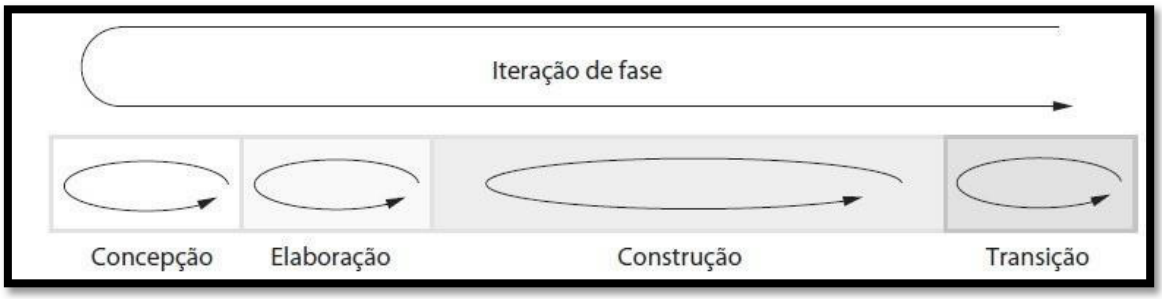
Ainda segundo Sommerville (2011, P.34), no RUP, "a iteração é apoiada de duas maneiras, cada fase pode ser executada de forma iterativa com os resultados desenvolvidos de forma incremental." Além disso, todo o conjunto de fases também pode ser executado de forma incremental, como indicado pela seta curva de 'transição' para concepção, demonstrado mais a frente na Figura 07. Abaixo estão reproduzidas as fases do RUP (SOMERVILLE2011, p.34).

- Concepção: O objetivo da fase de concepção é estabelecer um business case para o sistema. Você deve identificar todas as entidades externas (pessoas e sistemas) que vão interagir com o sistema e definir as interações;
- Elaboração: As metas da fase de elaboração são desenvolver uma compreensão do problema dominante, estabelecer um framework da arquitetura para o sistema, desenvolver o plano do projeto e identificar os maiores riscos do projeto;
- Construção: A fase de construção envolve projeto, programação e testes do sistema. Durante essa fase, as partes do sistema são desenvolvidas em paralelo e integradas;

- Transição: A fase final do RUP implica transferência do sistema da comunidade de desenvolvimento para a comunidade de usuários, e em seu funcionamento em um ambiente real.

Ao contrário do modelo em cascata, no qual as fases são igualadas com as atividades do processo, as fases do RUP são exatamente relacionadas ao negócio, e não à assuntos técnicos. A Figura 7 a seguir, mostra as fases do RUP, que foram descritas anteriormente de acordo com Sommerville.

Figura 7. Fases do RUP



Fonte: Sommerville, (2011, p.34).

Seu principal objetivo é atender as necessidades dos usuários garantindo uma produção de *software* de alta qualidade, que cumpra um cronograma e um orçamento previsível. Permiti ainda definir quem é o responsável pelo o que deve ser feito, como deve ser feito e quando deve ser feito, descrevendo todas as metas de desenvolvimento especificamente para que sejam alcançadas.

2.5 METODOLOGIA ÁGIL DE SOFTWARE

A migração de processo de desenvolvimento tradicional para o processo ágil, possui uma questão de dificuldades para alguns indivíduos, e para determinados cenários, esta mudança, se torna bem complexa, mas, no entanto, a insatisfação com essas abordagens pesadas classificadas como tradicionais, levou um grande número de desenvolvedores a buscar e propor novos métodos.

Conforme Sommerville (2011, p.39) a ideia de desenvolvimento rápido e ágil de software, foi apoiada no ano de 1980, no entanto, o conceito realmente decolou no final da década de 1990, com o desenvolvimento da noção de abordagens ágeis", como metodologia de desenvolvimento de sistemas dinâmicos (DSDM, do inglês *dynamic*

systems development method) (STAPLETON, 1997), Scrum (SCHWABER e BEEDLE, 2001) e XP-Extreme Programming (BECK, 1999; BECK, 2000).

Zenaro (2012, p.77), em seus estudos, afirma também que as metodologias ágeis, surgiram na década de 90, porém declara que "foi como uma revolução dos métodos tradicionais de desenvolvimento de *software*". Assim como também, Pressman (2016), declara que "as metodologias ágeis se desenvolveram em um esforço para sanar fraquezas reais e perceptíveis da engenharia de *software* convencional". Contudo, foi uma espécie de atualização de metodologias, surgiu atendendo aspectos não sustentados pelas metodologias tradicionais.

O objetivo dessas metodologias ágeis é, segundo Sommerville (2011, p.40), "reduzir a burocracia do processo, evitando qualquer trabalho de valor incerto de longo prazo e também, evitar qualquer documentação que provavelmente nunca será usada".

Pressman (2016), afirma que o que diferencia as metodologias ágeis das metodologias tradicionais, "é o foco em valores, ou seja, para uma metodologia ser considerada ágil deve ressaltar pessoas ao invés de documentação". Para ser realmente considerada ágil a metodologia deve aceitar a mudança ao invés de tentar prever o futuro.

A seguir é apresentada a Tabela 2, que contém algumas das vantagens e desvantagens a respeito dos modelos de processos ágeis quanto a sua utilização em desenvolvimento de software.

Tabela 2. Metodologia ágil: Vantagens e desvantagens

Desvantagens	**Vantagens**
Escalabilidade para equipes grandes ou dispersas. Mais indicado para pequenas equipes.	Revisão e teste de códigos durante o desenvolvimento.
Aberto a mudanças, requisitos e funcionalidades podem ser adicionadas ao longo do projeto.	Divisão de trabalho é realizada de acordo com as habilidades existentes entre os envolvidos na equipe.
Valor total do projeto é volátil, tornando-se desconhecido algumas vezes pelo cliente.	Existe a participação ativa do cliente ao longo do projeto.

Fonte: Elaborada pela autora

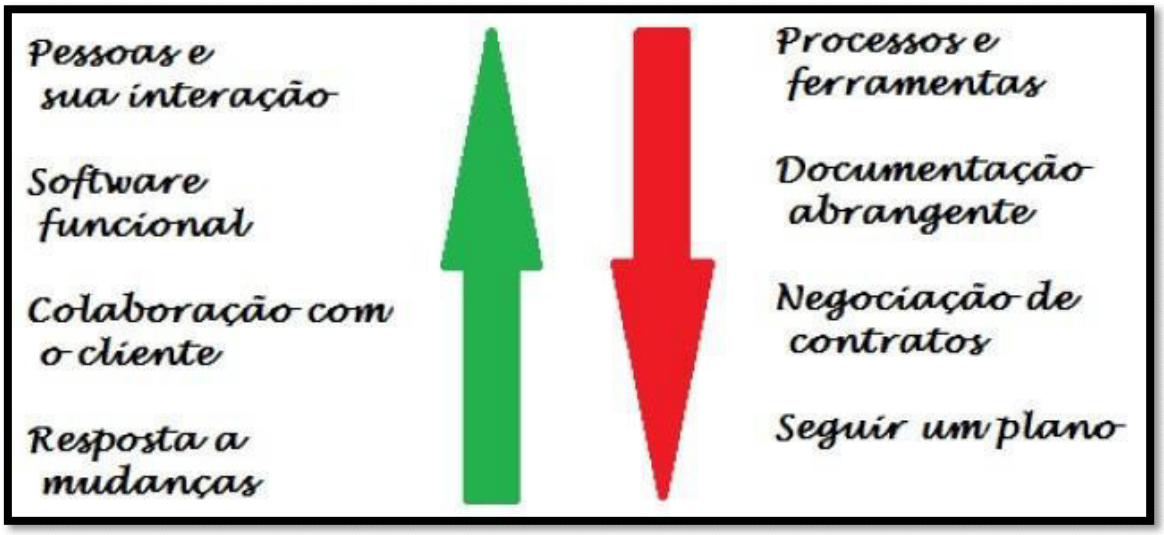
2.5.1 MANIFESTO ÁGIL

Em 2001, uns grupos de notáveis desenvolvedores, entre eles Kent Beck, assinaram um manifesto que expressa toda a ideia do desenvolvimento ágil, abaixo temos um trecho do mesmo.

Estamos descobrindo maneiras melhores de desenvolver softwares fazendo-onós mesmos e ajudando outros a fazê-lo. (MANIFESTO AGIL, 2001).

Através deste trabalho, passamos a valorizar, quatro valores do manifesto exibidos a seguir, e que se deve ter total compreensão, para um melhor desenvolvimento de software e entrega de produtos com qualidade, esses valores são apresentados a seguir na Figura 8.

Figura 8. Valores ágeis



Fonte: Gonzales (2015, p.07)

A partir da ideia de mudanças rápidas, surgi a abordagem chamada ágil que passou a ser utilizada, e algum tempo depois “foi divulgado o manifesto para o desenvolvimento ágil de software, que conta com doze princípios fundamentais” (MANIFESTO, 2001).

Os doze princípios estão presentes para auxiliar na compreensão dos valores já descritos anteriormente e são citados de acordo com o Manifesto (2001) a seguir:

1. A prioridade é satisfazer o cliente através de entregas contínuas e frequentes;
2. Receber bem as mudanças de requisitos, mesmo em uma fase avançada do projeto;
3. Entregas com frequência, sempre na menor escala de tempo;
4. As equipes de negócio e de desenvolvimento devem trabalhar juntas diariamente;
5. Manter uma equipe motivada fornecendo ambiente, apoio e confiança necessários;
6. A maneira mais eficiente da informação circular dentro do time de desenvolvimento é através de uma conversa presencial;

7. Ter o sistema funcionando é a melhor medida de progresso;
8. Processos ágeis promovem o desenvolvimento sustentável;
9. Atenção contínua à excelência técnica e a um bom design, aumenta a agilidade;
10. Simplicidade é essencial;
11. As melhores arquiteturas, requisitos e projetos provêm de equipes organizadas;
12. Em intervalos regulares, a equipe deve refletir sobre como se tornar mais eficaz.

Com base nesses princípios, as metodologias ágeis existentes apresentam um conjunto de atividades a serem adotadas durante o desenvolvimento de sistemas.

Segundo Sommerville (2011, p.40), os métodos ágeis, compartilham um conjunto de princípios, com base no manifesto ágil, e por isso têm muito em comum. Embora esses métodos sejam todos baseados na noção de desenvolvimento e entrega incremental, eles propõem diferentes processos para alcançar tal objetivo.

Existem diversas metodologias ágeis, mais aqui serão abordadas as mais comuns e utilizadas, que são: Extreme Programming (XP), (BECK, 1999; BECK, 2000), o Scrum (COHN, 2009; SCHWABER, 2004; SCHWABER e BEEDLE, 2001), Feature Driven Development (FDD), (PALMER e FELSING, 2002), Dynamic Systems Development Method (DSDM), (STAPLETON, 1997; STAPLETON, 2003), TDD (Test Driven Development), que serão discutidas na sessão a seguir.

2.5.2 XP: EXTREME PROGRAMMING

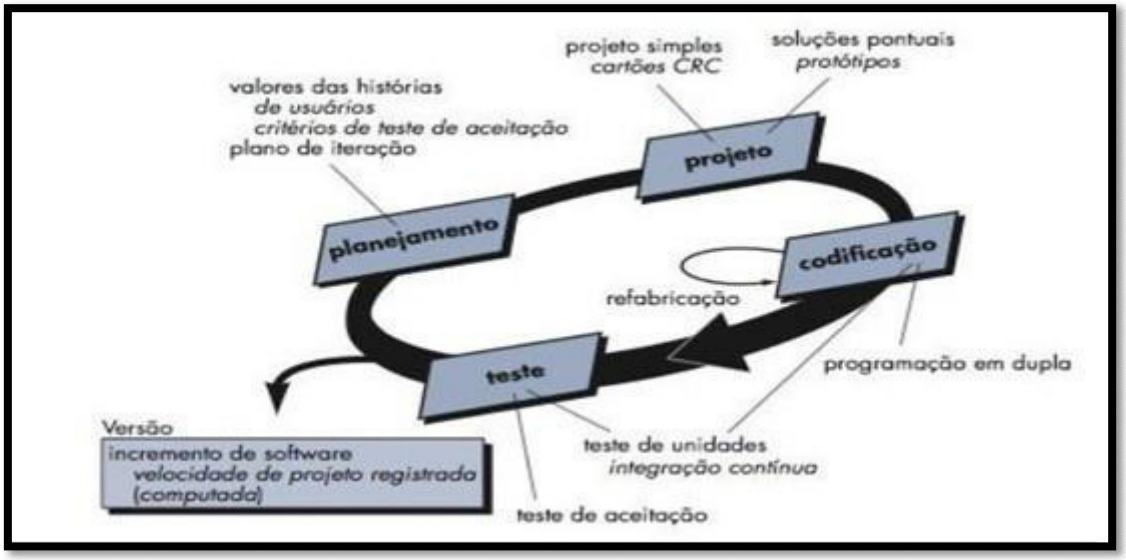
Extreme Programming (XP) “é talvez o mais conhecido e mais utilizado dos métodos ágeis, o nome foi cunhado por Beck (2000), pois a abordagem foi desenvolvida para impulsionar práticas reconhecidamente boas, como o desenvolvimento iterativo, a níveis ‘extremos’ (SOMMERVILLE, 2011, p.44).

Segundo Pressman (2016), XP, é uma abordagem destinada e disciplinada para o desenvolvimento do software. O primeiro projeto a usar o XP foi o C3, da Chrysler.

De acordo com Beck (2001), o “Extreme Programming foca no desenvolvimento rápido do sistema de software e procura garantir a satisfação do contratante, além de ainda favorecer o cumprimento das estimativas”. As práticas, valores e regras desse método proporcionam um bom cenário para o desenvolvimento de sistemas.

XP é composta por um conjunto de regras e práticas uniformes, possuindo quatro atividades metodológicas conforme Figura 9, as quais serão explicadas nas subseções a seguir.

Figura 9. Visão geral do processo de desenvolvimento XP



Fonte: Pressman (2011, p.88)

A estrutura apresentada na Figura acima, demonstra o processo de desenvolvimento desse modelo, com planejamento, projeto, codificação e teste, e em meio à eles, estão as práticas e valores que devem ser seguidas, "para que se obtenha um produto final de boa qualidade, esse método de desenvolvimento é sustentado por esses valores, práticas e ainda papéis" (SBROCCO; MACEDO, 2012), conforme serão descritos a seguir:

2.5.2.1 Práticas

Fernandes (2011, p.27), descreve abaixo as doze práticas do Extreme Programming:

1. Padrão na codificação: Os desenvolvedores devem escrever todo o código fonte alinhados as regras que enfatizam a comunicação durante a codificação. Esse padrão é definido antes de iniciar o projeto e deve ser seguido por toda a equipe de desenvolvimento;
2. Semanas de quarenta horas: O Extreme Programming possui uma regra, para cada elemento da equipe, que limita em quarenta horas o tempo total que se deve trabalhar em uma semana;

3. Cliente no mesmo local da equipe: Um usuário que responda pelo cliente deve integrar-se a equipe de desenvolvedores do sistema, disponível preferencialmente em tempo integral para responder as possíveis dúvidas dos elementos do projeto;
4. Integração contínua: São geradas versões internas e o sistema é integrado quantas vezes ao dia for preciso, após uma estória ser finalizada;
5. Propriedade coletiva: Os desenvolvedores da equipe possuem a permissão de alterar parte do código fonte em qualquer lugar do sistema e a qualquer momento;
6. Programação em pares: Dois desenvolvedores codificam o sistema em um mesmo computador. Geralmente, um é responsável por codificar enquanto o outro fica responsável por procurar erros;
7. Jogo de planejamento: Os desenvolvedores do sistema e o cliente entram em comum acordo para definirem o esforço necessário para implementar as estórias e a duração das iterações;
8. Incrementos curtos: Um incremento funcional e simples qualquer é gerado rapidamente com duração máxima de dois meses. Isso é feito para ter um retorno por parte do cliente em tempo hábil e poder incorporar correções e mudanças do sistema que está sendo desenvolvido;
9. Metáfora: Nessa prática é elaborado uma descrição que permite todos os envolvidos no projeto explicarem como o sistema funciona. A prática da metáfora é de grande ajuda para os envolvidos compreenderem os elementos básicos do sistema;
10. Projeto simples: Essa prática solicita que o sistema deve ser projetado da forma mais simples possível, respeitando as necessidades atuais do projeto;
11. Testes: Os testes funcionais são escritos pelo cliente, enquanto que os testes de unidade são escritos pela equipe do projeto e executados continuamente;
12. Reestruturação: Consiste na melhoria contínua do sistema. São realizadas atividades como remoção de códigos fontes duplicados, melhorias na comunicação, simplificações no processo, etc.

