UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO DE JOINVILLE
CURSO DE ENGENHARIA MECATRÔNICA

KARIANI MONTEIRO DIAS

AVALIAÇÃO DE METODOLOGIAS PARA GESTÃO DE PROJETOS DE ELETRÔNICOS

Joinville
2018

KARIANI MONTEIRO DIAS

AVALIAÇÃO DE METODOLOGIAS PARA GESTÃO DE PROJETOS DE ELETRÔNICOS

Trabalho de Conclusão de Curso apresentado como requisito parcial para obtenção do título de Bacharel em Engenharia Mecatrônica no curso de Engenharia Mecatrônica, da Universidade Federal de Santa Catarina, Centro Tecnológico de Joinville .

Orientadora: Dra. Janaina Renata Garcia

Joinville

2018

KARIANI MONTEIRO DIAS

AVALIAÇÃO DE METODOLOGIAS PARA GESTÃO DE PROJETOS DE ELETRÔNICOS

Este Trabalho de Conclusão de Curso foi julgado adequado para obtenção do título de Bacharel em Engenharia Mecatrônica na Universidade Federal de Santa Catarina, Centro Tecnológico de Joinville.

Joinville, 27 de novembro de 2018.

_____

Dr. Diego Santos Greff
Coordenador do Curso

Banca Examinadora:

_____

Dra. Janaina Renata Garcia
Orientadora
Presidente

_____

Dr. Carlos Mauricio Sacchelli
Membro
Universidade Federal de Santa Catarina

_____

Dr. Diego Santos Greff
Membro
Universidade Federal de Santa Catarina

Este trabalho é dedicado aos meus pais.

AGRADECIMENTOS

Agradeço a Deus pela força em todos os momentos dessa jornada, me iluminando e guiando pelo caminho correto. Por me prover companhia, mesmo quando eu pensei que estava sozinha e por me proporcionar momentos de felicidade independente das circunstâncias.

Aos meus pais, Pedro e Nilda, por serem o exemplo mais forte de união na minha vida, pelo apoio em todas as situações e decisões e por acreditarem em mim em todos os momentos. Por me amarem incondicionalmente mesmo quando estou errada.

Às minhas irmãs, Karina e Karol, por estarem sempre ao meu lado, por me ajudarem, me ouvirem e me motivarem a melhorar sempre.

A minha avó, Sirlei, pelos conselhos incríveis nos momentos certos e por mostrar sempre que se orgulha e apoia o meu crescimento. Por todas as orações e por ser uma matriarca incrível para toda a família Monteiro.

À professora Janaina Renata Garcia, minha orientadora, por ser uma mulher forte, inspiradora e uma educadora incrível! Por me ajudar no processo de construção desse trabalho, me provendo todo o suporte necessário.

Aos meus professores das instituições de ensino por onde passei, pois me auxiliaram a chegar até aqui. Especialmente aos meus professores da graduação, por me ajudarem na formação acadêmica e profissional e por dedicarem o seu tempo a ensinar assuntos de complexidade elevada.

Aos professores Diego Greff e Carlos Sacchelli por disponibilizarem parte do seu tempo e aceitarem ser parte da banca avaliadora.

A todos os meus amigos, dádivas que Deus me deu, que de alguma forma contribuíram para a realização do meu sonho. Por me entenderem nos momentos que nem eu conseguia, por me ajudarem quando eu achava que não tinha solução e por me proporcionarem momentos de alegria que serão sempre lembrados com carinho e amor.

Essa conquista só foi possível pela presença de cada um de vocês! Obrigada!

Tudo o que temos de decidir é o que fazer com o tempo que nos é dado.

– J.R.R. Tolkien

RESUMO

Os sistemas eletrônicos estão cada vez mais integrados aos diferentes produtos usados no cotidiano. A velocidade com que surgem novas tecnologias evidencia a necessidade de olhar atenciosamente para a maneira como os projetos são desenvolvidos. A gestão de projetos é aplicada, então, para facilitar esse processo e garantir entregas ótimas que satisfaçam o cliente em suas necessidades e sejam realizadas no menor tempo. Desta forma, os objetivos deste estudo são conhecer as particularidades de projetos de sistemas eletrônicos, identificar e apresentar as metodologias tradicionais e ágeis mais divulgadas em gerenciamento de projetos, compará-las quanto às particularidades de sistemas eletrônicos e avaliar a mais adequada para o desenvolvimento da maior parte desses projetos. A metodologia de pesquisa utilizada para o desenvolvimento desse trabalho foi dividida em duas etapas, a primeira pode ser classificada como qualitativa de natureza aplicada. A segunda pode ser considerada quantitativa, pois contou com técnicas bibliométricas para identificar quantitativamente as metodologias de gestão de projetos mais citadas na literatura, obtendo como resultado as metodologias PMBOK, PRINCE2, Modelo Cascata, Extreme Programming (XP) e Scrum, que foram abordadas nesse trabalho. Uma análise comparativa entre as metodologias abordadas foi proposta e a metodologia Scrum foi indicada como mais adequada para a maioria dos projetos de eletrônicos.

Palavras-chave: Gestão de projetos. Sistemas eletrônicos. Scrum.

ABSTRACT

Electronic systems are increasingly integrated into the different products used in daily life. The speed, which new technologies emerge, highlights the need to look closely at how projects are developed. Project management is applied to facilitate this process and ensure optimal deliveries that satisfy the customer in their needs and are carried out in the shortest time. Therefore, the objectives of this study are to know the particularities of electronic systems projects, identify and present traditional and agile methodologies in project management, compare them with regard to the particularities of electronic systems and to evaluate the most appropriate for the development of most of electronics projects. The research methodology used for the development of this work was divided in two stages; the first one can be classified as qualitative of applied nature. The PMBOK, PRINCE2, Waterfall, Extreme Programming (XP) and Scrum methodologies, which were addressed in this paper, were quantified using bibliometric techniques to quantitatively identify the most cited project management methodologies in the literature job in the second stage. A comparative analysis between the methodologies was proposed and the Scrum model was indicated as more suitable for most electronic projects.

Key-words: Project management. Electronics systems. Scrum.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

ANSI – *American National Standards Institute*

DT – *Development team*

EAP – Estrutura Analítica do Projeto

PMBOK – *Project Management Body of Knowledge*

PMI – *Project Management Institute*

PO – *Product Owner*

PRINCE2 – *Projects in Controlled Enviroments*

SM – *Scrum Master*

SoS – *Scrum of Scrums*

SXBOK – *Software extension to PMBOK*

TI – Tecnologia da informação

XP – *Extreme Programming*

SUMÁRIO

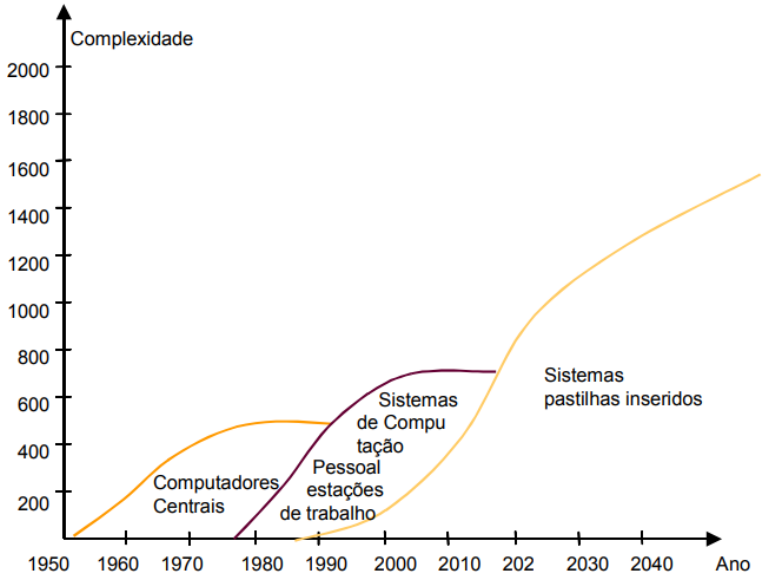
1 INTRODUÇÃO

A necessidade de sistemas eletrônicos surgiu nos Estados Unidos, com a preocupação do tempo gasto para tabular o censo. Em 1890, surgiu a primeira máquina acionada por energia elétrica e controlada por cartões perfurados, que permitiu uma redução de quatro anos na tabulação dos dados do censo. O ápice das máquinas eletromecânicas aconteceu nos anos 1940, com a construção de uma máquina com 800 km de fios e 3 milhões de conexões, essa máquina realizava multiplicações em 6 segundos e divisões em 12 segundos. Em 1904, surgiu o diodo e em 1906, uma melhoria significativa nesse dispositivo levou ao surgimento do tríodo (ZUFFO et al, 2018).

A segunda Guerra mundial e as demandas militares para realizar cálculos balísticos precisos, a necessidade do radar e da criptografia impulsionaram o desenvolvimento de sistemas eletrônicos analógicos e digitais. Foi nessa época que surgiu o ENIAC (*Eletronic Numerical Integrator and Calculator*), o primeiro computador eletrônico de uso geral. Em 1947 surgiu o transistor, em substituição às válvulas eletrônicas que ocupavam muito espaço e gastavam muita energia. Na sequência surgiram as placas de circuito impresso, para suportar e interconectar os componentes eletrônicos de qualquer circuito. Um problema era o tamanho extenso dessas placas, que podiam alcançar 1m², o que fazia com que os sistemas ficassem caros, grandes e de difícil manutenção. Levando ao surgimento dos Circuitos Integrados (CIs) em 1958 (ZUFFO et al, 2018).

O principal advento com o lançamento dos CIs foi o desenvolvimento de microprocessadores, em 1971, que deram impulso à informática. A Figura 1 representa graficamente a evolução da complexidade dos sistemas eletrônicos no decorrer do tempo. No gráfico, é possível observar a velocidade com que aumentam a quantidade e a complexidade dos sistemas eletrônicos, o que indica fortemente a necessidade de observação da gestão dos projetos desses sistemas.

Figura 1 Tendência da complexidade dos sistemas eletrônicos no decorrer dos anos.



Fonte: Zuffo et al, 2018.

Os sistemas eletrônicos estão cada vez mais integrados aos diferentes produtos utilizados no dia a dia. Bens de consumo, eletrodomésticos, produtos da indústria são apenas alguns exemplos que incorporam cada vez mais a eletrônica. Esse crescimento significativo exige que lançamentos de novas tecnologias sejam cada vez mais rápidos e impactantes. É vital que este setor incorpore uma estrutura incisiva voltada para o gerenciamento de projetos, garantindo assim o sucesso dos produtos e serviços apresentados ao mercado (BUZZETTO, 2008).

O Project Management Institute (PMI) indica que durante o século XIX as técnicas de gerenciamento de projetos ainda eram bastante primitivas, surgindo então a figura do supervisor de projetos que sabia ler e fazer contas (MARCONDES, 2017). No século XX, com a necessidade de maximizar as produções sem utilizar horas extras de serviço ou mais pessoas, surgiu o raciocínio científico, que basicamente quebrou os elementos de um processo em tarefas. Também no século XX, Henry Gantt (1861-1919) criou a técnica de sequenciar e definir a duração das tarefas, seu diagrama permitiu verificar as fases de um projeto (BERNARDO apud MARCONDES, 2017).

Na terceira revolução industrial, já com a utilização dos computadores e nas décadas após a segunda guerra mundial, foram criados novos elementos de gerenciamento de projetos, os gráficos *Program Evalluation and Review Technique* (PERT) e o método do caminho crítico (CPM). Na década de 1960, o gerenciamento de projetos foi oficializado como ciência. Nessa época, as empresas começaram a ver os benefícios da organização com os conhecimentos de

projetos (MARCONDES, 2017). Em 1969 surgiu o *Project Management Institute* (PMI), formado por um grupo de profissionais unidos para discutir e compilar as melhores práticas de gerenciamento dos projetos esse grupo cresceu e se tornou o maior instituto sem fins lucrativos no campo da gestão de projetos.