2.5.2.2 Valores

Segundo Sommerville (2011, p.47) “na prática, muitas empresas que adotaram XP não usam todas as práticas da Extreme Programming”. Tentam admitir para si, a maioria dos valores, no entanto as práticas, elas escolhem de acordo com sua organização.

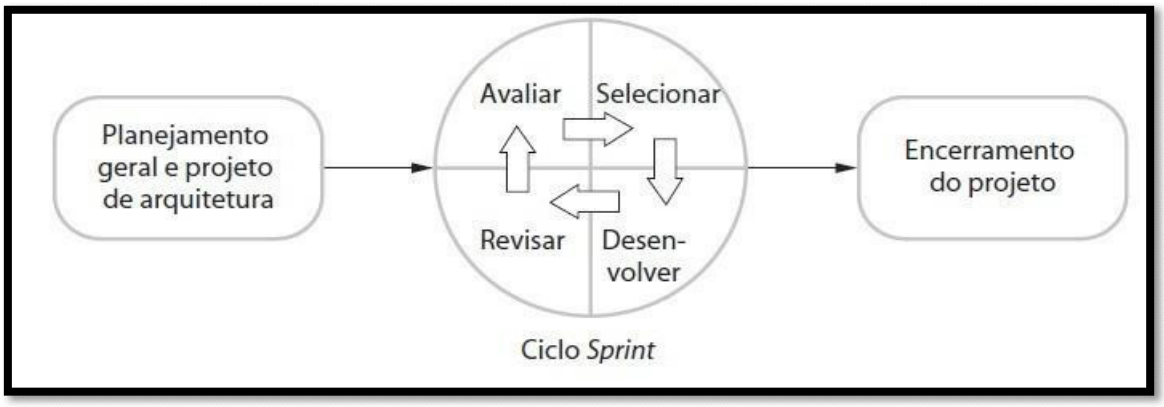
- Comunicação: O XP enfatiza a comunicação entre os integrantes do projeto e principalmente um contato direto com o cliente.
- Simplicidade: O importante é focar nos requisitos sem a necessidade de desenvolvimentos complexos que não agregam valor para o cliente, sendo que a simplicidade não quer dizer desenvolver de qualquer jeito;
- Coragem: Muitos requisitos podem aparecer durante o desenvolvimento, nesse caso, os desenvolvedores devem ter a coragem de enfrentar processos complexos ou de alto risco;
- Feedback: Esse valor enfatiza o feedback do cliente assim que cada parte do sistema é liberada;
- Respeito: O respeito deve estar presente em todas as relações com o grupo e com os clientes.

2.5.3 SCRUM

A abordagem Scrum (COHN, 2009; SCHWABER, 2004; SCHWABER e BEEDLE, 2001), "é um método ágil geral, mas seu foco está no gerenciamento do desenvolvimento iterativo, ao invés das abordagens técnicas específicas da engenharia de software ágil" (SOMMERVILLE, 2011, p 50.).

A seguir é exibido o processo Scrum, onde se inicia com uma fase de planejamento geral, em que se estabelecem os objetivos gerais do projeto de software. Em seguida, ocorre uma série de ciclos de Sprint, sendo que cada ciclo desenvolve um incremento do sistema e finalmente, a última fase do projeto encerra o projeto, e é completa com a documentação exigida. A Figura 10 abaixo, representa essa descrição.

Figura 10. Processo SCRUM



Fonte: Sommerville, (2011, p.50)

Schwaber e Sutherland (2013), definem o Scrum como “um framework para desenvolver e manter produtos complexos, onde é possível tratar e resolver problemas complicados e adaptativos, e entregar produtos com o mais alto valor possível”.

“Essa metodologia está dividida em três partes: papéis, artefatos e eventos, a estrutura de funcionamento é por ciclos, popularmente conhecida por Sprint, a duração de cada Sprint é normalmente de duas a quatro semanas” (SBROCCO; MACEDO, 2012).

Em seguida são abordados os três papeis que pertencem ao método Scrum.

3.5.3.1 Papéis

São divididos em três:

1. *Scrum Master*: é o facilitador, promove a comunicação de todos os envolvidos, representa o cliente no projeto;
2. *Product Owner*: Ou dono do produto, é quem transforma necessidades em valores de negócio para o cliente;
3. *Equipe Scrum*: É a equipe de desenvolvimento, normalmente possuí de cinco a nove pessoas.

3.5.3.2 Eventos

“Os eventos são prescritos e usados no Scrum com o objetivo de criar uma rotina e minimizar a necessidade de eventos não definidos pelo framework”. (SCHWABER; SUTHERLAND, 2013). E estão descritos a seguir:

- *Sprint*: “O coração do Scrum é o *Sprint*, que tem duração média de um mês, onde uma versão potencialmente utilizável do produto é criada para ser entregue ao cliente. Um novo Sprint é iniciado imediatamente após o término do anterior” (SCHWABER; SUTHERLAND, 2013, p.). “Dentro da Sprint existem atividades de controle que gerenciam os itens não entregues, os problemas no processo e na codificação, e montam-se estratégias para evitá-los” (SBROCCO; MACEDO, 2012);
- Reunião de Planejamento do *Sprint*: A reunião de planejamento é provavelmente o evento mais importante do Scrum. O propósito desta reunião é dar à equipe informação suficiente para trabalhar;

- Reunião Diária: “A Reunião Diária do Scrum é um evento de 15 minutos, para que o time de desenvolvimento possa sincronizar as atividades e criar um plano para as próximas 24 horas” (SCHWABER; SUTHERLAND, 2013, p.);
- Revisão do *Sprint*: Este evento ocorre ao final do Sprint. Nesta reunião é apresentado o que foi desenvolvido durante o Sprint e o produto gerado é avaliado;
- Retrospectiva do *Sprint*: Ocorre após a equipe mostrar o que conseguiu fazer no Sprint anterior, para que a equipe pense no que deu certo, o que poderia ter sido melhor e o que pode melhorar para o próximo Sprint.

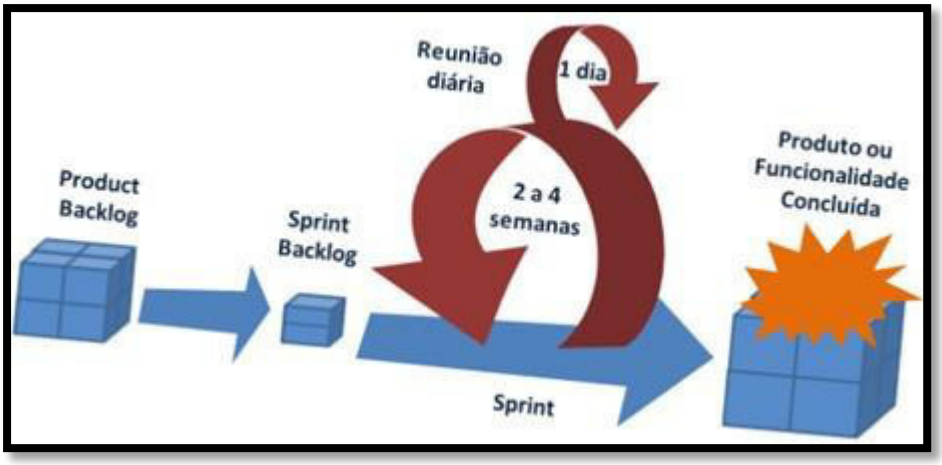
3.5.3.3 Atefatos do Scrum

“Os artefatos definidos para o Scrum são especificamente projetados para maximizar a transparência das informações chave, de modo que todos tenham o mesmo entendimento dos artefatos”. (SCHWABER; SUTHERLAND, 2013). A seguir são descritas alguns dos artefatos propostos pelo Scrum:

- *Product Backlog*: Corresponde a lista de requisitos e atividades do projeto que foram priorizados pelo product owner e possuem tempo estimado de entrega;
- *Release Burndown*: É considerado o principal gráfico de controle do Scrum e representa o trabalho restante dentro do Sprint de uma versão do sistema que será entregue;
- *Taskboard*: É um grande quadro que possibilita colocar várias informações relevantes para o acompanhamento do *Sprint*. O produto *backlog*, as atividades do Sprint juntamente com o seu estado (*To Do* (Para fazer), *In Process* (Em Desenvolvimento), *To Verify* (Para verificar) e *Done* (Realizado)), estes ficam sempre visíveis e disponíveis para todos os interessados no desenvolvimento do projeto.

A Figura 11 a seguir, apresenta o ciclo de vida do Scrum, esse ciclo de desenvolvimento tem como início a criação do *Product Backlog* pelo *Product Owner*, logo depois são realizadas as definições iniciais, em seguida vem ser a elaboração do *Sprint Backlog*, esse processo e realizado no *Sprint Planning Meeting*, nela é definido os itens que irão fazer parte do Sprint Backlog, a qual dá origem ao *Sprint*.

Figura 11. Ciclo da vida do Scrum



Fonte: Adaptado de desenvolvimento ágil, (2013)

2.5.4 DSDM: DYNAMIC SOFTWARE DEVELOPMENT METHOD

"Baseado nas práticas RAD – Rapid Application Development, o DSDM, Desenvolvimento Rápido de Aplicativos, consiste em um modelo de desenvolvimento incremental e iterativo para a construção de sistemas com prazos restritos" (PRESSMAN, 2016, p.).

O modelo DSDM "fornece um processo voltado a projetos que possuem restrição de prazo apertadas e que permita o uso da prototipagem incremental em um ambiente controlado por projeto" (PRESSMAN, p.2016).

Assim como o XP o DSDM, sugere um processo iterativo de software, "mas a cada iteração é levado em conta a regra dos 80%, isto significa, que somente um certo esforço será realizado para permitir o avanço do próximo incremento" (LEIDEMER,2013, p.42).

Conforme Loff (2010, p.35), "as práticas do DSDM são mantidas pelo dsdm Consortium", este é um grupo mundial de empresas que definiu o modelo ágil de processo, chamado ciclo de vida DSDM, este possui três ciclos iterativos, precedidos por duas atividades adicionais descritas a seguir.

- Estudo de viabilidade: Oferece os requisitos e as restrições necessárias para o projeto em construção e define se está aderente ao DSDM;
- Estudo de negócio: Estabelece os requisitos funcionais e não funcionais, que permitiram avaliar o negócio;
- Iteração do modelo funcional: Fornecem um conjunto de protótipos que são avaliados pelo cliente;

- Iteração de projeto e construção: Revisa os protótipos desenvolvidos na iteração do modelo funcional na visão de engenharia e permite a adição de valor de negócio pelo usuário;
- Implementação: Transforma o protótipo em um produto.

Segundo Scobro (2012 apud Leidemer, 2013, p.42-43), o DSDM possui um ciclo de vida composto por cinco fases para aplicação da metodologia. E são as seguintes fases: estudo da viabilidade, estudo de negócio, modelo de iteração funcional, projeto e construção de iteração e implementação.

2.5.5 FDD: FEATURE DRIVEN DEVELOPMENT

A metodologia FDD "foi criada em Singapura nos anos 90, mais especificamente entre 97 e 98, sendo utilizada pela primeira vez para o desenvolvimento de um sistema bancário internacional, que já havia sido considerado inviável dentro do prazo determinado" (SBROCCO; MACEDO, 2012, p.).

Foi criado por Jeff De Luca e Peter Coad, é considerada uma metodologia ágil, forte e muito utilizada atualmente (SCOBRO, 2012).

Segundo Pressman (2016), este método ágil, pode ser traduzido como desenvolvimento guiado por características, no contexto desse modelo, uma característica deve ser interpretada como "uma função valorizada pelo cliente". O FDD, diferentemente de outros métodos ágeis, "dá mais ênfase em diretrizes e técnicas de gestão de projeto" (LEIDEMER, 2013, P.42).

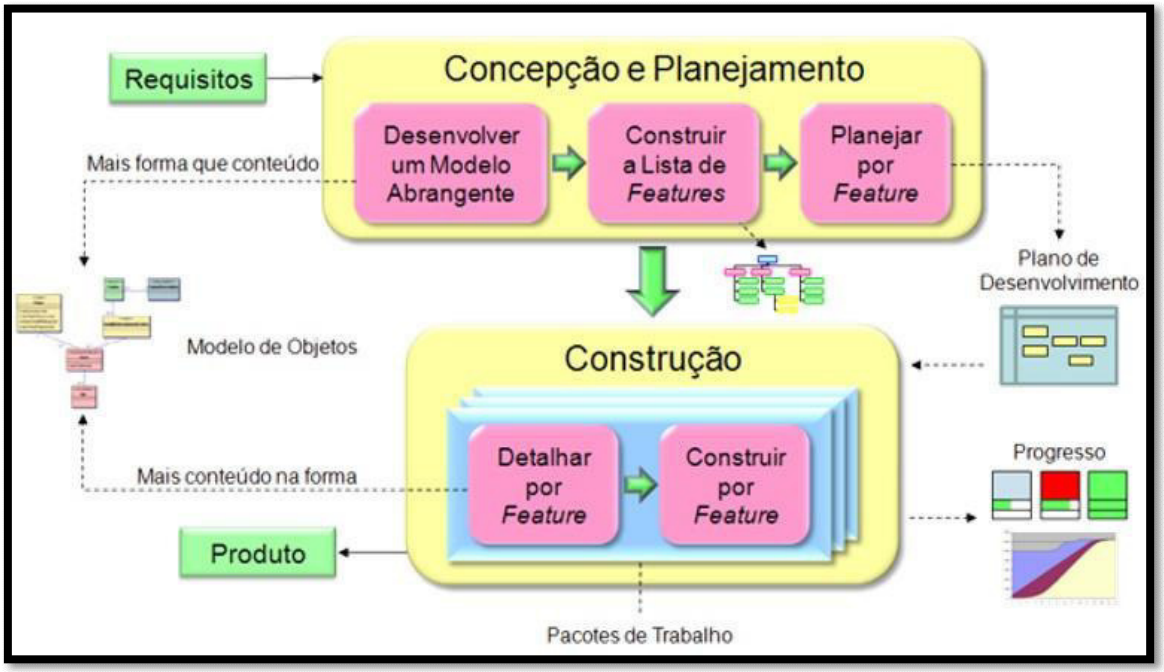
As características dessa metodologia, são pequenos blocos de funcionalidades passiveis de entrega, que permitem o usuário descrevê-las com facilidade e entender como elas relacionam-se umas com as outras.

Segundo Almeida (2015) em seu artigo, o FDD não é uma metodologia de gerenciamento de projetos de software. Ainda que exista atividades relacionadas a esse fim, "ele tem como principal foco cobrir os processos da engenharia de software, e não do gerenciamento".

Com base na Figura 12, pode ser observado que este processo possui de início duas fases: Uma etapa de planejamento (Concepção e Planejamento) e uma de construção. De acordo com Almeida (2015), a primeira etapa existe três processos, onde são desenvolvidos, uma lista de funcionalidades do software e um planejamento para cada funcionalidade. "A construção inclui dois processos: O detalhamento de uma

funcionalidade, que tem como resultado o modelo de domínio e os esqueletos de códigos prontos para serem preenchidos e a construção".

Figura 12. Processo de desenvolvimento com FDD



Fonte: Almeida, (2015)

O FDD possui ainda, as seguintes práticas associadas:

- Modelagem Orientada a Objetos do Domínio;
- Desenvolvimento por funcionalidade;
- Classe proprietária, ou seja, não temos propriedade coletiva do código. Cada classe tem seu dono;
- Equipes de recursos: destinadas a desenvolver recursos necessários ao projeto, de forma secundária;
- Inspeção é realizada constantemente para garantir a boa qualidade do código e do projeto;
- Gerenciamento de configuração;
- Integração contínua;
- Visibilidade de progressos e resultados.

3. TRABALHOS RELACIONADOS

Para o desenvolvimento deste trabalho, foi levado em consideração diversas referências bibliográficas de autores que abordam assuntos relevantes e até bem próximos do proposto. Estes foram tomados como base de pesquisa e estudos para o desenvolvimento desse estudo.

A bibliografia que serviu como base para identificar e obter informações e conhecimento a respeito dos modelos de processo de desenvolvimento de software, serão apresentados a seguir.

Antigamente quando se pensava em desenvolver um projeto de *software* não se fazia planos. Após algum tempo dessa época houve uma grande revolução quanto ao modelo de desenvolvimento de software, surgiram então métodos para auxiliar projetos. Informações como estas podem ser verificadas em (PRESSMAN 2011/2018, SOMMERVILLE 2007/2011, SBROCCO; MACEDO, 2012, YOSHIMA, 2011 e SCHWABER; SUTHERLAND, 2013).

Um pouco mais tarde surgiu um grupo de modelos de desenvolvimento de *software,* no qual foi atribuído o nome de metodologias ágeis, são abordadas neste trabalho as seguintes: XP, SCRUM, FDD, DSDM. Cada método tradicional e ágil pode ser consultado em (PRESSMAN 2006, 2011, 2016/ SOMMERVILLE 2007,2011 e Gudwin 2015), entre outros.

As metodologias ágeis foram bem aceitas em sociedade, e cada vez mais, ganham seu espaço no mercado. Alguns estudos analisam especificamente os impactos e benefícios da adoção e implantação de metodologias ágeis em organizações como em: “Benefícios das metodologias ágeis no gerenciamento de projetos de tecnologia da informação” (LIMA, 2015), em seu artigo o autor explora conceitos de metodologias ágeis. Para aprimoramento do seu estudo, destaca dois métodos: o SCRUM que abrange o gerenciamento de projetos de software e o XP, no âmbito de desenvolvimento.

A partir da exploração dos métodos realiza uma avaliação preliminar, buscando identificar na literatura algumas evidencias de aplicação dessas metodologias, resulta seu trabalho demonstrando como resultado a forma como o mercado utiliza e encara as metodologias ágeis nos dias atuais. Além deste, outros temas também relacionados com este trabalho são: “Gerenciamento de processos com metodologias ágeis” (KOSCHEVIC, 2011) e “Metodologia ágil com Scrum: Estudo do Processo de Implantação em uma Software House de Pequeno Porte” (GONZALES,2015).

Diversos relatos descrevem o uso de metodologias tradicionais e ágeis. E vem levantando uma dúvida a respeito de qual método deve-se utilizar? Qual o melhor? Qual vai sanar meus problemas na organização? Em “Estudo comparativo entre métodos ágeis e tradicionais de fabricação de software” (ALVES, 2012), Alves, coletou e efetuou uma pesquisa a respeito dos atuais métodos de fabricação de software assim como também foram citados métodos usados no início da modernização dos processos de desenvolvimento.

Realizou a comparação entre os métodos tradicionais e ágeis com base em entrevistas realizadas com especialistas da área, além de ter se obtido de leitura especializada e um estudo de caso que possibilitou o real conhecimento para a realização do mesmo.

Na atualidade, o uso de modelos, métodos de desenvolvimento de *software*, é um tema que desperta cada vez mais o interesse de pesquisadores e estudiosos, utilizam em sua pesquisa “Metodologias tradicionais e ágeis: Análise comparativa entre Rational Unified Process e Extreme Programming”, assim como (ALVES, 2012), um estudo das metodologias de desenvolvimento de *software* para a realização de uma comparação entre elas. Utiliza a metodologia ágil XP e a metodologia tradicional RUP como referência. Para realizar a comparação.

Entre essas metodologias ele expõe e compara as características que estes possuem em comum, além de explicar vantagens e desvantagens das metodologias citadas levando em consideração fatores externos que podem influenciar a implantação e desenvolvimento de projetos (UTIDA, 2012).

Estas referências tratadas acima são evidenciadas com principais, pois demonstram e remetem caminho semelhante, trilhado pelo presente trabalho, assim como (SOUZA,2016) com o seu trabalho “Abordagens de teste: uma comparação entre as metodologias tradicional e ágil”, (CABRAL,2006) “Comparativo entre metodologias tradicionais e ágeis”, (BARBOZA et al, 2016). “Análise comparativa entre as abordagens ágil e tradicional de gestão de projetos: um estudo de caso no setor industrial”,(EDER et al,2013)”Diferenciando as abordagens tradicional e ágil de gerenciamento de projetos” e (SILVA, CAMARGO & SOUZA, 2013) “Metodologias ágeis para o desenvolvimento de *software*: Aplicação e uso da metodologia Scrum em contraste ao modelo tradicional de gerenciamento de sistemas”.

Nesse mundo de informações e pesquisas, algumas dissertações e artigos também foram publicados com temas relacionados. Coelho, apresentou um relato de experiência na implantação de um método ágil em uma equipe de desenvolvimento de *software*, fez

uma verificação de como as práticas ágeis estão sendo tratadas por uma equipe de desenvolvimento de um setor de determinada organização (COELHO, 2011).

Oliveira descreveu metodologias ágeis, como alternativa aos processos tradicionais usados na engenharia de *software*. Apresenta ainda um estudo de caso – um sistema desenvolvido utilizando um processo baseado nas metodologias (OLIVEIRA, 2003) e Libardi e Vladimir, apresentam um estudo sobre metodologias ágeis de desenvolvimento. Um breve estudo das metodologias ágeis em geral e uma especificação detalhada dos papeis, regras e práticas da metodologia Scrum e da linguagem XP. Além disso, apresentam uma comparação dos processos propostos pelos métodos ágeis XP, SCRUM, FDD e ASD (LIBARDI, VLADIMIR 2010).

Além desses trabalhos, inúmeros artigos relatam casos de uso de metodologias ágeis na educação e na indústria (LEITÃO, 2010), (SOUZA, 2014), (BENZECRY, 2017), (UTIDA, 2012) e (ALVES, 2012) com resultados que contribuem para fortalecer as evidências de eficácia dos métodos ágeis e tradicionais. Os relatos incluem descrições de como as metodologias foram aplicadas e adaptadas para cada contexto.

Surgindo com uma forma de desenvolver *softwares* com velocidade e qualidade, as metodologias ágeis acabaram se tornando uma ferramenta para diversas áreas presentes no mundo, auxiliando diversos segmentos para responderem mais rápido às mudanças de mercado. Um exemplo disso, é que (LOPES,2017), em seu estudo analisa a aplicação da metodologia Scrum, que normalmente é utilizada para o desenvolvimento de *software*, Lopes realiza a experiência de aplicar a metodologia na gestão do trabalho em uma área empresarial do varejo brasileiro. O objetivo é chegar a conclusões a respeito da utilização do Scrum em tal cenário (LOPES,2017).

JESUS (2016), em sua pesquisa aborda algo relativamente novo, que é o uso de startups, com o tema “Metodologias Ágeis de Gerenciamento de Projetos Para Startups”, apresenta um estudo sobre as principais metodologias ágeis (XP e Scrum), e propõe uma metodologia híbrida para gerenciamento de projetos baseado em seus princípios e fundamentos, adaptado para startups. Quanto que (EKERT 2011), em seu trabalho aborda uma área bem próxima desta já citada, utiliza as mesmas metodologias em seu estudo, porém a diferença está nas características e maneiras de funcionamento.

O autor citado trata de microempresas, fez uma análise das metodologias ágeis que são utilizadas pelas microempresas de medianeira para o desenvolvimento dos projetos de *software* e propôs a adoção das práticas e valores do Scrum e do XP como uma alternativa que poderia ser utilizados para o desenvolvimento dos projetos de *software*. (EKERT, 2011).

Assim como (EKERT, 2011) e (JESUS, 2016), alguns autores utilizaram em suas pesquisas o uso da metodologia Xp e Scrum para implantação em algum ambiente organizacional. Como o (SANTOS,2014), com o tema de sua dissertação "O Impacto do uso das Metodologias Ágeis Scrum e XP na Satisfação dos Stakeholders" onde se baseia por questionários realizados para analisar quais são esses impactos causados pelo uso da metodologia, o autor (BORBOREMA, 2007) aborda a metodologia XP com o tema "Impacto da Aplicação da Metodologia XP nas Organizações de Desenvolvimento de *Software*" este Trata-se de um estudo, que apresenta o impacto do uso da metodologia ágil Extreme Programming (XP) com objetivo de obtenção de sucesso em desenvolvimento de sistemas. Baseado em estudos que demonstram falhas em projetos de *software*.

De com estudos realizados essas metodologias são consideradas as mais utilizadas, sendo que o Scrum se destaca quanto a outra, pois segundo estudiosos estas se adequam mais facilmente a maioria dos ambientes.

Modelos de processos de *software* estão presentes em diversas áreas de estudo, e são aplicadas e abordadas de muitas formas para resolução de problemas em muitos ambientes. (SILVA,2017) e (BARROS,2018) Tratam em seus trabalhos sobre a qualidade de *software* envolvendo o contexto de metodologias ágeis que é um modelo de processo de desenvolvimento de *software* que estar muito em alta nos últimos anos. (SILVA,2017) aborda e investiga o tema qualidade de *software* sobre o contexto do desenvolvimento ágil, para isso estuda quais métricas e ferramentas de controle e garantia de qualidade são empregadas nesse contexto. Já (BARROS,2018) por meio de uma revisão sistemática, investiga quais práticas presentes em metodologias ágeis auxiliam o atingimento da qualidade de *software*. Técnicas como TDD, programação em pares e constante interação do cliente foram vistas como bastante positivas pelo autor, contribuindo bastante para o atingimento da qualidade (BARROS,2018).

Dentre as diversos métodos de processamento de software, sendo tradicionais ou não, um em especial vem se destacando da família dos métodos ágeis, o método Srum tem se popularizado desde a chegada das metodologias ágeis, é definida como uma metodologia relativamente simples de implementar e de ser abordada em muitos desafios de gestão, talvez essa seja a razão para estar tão presente em diversos estudos como no do autor (ASSIS, 2016) com o tema" Gerenciamento de Projetos por Meio da Metodologia Scrum Aplicado em uma Empresa de Base Tecnológica",(LEITÃO,2010) "Aplicação de Scrum em Ambiente de Desenvolvimento de *Software* Educativo",

(LIMA,2011) “Scrum no Brasil”, (OLIVEIRA,2012) “Lista de Verificação Scrum para Desenvolvimento de *Software*”, (LEIDEMER,2013) “Implantação de Scrum em uma Empresa de Desenvolvimento de *Software*”, (SOUZA,2014) “Implantação da Metodologia Ágil Scrum em um ambiente de desenvolvimento” e muitos outros trabalhos que autores apresentam a metodologia Scrum que não se fizeram presentes nessa análise.

Por fim, com o aumento da presença da tecnologia em todos os meios e difusão pelo mundo todo, nasce a necessidade de escolha sobre como todo este processo de desenvolvimento que à envolve deverá ser conduzido, de modo a comportar as mudanças provenientes de uma sociedade cada vez mais exigente.

Autores como (XAVIER, 2017) aborda em seu trabalho, metodologias tradicionais e ágeis que auxiliam na condução e comportamento de processos no desenvolvimento de *software*. O autor elícita as características do *Rational Unified Process* – RUP, uma metodologia tradicional de desenvolvimento de *software*, e do SCRUM que é um framework estruturado de metodologia ágil, para fins de comparação e entendimento sobre o comportamento de ambas no processo de desenvolvimento de um novo *software*. (XAVIER, 2017).

3.1 COMPARAÇÃO E DISCUSSÃO DOS TRABALHOS

Nesta seção são abordados alguns trabalhos, dos já mencionados anteriormente, que serviram como base para a pesquisa realizada. Esses serão apresentados em resumos, para que seja realizada a análise para o comparativo entre os trabalhos aqui apresentados. 05 artigos foram analisados, tendo como foco o tema da pesquisa e como referências: seus objetivos, aplicação, metodologia, resultados, observando as vantagens e desvantagens do uso das metodologias. Os trabalhos mencionados, relatam discussões próximas da presente pesquisa.

Alguns artigos foram desconsiderados em função da falta de correlação e relevância em relação ao tema central da pesquisa, uma vez que abordam estudos variados acerca de gerenciamento de projetos e metodologias.

Silva, Souza e Camargo (2013), apresentam uma revisão da literatura abordando as principais diferenças sobre a metodologia ágil Scrum e a metodologia tradicional de gerenciamento de projetos.

O cenário abordado é o de empresas de desenvolvimento de software, principalmente no que refere-se a pequenas e médias empresas que são impactadas pelo

elevado custo de manutenção oriunda das metodologias tradicionais. (SILVA, SOUZA E CAMARGO, 2013).

Os autores fornecem um quadro comparativo entre as metodologias tradicional e ágil e concluem que a metodologia tradicional afirma a integração da organização em um nível mais estratégico, enquanto o Scrum norteia a operacionalização da gestão de projetos de maneira informal e colaborativa. Os autores ainda descrevem que as práticas podem ser complementares, onde pontos falhos de uma podem ser supridos por pontos fortes da outra.

O conteúdo deste trabalho se assemelha muito ao desta presente pesquisa, elabora a comparativa das metodologias para melhor entendimento da comunidade de desenvolvimento.

Uma das contribuições fundamentais deste trabalho foi o de fornecer a informações de nível bibliográfico que auxiliaram no desenvolvimento deste.

Ainda se tratando de diferenciar as metodologias é descrito a seguir o estudo de Eder et al. (2013). Onde os autores demonstram um trabalho, onde comparam às metodologias tradicionais de gerenciamento de projetos as metodologias ágeis. A pesquisa utilizou uma revisão bibliográfica com o intuito de identificar as práticas recomendadas em cada abordagem. O resultado foi à identificação de seis características críticas que diferenciam o uso de uma ou outra abordagem Eder et al. (2013).

Foi gerado um modelo conceitual, sendo apresentado como resultado da pesquisa, que a diferença principal das abordagens analisadas está nas técnicas usadas. Isso é, as ações de planejamento e controle são semelhantes, porém a forma como são feitas (técnicas e ferramentas) é que distinguem as duas abordagens.

Outra importante verificação realizada pelos autores, se refere à identificação em seis ações específicas que distinguem as abordagens. Eder et al. (2013) concluem o estudo afirmando que é possível identificar o uso ou não uso, da abordagem do gerenciamento ágil de projetos por meio da observação de seis características específicas, dentre as práticas adotadas pela organização.

Barboza et al. (2016), trataram em seu estudo sobre a verificação prática dos benefícios, limitações e impactos decorrentes da implantação do método ágil Scrum quando comparado ao gerenciamento tradicional de projetos. Sua pesquisa é realizada sobre um estudo de caso em uma empresa do setor industrial.

Para objetivar o estudo proposto os autores realizaram um estudo de caso, por meio do qual se pretendeu avaliar projetos desenvolvidos em uma organização. Utilizou-

se para isso, uma ampla variedade de evidências, como para definição dos projetos a serem analisados, centrou-se na abordagem de gestão adotada, assim como na equivalência do escopo entre eles (BARBOZA et al, 2016). Para assim pudesse investigar variáveis semelhantes viabilizando desta maneira, o estudo comparativo.

Os autores concluíram, ressaltando que os resultados alcançados indicam que todas as abordagens de gestão são eficientes. Todavia, o desenvolvimento ágil oferece benefícios importantes, no entanto, não é indicado para todos os projetos, produtos, equipes e situações, assim como métodos robustos e tradicionais também não são (BARBOZA et al. 2016).

Se tratando até então de cenários que demonstra o envolvimento e o confronto entre as metodologias, o estudo de Cabral (2006), apresenta uma proposta de incorporar práticas de metodologias ágeis, usando o XP como base, em metodologias tradicionais de desenvolvimento de software, com a finalidade de atenuar a burocracia existente nas metodologias tradicionais, que as tornam extremamente pesadas quando aplicadas a projetos de pequeno porte.

Cabral (2006), aborda em seu trabalho que o uso de uma metodologia para desenvolver *software* é o ponto chave para o sucesso do projeto. Porém, não existe uma única metodologia para desenvolver diversos sistemas.

A análise do presente trabalho foi feita tendo como base as fases de desenvolvimento de *software* das metodologias apresentadas neste estudo e nos trabalhos de (Oliveira, 2003) e (Neto, 2004). O autor aborda a incorporação das práticas XP no método RUP, Apresentada as atividades essenciais desta fase de transição e que podem ser usadas para pequenos projetos. A proposta é apresentar a possibilidade de adoção das práticas, redução de atividades e artefatos do RUP para ser possível sua aplicação em projetos de pequeno porte.

Cabral (2006), realiza a extensão do RUP para projetos pequenos, com a finalidade de atenuar a burocracia existente nas metodologias tradicionais, que as tornam extremamente pesadas quando aplicadas a projetos de pequeno porte.

Concluindo Cabral (2006), diz que as práticas de XP foram adotadas pelo fato de que a metodologia XP está mais consolidada atualmente. É uma metodologia que tem como foco as estórias, testes e codificação, e não em documentação como é feito no RUP.

E finalizando o estudado trabalho de Souza (2016), que apresenta as técnicas mais populares das metodologias ágeis e tradicionais realizando um comparativo entre as práticas adotadas, ciclo de vida e resposta de mercado. Buscou-se apresentar neste estudo

uma comparação entre as diversas formas de assegurar a qualidade do software durante seu ciclo de desenvolvimento, apresentando diversas técnicas e abordagens dentro das metodologias de teste tradicional e ágil.