O gerenciamento de projetos pode ser definido como a aplicação de conhecimentos, habilidades e técnicas para a execução de projetos de forma efetiva e eficaz (PMI, 2013a). O PMI (2013a), através do guia Project Management Body Of Knowledge – PMBOK, completa ainda que esse gerenciamento é uma competência estratégica para as organizações, permitindo que os resultados dos projetos sejam ligados aos objetivos do negócio, garantindo a sustentação no mercado. Para Vargas (2005), a proposta do gerenciamento de projetos estabelece um processo estruturado e lógico para trabalhar com eventos novos e complexos no aspecto de desenvolvimento de produtos e serviços.

Considerando a importância de garantir o sucesso dos projetos no setor eletrônico, é importante que sejam buscados métodos de gestão de projetos que se adequem à área, de forma que sejam consideradas as particularidades e requisitos empregados. Assim, esse trabalho procura apontar uma metodologia adequada para gestão de projetos de eletrônicos. Para isso, foram buscadas metodologias de gestão tradicionais e ágeis e comparadas entre si considerando as particularidades de sistemas eletrônicos.

A metodologia de desenvolvimento desse trabalho envolve identificar metodologias usuais para gestão de projetos, comparar com elementos essenciais de projetos na área de eletrônica e identificar qual a metodologia mais adequada para esse campo. A abordagem da pesquisa será qualitativa, uma vez que analisa os dados através de comparações subjetivas, sujeitas a considerações do autor, usando quatro bases de dados diferentes Repositório UFSC, Catálogo de Teses Capes, Scopus e IEEE. Para definir as principais metodologias de gestão de projetos será aplicada a bibliometria em duas das bases de dados utilizadas, Scopus e IEEE.

1.1 OBJETIVOS

1.1.1 Objetivo Geral

Avaliar a metodologia de gestão de projetos mais adequada para gestão de projetos eletrônicos.

1.1.2 Objetivos Específicos

- Conhecer as particularidades de projetos de eletrônicos;
- Comparar metodologias usuais de gestão de projetos;
- Comparar as metodologias aplicadas às particularidades dos projetos de eletrônicos.

1.2 JUSTIFICATIVA

Os projetos de sistemas eletrônicos apresentam particularidades que não estão presentes em outros produtos. As principais delas são o ciclo de vida dinâmico e as mudanças de requisitos que podem acontecer durante o desenvolvimento. Muitas empresas criam o seu próprio modo de desenvolver software e/ou hardware para absorver essas particularidades e entregar o melhor resultado aos clientes. Outras empresas não utilizam metodologias específicas para a gestão do desenvolvimento, apenas aplicam técnicas isoladas de controle que auxiliam no processo como um todo (LAYTON and OSTERMILLER, 2017).

Devido a erros possíveis na criação de uma metodologia nova ou a inadequação dos projetos às metodologias existentes ou aos métodos adotados para a gestão dos projetos, muitas empresas acabam por falhar no desenvolvimento de sistemas eletrônicos. Para reduzir as falhas, esse trabalho busca entender as particularidades de sistemas eletrônicos e avaliar uma metodologia estruturada, conhecida e mais adequada para a aplicação em projetos de natureza dinâmica e características singulares como as de sistemas eletrônicos.

É de extrema importância que sejam desenvolvidos estudos com tema de gestão de projetos para facilitar seu gerenciamento, manter equipes motivadas, reter conhecimentos sobre o tema e principalmente permitir às indústrias a realização de entregas mais precisas e convergentes às necessidades dos clientes.

1.3 ESTRUTURA DO TRABALHO

Esse trabalho é composto por cinco capítulos principais. No primeiro deles, o tema é introduzido, expondo os objetivos do presente estudo, justificando e mostrando as motivações de desenvolvimento desse estudo.

O segundo capítulo, indica a metodologia de pesquisa utilizada para a captação dos materiais nos quais o trabalho é baseado, indicando os passos principais de busca e construção

do portfólio bibliográfico. Esse capítulo destaca as razões pelas quais foram consideradas as metodologias de gestão como as principais, através da aplicação de bibliometria.

O terceiro capítulo apresenta a fundamentação teórica desse trabalho, formado pelo portfólio bibliográfico e complementado por referências externas relevantes para o tema. Esse capítulo divide-se em duas partes principais: metodologias tradicionais e metodologias ágeis. Buscando-se assim identificar e mostrar as principais metodologias abordadas na literatura e as particularidades observadas no desenvolvimento de sistemas eletrônicos.

O quarto capítulo é formado pela comparação entre diferentes metodologias, buscando evidenciar os benefícios e pontos de melhoria de cada uma e, finalmente, apresentar uma proposição da metodologia abordada que mais se adeque aos projetos de sistemas eletrônicos.

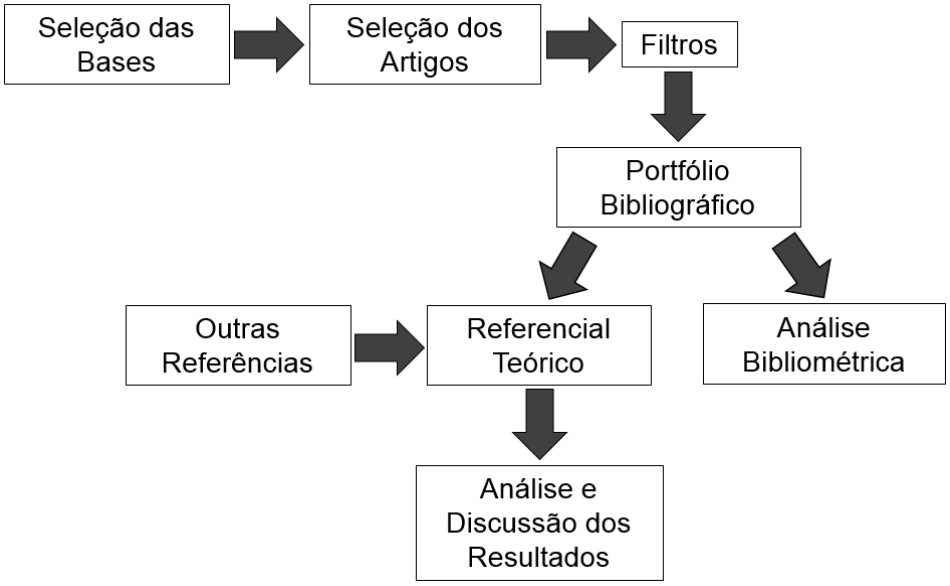
Finalmente, o quinto capítulo mostra as principais considerações feitas durante o desenvolvimento desse estudo e propõe estudos futuros relacionados a esse trabalho. Em seguida, são apresentadas as referências utilizadas para compor esse estudo.

2 METODOLOGIA DE PESQUISA

Para que um trabalho possa ser reconhecido como científico, precisa ser lógico, sistemático, coerente e bem argumentado. Isso o diferencia de outros conhecimentos como senso comum, sabedoria ou ideologia. De acordo com Silveira e Córdova (2009), a pesquisa científica é resultado de um exame minucioso realizado com o objetivo de resolver um problema. Os autores referem-se à pesquisa como sendo o procedimento sistemático e intensivo para descobrir e interpretar os fatos inseridos em determinada realidade. Assim, é feita uma fundamentação teórica sólida e bem estruturada.

Assim, a metodologia proposta por Vasconcelos (2014) foi utilizada para o desenvolvimento desse trabalho, por detalhar todas as etapas de busca e análise. A Figura 2, abaixo, apresenta o fluxograma de seleção e análise dos materiais utilizados.

Figura 2 Fluxograma da seleção e análise do referencial teórico.



Fonte: adaptado de Vasconcelos (2014, p. 12).

Conforme apresentado na Figura 2, quatro etapas foram utilizadas para o desenvolvimento do portfólio bibliográfico: seleção das bases, seleção dos estudos nas bases, filtros e pré-análise dos materiais selecionados, as quais são detalhadas na subseção a seguir. Após a definição do portfólio bibliográfico, outros dados relevantes foram utilizados para complementar a fundamentação teórica. Essas referências foram selecionadas considerando a

vasta importância das mesmas para o conhecimento do tema abordado, de forma que, sem essa complementação, o trabalho ficaria com lacunas ou incompleto.

2.1 SELEÇÃO DAS BASES

A disponibilização de estudos científicos online facilita o processo de busca e seleção de materiais sobre diferentes áreas do conhecimento (DELLA BRUNA JUNIOR; ENSSLIN, L.; ENSSLIN, S. R., 2012). Para Campbell et al (2010), as bases de dados online se configuram como a principal forma de difusão dos resultados científicos. Costa (2007) complementa essa visão afirmando que isso permite o acesso a um grande número de pesquisadores de qualquer parte do planeta rapidamente.

Apesar do grande número de bases de dado disponíveis, esse trabalho limitou-se a quatro bases: *Repositório Institucional UFSC*, *Catálogo de Teses da Capes*, *IEEE* e *Scopus*. Essas bases de dados foram escolhidas devido aos seguintes critérios:

a. Disponibilidade de acesso gratuito livre ou utilizando a rede da universidade;
b. Alinhamento com o tema de estudo do trabalho e muitos estudos relacionados a engenharia;
c. Acesso livre aos textos completos;
d. Bases de dados recomendadas pela Biblioteca UFSC;
e. Possibilidade de desenvolver o estudo bibliométrico com duas delas IEEE e Scopus.

Entende-se que utilizando essas bases de dados é possível desenvolver um estudo com diferentes domínios e acervo variado, sem sobrecarregar os dados analisados com informações de difícil tratamento e quantidade exaustiva de materiais.

2.2 SELEÇÃO DOS ESTUDOS NAS BASES

Para selecionar os materiais relevantes dentro das bases de dados, foram realizadas pesquisas em duas etapas bem definidas. Na primeira etapa, foram utilizados o Repositório Institucional UFSC e o Catálogo de Teses da Capes e as buscas se restringiram aos termos chave *gestão de projetos de software* e *gestão de projetos de hardware*, que poderiam aparecer em qualquer parte do trabalho. Essa pesquisa localizou uma quantidade muito extensa de títulos,

aproximadamente dois milhões de trabalhos (considerando títulos repetidos), o que evidenciou a necessidade de filtragem dos dados encontrados.

A segunda etapa da pesquisa foi realizada nas bases de dados da Scopus e IEEE e buscou os termos *project AND management AND methodology* ou o termo *project management* no título, no resumo ou nas palavras chaves. Essa busca teve a intenção de captar dados relacionados a gestão de projetos independentes da área de aplicação, para conhecer as metodologias mais citadas através da análise bibliométrica.

2.3 FILTROS DE PESQUISA

Os títulos encontrados na primeira etapa de pesquisa passaram por uma filtragem, da qual foram utilizados somente trabalhos de gerenciamento, sendo excluídos estudos referentes a métodos de desenvolvimento de software e hardware, ou seja, foram excluídos trabalhos que detalhassem o desenvolvimento de sistemas específicos e não consideravam a gestão desse processo. Outros filtros aplicados foram a data – somente foram considerados trabalhos realizados entre 2016 e 2018 e a área de conhecimento – somente foram considerados trabalhos de engenharia.

Para a segunda etapa, somente filtros relacionados a data foram aplicados, restringindo os trabalhos realizados entre 2016 e 2018. Com esse filtro, foram encontrados 2402 (dois mil quatrocentos e dois) trabalhos na IEEE e 805 trabalhos na Scopus – títulos não duplicados.

2.4 ANÁLISE BIBLIOMÉTRICA

Como parte da composição do portfólio bibliográfico para a construção do referencial teórico, está a análise bibliométrica dos materiais considerados relevantes para essa etapa após a aplicação de filtros. Essa análise objetiva evidenciar os principais aspectos sobre o portfólio bibliográfico, quantificando suas características (LACERDA; ENSSLIN, L.; ENSSLIN, S. R., 2012).

A bibliometria pode ser definida como uma técnica qualitativa e estatística das produções e da disseminação do conhecimento científico. Essa técnica permite organizar grandes massas de dados, auxiliando a criar resultados valiosos provenientes da análise (ARAUJO, 2006; YOSHIDA, 2010).

A intenção principal da análise bibliométrica desenvolvida nesse estudo foi a concepção dos principais modelos aplicados na gestão de projetos em quaisquer setores. Afim

de compreender a influência de cada metodologia nos desenvolvimentos de projetos, os trabalhos filtrados foram separados em diferentes grupos, observando-se o título, resumo e palavras-chave, sendo possível inferir que:

a. Cento e cinco (105) trabalhos citaram PMBOK;
b. Cento e dezessete (117) estudos citaram SCRUM;
c. Duzentos e sessenta e três (263) citaram metodologias ágeis;
d. Dezesseis (16) citaram o método PRINCE2;
e. Trinta e um (31) citaram o modelo XP; e
f. Trinta e um (31) citaram o Modelo Cascata.

O restante dos modelos não chegava a ter número significativo de citações. Assim, definiu-se que o referencial teórico seria fundamentado nas metodologias supracitadas.

2.5 CLASSIFICAÇÃO DA PESQUISA REALIZADA

A primeira etapa da pesquisa realizada pode ser classificada quanto a abordagem como qualitativa de natureza aplicada, uma vez que as análises desenvolvidas não se preocupam com a representatividade numérica dos dados, mas sim com o aprofundamento da compreensão do assunto e com a solução de um problema específico. Nessa forma de abordagem, o autor não quantifica os valores nem se submete à prova de fatos, pois os dados analisados são não-métricos e se valem de diferentes abordagens, envolvendo interesses locais (SILVEIRA; CÓRDOVA, 2009).

A segunda etapa da pesquisa pode ser classificada como quantitativa e qualitativa, pois preocupa-se com a quantidade de referências sobre os assuntos para a definição de quais metodologias serão abordadas. A característica qualitativa fica evidenciada no aprofundamento das metodologias determinadas pela bibliometria.

A classificação quanto aos objetivos pode ser definida como uma pesquisa exploratória explicativa, já que sua intenção é compreender o desenvolvimento de software e hardware com suas especificidades e propor uma metodologia de gestão através de métodos existentes descritos na literatura. Esse estudo é puramente bibliográfico, levantando dados de outras referências teóricas já analisadas para propor uma solução observando diversas posições sobre gerenciamento de projetos conforme descrito na literatura.

3 FUNDAMENTAÇÃO TEÓRICA

O dicionário de Cambridge (COLIN, 2013) define hardware como “as partes físicas e eletrônicas de um computador”, para conseguirmos o conceito aqui utilizado, é importante definir também computador. De acordo com a mesma fonte, computador é um dispositivo eletrônico que pode armazenar grandes quantidades de informação e realizar ações sobre elas de acordo com instruções definidas, esse dispositivo pode ainda controlar outros dispositivos.

A definição de software é relacionada às instruções descritas em um computador, pelo dicionário de Cambridge (COLIN, 2013), pode-se afirmar que “software é a instrução que controla o que um computador faz; programas de computadores” ou ainda “programas inseridos em um computador para fazer trabalhos específicos”.

A junção de software e hardware com propósito específico pode ser considerada sistema eletrônico ou eletrônico. Os sistemas eletrônicos podem ou não ser embarcados, sendo considerados embarcados quando possuem uma função específica dentro de um sistema maior e não são vistos nem programados pelo usuário (SANTOS, 2002). Os projetos de eletrônicos englobam toda a construção de software e/ou hardware, o que faz com que o desenvolvimento esteja sujeito a inúmeros imprevistos, que podem levar a atrasos e falhas.

Drechsler e Breiter (2007) consideram que as etapas de desenvolvimento de hardware são muito semelhantes às de software. Inicialmente tem-se uma ideia, para a qual são descritas especificações, em seguida, tudo isso é formalizado para ser simulado e, finalmente, são desenhados os layouts e os itens são produzidos. Outras semelhanças podem ser encontradas, como a linguagem de escrita e a boa prática de dividir um sistema complexo em partes menores para simplificar o processo de desenvolvimento. Os autores consideram ainda que os mesmos modelos de gerenciamento de fluxo de projetos aplicados a software podem ser aplicados a hardware, pois as características dominantes nos dois desenvolvimentos são bastante similares.

É essencial definir as etapas envolvidas nos processos de projeto de software e hardware. Para Sommerville (2004 apud LUDVIG; REINERT, 2007), quatro atividades são comuns a todos os projetos de software:

1. Especificação: definição das atribuições, requisitos e restrições do sistema;
2. Desenvolvimento: projeto e implementação do sistema de acordo com as especificações da primeira etapa. Essa fase inclui também a documentação do projeto e conhecimentos adquiridos;

3. Validação: essa etapa é necessária para garantir as funcionalidades da aplicação e o atendimento das especificações;
4. Evolução: atualizações e evoluções da solução final de acordo com as necessidades do cliente.

Por outro lado, observando as atividades envolvidas nos projetos de hardware, seis fases podem ser destacadas (DRECHSLER E BREITER, 2007):

1. Inicialização: definição das restrições, requisitos e pessoas envolvidas no projeto;
2. Oferta/Contrato: baseado nas definições da primeira etapa, o contrato deve descrever as entregas e detalhar o máximo possível o projeto;
3. Design: é a fase responsável pela construção e desenvolvimento do hardware, deve-se concentrar nas principais características definidas nas fases anteriores e verificar características adicionais que podem ser acrescentadas ao sistema. Importante considerar documentação para reuso de conhecimento nessa fase;
4. Protótipo: essa fase é mais aplicável a projetos extensos, dependendo do tipo de hardware, um protótipo pode ser construído para em etapas futuras transformar-se no projeto completo;
5. Manutenção: para hardware, idealmente não há mudanças depois que o sistema está em uso, mas os requisitos podem mudar. Atualmente, os dispositivos de hardware têm vida de aproximadamente dois a três anos e pequenas alterações/melhorias, costumam ser o suficiente para atender a novas demandas;
6. Reuso: muitos componentes e conhecimentos podem ser reutilizados durante muitos anos, aplicar técnicas de reuso pode reduzir muito o tempo de projeto, além de reduzir custos.

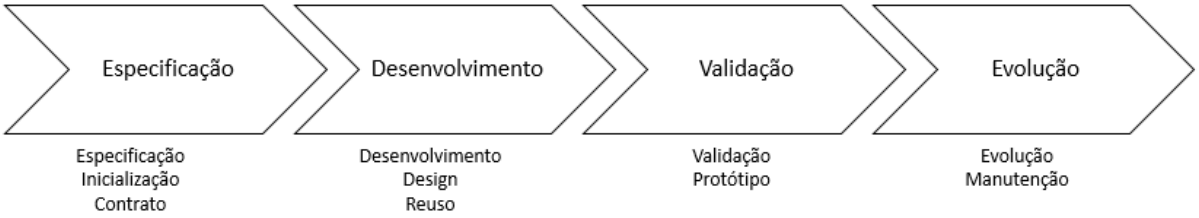
Fazendo uma analogia entre os passos de software e hardware, as quatro etapas definidas em software são suficientes para englobar as etapas de hardware da seguinte maneira:

1. A etapa de Especificação englobará as etapas de inicialização e oferta/contrato de hardware;
2. A etapa de desenvolvimento será considerada análoga à etapa de design e incluirá também a fase de reuso de hardware, que é uma fase específica para captação de conhecimentos e em software isso é parte do desenvolvimento de software;

3. A etapa de protótipo de hardware será considerada parte da validação em software;
4. A etapa de evolução será unida à etapa de manutenção de hardware.

Considerando essas etapas comuns de projeto e a semelhança entre desenvolvimento de software e hardware, as metodologias de gestão de projetos de software podem também ser implementadas aos projetos de hardware e, serão nesse trabalho assim consideradas. Ainda considerando as semelhanças entre os desenvolvimentos de software e hardware, todas as considerações feitas sobre os desenvolvimentos de software poderão ser estendidas para hardware. A Figura 3, mostra um diagrama com as fases dos projetos de eletrônicos:

Figura 3 Diagrama das etapas de desenvolvimento de sistemas eletrônicos.



Fonte: A autora.

É importante salientar projetos eletrônicos são diferentes entre si, em função de tamanho, importância e criticidade para os negócios. Com base nessas diferenças muitas empresas aplicam técnicas diferenciadas para cada desenvolvimento, sendo necessário avaliar a forma mais adequada para atender ao maior número de desenvolvimentos possível (COCKBURN, 2000 apud CORDEIRO, 2014).

3.1 PARTICULARIDADES DE PROJETOS DE ELETRÔNICOS

Existem muitos fatores que fazem com que o desenvolvimento de software seja especial e desafiador. O PMI (2013, p. 6), apresenta na Extensão de Software para o PMBOK (SXBOK), considera que os principais desafios são:

- Software é um produto intangível e maleável; o código fonte para software é escrito em texto. Na maioria dos casos, os times de desenvolvimento de software geram e revisam documentos compartilhados (por exemplo: requisitos, especificações, código e planos de teste). O desenvolvimento de software é também, por vezes, caracterizado

como um processo de aprendizagem, no qual, informações são geradas durante o projeto;

- Os atributos chaves que fazem o processo de software desafiador são a complexidade do projeto e do produto, o dimensionamento não linear dos recursos, das medições do projeto e do produto, as incertezas iniciais no escopo e o conhecimento envolvido no desenvolvimento do projeto;
- Os requisitos de software podem apresentar mudanças durante o projeto, conforme o conhecimento do escopo é aumentado e o produto emerge;
- Os requisitos para novos ou alterações de softwares sofrem influência dos processos de negócios da organização e dos fluxos de processos dos colaboradores;
- O capital intelectual da equipe é o principal ativo para projetos e organizações de desenvolvimento de software, por ser um produto direto dos processos cognitivos humanos;
- A comunicação e a coordenação dentro das equipes de software e com os participantes do projeto muitas vezes carecem de clareza. Muitas das ferramentas e técnicas usadas na engenharia de software destinam-se a melhorar a comunicação e coordenação;
- A criação de software requer uma solução inovadora de problemas para criar soluções exclusivas. A maioria dos projetos de software desenvolve produtos exclusivos porque a replicação de software existente é um processo simples, em comparação com a replicação de artefatos físicos. Os projetos de software são mais parecidos com projetos de pesquisa e desenvolvimento do que com projetos de construção ou manufatura;
- Os projetos de software envolvem risco e incerteza, porque exigem inovação, o produto é intangível e as partes interessadas podem não articular ou concordar efetivamente com as necessidades a serem satisfeitas pelo produto de software;
- O planejamento e a estimativa básicos para projetos de software são desafiadores porque essas atividades dependem de requisitos, que geralmente são imprecisos ou fazem parte de dados históricos que geralmente estão faltando ou são inaplicáveis. A preparação de estimativas precisas também é um desafio, porque a eficiência e eficácia dos desenvolvedores de software são amplamente variáveis;
- A complexidade do produto torna o desenvolvimento e a modificação de software desafiadores devido ao enorme número de caminhos lógicos dentro dos módulos do

programa combinados com valores de dados que exercitam os caminhos e as combinações de detalhes da interface entre os módulos do programa;

- Testes exaustivos de software são impraticáveis devido ao tempo que seria necessário para testar todos os caminhos e interfaces lógicos em todas as combinações de dados de entrada e outros estímulos de entrada;
- Desenvolvimento de software envolve frequentemente a inclusão de diferentes produtos de fornecedores e o desenvolvimento de interfaces para outro software; isso pode resultar em problemas de integração e desempenho;
- Como a maioria dos softwares é interconectada, técnicas de segurança da informação são necessárias. A segurança de software é um desafio grande e crescente;
- Quantificação objetiva e medição da qualidade de software são difíceis por causa da natureza intangível do produto. Devido à natureza intangível do produto, não é possível realizar medições pelas unidades de medida tradicionais;
- Desenvolvedores de software usam processos, métodos e ferramentas que estão em constante evolução e são atualizados com frequência;
- O software geralmente é o elemento de um sistema que é alterado quando atributos de funcionalidade, comportamento ou qualidade devem ser alterados;
- Um produto de software pode precisar operar em uma variedade de plataformas de hardware e de infraestrutura de software;
- O software executável não é um produto autônomo. Ele é executado em hardware de computação e geralmente é um elemento de um sistema que consiste em diversos hardwares, outros softwares e procedimentos manuais;
- As tecnologias de plataforma, as infraestruturas de software e o software fornecido pelo fornecedor são frequentemente alterados ou atualizados, o que pode exigir alterações no software que está sendo desenvolvido.