Conforme Souza (2016), Ficou claro que no diagrama da abordagem tradicional os esforços de teste acontecem no final da codificação e antes da publicação. Por outro lado, o ágil é iterativo e incremental, portanto os testadores podem testar cada incremento de codificação, assim que a funcionalidade é publicada.

O autor finaliza mencionando, que a partir do seu trabalho é possível ter como conclusão, que apesar dos testes na metodologia ágil possuir mais resultados de sucesso do que críticas, ela própria, não tem o poder de avaliar se o sistema em questão está livre de falhas, bem como os testes da metodologia tradicional não os tem (SOUZA, 2016). A seguir é apresentada a relação completa dos trabalhos.

Tabela 3. Relação completa dos trabalhos relacionados

Autor	**Ano**	**Aplicação Método Ágil**	**Aplicação Método Tradicional**	**Aplicação prática em empresa**	**Métodos comparados**	**Metodologia**
Silva, Souza e Camargo	2013	✓		✓	2 Scrum e Tradicional.	Estudo de caso
Eder et al	2013	✓	✓	✓		Pesquisa discursiva.
Barboza et al.	2016	✓		✓	2 Nome não especificado.	Estudo de caso
Cabral	2006	✓	✓		2 XP e RUP	Pesquisa discursiva.
Souza	2016	✓	✓		2 XP e RUP	Pesquisa discursiva.

Fonte: Elaborada pela autora

A Tabela 4 a seguir, apresenta pontos fortes e francos, observados a partir da análise dos trabalhos citados neste capítulo, explicitando os principais pontos analisados durante a execução desta monografia.

Tabela 4. Pontos fortes e fracos dos trabalhos

Autor	Pontos Fortes	Pontos Francos
SOUZA (2016)	✓ Realizou o comparativo; ✓ Realizou abordagem sobre metodologias de teste tradicional e ágil.	✓ Abordagem de poucos tipos de metodologia; ✓ Comparativo entre dois tipos.
Eder et al. (2013)	✓ Realizou o comparativo; ✓ Obteve o resultado a partir de características críticas que afirmaram que a diferença está em como cada uma trabalha.	✓ Não menciona que tipo de metodologia está comparando; ✓ Comparativo somente a respeito das abordagens.
Barboza et al. (2016)	✓ Realiza a verificação de benefícios e impactos a partir da implantação do método ágil; ✓ Realiza o comparativo; ✓ Faz a implantação do método ágil em uma empresa.	✓ Não especificou a qual método tradicional comparou o Scrum; ✓ Comparativo somente entre dois tipos de metodologias.
Silva, Souza e Camargo (2013)	✓ Realizou o comparativo; ✓ Forneceu um quadro comparativo com as principais características dos métodos; ✓ Conclui que as práticas dos métodos podem ser complementares.	✓ Realizou o comparativo apenas entre dois tipos de métodos; ✓ Não mencionou qual metodologia tradicional comparou ao Scrum.
Cabral (2006)	✓ Incorporação das práticas XP no método RUP; ✓ Realiza o comparativo.	✓ Não dá ênfase no comparativo dos métodos; ✓ Realiza o comparativo entre dois métodos.

Fonte: Elaborado pela autora

Mediante as análises dos trabalhos, foi possível extrair as seguintes similaridades entre a presente pesquisa e os trabalhos citados anteriormente. Que foram agrupadas pelos autores, conforme a Tabela 5.

Tabela 5. Comparação com a monografia

Critério	Artigos
Abordagem de metodologias ágeis e tradicionais.	(SOUZA, 2016); (BARBOZA et al., 2016); (SILVA, SOUZA E CAMARGO 2013); (CABRAL,2006); (EDER ET AL. ,2013).
Comparativo entre metodologias	(SOUZA, 2016); (BARBOZA et al., 2016); (SILVA, SOUZA E CAMARGO 2013); (CABRAL,2006); (EDER ET AL. ,2013).
Comparativa entre mais de um tipo de metodologia ágil ou tradicional	
Comparativa entre metodologias do mesmo tipo.	
Fornece quadro comparativo	SILVA, SOUZA E CAMARGO (2013)
Fornece fatores para escolha de métodos.	

Fonte: Elaborada pela autora

A partir das tabelas obtidas, foi possível se fazer uma análise a respeito das distinções do presente estudo com os trabalhos mencionados anteriormente. Em geral, a maioria realiza a abordagem com respeito as metodologias de desenvolvimento de software ágil e tradicional, geram o comparativo, no entanto, apenas os autores Silva, Souza e Camargo (2013) oferecem em seu trabalho com evidencia um quadro colocando em contraposto as metodologias.

Os trabalhos dos referidos autores, pecam em não oferecer ao leitor, interessados no assunto, condições mais precisas a respeito de qual metodologia usar, não esclarecem fatores para que tomem tal decisão de escolha.

Alguns dos autores, nem se quer mencionam qual metodologia tradicional está sendo comparada no seu estudo, abordam de forma geral, sendo que os tipos de metodologias tradicionais são muito parecidas, porém possuem suas particularidades que deveriam ser ressaltadas.

Um ponto a ser destacado e que objeto de é que alguns dos trabalhos como os dos autores Silva, Souza e Camargo (2013); Eder et al(2013) e Barboza et al.(2016), forneceram detalhes sobre a implantação de metodologias ágeis em alguma empresa, esta situação é objeto de distinção dos trabalhos mencionados com o atual.

A comparativa elaborada na presente pesquisa, se utiliza de mais de uma metodologia para realizar o confronto, usa suas características de processos de gerenciamento de projetos, aspectos de desenvolvimento de um software e ainda principais características de cada tipo de metodologia, desta forma auxilia no aprimoramento do conhecimento e da compreensão, facilitando na escolha de um dos métodos.

Pode-se concluir manifestando, que o presente trabalho se propõe a melhorar os já citados, pois aborda mais conteúdo e com mais detalhes se comparado com os demais, em cumprimento do tema abordado que coincide entre eles.

O presente estudo inova quando visa realizar o comparativo de maneira geral e de forma mais especifica, no dado momento que apresenta o comparativo entre os tipos de metodologias de cada processo de desenvolvimento de software.

Contudo, os trabalhos argumentados foram de extrema importância para o desenvolvimento deste, contribuindo com informações necessárias que serviram de base do exposto tema.

4. COMPARAÇÃO DE MÉTODOS ÁGEIS E TRADICIONAIS

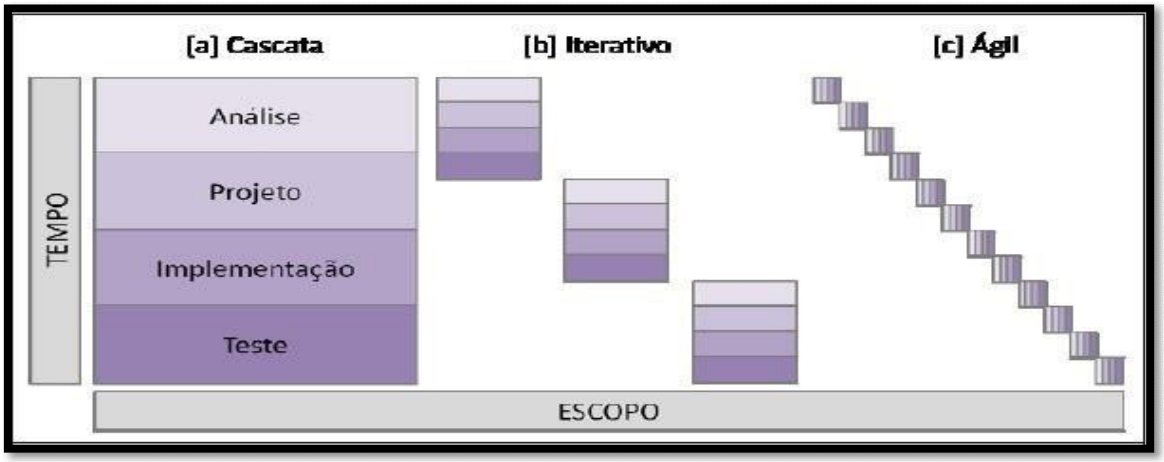
A presente seção traz uma análise comparativa entre os métodos citados anteriormente ao longo do estudo, onde são confrontados destacando suas similaridades e diversidades a partir de suas características, e ao final são descritos os resultados obtidos desta comparação de forma discursiva, ressaltando ainda fatores para a escolha de uma determinada metodologia.

Segundo Pressman (2016), essas metodologias de processo são a base de todo o projeto, sendo esta, a forma pela qual o projeto alcançará seu objetivo final e com qualidade.

Para se inteirar e analisar qual será a melhor abordagem a ser utilizada, o presente trabalho pode auxiliar com a comparação dos modelos de desenvolvimento a seguir. É importante conhecer e compreender como cada uma delas funcionam. A partir deste ponto, deve-se alinhar o que cada uma pode oferecer, de acordo com as necessidades de empresas, desenvolvedores e organizações.

Para iniciar a comparação, a Figura 13 abaixo, apresenta a evolução quanto a abordagem de gestão do ciclo de vida de projetos de desenvolvimento de software, presente em diferentes modelos. De acordo com Pedroso, Jouger e Oliveira em seu artigo, “Inicialmente a execução do ciclo de vida dos projetos é proposta em processos encadeados”, que é característica do modelo cascata, “onde cada etapa do ciclo é realizada por completo antes de iniciar a próxima, em seguida estão os processos mais Iterativos, com desenvolvimento incremental, e suas quatro etapas possuindo entregas incrementais.

Figura 13. Evolução na forma de execução de projetos de desenvolvimento de software



Fonte: BECK, (1999)

Por fim os autores abordam os atuais processos ágeis, “com pequenos ciclos repetitivos nas quatro etapas, sendo capaz de gerar produtos mais cedo” como pode ser analisado na Figura 13 a seguir.

Dando continuidade para as análises comparativas, na Tabela 06 a seguir, é realizada a comparação dos modelos de metodologias, tendo como critério e base para esta análise, características de gerenciamento de projetos.

Tabela 6. Análise comparativa entre metodologias de desenvolvimento de software

Tópicos	Métodos Tradicionais	Métodos Ágeis
Objetivo Principal	Orientado por atividades e centrado em processos.	Orientado por produto e centrado em pessoas.
Tipo de Projeto	Estáveis com baixos níveis de mudanças	Projeto com mudanças constantes e que necessitem de respostas rápidas.
Tamanho	Aplicável em projetos de todos os tamanhos.	Mais efetivo em projetos pequenos (5-10 pessoas).
Equipe	Atuação com papeis claros e bem definidos.	Atuação colaborativa em todos as atividades do projeto.
Cliente	Participa das fases iniciais de requisitos e das validações do projeto.	Essencial. Deve ser parte integrante da equipe do projeto.
Comunicação	De maneira formal	De maneira informal
Planejamento	Detalhado e os envolvidos têm o papel de validação.	Curto e com a participação de todos os envolvidos na elaboração do planejamento.
Execução	Deve seguir à risca o planejamento inicial, qualquer mudança deve passar por avaliação.	É feita em iterações (semanais, mensais, etc.). Toda mudança é bem-vinda.
Foco	Em seguir os processos corretos de um gerenciamento de projetos bem controlado, planejado, executado, organizado e documentado.	No produto final e satisfação do cliente.
Estilo de Gestão	Comando e controle	Liderança e colaboração
Atribuição de Papeis	Individuais, tem favorecimento por especialização.	Times autogeridos e ainda encorajam troca de papeis.
Sucesso	Entregar o Planejado	Entregar o desejado

Fonte: Elaborado pela autora

A tabela acima, demonstra algumas das principais características dos métodos. A partir dela, pode-se observar, que o método tradicional é centrado em processos, em projetos estáveis, tendo baixos níveis de mudanças, pode ser aplicado para qualquer tamanho de projeto, os envolvidos no projeto possui papeis bem definidos e a comunicação entre eles é de maneira formal, possui seu planejamento detalhando, e seguindo à risca esse planejamento pretende chegar ao sucesso que para esta metodologia, está em entregar o que foi planejado.

Quanto ao métodos ágeis, são centrados em pessoas, seus processos podem sofrer mudanças constantes, obtém efeito positivo mais em projetos pequenos, as atividades de seus papeis são realizadas de maneira colaborativa, mantendo uma comunicação informal, seu planejamento é curto e dispõe da participação de todos os envolvidos e assim chega ao seu sucesso que é entregar o produto desejado.

Com base na tabela apresentada, como característica de um gerenciamento de projeto é relatado um dos primeiros passos para o desenvolvimento que é o planejamento. Em todas as metodologias de desenvolvimento de projetos, existem planejamento antes da execução, não existe metodologia que recomende executar algo sem planejar. A diferença entre as metodologias é o tanto de planejamento que se faz, ou seja, o que foi programado antes do agir.

Nas metodologias tradicionais, se planeja com antecedência, início do preto, esse planejamento é todo detalhado, percorrido todo o projeto a ser feito e os envolvidos ainda têm o papel de valida-lo. Já nas metodologias ágeis o planejamento é feito de forma iterativa e incremental, é curto, e com a participação de todos os envolvidos na elaboração, desta forma tentando descobrir o trajeto do caminho a ser seguido.

De acordo com Soares (2005 apud Lutz 2017, p.35) as metodologias ágeis não apresentam grandes mudanças quando comparadas com as metodologias tradicionais, “o que as diferencia é a forma como são aplicadas e os seus valores”. Nas metodologias ágeis o foco é nas pessoas, na satisfação do cliente, enquanto que o modelo tradicional foca em seguir os processos corretos, planejados, executados, organizados e documentados. Como é demonstrado na tabela anterior.

Muitos dados e informações, podem remeter a diversas comparações a respeito das metodologias presentes neste trabalho, como pode ser visto a seguir, a comparação dos métodos a partir de elementos identificados no início do projeto por cada um deles.

Na metodologia tradicional, os elementos com maior prioridade, ao dar início ao projeto são: escopo, prazo e custo, questões de qualidade neste momento são

desconsideradas para um segundo plano, ou seja, para ela, o produto deve ser entregue conforme o planejado, na data determinada e como o valor acertado, para a qualidade não é dado muita ênfase, o que pode ao final do projeto não agradar muito o cliente.

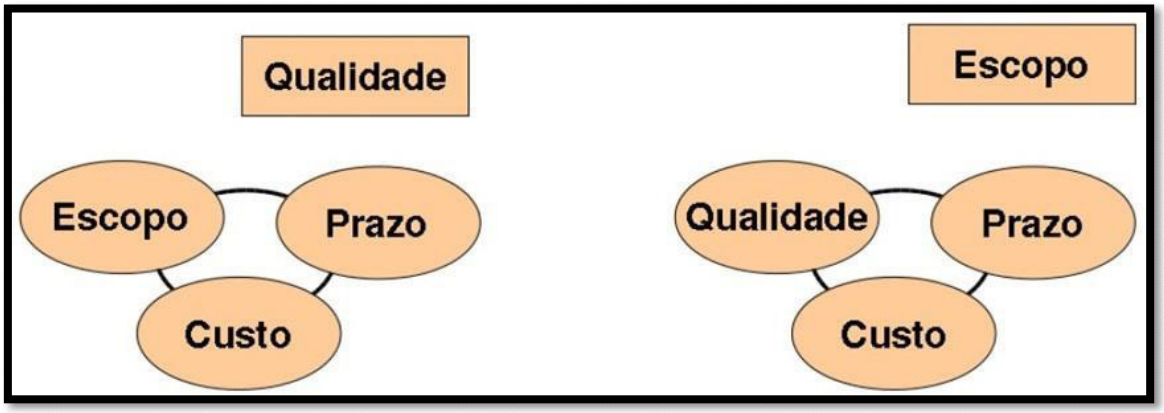
No escopo do modelo tradicional, a obtenção dos requisitos é volumosa, detalhada e demasiada nessa fase. “A equipe de desenvolvimento fará entrevistas com o cliente e com o usuário final, obtendo todas as informações a respeito das funcionalidades do software desejado, não podendo haver erro algum” (ALVES, 2012. p.26). Quanto ao custo, refere-se ao controle dos orçamentos realizados dos valores que envolverão a realização do projeto, o prazo remete-se as datas estipuladas de início e termino do produto e a qualidade contempla processos que conduzirão o projeto por um caminho que obedeça a padrões e normas de qualidade desejados.