Alguns requisitos do software são normalmente padronizados pelas definições do hardware no qual será implementado, como capacidade de processamento, memória disponível, e largura de banda de comunicação (PMI, 2013). Em comparação com hardware, software é mais fácil e rápido de alterar os requisitos de acordo com as necessidades do cliente ou com fatores do ambiente de desenvolvimento.

3.2 METODOLOGIAS DE GESTÃO DE PROJETO

As metodologias de gestão de projetos podem ser basicamente divididas em dois grupos diferentes, metodologias tradicionais e metodologias ágeis. As metodologias tradicionais ou clássicas, de acordo com Cordeiro (2014), possuem similaridades com as estruturas organizacionais Tayloristas, com alocação de profissionais especializados e racionalização das tarefas. Já as metodologias ágeis, para o mesmo autor, aceitam a imprevisibilidade, não se apoiam em processos e tem foco no potencial dos indivíduos envolvidos no projeto para levar ao sucesso. Aqui serão abordadas metodologias gerais de desenvolvimento gestão de projetos e metodologias direcionadas para projetos de software/hardware em ambos os grupos – tradicionais e ágeis.

3.2.1 Metodologias tradicionais

As metodologias gerais mais difundidas são disponibilizadas por institutos direcionados especificamente para estudos de projetos (BARACAT, 2016). Os métodos para gestão de projetos gerais mais amplamente utilizados podem ser vistos no Quadro 1.

Quadro 1 Métodos de gestão de projetos e suas principais características.

Conjunto de Métodos	Características
PMBoK – Project Management Body of Knowledge	Conjunto de métodos bastante genérico. Estruturado por áreas de conhecimento de um projeto. Complementado por gerenciamento de programa e gerenciamento de portfólio.
PRINCE2 – Projects In Controlled Environments	O documento tem um enfoque muito grande nos aspectos contratuais de um projeto.
Waterfall ou Cascata e suas variantes	Define fases consecutivas que só devem se iniciar após a finalização da anterior. Permite clareza de entregas entre etapas.

Fonte: Adaptado de Patah, Patah e Carvalho (2012); Baracat (2016).

As abordagens de desenvolvimento tradicionais começaram a ser introduzidas como metodologias de desenvolvimento de software em 1970 e são centradas em processos e na crença de que as fontes de variações são identificáveis e eliminéveis através de refino de processos (CORDEIRO, 2014). Essas metodologias são conhecidas como pesadas ou orientadas a documentação, de acordo com Ludvig e Reinert (2007), uma característica marcante dessas metodologias é a divisão do processo em etapas bem definidas. Alguns dos principais modelos direcionados a software são: Modelo Cascata, modelo de prototipação e modelo de desenvolvimento iterativo. Os modelos de prototipação e desenvolvimento iterativo

são considerados, neste trabalho, variantes do Modelo Cascata, dada a similaridade entre as metodologias.

As metodologias de gestão relacionadas a desenvolvimento específico de hardware são pouco encontradas na literatura devido à semelhança com o desenvolvimento de software, conforme explicado anteriormente. Aqui são abordadas metodologias para gestão de projetos de produtos gerais e metodologias para gestão de software, as quais podem ser aplicadas a hardware.

3.2.1.1 PMBOK

Para Baracat (2016), o PMBOK não deve ser considerado uma metodologia, pois o guia indica as melhores práticas de desenvolvimento de projeto sem apontar uma sequência na qual os processos devem ser realizados. Nos EUA, o PMBOK é reconhecido como um padrão desde 1999 pelo American National Standards Institute (ANSI) (GONÇALVES, 2016). O objetivo do guia é identificar um conjunto de conhecimentos sobre gerência de projetos, não fornecendo ações e formas de realização para a execução dos projetos (PROJECT MANAGEMENT INSTITUTE - PMI apud GONÇALVES, 2016). Moraes (2016), entretanto, afirma que essa metodologia é a mais conhecida e utilizada para gestão de projetos.

No PMBOK, os projetos são divididos em fases, conhecido também como ciclo de vida do projeto. Os ciclos de vida ainda são divididos em fases menores, que ligam o início e o fim do projeto. As transições entre as fases normalmente envolvem entregas ou transferência técnica. Os ciclos de vida, normalmente, representam os trabalhos, os processos e envolvidos em cada fase (PMI apud GONÇALVES, 2016).

O PMI tem como elementos recorrentes, cinco grupos de processos:

1. Grupo de processos de Iniciação: Relacionados a definição e autorização do projeto ou de uma fase;
2. Grupo de processos de Planejamento: Definem os objetivos, planejamento das ações necessárias e escopo para o projeto;
3. Grupo de processos de Execução: Caracterizados pela integração dos recursos do projeto para realizar o plano de gerenciamento;
4. Grupo de processos de Monitoramento e Controle: Realizam as medições e monitoramento do progresso, permitindo identificar variações no plano de gerenciamento.

5. Grupo de processos de Encerramento: Formaliza as entregas e encaminha o projeto para a finalização.

O Guia PMBOK sugere que as fases podem se relacionar de maneira sequencial ou sobreposta (PMI, 2013 apud BARACAT, 2016). Para Gonçalves (2016), normalmente o relacionamento é sequencial e as saídas de um grupo se transformam em entradas do grupo subsequente, esse tipo de relacionamento reduz incertezas, mas elimina chances de redução do cronograma total. Na relação sobreposta, uma fase tem início antes da finalização de outra, acontecendo em paralelo, isso pode reduzir o cronograma, mas pode acarretar em retrabalhos, atrasos e recursos extras.

O PMI (2013 apud BARACAT, 2016), lista quarenta e sete processos divididos em dez áreas de conhecimento, as dez áreas de conhecimento são: Integração, Escopo, Tempo, Custos, Qualidade, Recursos Humanos, Comunicações, Riscos, Aquisições e Partes Interessadas. Baracat (2016) indica que as áreas de conhecimento detalham as entradas e saídas do processo e explicam as técnicas e ferramentas usadas com maior frequência para produzir os resultados esperados. A Figura 4 mostra o mapeamento dos processos, relacionando-os aos grupos e áreas de conhecimento.

Figura 4 Mapeamento dos processos por área de conhecimento e grupo.

Áreas de conhecimento	Grupo de processos de gerenciamento de projetos				
	Grupo de processos de iniciação	Grupo de processos de planejamento	Grupo de processos de execução	Grupo de processos de monitoramento e controle	Grupo de processos de encerramento
4. Gerenciamento da integração do projeto	4.1 Desenvolver o termo de abertura do projeto	4.2 Desenvolver o plano de gerenciamento do projeto	4.3 Orientar e gerenciar a execução do projeto	4.4 Monitorar e controlar o trabalho do projeto 4.5 Realizar o controle integrado de mudanças	4.6 Encerra o projeto ou fase
5. Gerenciamento do escopo do projeto		5.1 Planejar o escopo 5.2 Coletar os requisitos 5.3 Definir o escopo 5.4 Criar a EAP		5.5 Verificar o escopo 5.6 Controlar o escopo	
6. Gerenciamento do tempo do projeto		6.1 Planejar o cronograma 5.2 Definir as atividades 6.3 Sequenciar as atividades 6.4 Estimar os recursos das atividades 6.5 Estimar as durações das atividades 6.6 Desenvolver o cronograma		6.7 Controlar o cronograma	
7. Gerenciamento dos custos do projeto		7.1 Planejar o custo 7.2 Estimar os custos 7.3 Determinar o orçamento		7.4 Controlar os custos	
8. Gerenciamento da qualidade do projeto		8.1 Planejar a qualidade	8.2 Realizar a garantia da qualidade	8.3 Realizar o controle da qualidade	
9. Gerenciamento dos recursos humanos do projeto		9.1 Desenvolver o plano de recursos humanos	9.2 Mobilizar a equipe do projeto 9.3 Desenvolver a equipe do projeto 9.4 Gerenciar a equipe do projeto		
10. Gerenciamento das comunicações do projeto		10.1 Planejar as comunicações	10.2 Gerenciar as comunicações	10.3 Controlar as comunicações	
11. Gerenciamento dos riscos do projeto		11.1 Planejar a gestão de riscos 11.2 Identificar os riscos 11.3 Realizar a análise qualitativa dos riscos 11.4 Realizar a análise quantitativa dos riscos 11.5 Planejar as respostas aos riscos		11.6 Monitorar e controlar os riscos	
12. Gerenciamento das aquisições do projeto		12.1 Planejar as aquisições	12.2 Conduzir as aquisições	12.3 Administrar as aquisições	12.4 Encerrar as aquisições
13. Gerenciamento das partes interessadas do projeto	13.1 Identificar as partes interessadas	13.2 Desenvolver o plano das partes interessadas	13.3 Gerenciar o engajamento das partes interessadas	13.4 Controlar o engajamento das partes interessadas	

Fonte: PMI (2013 apud BARACAT, 2016 p.30)

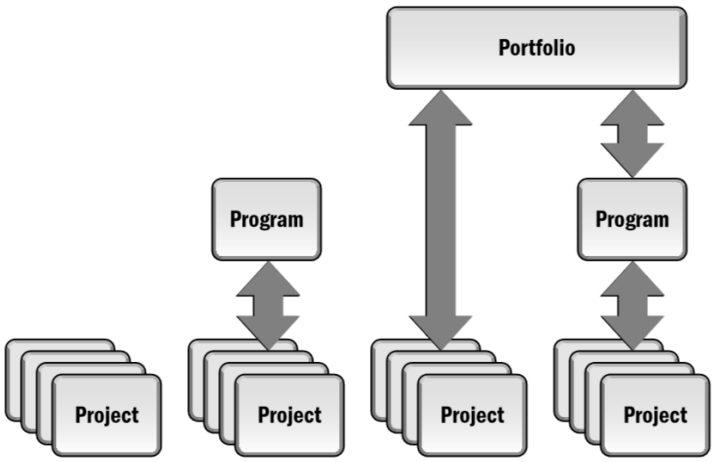
É importante ressaltar que 24 dos 47 processos estão concentrados no grupo de planejamento, mostrando que o enfoque do Guia é planejar (BARACAT, 2016). Baracat (2016) afirma ainda que normalmente as práticas indicadas pelo PMBOK são adaptadas para atenderem completamente as necessidades dos projetos.

O PMI (2013) disponibiliza uma extensão para o Guia PMBOK (SXBOK) específica para software, nela estão descritas as principais características, os maiores desafios de gerenciamento em projetos de software e são utilizados termos e processos específicos para a gestão de projetos de software. As considerações da extensão de software do PMBOK serão descritas a seguir.

De acordo com o PMI (2013), através do SXBOK, os projetos de software podem acontecer em três frentes de trabalho diferentes, que podem acontecer juntas ou separadas, de acordo com as necessidades do projeto, são elas: *System Software* – o software que serve como infraestrutura para desenvolvimento de outros softwares, *Software-Intensive* – o agrupamento de software e hardware, nesses sistemas, o software é o principal agente integrador entre todas as partes e o *Application Software* – softwares que proveem a interface com o usuário em diferentes aplicações e utilidades.

Quando se considera o gerenciamento dos projetos, o SXBOK (PMI, 2013) define a relação de projetos de software com portfólio e programa. Portfólio conforme a definição do PMBOK, é uma coleção de projetos ou programas e outros trabalhos que são agrupados para facilitar o gerenciamento efetivo dos trabalhos para convergir com o objetivo estratégico da empresa. Esse tipo de gerenciamento normalmente é realizado por empresas específicas de software. Quanto aos programas, são usados quando o software é um componente de sistemas que envolvem outros desenvolvimentos, assim, pode não existir gerenciamento específico do projeto de software. A relação dos projetos de software com programas e portfólio pode ser visto na Figura 5.

Figura 5 Relação entre projetos, programas e portfólio.



Fonte: PMI (2013).