Diferentemente da metodologia tradicional, a partir do estudo de Alves, as metodologias ágeis priorizam a qualidade, custo e o prazo. Neste momento coloca-se o destaque na qualidade, “o produto deve sempre possuir boa qualidade para alcançar a satisfação do cliente. E em relação de levantamento de requisitos, é dado a partir das estórias passadas pelo cliente de maneira superficial”. (ALVES, 2012 p. 26). Estórias são informações passadas pelos clientes, de como deseja que seu produto funcione, o que espera ter nele, ou seja, é algo mais básico, resumido quando comparado ao do modelo tradicional que é mais detalhado obtido através de uma entrevista.

A respeito do custo, é dado prioridade, pois nesse modelo os valores podem ser alterados ao longo do projeto, a partir de modificações realizadas em funcionalidades e outros aspectos, sendo que o cliente nunca terá uma precisão quanto a valores, isto será visto somente no fim do projeto. E o prazo é indeterminado, este modelo não prevê o fim do projeto, em razão de que a cada etapa do projeto o cliente recebe uma parte já implementada e funcionando do mesmo, sem falar ainda, que é considerado aberto a mudanças, então essas funcionalidades podem ser modificadas a qualquer instante aumentando ainda maia o desconhecimento acerca de prazos.

Fundamentada nas considerações acima, a Figura 14 a seguir, demonstra os fatores mencionados das metodologias e que são levados em consideração na gerencia de um projeto de acordo com cada metodologia, permitindo uma comparação entre esses fatores.

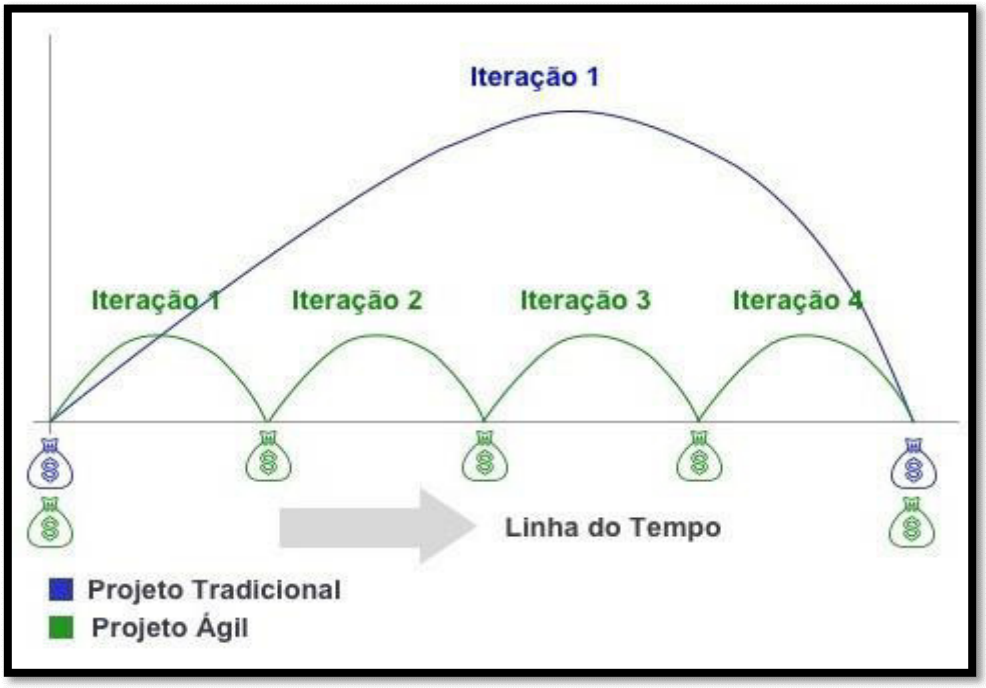
Figura 14. Figuras de projeto



Fonte: Barnett (2006)

Em seguida é ilustrado melhor o elemento custo, onde no modelo ágil, ao final de cada entrega o valor é negociado, sendo que cada entrega será agregado algum valor ao cliente. Já no modelo tradicional, como ele somente terá dois marcos, duas entregas, no início e no fim do projeto, provavelmente valores são entregues nesses dois momentos, com isso o cliente não terá mais nenhum valor agregado (LEONI,2017), diferentemente do ágil só receberá seu produto quando estiver totalmente pronto. Estas considerações podem ser demonstradas na Figura 15 a seguir.

Figura 15. Gráfico de custo de metodologias ágeis e tradicionais



Fonte: Leoni (2017)

Lutz (2017, p.36) em seu trabalho ressalta que, uma das principais características que diferencia as metodologias de desenvolvimento, “é a forma e o momento em que a documentação é gerada”, sendo que no método tradicional, “o software precisa estar todo especificado para que o desenvolvimento possa ter início”, enquanto que os métodos ágeis “focam em entregas e desenvolvimento”.

Como foi verificado anteriormente, quando se tratada do planejamento, desde esse momento pode-se perceber que a metodologia tradicional dá mais ênfase para documentação detalhada, quanto a metodologia ágil, realiza essa documentação meio que superficial, somente o que juga necessário.

Dando continuidade à análise comparativa, a seguir é apresentado uma tabela (Tabela 7) onde esses dois métodos são confrontados novamente. Neste momento a comparativa tem como base um desenvolvimento de *software*.

Tabela 7. Análise comparativa entre características das metodologias de desenvolvimento de software

Critério	**Descrição**	**Metodologias**	
		Tradicionais	**Ágeis**
Aberto a mudanças	Seu desenvolvimento pode ser modificado ou sofrer alterações em qualquer etapa.		✓
Comunicação com o cliente	Realiza comunicação com o cliente antes, durante e depois do desenvolvimento do projeto. Todos os requisitos são identificados junto ao cliente.	✓	✓
Documentação completa	É realizada uma documentação completa do software desde de seu planejamento.	✓	
Planejamento Iterativo	O planejamento é realizado de forma interativa entre desenvolvedores e cliente.		✓
Planejamento pré-definido	O seu planejamento é realizado logo no início do desenvolvimento.	✓	
Tempo de ciclo	São desenvolvidos em um tempo de ciclo sendo guiado por tarefas ou funcionalidades.	✓	✓
Testes unitários	Durante todo o desenvolvimento são realizados testes unitários de unidade.	✓	✓
Teste final	O software é testado como um todo ao final de seu desenvolvimento.	✓	✓

Fonte: Elaborada pela autora

A comparação dos modelos de metodologias, expondo para isto os principais aspectos de um desenvolvimento de software que diferenciam uma da outra. A tabela contém os critérios de comparação juntamente com a sua descrição respectivamente.

Por meio da análise comparativa entre os modelos de processos tradicionais e os modelos de processos ágeis, conclui-se que as duas metodologias resultam em um produto de software de qualidade.

Conforme acima representado, os dois modelos atendem a maioria dos critérios identificados para comparação. De acordo com o quadro apresentado as metodologias tradicionais atendem 75% dos critérios informados. E as metodologias ágeis, também atendem 75% dos critérios identificados assim como a anterior.

Observando a Tabela 7, a primeira impressão que se tem, é que esse comparativo levar à uma falsa impressão de que os dois métodos representam a melhor alternativa para o desenvolvimento de software, todavia, de fato ambos atenderam ao mesmo número de critérios, mas cada um possuindo algumas particularidades como o caso dos tradicionais que possuem uma documentação extensa do produto e aos ágeis são abertos a mudanças. Na verdade, a seleção da opção mais adequada dependerá de diversos fatores. Dentre as questões a serem consideradas estão as características específicas de cada projeto.

Segundo Alves (2012), a primeira diferença das metodologias ágeis para as tradicionais apontada, é a priorização das pessoas e não dos documentos. Pode-se confirmar está constatação a partir da tabela apresentada, onde a metodologia tradicional tem como assertiva de critério a documentação que é realizada desde o planejamento do projeto como foi relatado.

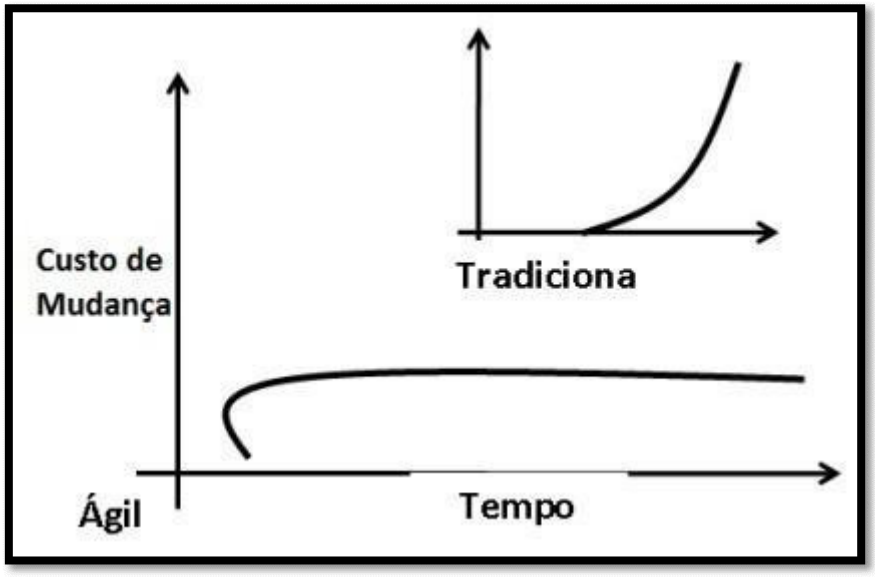
Ressaltando está afirmativa, Semedo diz que “para as metodologias ágeis a documentação não tem muita relevância. Elas têm como uma das suas principais preocupações, gastar menos tempo com a documentação e mais com a implementação”. (SEMEDO, 2012.p 76).

Dessa forma, percebe-se que as metodologias ágeis conseguem ser mais dinâmicas às mudanças de requisitos, pois conseguem realizar rápidos planejamentos e reuniões, visto que não existe a criação de uma documentação propriamente dita.

O modelo tradicional de *software* é mais restrito a mudanças, visto que exige e implica um aumento do orçamento e a ampliação do cronograma (SEMEDO, 2012.p.78). Já o modelo ágil diferentemente do tradicional, é aberto a mudanças, pode sofrer alterações em todos os momentos do processo de desenvolvimento, se necessário. Todas essas mudanças podem ser feitas sem ter que aumentar o orçamento (SEMEDO,

2012.p.78). A Figura 16 a seguir, demonstra esse aumento no valor que já estava planejado e as datas previstas são alteradas para prazos mais longos.

Figura 16. Custo das mudanças no desenvolvimento ágil e tradicional



Fonte: Castro (2007)

Para continuar comparando as duas metodologias, é voltado o foco neste momento para os tipos de métodos tradicionais e ágeis. Foram escolhidos alguns tipos como: Para os modelos de processos Tradicionais: Cascata, Incremental, RUP, RAD, e para comparar as metodologias ágeis, foram escolhidos os modelos Scrum, XP, DSDM e FDD. Estes foram comparados por meio de critérios referenciados nos autores Pressman (2016) e Sommerville (2011).

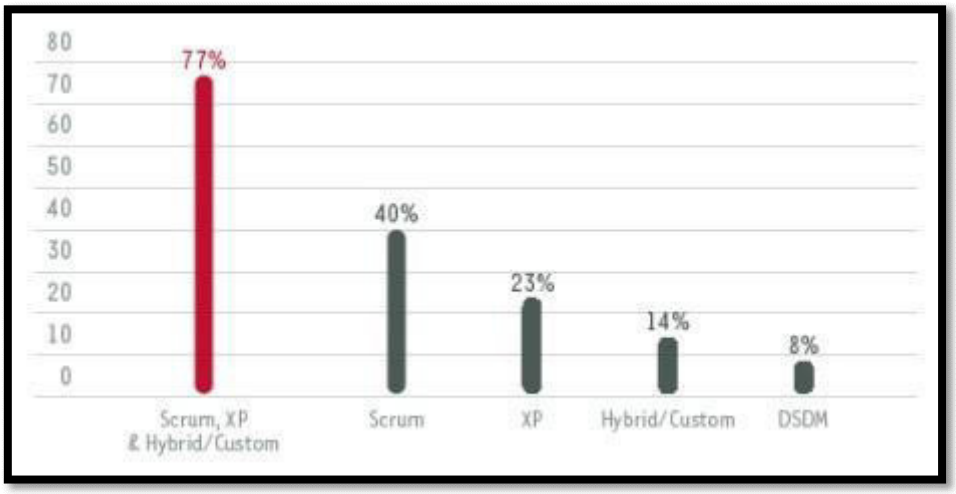
4.1 COMPARANDO MÉTODOS ÁGEIS

O ato de adotar processos mais simplificados, como é o caso das metodologias ágeis que procuram desenvolver *software* de maneira mais ágil, direta e pouca rigidez (CAMARA, 2005), tem despertado um grande interesse entre as comunidades de desenvolvimento de software nacional e até internacional como demonstra a pesquisa a seguir.

Entre os tipos de metodologia ágil, uma em especial se destaca no universo de desenvolvimento de software internacional, o Scrum (SCRUM ALLIANCE, 2007).

Pesquisas mostram que a metodologia Scrum possui uma grande aceitação pela comunidade de desenvolvimento de software internacional, como pode ser visto na Figura 17 a seguir.

Figura 17. Utilização de metodologias ágeis



Fonte: Versionone, State of Agile Development, (2016)

A metodologia Scrum possui o maior percentual de uso, logo após vem o modelo XP e em seguida o DSDM, com percentuais menores, mais bem utilizados ainda hoje. Quanto ao cruzamento de duas metodologias, como é demonstrado será um assunto tratado no final desta pesquisa.

A presente seção tem como objetivo confrontar metodologias ágeis fundamentada em suas características e elementos envolvidos no desenvolvimento de software.

Para comparar as Metodologias Ágeis, foram escolhidos os modelos SCRUM, XP, TDD, FDD e DSDM respectivamente. Foram contrapostos por meio de critérios referenciados pelo autor Pressman (2016) e Sommerville (2011).

De acordo com o é apresentado na Tabela 8, a seguir, o método XP que é encarado como uma metodologia ágil popular, e é utilizada quando os requisitos não são uniformes, ou seja, podem alterar em algum momento (SBROCCO; MACEDO, 2012). Foi o modelo que atendeu a maioria dos critérios identificados para comparação, como contato constante com clientes, planejamento iterativo, reuniões diárias, ciclo de desenvolvimento, equipes pequenas e outras, que o colocaram como o método que atendeu quase 100% dos critérios definidos.

Tabela 8. Análise comparativa entre metodologias ágeis

Critérios/Métodos	SCRUM	XP	FDD	DSDM
Quadro de Tarefas	✓			
Contato constante com o cliente.	✓	✓	✓	✓
Planejamento Iterativo	✓	✓	✓	✓
Reuniões Diárias	✓	✓		
Ciclo de desenvolvimento	✓	✓	✓	✓
Equipes pequenas	✓	✓		✓
Teste Unitário		✓		✓
Desenvolvimento incremental	✓	✓	✓	✓
Dupla de programadores		✓		
Inspeção de códigos			✓	

Fonte: Elaborada pela autora

Em seguida, vem o método Scrum, que é um processo bastante leve para gerenciar e controlar projetos de desenvolvimento de software e para criação de produtos (SANTIAGO, 2012), com sete critérios atendidos. Logo após, há o método DSDM (*Dynamics Systems Development Method*) que é uma metodologia de desenvolvimento de software originalmente baseada em "Desenvolvimento Rápido de Aplicação", defende as necessidades do negócio, participação ativa dos utilizadores, equipes capacitadas, entrega frequentes, testes integrados e colaboração das partes interessadas (ABRAHAMSSON et al., 2002).

E por último, atendendo o menor número de critérios do quadro, tem o FDD que segundo Martins (2006), é definido como um processo ágil, interativo que enfatiza a qualidade e realiza entregas frequentes.

É possível constatar que muitos destes métodos, demonstram similaridades entre si, como por exemplo, é claramente perceptível no quadro comparativo acima, que os critérios ciclo de desenvolvimento, planejamento iterativo e contato constante com os clientes estabelecidos para a comparação, são característica presentes em todas as metodologias ágeis apresentadas nesta tabela.

Pode se notar também, que, entretanto, algumas particularidades foram notadas em alguns dos métodos, como dupla de programadores no XP (SBROCCO; MACEDO, 2012), quadro de tarefas no Scrum (SANTIAGO, 2012), e inspeções de código no FDD (MATIAS, 2006). Assim como também apenas no Scrum e no XP, foi constatado a

reunião diária em seus projetos, onde realizam a troca de ideias e planejamento para a próxima funcionalidade a ser desenvolvida.