A composição dos times de desenvolvimento nessas frentes de trabalho deve ser sempre balanceada entre recursos focados e compartilhados, de forma que o trabalho possa ser otimizado. O SXBOX (PMI, 2013) apresenta alguns *trade-offs* principais relacionados aos times de trabalho, os quais podem ser vistos Quadro 2:

Quadro 2 Trade-offs considerados ao montar equipes de projetos (continua).

Tipo de Recurso	Descrição
Times dedicados X Times compartilhados	Usar times dedicados a um projeto evita sobrecarga dos membros da equipe, além de garantir foco em tarefas do projeto e eficiência. Entretanto, muitos projetos não possuem tarefas ou orçamento suficientes para manter pessoas dedicadas, devendo-se buscar o equilíbrio entre os dois formatos de times.
Times colaborativos X Divisão funcional	Em desenvolvimento de software, os times colaborativos devem envolver pessoas que consigam trabalhar com maestria desde a interface com o usuário até as partes mais internas do software, enquanto trabalhar com divisão funcional permite que cada time seja responsável pela entrega de uma parte do total. Times colaborativos permitem que exista uma maior troca entre os membros, melhorando o tempo de entrega e reduzindo retrabalhos. A recomendação nesse caso é trabalhar com times colaborativos e direcionar tarefas para especialistas em casos de complexidade intensa no projeto.
Alocação virtual X Alocação física	Muitos times são beneficiados por discussões presenciais, o que permite maior troca de conhecimentos tácitos e uma maior proximidade da equipe. Em contraste, há dificuldades para detalhar problemas encontrados quando se utiliza o ambiente virtual, mas isso permite que recursos externos sejam aplicados ao projeto em um custo menor. Assim, quando o projeto envolve pessoas alocadas em diferentes áreas, é interessante que as discussões principais – como iniciação do projeto, planejamento, orientação e treinamentos – sejam feitas presencialmente e o trabalho diário seja feito de maneira virtual.

Fonte: PMI (2013).

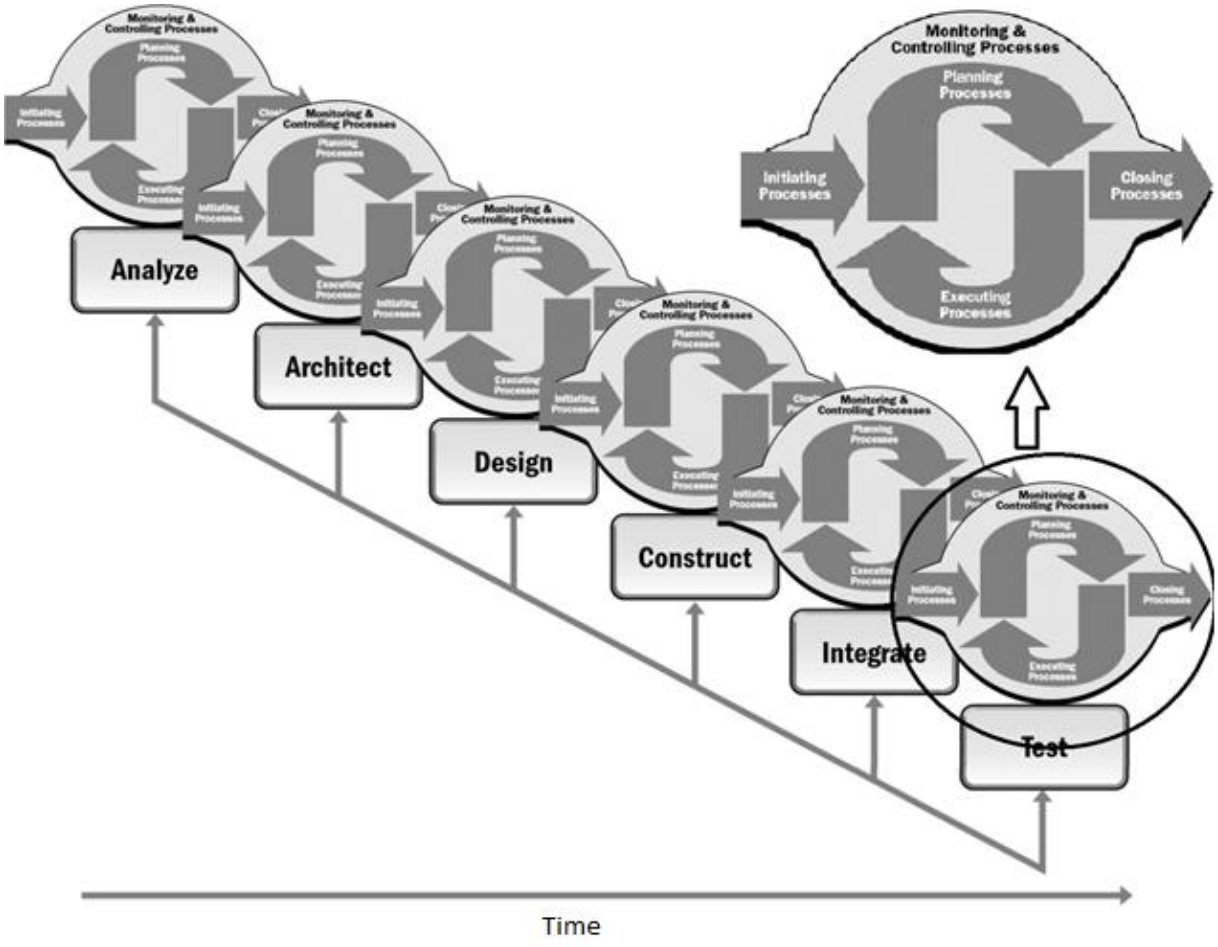
Quadro 2 Trade-offs considerados ao montar equipes de projetos (conclusão).

Tipo de Recurso	Descrição
Especialistas X Generalistas	Como os preços para a contratação de um especialista são altos, normalmente, os gerentes de projetos utilizam equipes generalistas e, periodicamente, chamam um especialista para avaliar e prestar assistência aos membros generalistas. Essa abordagem permite encontrar soluções diferenciadas.
Equipes fixas X Temporárias	Manter uma equipe entre projetos permite maior retenção de conhecimento e garante melhor desempenho no desenvolvimento. A troca entre equipes, por outro lado, é mais barata. Assim, o SXBOX recomenda equipes fixas para melhorar o desenvolvimento do projeto.

Fonte: PMI (2013).

O ciclo de vida de um projeto de software é composto por seis processos principais: Análise – processo de análise dos requisitos de software; Arquitetura – processo de definição da arquitetura do software; Design – detalhamento do design do software; Construção – desenvolvimento do software; Integração – processo de integrar as diferentes partes desenvolvidas; e Testes – processo de testar e qualificar o software desenvolvido (PMI, 2013). Normalmente as fases acontecem sequencialmente, mas podem se sobrepor, como descrito também no guia PMBOK. As sobreposições acontecem parcialmente quando a fase anterior tiver provido saídas suficientes para a fase seguinte ser iniciada e houver recursos para duas fases acontecerem simultaneamente (PMI, 2013). A Figura 6 mostra um esquemático de processos sobrepostos no desenvolvimento de softwares:

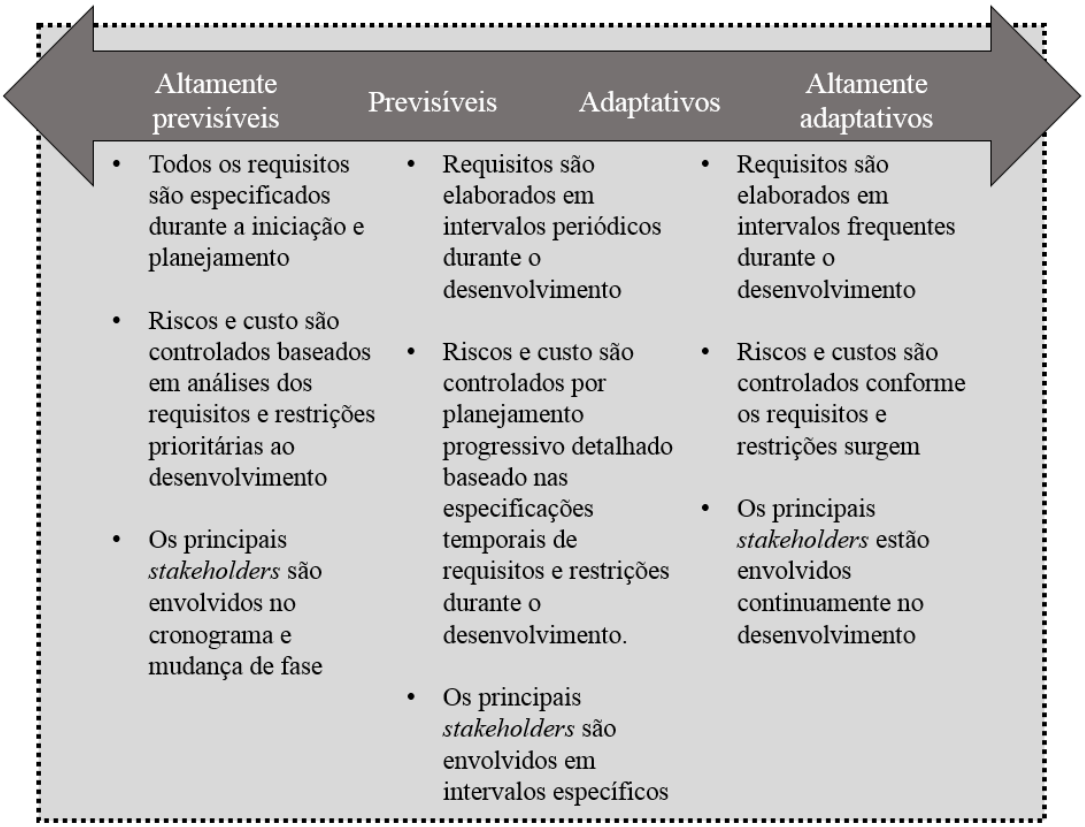
Figura 6 Fluxo de projeto com fases parcialmente sobrepostas.



Fonte: PMI (2013).

O PMI (2013) define que os ciclos de vida dos projetos de software estão em uma linha contínua que varia entre projetos altamente previsíveis e altamente adaptativos. Os projetos altamente previsíveis são caracterizados pela ênfase na especificação dos requisitos e planos baseados em requisitos conhecidos e riscos e custo reduzidos. Já os projetos altamente adaptativos são compostos por especificação progressiva dos requisitos e apresenta pequenos ciclos iterativos de desenvolvimento. Os riscos e custo reduzem de acordo com os avanços do plano inicial. É fundamental que nos processos altamente adaptativos os *stakeholders* estejam envolvidos no desenvolvimento. A Figura 7 mostra o espectro dos ciclos de vida de software.

Figura 7 O espectro contínuo de ciclo de vida de software.



Fonte: PMI (2013).

Os processos de desenvolvimento de projetos de software no SXBOX são divididos e estudados em dez áreas de conhecimento e cinco grupos de processos, como é descrito no guia PMBOK (PMI, 2013) e na Figura 4 anteriormente. Assim, o PMI (2013) considera que poucos ajustes são necessários no gerenciamento de software com relação ao gerenciamento de outros projetos diversos.

3.2.1.2 PRINCE2

O PRINCE2, assim como o PMBOK também pode ser aplicado a qualquer projeto, pois isola aspectos de gerenciamento de projetos das contribuições especializadas (SILVEIRA, 2017). Essa metodologia surgiu no Reino Unido em 1989, apesar de ter sido lançada somente em 1996 (BARCELOS, 2016) e é utilizada como padrão para gestão de todos os projetos governamentais no país (OFFICE OF GOVERNMENT COMMERCE, 2011 apud BARCELOS, 2016). O método surgiu para o desenvolvimento de projetos de tecnologia de informação (TI), mas foi expandida para projetos de quaisquer tipos (BARACAT, 2016).

Essa metodologia é estruturada sobre quatro elementos que se integram: sete princípios, sete temas (análogos às áreas de conhecimento do PMBOK), sete processos e, finalmente, o ambiente do projeto (SILVEIRA, 2017). O foco é o gerenciamento de seis objetivos principais do projeto: escopo, tempo, custo, qualidade, riscos e benefícios (FERNANDEZ, 2016). O Quadro 3, abaixo, mostra os elementos acima citados (princípios, temas e processos).

Quadro 3 Princípios, temas e processos do PRINCE2

PRINCÍPIOS	TEMAS	PROCESSOS
1 Justificação de negócio contínua	**1** Caso de negócio	**1** Pré-projeto
2 Aprender com a experiência	**2** Organização	**2** Dirigir o projeto
3 Papéis e responsabilidade definidos	**3** Qualidade	**3** Iniciar o projeto
4 Gerenciar por estágios	**4** Planos	**4** Controlar os estágios
5 Gerenciar por exceção	**5** Risco	**5** Gerenciar a entrega do produto
6 Foco em produtos	**6** Mudanças	**6** Gerenciar o limite de estágios
7 Adequar o ambiente do projeto	**7** Progresso	**7** Encerramento do projeto

Fonte: Barcelos (2016)

Os princípios do PRINCE2 devem ser cumpridos em sua totalidade e podem ser considerados boas práticas para garantirem que o projeto está sendo gerenciado nessa metodologia (ALEXOS, 2009 apud FERNANDEZ, 2016). São as definições de cada princípio:

1. O primeiro princípio – justificação de negócio contínua – requer uma razão documentada para o início do projeto e a garantia de validade da justificativa para todo o ciclo de vida do projeto (BARCELOS, 2016; FERNANDEZ, 2016);
2. O segundo princípio – aprender com a experiência – indica uma forma colaborativa de desenvolver o projeto incorporando conhecimento tácito e explícito e documentando-o continuamente (BARCELOS, 2016; FERNANDEZ, 2016);
3. O terceiro princípio – papéis e responsabilidade definidos – define as pessoas certas que precisam estar no projeto, mantendo times com uma estrutura organizacional clara (BARCELOS, 2016; FERNANDEZ, 2016);
4. O quarto princípio – gerenciar por estágios – define que os projetos devem ser planejados, monitorados e controlados em etapas ou fases (BARCELOS, 2016; FERNANDEZ, 2016);
5. O quinto princípio – gerenciar por exceção – estabelece que um projeto tem riscos e os limites aceitáveis para os objetivos de entrega (BARCELOS, 2016; FERNANDEZ, 2016);

6. O sexto princípio – foco em produtos – define que um projeto concentra o foco na definição e entrega dos produtos e na qualidade e atendimento dos requisitos (BARCELOS, 2016; FERNANDEZ, 2016);
7. O sétimo princípio – adequar ao ambiente do projeto – declara que o PRINCE2 se adequa ao ambiente, porte, complexidade, importância, capacidade e risco do projeto, de acordo com as necessidades (BARCELOS, 2016; FERNANDEZ, 2016).

Os temas do PRINCE2 são análogos às áreas de conhecimento definidas no PMBOK e definem os aspectos que precisam ser gerenciados continuamente ao longo do projeto. Cada tema pode ser explicado, de acordo com Barcelos (2016) da seguinte maneira:

1. Business Case – trata da justificativa do projeto e busca responder que valor o projeto entrega para a organização.
2. Organização - responsabiliza-se pelos papéis dos envolvidos e pela definição dos responsáveis em cada área do projeto, respondendo, de acordo com Fernandez (2016), à pergunta "como os papéis e responsabilidades individuais de cada membro da equipe podem ser definidos visando uma gestão eficaz do projeto? ".
3. Qualidade – descreve o produto ou a entrega final do projeto. Em outras palavras, define quais os requisitos da entrega final e como serão mensurados.
4. Planos – consiste na definição estrutural do projeto, para Fernandez (2016), deve responder à pergunta "qual o planejamento dos passos e técnicas? "
5. Riscos – está relacionado ao gerenciamento das ameaças e oportunidades, definindo como serão avaliadas e abordadas as incertezas no ambiente de projeto e levando ao sexto tema;
6. Mudanças – determina um gerenciamento de possíveis mudanças e imprevistos;
7. Progresso – é o controle do andamento do projeto e a constante avaliação da viabilidade, dos planos e se o projeto deve prosseguir.

O PRINCE2 define para cada um dos processos, uma lista de verificação de atividades com as principais recomendações de processos e descrições dos produtos, relatórios, registros, notas de lição e as responsabilidades relacionadas que devem ser geradas ao final de cada processo (BARCELOS, 2016; FERNANDEZ, 2016). O Quadro 4 apresenta a lista de verificação sugerida pelo método.

Quadro 4 Lista de verificação de atividades PRINCE2

Iniciar um Projeto (SU)
Nomear o gerente executivo e o gerente de projeto
Capturar lições anteriores
Desenhar e apontar a equipe de gerenciamento do projeto
Desenvolver o business case
Selecionar a abordagem e montar o sumário do projeto
Planejar o estágio inicial
Iniciação do Projeto (IP)
Preparar a estratégia de gestão de risco
Preparar a estratégia de gerenciamento da configuração
Preparar a estratégia de gestão da qualidade
Preparar a estratégia de gestão da comunicação
Definir os controles do projeto
Criar o plano de projeto
Refinar o business case
Montar a documentação de iniciação do projeto
Dirigir o Projeto (DP)
Autorizar a iniciação
Autorizar o projeto
Autorizar um plano de estágios ou exceção
Direcionar o projeto
Autorizar o encerramento do projeto
Controlar os Estágios (CS)
Autorizar os pacotes de trabalho
Revisar o status do pacote de trabalho
Revisar o pacote de trabalho completado
Monitoração e relatórios
Relatórios gerenciais
Identificar e analisar riscos e questões (issues)
Escalonar issues e riscos (comitê diretor do projeto)
Tomar ações corretivas
Gerenciar Entrega do Produto (MP)
Aceitar um pacote de trabalho
Executar um pacote de trabalho
Entregar um pacote de trabalho
Gerenciar a *Stage Boundary* (SB)
Planejar o estágio seguinte (plano de estágio)
Atualizar o plano de projeto
Atualizar o business case
Relatório do estágio final
Produzir um plano de exceção
Encerrando um Projeto (CP)
Encerramento planejado ou prematuro do projeto
Preparar encerramento planejado
Preparar encerramento prematuro
Avaliar o projeto
Recomendar o encerramento do projeto

Fonte: Barcelos (2016)

De acordo com Fernandez (2016), o método PRINCE2 direciona o projeto para um business case incorporado a uma estrutura organizacional de projeto utilizando duas técnicas: planejamento baseado em produto e revisão da qualidade. Para Ribeiro (2011) apud Fernandez (2016), apesar da definição de duas técnicas no método, não há impedimentos para que outras técnicas e ferramentas sejam abordadas e usadas em conjunto. O autor indica ainda que o PMBOK pode complementar essa metodologia.

3.2.1.3 Modelo Cascata

O Modelo Cascata apresenta fases definidas e seguidas de maneira linear. Ao final de cada fase associa-se uma documentação padrão que deve ser aprovada antes do início da próxima fase. Suas diferentes etapas podem ser visualizadas no esquemático da Figura 8.

Figura 8 Esquemático Modelo Cascata

Fonte: Vavassori (2002).

Para Vavassori (2002), o Modelo Cascata apresenta sete etapas diferentes, enquanto que Cordeiro (2014) não apresenta a etapa de instalação. Para o autor, as etapas básicas do modelo são somente seis, sendo o modelo adaptável de acordo com a necessidade do projeto. Quanto a cada uma das fases, ambos os autores concordam na descrição, para melhorar o entendimento, as etapas são brevemente descritas a seguir:

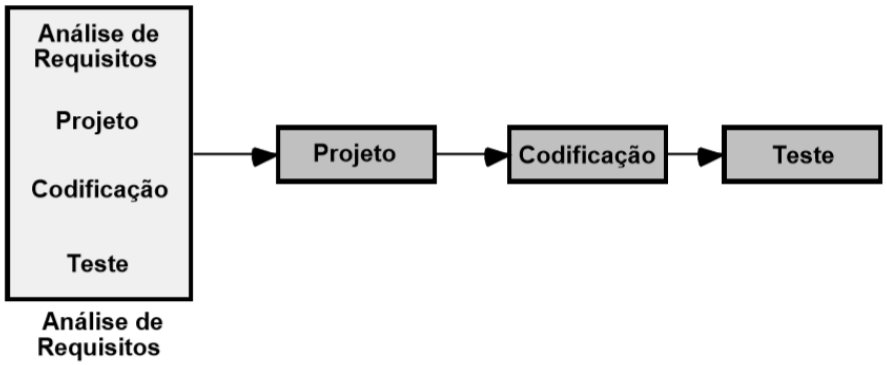
1. Estudo de viabilidade: Ao final dessa fase, tem-se definido em um documento a observação do problema e a avaliação dos custos e benefícios da aplicação proposta;
2. Análise de requisitos: Compreende os requisitos funcionais e não funcionais do sistema, funcionalidade, modularidade e facilidade de uso;
3. Projeto: Elabora as principais recomendações de construção e elabora o projeto detalhado, definindo como a solução será implementada;
4. Codificação: É basicamente a tradução da solução em códigos. Somente é concluída quando toda a codificação está pronta e documentada, além de padronizada e livre de erros de compilação;
5. Testes: Nessa etapa, o código é rigorosamente testado para validar seu funcionamento;
6. Instalação: É o momento no qual o código é distribuído aos usuários;
7. Manutenção: Essa fase inicia-se no momento de entrega do projeto ao cliente e considera as atualizações e alterações solicitadas pelo cliente, independente de considerar ajuste de erros anteriores ou incremento de funcionalidades.

Apesar de ser bastante comum, esse modelo apresenta algumas limitações como a consideração de que os requisitos e funções específicas dos colaboradores não serão alterados durante o desenvolvimento do projeto (CORDEIRO, 2014). Não considerar possíveis mudanças de requisitos pode afetar diretamente na escolha do harware utilizado no sistema, uma vez que os projetos podem levar muito tempo, pode acontecer de se utilizar um hardware não atualizado, aplicando esse modelo.

Para tentar eliminar as limitações do Modelo Cascata, duas variações importantes e que devem ser destacadas foram criadas: O modelo de prototipação e o modelo de desenvolvimento iterativo. Na sequência, os dois modelos são explicados com mais propriedade.

O modelo de prototipação considera possíveis alterações de requisitos nas fases de projeto e codificação, de forma que os requisitos possam ser mais bem compreendidos (VAVASSORI, 2002). O protótipo passa pelas fases de projeto, codificação e teste, mas não formalmente. A intenção do protótipo é dar ao cliente a capacidade de entender melhor os requisitos do programa e, assim, ter requisitos mais estáveis. Um esquemático do modelo de prototipação pode ser visto na Figura 9.

Figura 9 Esquemático do modelo de prototipação.

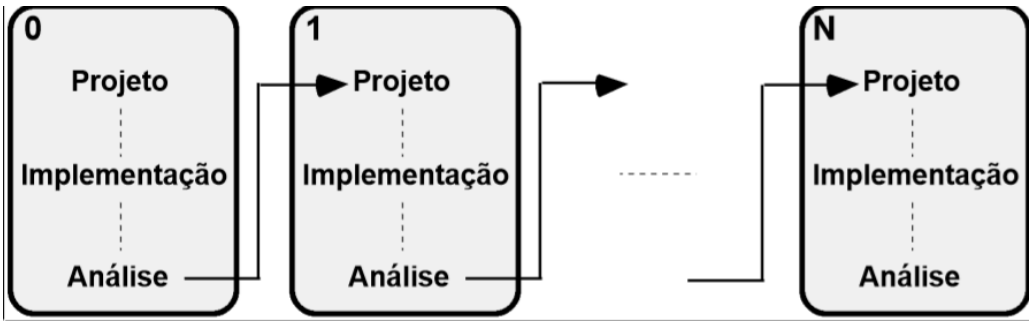


Fonte: Jalote (1997) apud Vavassori (2002)

As principais críticas a esse modelo consideram que o cliente não tem a percepção de que o protótipo ainda não leva em conta as questões de qualidade e manutenção do sistema a longo prazo. Entre os benefícios desse modelo, são citados que seu uso é interessante para sistemas complexos, uma vez que a prototipação permite verificar a viabilidade da solução e melhora a descrição de requisitos difíceis de exprimir (VAVASSORI, 2002).