Todavia, o Scrum não indica nenhuma prática de programação específica, como o XP, por exemplo. Porém, possui um grande número de práticas, ou regras, para o controle gerencial do projeto (SANTOS, 2007.p. 53).

Sommerville (2011), afirma que os métodos ágeis são fundamentados no desenvolvimento Incremental. Isso explica o atendimento de todos os métodos neste critério, no quadro de comparação.

No que diz respeito a análise comparativa de métodos ágeis, podemos concluir que existe bastante igualdades entre si, fica provado após essa comparação que em todas as metodologias as etapas do processo de desenvolvimento são bem semelhantes, o que pode vir a diferencia-las é a forma do gerenciamento de seus projetos.

4.2 COMPARANDO MÉTODOS TRADICIONAIS

Para comparar as metodologias tradicionais, foram escolhidos os modelos Incremental, RUP, RAD e Cascata respectivamente. Foram comparados por meio de critérios referenciados pelos autores Pressman (2016) e Sommerville (2011). Na Tabela 9, foi realizada a análise comparativa.

Tabela 9. Análise comparativa entre metodologias tradicionais

Critérios	Incremental	RUP	RAD	Cascata
Resistente a mudanças			✓	✓
Planejamento pré-determinado	✓	✓	✓	✓
Foco na documentação	✓	✓		✓
Comunicação com o cliente.	✓	✓	✓	
Prioridade das necessidades de acordo com o cliente		✓	✓	✓
Testes unitários	✓	✓	✓	✓
Desenvolvimento interativo	✓	✓	✓	
Software é testado apenas no final		✓		

Fonte: Elaborada pela autora

Conforme acima apresentado, o modelo RUP (*Rational Unified Process*), que é definida como uma plataforma de processos de desenvolvimento de software, Foi o que atendeu o maior número dos critérios identificados para comparação como: ser resistente a mudanças, planejamento pré-determinado, foco na documentação entre outros. Em seguida, o modelo RAD, que possui a proposta de concluir o projeto, em um período preestabelecido e relativamente curto (BASSI FILHO, 2008). Atendeu à seis critérios identificados para comparação, logo após o primeiro modelo citado.

Já o modelo incremental, que é fundamentado em repetições de etapas, em que estas são apenas melhoradas, e uma nova versão surge ao final de cada iteração (SANTIAGO; 2012), juntamente com o modelo cascata usualmente chamado de ciclo de vida clássico ou modelo em cascata, é o método mais antigo e o mais amplamente utilizado das metodologias tradicionais (SBROCCO; MACEDO, 2012), atenderam ao menor número de critérios estabelecidos.

Conforme pode ser verificado no quadro, o modelo em cascata possui resistência a mudança, é rígido quanto a documentação do projeto, seu desenvolvimento deve ser dirigido a planos, ou seja, deve haver um planejamento de todas as atividades do processo (SOMMERVILLE, 2011).

Souza (2014), Aborda em seu estudo que o modelo incremental segue uma ideia parecida com o modelo em cascata, tanto que atenderam ao mesmo número de critérios no comparativo feito, porém relata que o primeiro citado é mais iterativa.

Foi possível notar, a partir da Tabela 09 que os métodos têm bastante semelhanças entre si, no entanto o modelo RUP possui uma particularidade, a de testar o software apenas no final. Utida (2012), confirma essa ideia, quando diz que nesta “testa-se o sistema para validá-lo contra as expectativas dos usuários finais. Nesta fase ocorre também, o treinamento e capacitação dos usuários” (UTIDA,2012).

Ao término desta comparativa, pode-se concluir que assim como os métodos ágeis, os métodos aqui citados possuem semelhanças e particularidades, porém, suas práticas podem ser complementares, onde pontos falhos de uma podem ser supridos por pontos fortes da outra. (SILVA, SOUZA & CAMARGO, 2013).

Tendo em vista as comparações realizadas, fica incontestável, que não existe uma metodologia considerada melhor entre elas, existe a que melhor se enquadra na necessidade da organização naquele momento. Tudo isso será verificado melhor a seguir, onde são dadas ideias, estatísticas e análises de autores a respeito do uso de metodologias.

4.3 FATORES DE UTILIZAÇÃO DE METODOLOGIAS DE DESENVOLVIMENTO

Nerur, Mahapatra e Mangalarg (2005), afirmam que metodologias de desenvolvimento de software, estão evoluindo constantemente devido aos avanços tecnológicos. Os autores Também ressaltam que “embora metodologias tradicionais continuem a dominar a área de desenvolvimento de software, inúmeras opiniões e questionários veem demonstrando a crescente popularidade das metodologias ágeis”.

Eder et al (2013), apresentaram como resultado de sua pesquisa, que a diferença principal das abordagens analisadas está nas “técnicas empregadas, ou seja, seu planejamento pode até ser parecido, mas sua maneira de executar é de certa forma diferente”.

Segundo Souza (2014), a escolha de uma metodologia a ser utilizada no desenvolvimento de software é de suma importância. “Deve ser realizada com base na natureza do projeto e do produto a ser desenvolvido, dos métodos e ferramentas a serem utilizadas e dos controles e produtos intermediários desejados” (WILLIAMS, 2010).

É notável que a diferença entre as metodologias ágeis e tradicionais, está relacionada com a maneira de gerenciar seus processos e a forma como acontece o envolvimento das pessoas nesse mesmo processo.

Contudo, escolher uma metodologia de projetos eficiente, é um passo fundamental para garantir o sucesso de um negócio. Não existe uma regra que defina quando se deve usar uma metodologia de projetos ágil ou tradicional. É importante observar e selecionar aquela que vai se encaixar melhor nos objetivos determinados. É necessário analisar as diferenças entre elas e ajustar de acordo com a característica e os objetivos do projeto.

Em seus resultados de trabalho, os autores Laanti, Salo e Abrahamsson (2011), exibem que a aceitação em favor de métodos ágeis, pode ser considerada como positiva, já que “60% dos entrevistados em seu estudo, preferem continuar no modo de trabalho ágil, enquanto que apenas 9%, preferem os métodos tradicionais de trabalho”. Todavia, segundo os autores também existi aqueles entrevistados que não veem nenhuma diferença ou não tem uma opinião acertada sobre a análise das metodologias.

Para esclarecer ainda mais estas duas abordagens, possuindo uma opinião mais acertada sobre, é proposto abaixo exemplos de aplicabilidade na atualidade das metodologias tradicionais e ágeis.

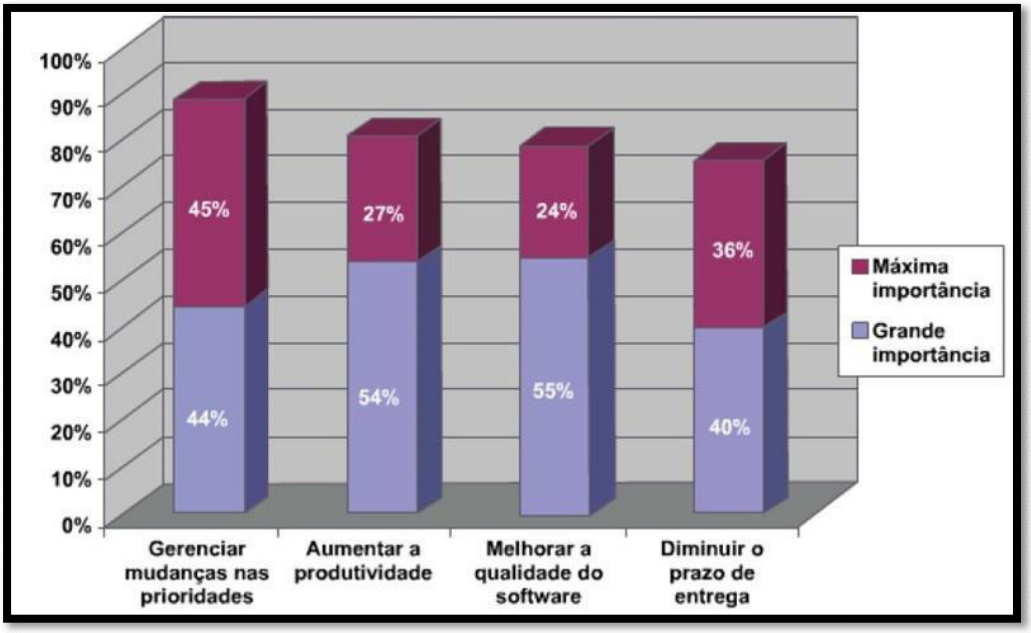
Tonon (2015) apresenta em seu artigo, um bom exemplo de aplicabilidade da metodologia tradicional, ele menciona os projetos ou licitações do governo. “Diz que nesses dois casos as metodologias tradicionais funcionam bem, dado que é preciso apresentar um escopo totalmente detalhado nessas situações, de acordo com o que o edital pede”.

Nessa situação, se observa que será necessária mais rigidez, característica predominante em métodos tradicionais. Logo, os métodos tradicionais podem ser mais aplicáveis sem maiores problemas ou custos extras, como foi mencionado nas tabelas anteriores, pois geralmente seu custo é especificado no início do projeto.

Um outro exemplo citado por Tonon (2015), mas agora apontado para metodologias ágeis, segundo o autor, é o de um projeto novo e com inovações para uma marca de celulares por exemplo, “neste caso é bem mais interessante utilizar os métodos ágeis, uma vez que não há a necessidade de se fazer um produto totalmente completo logo no primeiro momento”. Pode- se lançar uma versão inicial do aparelho tendo algumas funcionalidades diferenciadas. (TONON, 2015). Esses acontecimentos remetem ao uso de características dos métodos ágeis.

Leitão (2010), aborda em seu trabalho uma análise de fatores que levaram muitas pessoas à adoção de uma metodologia ágil, sendo assim a Figura 18 a seguir, demonstra os fatores identificados.

Figura 18. Fatores de adoção de uma metodologia ágil



Fonte: Barnett (2006)

Observando os resultados apresentados na figura, é possível destacar o aumento da produtividade e da qualidade do *software*, “associado ao gerenciamento de requisitos em constante variação, sendo o principal motivador da adoção de uma metodologia ágil de desenvolvimento” (LEITÃO, 2010 e Barnett, 2006).

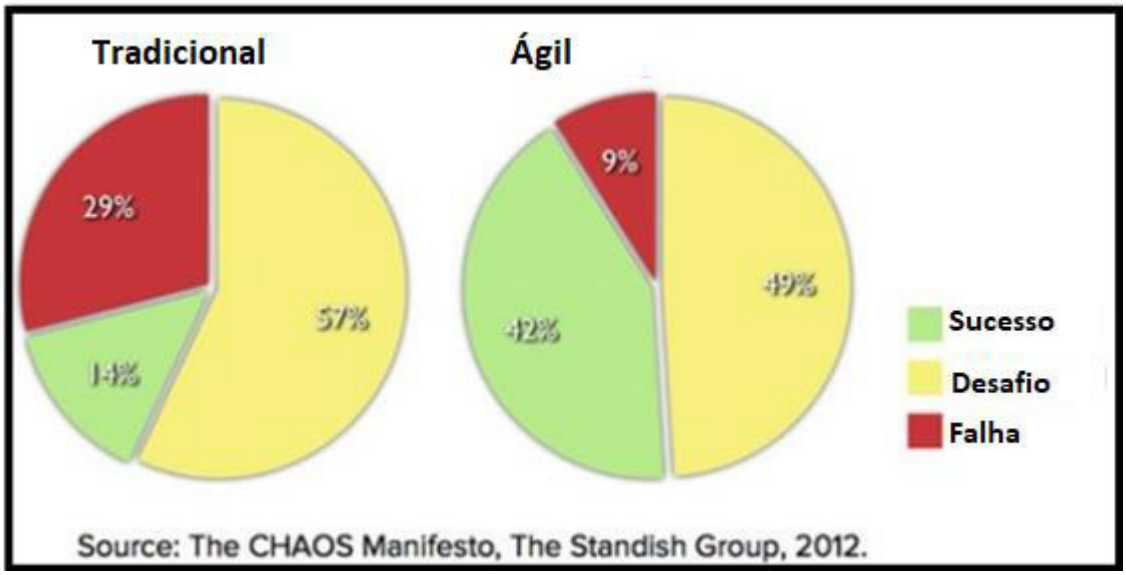
Bastos (2013), em sua publicação na CIO, afirma que a cada ano o relatório CHAOS Manifesto da consultoria Standish Group, mostra mais vantagens para o ágil em relação ao tradicional. Relata que no ano de 2011 o percentual de sucesso dos projetos ágeis foi de 42% contra 14% dos tradicionais. E os projetos ágeis falharam em 9% contra 29% dos projetos tradicionais.

Ainda Conforme Bastos (2013), “esse crescimento à adesão aos métodos ágeis se justifica pela vantagem inegável a adaptabilidade às mudanças, ou seja, nesta pesquisa os envolvidos agregam valor a esta característica dos métodos ágeis, que é um objeto de comparação neste trabalho”.

A pesquisa ágil de Xebia (2012), apresentou que 80% dos entrevistados concordaram em fazer uso da metodologia Ágil para o desenvolvimento de software. Além disso, a pesquisa também concluiu que 92% dos entrevistados seguem o Scrum e o Extreme Programming (XP).

De acordo com o relatório da CHAOS (2012), a Figura 19 a seguir, demonstra o claro aumento de índice de sucesso dos projetos que utilizaram metodologias ágeis em seus processos.

Figura 19. Representação de índice de sucesso e falhas em projetos tradicionais e ágeis



Fonte: Standish Group - Chaos Report (2013,

Com base no gráfico acima, é apresentado uma tabela retirada do relatório da CHAOS Manifesto (2012), Essa aponta um dado bastante interessante, onde o Standish Group revela projetos de software, que foram desenvolvidos seguindo paradigmas de metodologias ágeis, "obtiveram taxa de sucesso três vezes maior quando comparados às formas tradicionais" (MANIFESTO,2012), conforme Tabela descrita a seguir.

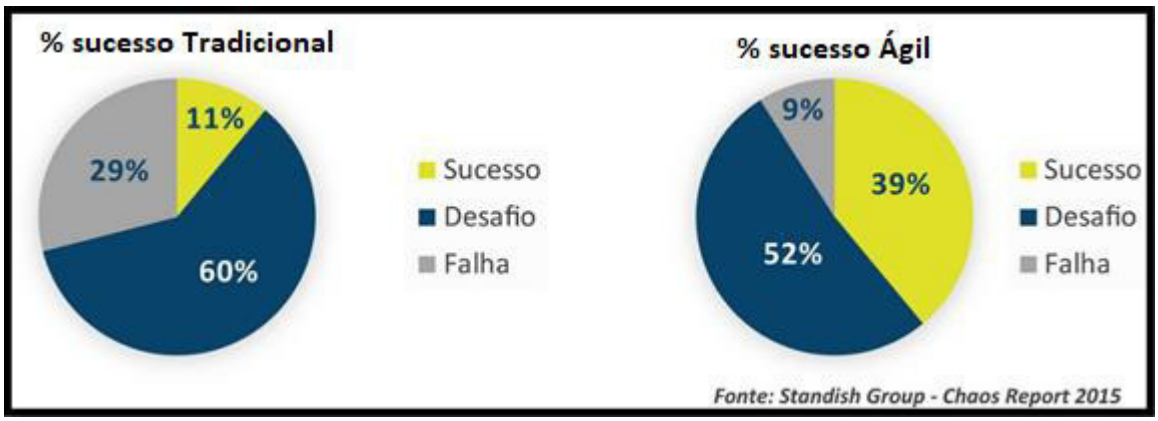
Tabela 10. Índice de sucesso por método

Classificação / Abordagem	Tradicional	Ágil
Bem-sucedido	14%	42%
Posto em risco	57%	49%
Cancelado	29%	9%

Fonte: Standish Group- Chaos Report (2013, p25)

A tabela acima, demonstra claramente a afirmação feita anteriormente, que demonstra que o índice de sucesso das metodologias ágeis é bem maior quando comparado ao índice de sucesso das metodologias tradicionais. A partir de um novo relatório de pesquisa realizada pela CHAOS, só que agora passando-se três anos depois do citado acima, é possível perceber e ainda confirmar o que foi apresentado anteriormente, é perceptível o sucesso dos métodos ágeis, continuando com o índice maior de sucesso, desafio e menor de falhas, quando comparado com os métodos tradicionais (CHAOS,2015). Tudo isso pode ser verificado a partir da Figura 20 a seguir.