Quanto ao modelo de desenvolvimento iterativo, esse, é caracterizado por unir os benefícios do Modelo Cascata e do modelo de prototipação, a ideia principal é que o desenvolvimento seja incremental, a cada incremento, uma nova funcionalidade é adicionada ao sistema (VAVASSORI, 2002). Um esquemático para o modelo de desenvolvimento iterativo pode ser visto na Figura 10.

Figura 10 Esquemático Modelo iterativo



Fonte: Jalote (1997 apud Vavassori, 2002).

Pela Figura 10, é possível verificar que inicialmente o sistema recebe uma primeira versão de implementação e nas iterações seguintes, acontecem os incrementos que são realizados até que a solução final do problema seja alcançada. Um problema desse modelo, de

acordo com Jalote (1997 apud VAVASSORI, 2002) é referente ao fechamento de contrato na consideração de como os valores das características adicionadas serão negociados.

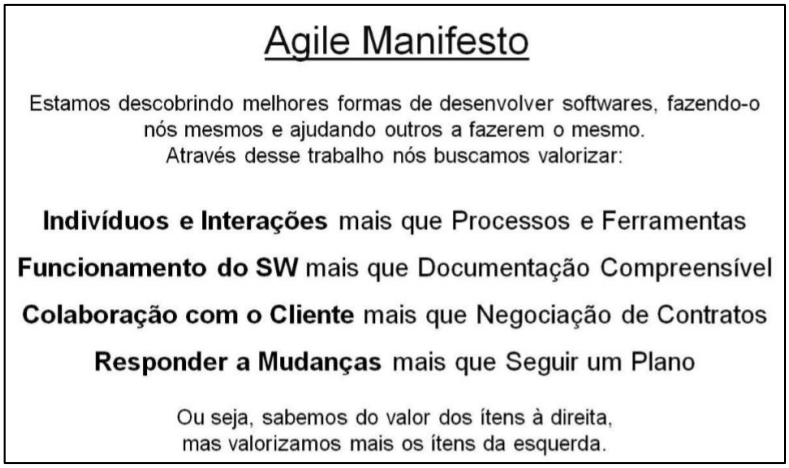
3.2.2 Metodologias ágeis

As metodologias ágeis começaram a surgir durante a década de 1990 para permitir processos adaptáveis a mudanças de acordo com as necessidades de projeto (FAGUNDES, 2005 apud LUDVIG; REINERT, 2007). Essas metodologias vieram principalmente com o objetivo de criar alternativas ao Modelo Cascata e suas variantes (HILMAN, 2004 apud LUDVIG; REINERT, 2007). As metodologias ágeis aceitam a imprevisibilidade, sendo adequadas a cenários de instabilidade, nos quais os requisitos não estão completamente definidos desde o princípio e estão sujeitos a mudanças e redefinições ao longo do desenvolvimento (CORDEIRO, 2014).

Uma boa forma de explicar as metodologias ágeis é indicando que os processos desenvolvidos são os mesmos das metodologias tradicionais: são criados requisitos, é feito o design, o produto é desenvolvido e documentado e, se necessário, há a integração com outros produtos. Em seguida, o produto é testado, problemas são resolvidos e é realizada a entrega. A diferença é que nos métodos tradicionais, o produto é desenvolvido para todas as características de uma vez, enquanto nos processos ágeis os requisitos são divididos em entregas menores ou iterações (LAYTON and OSTERMILLER, 2017).

O termo metodologia ágil popularizou-se em 2001, quando um grupo de especialistas criou a Aliança Ágil e estabeleceu o Manifesto Ágil (HIGHSMITH, 2002 apud LUDVIG; REINERT, 2007). Os termos do Manifesto Ágil podem ser vistos a seguir, na Figura 11.

Figura 11 Manifesto Ágil da Agile Alliance - 2001



Fonte: Cordeiro, 2014.

Observando a Figura 11, é possível inferir que enquanto as metodologias tradicionais enfatizam um plano rígido, evitam mudanças durante o desenvolvimento, documentam tudo e incentivam um controle hierárquico, o manifesto ágil foca em pessoas, comunicações, produto e flexibilidade (LAYTON e OSTERMILLER, 2017). De acordo com Highsmith (2002 apud LUDVIG; REINERT, 2007), os valores do Manifesto Ágil são explicados da seguinte forma:

- Indivíduos e interações valem mais que processos e ferramentas – equipes unidas constroem sistemas. O ponto é considerar a importância das pessoas e como elas trabalham juntas, de forma que as melhores ferramentas não adiantam sem a compreensão da importância do trabalho em equipe (AMBLER, 2004 apud LUDVIG; REINERT, 2007);
- Um software funcionando vale mais que documentação extensa – em diversas ocasiões é mais interessante um protótipo simples que demonstre o funcionamento do sistema do que um diagrama complexo. Um protótipo facilita a análise do cliente, que muitas vezes tem dificuldade para analisar diferentes diagramas. As metodologias ágeis não falam sobre o abandono das documentações, apenas sobre o uso correto das ferramentas disponíveis (SANTOS; MARTINS; LEAL, 2003 apud LUDVIG; REINERT, 2007);
- A colaboração do cliente vale mais que negociação de contrato – quem define as funcionalidades do sistema é o cliente. Ainda que o cliente não tenha certeza do que precisa e que mude os requisitos, essa é a realidade do processo de desenvolvimento. O contrato, apesar de importante não substitui a comunicação. Desenvolvedores de

sucesso trabalham próximos ao cliente e se esforçam para entender sua necessidade (AMBLER, 2004 apud LUDVIG; REINERT, 2007);

- Responder a mudanças vale mais que seguir um plano – à medida que o sistema é construído, o cliente tem mais percepções sobre o domínio do problema e sobre a solução proposta. A mudança é uma realidade no desenvolvimento, então, embora seja importante ter um plano de projeto, é extremamente importante ser maleável a alterações no decorrer do projeto.

Para apoiar as equipes a se desenvolverem corretamente, os criadores do manifesto ágil incrementaram os quatro valores com doze princípios que servem como um guia prático para as equipes de desenvolvimento e são apresentados por Layton e Ostermiller (2017), no texto original[1]:

1. Nossa prioridade mais alta é satisfazer os consumidores através de rápidas e contínuas entregas de software de grande valor.
2. Mudanças nos requisitos são bem-vindas, mesmo ao final do desenvolvimento. Os processos ágeis aproveitam as mudanças para a vantagem competitiva do cliente.
3. Entregar trabalhos de software com frequência entre algumas semanas e alguns meses, com preferência por escalas de tempo menores.
4. Empresários e desenvolvedores devem trabalhar juntos diariamente durante todo o desenvolvimento do projeto.
5. Desenvolva projetos em torno de pessoas motivadas. Dê a eles o ambiente e o suporte necessário e confie que o trabalho será realizado.
6. O modo mais efetivo de levar e trazer informações do time de desenvolvimento é através de conversas presenciais.
7. O software funcionando é a primeira medida de progresso.
8. Processos ágeis promovem ambientes de desenvolvimento sustentáveis. Os responsáveis, desenvolvedores e usuários devem ser capazes de manter um ritmo regular constante indefinidamente.
9. Atenção contínua à excelência técnica e ao bom design aumenta a agilidade.
10. Simplicidade – a arte de maximizar a soma de trabalho não realizado – é essencial.
11. As melhores arquiteturas, requisitos e projetos emergem de times auto organizados.

[1] Tradução da autora.

12. Em intervalos regulares, a equipe reflete sobre como se tornar mais eficiente e ajusta seu comportamento de acordo à reflexão.

O Manifesto ágil não é contra as abordagens tradicionais de desenvolvimento de projeto, ou seja, não rejeita a formalização adotada pelas metodologias tradicionais, apenas aponta que a importância da formalização é secundária se comparada a indivíduos e iterações, como a um software funcional e a abertura a mudanças (LUDVIG; REINERT, 2007). De acordo com Cordeiro (2014):

> É fácil perceber que um dos aspectos críticos para o sucesso das metodologias ágeis é a participação ativa, colaborativa e presencial do cliente junto à equipe de desenvolvimento. É essa participação ativa que irá viabilizar o comprometimento entre as partes quanto a prazos e escopo, a troca das especificações documentadas pela comunicação oral, e a rápida validação, teste e aceitação de cada componente desenvolvido.

Assim, é válido comentar ainda que as metodologias ágeis focam sua aplicação na satisfação do usuário, com flexibilidade a mudanças de requisitos ao longo do desenvolvimento, menor número de defeitos e menor tempo de desenvolvimento, utilizando entregas incrementais (CORDEIRO, 2014). Esses métodos são aplicados a projetos não muito complexos, aplicando, dentre outras características, ciclos iterativos curtos, retroalimentação constante e proximidade da equipe (HIGHSMITH, 2002 apud LUDVIG; REINERT, 2007). Dentre os métodos ágeis, os que mais se destacam são o eXtreme Programming (XP) e o Scrum, com mais de 50% dos usos, de acordo com Almeida (2016).

3.2.2.1 Extreme Programming (XP)

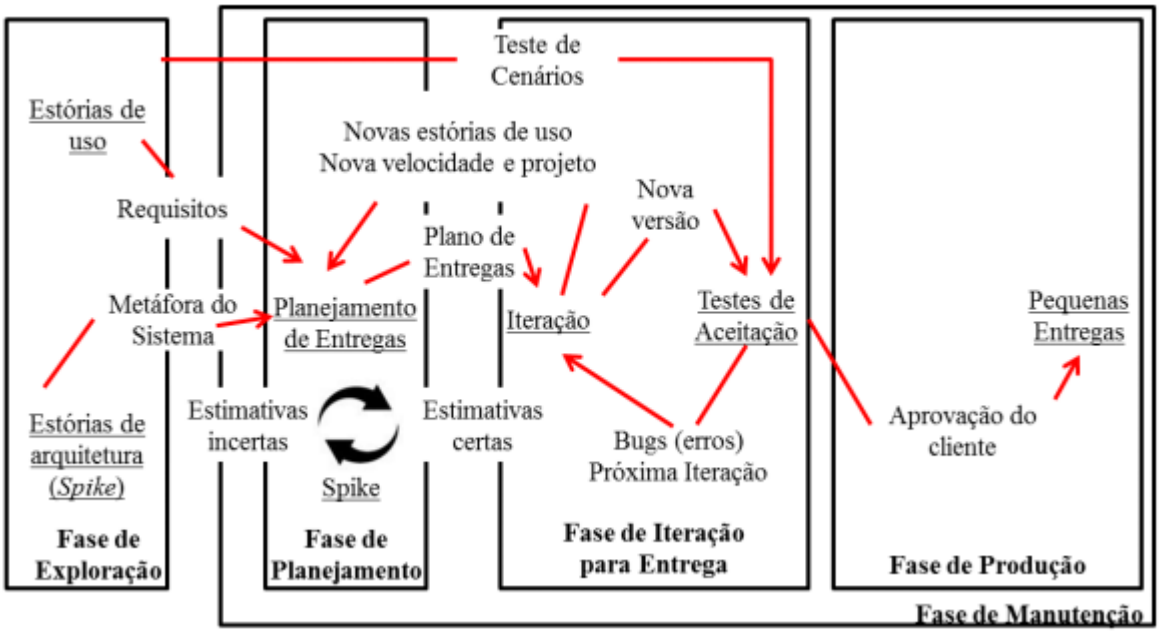
O método XP é definido por alguns autores como a perseguição da mais clara simplicidade em desenvolvimento de software. É uma metodologia voltada para projetos em que os requisitos estão frequentemente sendo alterados, equipes pequenas (geralmente de cinco a dez programadores) e desenvolvimento incremental (LUDVIG; REINERT, 2007). Esse método, de acordo com Almeida (2016), apresenta cinco fases, conforme mostrado na Figura 12. Essas fases podem ser descritas da seguinte maneira:

1. Fase de Exploração: nessa fase são desenvolvidos os fundamentos da arquitetura e os primeiros requisitos do usuário, o suficiente para produzir uma boa primeira versão.