Figura 20. Índice de sucesso, desfio e falhas de métodos tradicionais e ágeis



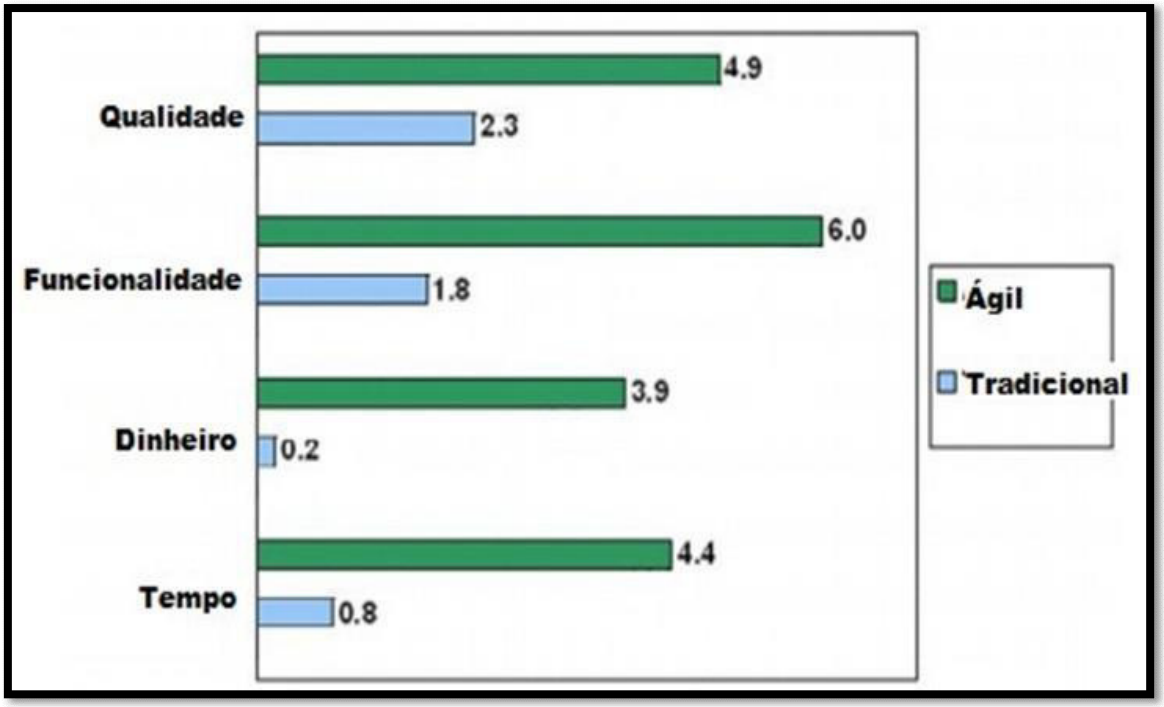
Fonte: Standish Group - Chaos Report (2015)

Pode-se observar em outras pesquisas como a de Ambler (2010), outros resultados a respeito do desfecho de projetos bem-sucedidos com o uso de metodologias tradicionais e ágeis. Esta pesquisa foi realizada com base em quatro fatores que o autor julga importante. Que, "para que se obtenha um projeto de qualidade deve se possuir: Funcionalidade, qualidade, tempo e dinheiro" (AMBLER,2010).

Na figura a seguir, Scott apresentou índices de percentagens entre as duas metodologias de desenvolvimento de software, e a partir disso pode-se concluir que os índices de sucesso das metodologias ágeis ainda são maiores que as das metodologias tradicionais. "Visto que para além de serem mais rápidas a apresentar resultados, fornecem rapidamente funcionalidades" (SEMEDO, 2012. P. 87).

As metodologias ágeis quanto a custo, tempo, funcionalidade e qualidade estão à frente das metodologias tradicionais, visto que qualidade é um valor que o método ágil mais preza em um projeto, funcionalidades são entregues incrementalmente, o tempo e o custo são termos que não são especificados no início do projeto, algumas vezes podem acabar prejudicando o produto final. Tudo isso pode ser verificado na Figura 21.

Figura 21. Fatores de sucesso das metodologias de desenvolvimento de software.



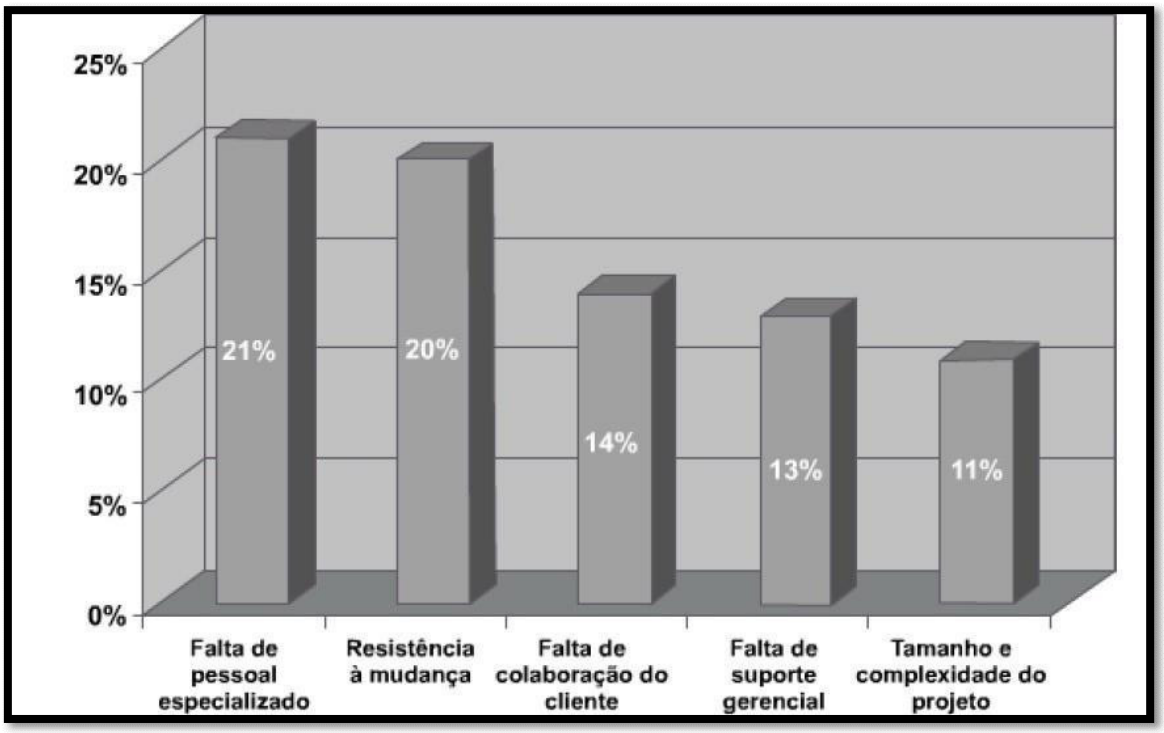
Fonte: Adaptado de Scott w. Ambler (2010)

Apesar das metodologias ágeis fazerem muito sucesso e obterem bons resultados, existem ainda barreiras que dificultam a sua implantação em qualquer organização, como será visto a seguir.

De acordo com Leitão (2010), existem algumas barreiras que não permitem que empresas adotem o método ágil. Conforme o autor "a resistência dos próprios desenvolvedores à mudança, são agora apontados como os principais fatores".

É possível observar, que as barreiras presentes na Figura 22, fazem parte de características relacionadas ao desenvolvimento tradicional, como a resistência a mudanças, ou seja, para que empresas desejem optar por uma metodologia ágil deverão se desprender de atividades tradicionais.

Figura 22. Barreiras para adoção do desenvolvimento ágil



Fonte: Barnett (2006)

Em síntese, a partir dos resultados demonstrados anteriormente, conclui-se que não existe a possibilidade de definir que uma é melhor que a outra, o fator determinante tem a ver com o ambiente, cultura, equipe e entre outros. Lembrando que as metodologias possuem características que podem ser adaptadas à necessidade de cada empresa, se notarmos o foco principal em todas as metodologias, temos o desenvolvimento por incremento, a comunicação constante com o cliente.

Para a seleção da abordagem mais adequada, dependerá de diferentes variáveis, como: características de cada projeto, da empresa e do que é requisitado pelo cliente. Em ambientes mais formais com equipes grandes costuma-se indica as metodologias tradicionais.

Segundo estudiosos um fator de importância decisivo para se obter ou não sucesso em um projeto, é o de definir qual a metodologia e ainda saber utiliza-la. Uma escolha ruim ou, ainda, uma má utilização dos recursos oferecidos por elas, aumenta consideravelmente os riscos de os projetos não obterem sucesso. Saber utilizar é tão importante quanto a escolha de qual delas utilizar.

Porém se faz necessário estudos mais exatos a respeito tanto das metodologias ágeis quanto das tradicionais, pois cada uma atende um público diferente da comunidade de desenvolvimento, e ainda existe aqueles que preferem dizer não a mudança e persistir com o tradicional, por isso é preciso compreender o ambiente para assim poder optar por um dos métodos para facilitar desta forma o desenvolvimento de produtos.

5. CONSIDERAÇÕES FINAIS

Após a leitura do presente trabalho, pode-se chegar à conclusão de que todo e qualquer tipo de metodologia de desenvolvimento pode dinamizar os processos e no final resultar em bons produtos, isso desde que se tenha feito a escolha e aplicação da metodologia correta, a que mais traduza o cenário pretendido.

Isso tendo associado essas metodologias à engenharia de *software*, que permite um total planejamento, organização e no fim resultar em projeto de boa qualidade.

Logo, o presente trabalho trouxe consigo a proposta de auxiliar indivíduos, organizações e desenvolvedores a escolherem a metodologia que melhor vai auxilia-los nos processos de desenvolvimento, para isso realizou-se uma análise comparativa entre os modelos de processo de desenvolvimento de *software*.

Foi abordado ao longo do estudo diversos tipos de metodologias, sendo elas classificadas como tradicionais e ágeis, que neste trabalho foram objetos de comparação. Foi realizado um comparativo entre elas, ressaltando suas características e aspectos que envolvem o seu processo em meio ao desenvolvimento de software, além de apresentar ideias, fatores e resultados que auxiliaram na escolha da metodologia.

A ideia da pesquisa foi a de contribui para a solução de um problema comum na área de gerenciamento de projetos: como o de identificar o uso ou não da abordagem de gerenciamento ágil e tradicional por uma organização e ainda contribui tentando esclarecer quais os problemas que estão envolvidos na insatisfação quanto ao desenvolvimento de *software*.

Foi demonstrado que ambas as metodologias atenderam a 75% dos critérios exigidos no comparativo, revelando que possuem similaridades e foi possível destacar algumas particularidades a partir do confronto entre os tipos de metodologias, como por exemplo, o método ágil Scrum que foi o único que atendeu o critério de possuir um quadro onde são colocadas, fases, atividades do processo para organização de equipe.

Foi possível constatar, a partir da comparação realizada entre os tipos de métodos ágeis, que a maioria são bastante semelhantes quanto suas características, e que os critérios de ciclo de desenvolvimento, planejamento iterativo e contato constante com os clientes estabelecidos para a comparação, são as características que foram correspondentes a todas as metodologias ágeis apresentadas na tabela. Porém o método XP foi o que atendeu o maior número de critérios, quanto o FDD atendeu ao menor número de critérios estabelecidos para o comparativo.

5.1 PRINCIPAIS ANÁLISES

Ficou perceptível algumas particularidades em alguns dos métodos, como dupla de programadores no XP (SBROCCO; MACEDO, 2012), quadro de tarefas no Scrum (SANTIAGO, 2012), e inspeções de código no FDD (MATIAS, 2006). Assim como também apenas no Scrum e no XP, foi constatado a reunião diária em seus projetos, onde realizam a troca de ideias e planejamento para a próxima funcionalidade a ser desenvolvida.

Quanto ao comparativo realizado entre os tipos de metodologias tradicionais, pode-se ser verificado, assim como nos métodos ágeis, que essas metodologias possuem diversas características similares, mas que no entanto somente o método RUP atendeu ao maior número de critérios estabelecidos para comparação, e os métodos incremental e modelo cascata, obtiveram o mesmo número de critérios atendidos, concluindo que estes atenderam o menor número de critérios.

Ao término destas comparativas, pode-se concluir que todas a metodologias aqui apresentadas, possuem semelhanças, distinções e particularidades, Porém, suas práticas podem ser complementares, onde pontos falhos de uma podem ser supridos por pontos fortes da outra. (SILVA, SOUZA & CAMARGO, 2013), afirmação que será aprofundada e confirmada a parir de trabalhos futuros.

Diversas pesquisas evidenciaram que as metodologias ágeis vem adquirindo bastante aceitação no universo de desenvolvimento, foi verificado no estudo de Laanti, Salo e Abrahamsson (2011), que a aceitação dos métodos ágeis pode ser considerada como positiva, já que 60% dos entrevistados em seu estudo preferem continuar com a metodologia ágil, enquanto que apenas 9% preferem a metodologia tradicional.

Bastos (2013), em sua publicação no CIO, através do relatório da CHAOS, demonstra que à mais vantagem para métodos ágeis do que tradicionais. Explica que o sucesso dos projetos ágeis foi de 42% contra 14% dos tradicionais. E os projetos ágeis falharam em 9% contra 29% dos projetos tradicionais, resultados que confirmam a afirmação dada anteriormente. Em conformidade com as demais estatísticas apresentadas, a Pesquisa ágil de Xebia (2012), demonstra também que as metodologias ágeis estão na frente quanto a aceitação por usuários, diz que 80% dos entrevistados concordaram em fazer uso da metodologia ágil para o desenvolvimento de software. Além disso, a pesquisa também concluiu que 92% dos entrevistados já seguiam o modelo Scrum e o Extreme Programming (XP).

Por outro lado, foi demonstrado também, que ainda existe um bom número de organizações que são adeptas as metodologias tradicionais e que não pretendem realizar mudanças quanto a seu padrão de uso de métodos. Como pode ser verificado na Figura de barreiras de adoção ao desenvolvimento ágil, que cujo as características evidenciadas como barreiras, traduzem ao uso de metodologias tradicionais, ou seja, ainda hoje a barreira principal é o não desprendimento de organizações ao uso padrão de características tradicionais.

Complementando pode-se dizer que a melhor metodologia para ser aplicada em um projeto é a que mais se adequa a necessidade do projeto proposto. Metodologia ágil funciona bem, realiza entregas rápidas, porém, possui suas limitações. Quando existem requisitos complexos, que necessite de rigidez, e o projeto não pode ser ajustável, então é mais correto utilizar as metodologias tradicionais.

É importante ressaltar para o desfecho dessa comparação, que mesmo com as metodologias ágeis difundidas pelo mundo todo e com crescente aceitação, a engenharia de *software,* não deixará as metodologias tradicionais de lado, ou esquecidas, estas ainda possuem um grande valor e continuam tendo um papel muito importante no processo de desenvolvimento, existem lacunas que somente elas podem preencher e depois, funcionam também como uma espécie de complemento para metodologias ágeis e vice-versa.

5.2 DIFICULDADES E TRABALHOS FUTUROS

As principais dificuldades encontradas na execução desta monografia foi em nortear o estado da arte, onde fora necessário filtrar conteúdos e retirar o que foi considerado importante, a respeito das metodologias tradicionais, onde ocorreu certa escassez de dados. A maioria dos autores avaliados abordam partes bem pequenas do conteúdo da abordagem tradicional. Além disso, o processo de busca nas comunidades e sítios particulares financiados pela Universidade para obter trabalhos científicos e acadêmicos da área relacionada, além de um grande filtro para trabalhar com estes.

No futuro, pretende-se aplicar os conhecimentos obtidos, aperfeiçoados e relembrados através desta pesquisa em ambientes que utilizam o processo de desenvolvimento de *softwares* em empresas, indústrias e em locais onde estes são continuamente aplicados.

Pretende-se também avaliar o envolvimento de mais de um tipo de metodologia, ou metodologias híbridas em um mesmo processo de desenvolvimento de *software*.

REFERÊNCIAS BIBLIOGRÁFICAS

ALMEIDA, G.A.M. Fatores de escolha entre metodologias de software tradicionais e ágeis. 2017. Dissertação (Mestrado)-Escola Politécnica, Universidade de São Paulo.

AGILE MANIFESTO (2015). Manifesto for Agile Software Development. Disponível em:<http://agilemanifesto.org>. Acessado em 03 de abril de 2018.

Ambler, S. – Agile adoption rate survey. 2010. Disponível em: http://www.ambysoft.com/surveys/. Acessado em 09/08/2018.

ALMEIDA, Vitor FDD - Desenvolvimento Guiado por Funcionalidades. 2015. Disponivel:http://www.itnerante.com.br/profiles/blogs/fdd-desenvolvimento-guiado-por-funcionalidades.Acessado em: 23/11/2018.

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. NBR ISO 8402/1994 – Gestão da qualidade e garantia da qualidade - Terminologia. Rio de Janeiro: ABNT, 1995.
ALVES, Bruno Junqueira. Estudo Comparativo entre Métodos ágeis e Tradicionais de Fabricação de Software. Recife, 2012.

ABRAHAMSSON, P.; SALO, O.; RONKAINEN, J. & WARSTA, J. Agile Software Development Method: Reviews and Analysis. Espoo: VTT Publications, 2002. Disponível em: http://www.inf.vtt.fi/pdf/publications/2002/P478.pdf>. Acesso em: 20 set. 2018.

ASSIS, Frederico Souza Picorelli. Gerenciamento de Projetos por Meio da Metodologia Scrum Aplicada em uma Empresa de Base Tecnológica. Juiz de Fora, 2016.

AMBLER, S. W.; JEFFRIES, R. Agile Modeling: Effective Practices for Extreme Programming and the Unified Process. Nova York: John Wiley & Sons, 2002.

BECK, K. Agile Manifesto. 2001. Disponível em: http://www.agilemanifesto.org/. Acesso em: 10 mar. 2018.

BECK, K. (1999). Embracing Change with Extreme Programming. IEEE 1999. Disponível: em http://www.cs.umd.edu/class/spring2003/cmsc838p/Process/extreme.pdf. Acesso em: 11/2008.

BECK, K.; Extreme Programming Explained; 1ª ed., 1999.

BECK, K. Extreme Programming Explained: Embrace Change. New York. 2001.

Barboza, et al. Análise comparativa entre as abordagens ágil e tradicional de gestão de Projetos: Um estudo de caso no setor industrial. São Paulo, 2016.

BENZECRY, Fernando Salztrager. Metodologias Ágeis para Gerenciamento de Projetos de Inovação e Pesquisa e Desenvolvimento. Rio de Janeiro, 2017.

BORBOREMA, Thiago. Impacto da Aplicação da Metodologia XP nas Organizações de Desenvolvimento de Software. Pouso Alegre,2007.

BARROS, Dennys de Souza. Qualidade de Software em Métodos Ágeis: Uma Revisão Sistemática. Recife, 2018.

BARNETT, 2006 BARNETT, L. Agile Survey Results: Solid Experience And Real Results. Agile Journal. 2006.

BASSI FILHO, D. L. Experiências com desenvolvimento ágil. 2008. Tese de Doutorado. Universidade de São Paulo.

COHN, M. Succeeding with Agile: Software Development Using Scrum. Boston: Addison-Wesley, 2009.fer

COHN, M. (2011) Desenvolvimento de Software com Scrum. Porto Alegre: Bookman.

COSTA. FILIPE SOUSA, Fusão de um processo tradicional de desenvolvimento de software com uma metodologia ágil: um estudo de caso. Universidade Federal de Uberlândia, 2017.

COCKBURN, Alistair. Agile Software Development: The Cooperative Game. Addison-Wesley: 2002.

COCKBURN, A e HIGHSMITH, J. Agile Software Development: The Business of Innovation. IEEE Computer, 2001.

CASTRO, Vinicius. Introdução ao desenvolvimento.2007. Disponível em: ágil.http://devagil.wordpress.com/2007/07/07/introducao-ao-desenvolvimento-agil/. Acessado em: 26/07/2018.

CABRAL. Estudo comparativo entre metodologias ágeis e tradicionais. 2006.

COELHO, Cristiane dos Santos. Relato de Experiência na Implantação de um Método Ágil em uma Equipe de desenvolvimento de software. 2011. DESENVOLVIMENTO ÀGIL, Scrum. Disponível em: http://desenvolvimentoagil.com.br/scrum. Acesso em 22 /04/ 2018.

EKERT, VALTER RODRIGO. Identificação dos processos ágeis para desenvolvimento de software em microempresas na cidade de Medianeira/PR. 2011. 59 páginas. Monografia (Especialização em Engenharia de Software). Universidade Tecnológica Federal do Paraná, Medianeira, 2011.

EDER, S. et al. Diferenciando as abordagens tradicional e ágil de gerenciamento de projetos. Production, v.25, n. 3, 2013, pp. 498-509.

FERNANDES, MATHEUS RAMOS. SCRUM E XP: Um comparativo no processo de desenvolvimento de software.2011. Universidade fumec faculdade de ciências empresariais – FACE. Belo Horizonte.

FOWLER, Martin, The New Methodology, 2003. Disponível em: Acessado em: 18/04/2018.Disponível:http://www.martinfowler.com/articles/newMethodology.html. Acesso em: 04/04/2018.

GONZALES, Thalisa Silva de Sotomayor. Metodologia Ágil com Scrum: Estudo do Processo de Implantação em uma Software House de Pequeno Porte. Anápolis, 2015.
GUDWIN, RICARDO R. Engenharia de Software: Uma Visão Prática 2ª ed. DCA-FEEC-Unicamp, 2015.

GOOGLE. Google Acadêmico. 2018. Disponível em: <https://scholar.google.com.br/intl/ pt-BR/scholar/about.html>. Acesso em: 10/01/2018.

IEEE, Institute of Electrical and Electronics Engineers. IEEE Standard for Software Test Documentation. New York, USA, 1990.

IEEE - Institute of Electrical and Electronics Engineers. ISO/IEEE Std. 12207: Systems and software engineering - Software life cycle processes. Montréal, 2008.

JESUS, Metodologias Ágeis de Gerenciamento de Projetos Para Startups. 2016.

KOSCHEVIC, MARCELA TURIM. Gerenciamento de processos com metodologias ágeis. 2011. Monografia (Especialização em Engenharia de Software). Universidade Tecnológica Federal do Paraná, Medianeira, 2011.

LEONI, André Luiz. Quando devo usar metodologia Ágil para Gerenciar meus Projetos? Artigo,2017. Disponível em: https://medium.com/@a_luizleoni/quando-devo-usar-metodologia-%C3%A1gil-para-gerenciar-meus-projetos-513534374c70. Acessado em: 12/06/2018.

LIMA, Greick Roger de Carvalho. Benefícios das metodologias ágeis no gerenciamento de projetos de tecnologia da informação. 2015.

LIBARDI, Paula L O e VLADIMIR, Barbosa. Métodos Ágeis. Limeira - SP2010.

LIMA, Wilian Vargas. Scrum no Brasil. Florianópolis 2011.

LEITÃO, Michele de Vasconcelos. Aplicação de Scrum em Ambiente de Desenvolvimento de Software Educativo. Recife, junho de 2010.

LEIDEMER, Rômulo Henrique. Implantação de Scrum em uma Empresa de Desenvolvimento de Software. Lajeado, 2013.

LUTZ, DOUGLAS. Implantação de Novo Processo de Trabalho em Uma Fábrica de Software Baseado nos Modelos Ágeis de Desenvolvimento. Universidade do vale do taquari - Curso de sistemas de informação. Lajeado, novembro de 2017.

LAANTI, M; SALO, O; ABRAHAMSSON, P. Agile Methods Rapidly Replacing Traditional Methods at Nokia: A Survely of Opinions on Agile Transformation. Information And Software Technology, V.53, 2011.

LOFF, DIOGO ANDRÉ. Aplicação de métodos ágeis de engenharia de software em um sistema caótico. Universidade do sul de santa catarina. Araranguá 2010.

MATIAS. LORIVAL WARMELING. Avaliação dos métodos de desenvolvimento tradicionais e ágeis para aplicação em uma empresa de software. Disponível em: http://www.projedata.com.br/images/notificacoes/artigo_lorival_metodos_de_desenvolv imento.pdf. Acesso em: 23.07.2018.

MANIFESTO ÁGIL. 2001. Disponível em: <http://www. manifestoagil.com.br/>. Acesso em: 08/03/2018.

NERUR, S; MAHAPATRA, R; MANGALARAJ, G. Challenges of Migrating To Agile Methodologies. Communications of The Acm, V.38, 2005.

OLIVEIRA, Bruno Carazato. Gaia protótipo: um modelo de prototipação Para processos de desenvolvimento de Software. Londrina 2018.

OLIVEIRA, Ebenezer Silva. Uso de Metodologias Ágeis no Desenvolvimento de Software. Belo Horizonte, 2003.

PEDROSO. L.S, OLIVEIRA, L, R e Jaeger, J.I -Análise da Aplicação de Métodos Ágeis na Gestão de Projetos em Empresas Desenvolvedoras de Software. VII Convibra Administração – Congresso Virtual Brasileiro de Administração – Artigo. Disponível em: http://www.convibra.com.br/upload/paper/adm/adm_1118.pdf. Acesso em: 19/05/2018.

PRESSMAN, Roger S. (2006) Engenharia de Software. 6ª ed. McGraw-Hill.

PRESSMAN, Roger. S. (2011) Engenharia de Software. 7. ed. Porto Alegre: Pearson Makron Books.

PRESSMAN, Roger S.; Bruce R. Maxim. Engenharia de Software, Uma Abordagem Profissional, 8° ed. Porto Alegre: AMGH, 2016.

PAULA FILHO, W. Engenharia de Software: Fundamentos, Métodos e Padrões. 2ª Ed., Rio de Janeiro: LTC, 2000.

PFLEEGER, S. Engenharia de Software: Teoria e Prática. 2ª Ed., São Paulo: Pearson Prentice Hall, 2004.

PRIKLADNICKI, Rafael. Problemas, desafios e abordagens do processo de desenvolvimento de software. Porto Alegre, 2004.

PALMER, S. R.; FELSING, J. M. A Practical Guide to Feature-Driven Development. Englewood Cliffs, NJ: Prentice Hall, 2002.

PMSURVEY.ORG 2014 Edition. Project Management Institute. 2014.

RODRIGUES. Edson Junior Lobo. Curso de Engenharia de Software: Métodos e Processos para Garantir a Qualidade no Desenvolvimento de Software. São Paulo: Digerati Books, 2008.

REZENDE, D. Engenharia de Software e Sistemas de Informação. 3ª Ed., Rio de Janeiro: Brasport, 2005.

ROYCE, W. W. Managing the development of large software systems: concepts and techniques. In: IEEE COMPUTER SOCIETY PRESS. Proceedings of the 9th international conference on Software Engineering, 1987.

ROYCE, Winston. Managing the Development of Large Software Systems. Proceedings of IEEE WESCON 26, 1970.

SOUZA, Diogo Rodrigues. Implantação da Metodologia Ágil Scrum em um Ambiente de Desenvolvimento. Araranguá, dezembro de 2014.

SANTOS, Rodrigo. Estudo de caso sobre a utilização do extreme programming em uma pequena empresa de desenvolvimento para web. Santa Catarina: Universidade do Sul de Santa Catarina, 2013.

SCRUM ALLIANCE.2007. Disponível em: http://www.scrumalliance.org. Acesso em Setembro de 2018.

SBROCCO, José Henrique Teixeira de Carvalho; MACEDO, Paulo Cesar de. Metodologias ágeis: Engenharia de software sob medida. São Paulo: Érica, 2012.

STAVARENGO, Rafael. Desenvolvimento ágil. Revista Clube Delphi. Ano 9, 125ª ed. DevMedia, 2011.

SOARES, Michel dos Santos. Comparação entre Metodologias Ágeis e Tradicionais para o Desenvolvimento de Software. Unipac - Universidade Presidente Antônio Carlos Faculdade de Tecnologia e Ciências de Conselheiro Lafaiete. Gigante, MG. 2005.

SCHWABER, K.; BEEDLE, M. Agile Software Development with Scrum. Englewood Cliffs, NJ: Prentice Hall, 2001.

STAPLETON, J. DSDM Dynamic Systems Development Method. Harlow, Reino Unido: Addison-Wesley, 1997. DSDM: Business Focused Development, 2. ed. Harlow, Reino Unido: Pearson Education, 2003.

SANTIAGO, Alan Moitinho. Desenvolvimento de software: aplicabilidade de metodologias ágeis no mercado de software em vitória da conquista - ba. Vitória da conquista - ba 2012.

SANTOS, Elton Andrade. Escolha e implantação de uma metodologia de Desenvolvimento de software: Um estudo de caso para o laboratório de Aplicação em tecnologia da informação. Florianópolis, 2007.

SCHWABER, K. Agile Project Management with Scrum. Seattle: Microsoft Press, 2004.
SUTHERLAND, Jeff. Scrum: A arte de fazer o dobro do trabalho na metade do tempo. Traduzido por Natalie Gerhardt. São Paulo: Leya, 2014.

SCHWABER, Ken; SUTHERLAND, Jeff. Guia do Scrum. Um guia definitivo para o Scrum: As regras do jogo. Jul. 2013. Disponível em: <http://goo.gl/KbiaTS>. Acesso em junho, 2018.

SANDRINO, Keila M. Análise da aplicação de metodologias ágeis: um estudo de caso em empresa de desenvolvimento de software. 2014. 59f. Trabalho de Conclusão de Curso de Tecnologia em Análise e Desenvolvimento de Sistemas- Universidade Tecnológica Federal do Paraná. Ponta Grossa, 2014.

SILVA, D. E. S.; SOUZA, I. T.; CAMARGO, T. Metodologias Ágeis Para O Desenvolvimento De Software: Aplicação E O Uso Da Metodologia Scrum Em Contraste Ao Modelo Tradicional De Gerenciamento De Projetos. Revista Computação Aplicada, v.2, n.1, 2013, pp. 39-46.

SOMMERVILLE, I. (2007) Engenharia de Software. 8. ed. São Paulo: Pearson Addison Wesley.

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

SANTOS, Melquizedequi Cabral dos. O impacto do uso das metodologias ágeis Scrum e XP na satisfação dos stakeholders – Recife, 2014.

SOUZA, DIOGO RODRIGUES. Implantação da Metodologia Ágil Scrum em um Ambiente de Desenvolvimento. Araranguá, dezembro de 2014.

SILVA, Bruno Lopes Ribeiro. Qualidade de Software no Desenvolvimento Utilizando Metodologias Ágeis. Brasília, 2017.

SOUZA. Abordagens de testes: uma comparação entre as metodologias tradicional e ágil.2016.

SEMEDO, Maria João Moreno. Dissertação- Ganhos de produtividade e de sucesso de Metodologias Ágeis VS Metodologias em Cascata no desenvolvimento de projetos de software. Universidade Lusófona de Humanidades e Tecnologias ECATI. Lisboa, 2012.

TRECCANI, P. J. F.; SOUZA, C. R. B. (2010) Utilização de Metodologias Ágeis no Desenvolvimento de Software: Resultados de um Estudo Empírico. Universidade Federal do Pará. Disponível em: <http://www.lbd.dcc.ufmg.br/colecoes/eselaw/2010/005.pdf>. Acesso em: agosto de2018.

THE STANDISH GROUP. Chaos manifesto 2013.Disponivel em: http://versionone.com/assets/img/files/ChaosManifesto2013.pdf. Acesso em 09 nov. 2013.

THE STANDISH GROUP. Chaos manifesto 2013.Disponivel em: http://versionone.com/assets/img/files/ChaosManifesto2013.pdf. Acesso em 09 nov. 2015.

Tonon, Giulia. Metodologias Ágeis x Tradicionais- qual aplicar? Artigo 2015. Disponível:http://dtidigital.com.br/blog/metodologias-ageis-x-metodologias-tradicionais. Acessado em: 28.10.2018.

UTIDA, Kleber Hiroki. Metodologias Tradicionais e Ágeis: Análise Comparativa entre Rational Unified Process e Extreme Programming. 2012.

VAZQUEZ, Caros Eduardo; SIMÕES, Guilherme Siqueira. Engenharia de requisitos: software orientado ao negócio. 1. ed. Rio de Janeiro: Brasport, 2016.

VERSION ONE. 10th Annual State of agile Report. 2016. Disponível em: https://versionone.com/pdf/VersionOne-10th-Annual-State-of-Agile-Report.pdf. Acesso: 09 out. 2018

WILLIAMS, L. Agile software development methodologies and practices. Advances in Computers, v. 80, 2010.

XAVIER, Roni Lucas Francisco. Metodologias Tradicionais e Metodologias Ágeis: uma Análise Comparativa entre Rational Unified Process (RUP) e Scrum (Framework Estruturado).2017

YOSHIMA, Rodrigo. O mito da Cultura Ágil. 03 abr. 2011. Disponível em: <http://blog.aspercom.com.br/2018/04/03/o-mito-da-cultura-agil>. Acesso em: 30 mai. 2018.

ZENARO, Flávia dos Santos. A utilização do Scrum em um sistema web: um estudo de caso. jul. 2012.