Esses requisitos devem ser de alto nível para o sistema. A equipe de projeto nessa fase realiza testes de arquitetura para explorar todas as possibilidades do sistema (BECK, 1999; AMBLER, 2002 apud ALMEIDA 2016);

2. Fase de Planejamento: essa fase tem como objetivo o alinhamento entre o cliente e a equipe de projeto. Aqui são realizados os acordos de datas e entregas dos requisitos mais prioritários. Nessa fase, o cronograma é acordado a partir da priorização e ordenação das estórias (BECK, 1999);
3. Fase de Iteração: nessa fase acontece toda a parte de modelagem, programação, teste e integração. Daqui saem as principais lições aprendidas da implementação dos requisitos definidos pelo usuário (ALMEIDA, 2016);
4. Fase de Produção: nessa fase, o foco é a entrega de partes de software prontas para o uso, assim, são realizados testes, de sistema, de carga e de instalação para garantir a funcionalidade completa. Novas ideias podem ser adicionadas na iteração ou postergadas para a execução em outro momento (BECK, 1999);
5. Fase de Manutenção: na última fase do projeto, continuam a acontecer desenvolvimentos ao longo do tempo, aplicando-se as fases de planejamento, iteração e produção. É importante que o cliente esteja envolvido apoiando os desenvolvimentos (AMBLER, 2002 apud ALMEIDA, 2016).

Figura 12 Fases da metodologia XP - modelo de referência.



Fonte: Almeida (2016)

A metodologia XP destaca seis atores mais importantes para o desenvolvimento do projeto: programador – responsável pelo desenvolvimento e testes do software, devem manter o mais simples possível; cliente – determina os testes funcionais, estabelece prioridade para o desenvolvimento e define quando um requisito é satisfeito; analista de testes – auxiliam o cliente na definição e desenvolvimento dos testes e se responsabilizam por ferramentas usadas para testar o software; *tracker* – é o responsável pelo feedback e previsões dadas aos clientes; *coach* – é uma forma de gerente que cuida do processo em sua totalidade; gerente – é o responsável pelas decisões, deve comunicar a equipe sobre a situação do projeto e extinguir deficiências no projeto (ALMEIDA, 2016).

Para uma empresa utilizar esse método, deve seguir seus valores e princípios, que prezam pela simplicidade, comunicação e feedback sobre o desenvolvimento em todas as etapas do projeto. As práticas dessa metodologia focam em pequenas reuniões para acertos de diretrizes e pela presença do cliente no desenvolvimento do sistema, as principais são descritas no Quadro 5:

Quadro 5 Principais práticas da metodologia XP

Prática	**Descrição**
Jogo de planejamento	Interação próxima dos clientes e equipe de projetos para a estimativa de estórias
Entregas pequenas	Entregas parciais e novas versões atualizadas rapidamente
Uso de metáforas	Uso de metáforas entre o cliente e a equipe para descrever como o sistema deve funcionar
Projeto simples	Ênfase em projetos com as soluções mais simples que podem ser executadas no momento. Toda a complexidade extra e códigos desnecessários devem ser removidos
Orientação a testes	Desenvolvimento de software dirigido por testes
Remanufatura	Reestruturar os sistemas removendo duplicações, aumentando a comunicação, simplificando e adicionando flexibilidade.
Propriedade coletiva	Qualquer um pode mudar qualquer parte do código a qualquer momento
Programação em pares	Duas pessoas escrevem o código em um computador
Integração contínua	Novas partes do código são integradas ao código base assim que os testes são realizados
Semana de 40 horas	Máximo de 40 horas de trabalho semanais e impossibilidade de duas semanas consecutivas com horas extras
Cliente presente	Cliente presente e disponível para interagir com a equipe quando necessário
Padrões de programação	Regras de programação existem e são seguidas
Espaço de trabalho aberto	Salas grandes com cubículos desejáveis. Equipe trabalha co-localizada
Apenas regras	A equipe tem suas próprias regras que devem ser seguidas, mas podem ser mudadas a qualquer momento.

Fonte: Almeida (2016).

A metodologia XP busca levar à equipe de desenvolvimento discussões e trocas de conhecimento, para o trabalho mais coletivo possível, de forma que o fracasso ou sucesso do projeto sejam compartilhados por todo o time. A qualidade das entregas é melhorada constantemente pela programação em pares, a integração contínua e a exclusão de possíveis partes duplicadas. A presença do cliente também faz com que a precisão das entregas seja melhor. Além disso, o desenvolvimento é feito completamente como um processo de melhoria contínua.

3.2.2.2 Scrum

O Scrum foi criado com o foco de desenvolvimento de software, por Jeff Sutherland, Ken Schwaber e Mike Beedle (SCHWABER, 2004 apud ALMEIDA, 2016). O método é bastante conhecido e considera em sua implementação o gerenciamento de tarefas complexas através de uma abordagem simples (CRUZ, 2016 apud LUDVIG; REINERT, 2007). Suas principais características são a consideração do sistema como processo empírico, iterativo e incremental. Primariamente, o Scrum considera que o sistema é muito complexo e imprevisível para ser planejado completamente no início do desenvolvimento. Em vez disso, o processo é controlado empiricamente para garantir a visibilidade, inspeção e adaptação (LUDVIG; REINERT, 2007).

O ciclo de vida do Scrum inicia-se com a listagem dos requisitos prioritários para o desenvolvimento. Esses requisitos são estabelecidos em conjunto com os *stakeholders* do projeto e não são muito detalhados, é recomendável que apenas as principais características do requisito sejam descritas, essa lista é chamada de *product backlog*. Em seguida, inicia-se o processo de desenvolvimento iterativo através de *sprints* (SCHWABER e SUTHERLAND, 2013 apud PEREIRA, 2014).

O *sprint* é a unidade básica de desenvolvimento do Scrum, um subconjunto do *backlog* e é caracterizado por ser um ciclo de desenvolvimento com período de tempo específico determinado com antecedência para cada *sprint*. Esse período tem duração de uma semana a um mês, normalmente, duas semanas é o tempo utilizado, dentro de cada *sprint*, não são permitidas alterações de requisitos (SATO, 2007 apud PEREIRA, 2014);(ZAVAN, 2013). Cada *sprint* é iniciado por uma reunião de planejamento, na qual são identificadas as tarefas e é feito um compromisso para o objetivo *sprint*. Ao longo do desenvolvimento são realizadas reuniões diárias (*daily Scrum meeting*) para acompanhamento, nessas reuniões discutem-se brevemente os resultados das tarefas realizadas no dia anterior e as tarefas a serem realizadas

no dia. Esses encontros seguem alguns princípios: a reunião começa na hora marcada mesmo sem a presença de alguns membros da equipe; a reunião acontece diariamente no mesmo local e horário; duração máxima de 15 minutos; somente os papéis principais falam, mas todos podem participar (ZAVAN, 2013).

Ao final dos sprints, são realizadas duas reuniões. A primeira – *sprint review meeting* – para revisar o trabalho feito ou não e adaptar o produto em construção. A outra – *retrospective meeting* – para promover a reflexão dos membros da equipe sobre o que pode ser melhorado nos próximos *sprints* a partir do que aconteceu no último, ou seja, serve para adaptar o processo. Cada sprint espera entregar ao cliente partes funcionais do projeto. Sendo considerado nas entregas o software completamente testado e documentado (RISING e JANOFF, 2000); (ALMEIDA, 2016).

O Scrum determina três papéis principais dentro da equipe de projetos, o *project owner* (PO), o *development team* (DT) e o *Scrum master* (SM) e algumas funções auxiliares. Outras funções podem ser vistas em projetos reais, mas a metodologia foca nas principais pois são elas que garantem o desenvolvimento do projeto (PEREIRA, 2014).

O *project owner* é o responsável por maximizar o valor do produto e do trabalho realizado pelo DT. É também o encarregado por gerir o *product backlog* e a pessoa que garante que a equipe se dedicará ao objetivo correto. Assim, ele deve, de acordo com Pereira (2014):

- Representar os interesses dos clientes e stakeholders e expressar claramente os itens/requisitos do *product backlog*;
- Definir data de lançamento e qual será o retorno do investimento do produto.
- Ordenar e priorizar os itens do product backlog para melhor atingir os objetivos do projeto. Não é objetivo do PO descrever como os itens serão implementados em nível técnico;
- Otimizar o valor do trabalho do DT;
- Priorizar e validar as atividades ao final de cada sprint;
- Garantir a compreensão do *product backlog* a todos os membros do projeto no nível necessário.

O *development team* é a equipe responsável pela entrega de incrementos de produtos ao final de cada *sprint*. A sugestão do Scrum é que essa equipe possua de três a nove pessoas com habilidades multifuncionais para execução de análise, projeto, desenvolvimento, testes,

documentação e outras atividades necessárias (PEREIRA, 2014). Outras características importantes para a equipe de desenvolvimento são ser autogerida e auto organizada, garantindo responsabilidade sobre todo o time, melhor desenvolvimento dos objetivos dos *sprints* e o sucesso do projeto (ALMEIDA, 2016).

O *Scrum master* garante os conceitos do ciclo de vida do Scrum. Ele é responsável pela remoção de obstáculos que dificultem a equipe de entregar as metas de produtos e as entregas dos *sprints*. "O SM não é um líder de equipe ou gerente de projeto tradicional, e sim um facilitador para o time e, principalmente, um analista do processo Scrum" (SOMMERVILLE, 2009 apud PEREIRA, 2014). É essa figura que garante a execução do processo Scrum, preside as reuniões-chave e desafia a equipe a melhorar continuamente. O *Scrum master* não tem autoridade sobre as pessoas envolvidas no projeto, mas tem autoridade sobre o processo e existe para garantir que a equipe trabalhe bem em conjunto (ZAVAN, 2013).

Quanto à documentação, quatro aspectos são mais importantes e devem ser considerados: *product backlog*, *sprint backlog*, *increment* e *burndown chart*. De acordo com Schwaber (2004 apud Almeida, 2016 p. 22):

> [...]O primeiro é a lista do product backlog no início da sprint previamente executada. O segundo é uma lista para a próxima sprint. Também são elaborados o relatório de mudanças, que detalha todas as diferenças entre as duas listas anteriores, além de sumarizar o que aconteceu durante a sprint e as adaptações feitas no projeto após a sprint review, e o quarto é o relatório burndown do product backlog.

Para Pereira (2014), o *product backlog* é uma lista ordenada dos requisitos do projeto. Inicialmente são estabelecidos somente requisitos conhecidos e bem entendidos, ao passo que o desenvolvimento evolui, o documento é incrementado. É um artefato dinâmico que muda constantemente de acordo com o que o produto precisa. O *sprint backlog* é a lista de itens a serem trabalhados durante cada sprint. Habitualmente, o *sprint backlog* é acompanhado por um painel de atividades – a executar, em execução e concluídas. O *increment* é a sumarização dos itens completados nos *sprints* anteriores. Esse documento precisa ser utilizável independente da liberação do PO. O *burndown chart* é um gráfico exibido mostrando o trabalho restante no *sprint backlog*. É atualizado diariamente para dar uma visualização simples do progresso do *sprint*.

O Scrum define alguns eventos ou cerimônias principais e que devem ocorrer a cada ciclo ou dentro de cada sprint, em momentos apropriados. De acordo com Pereira (2014),

devem ser destacados o *Sprint planning meeting*, *Daily Scrum*, *End meetings* (considera *review meeting* e *sprint retrospective*) e *Backlog refinement*. Já Almeida (2016), não destaca o *Backlog refinement*. O Quadro 6 apresenta as definições de cada uma das principais reuniões:

Quadro 6 Descrição dos eventos do Scrum.

Sprint Planning Meeting	Acontece no início de cada sprint e é uma reunião para planejar o sprint e as atividades que serão realizadas. Tem limite de oito horas de duração, nas quais as quatro primeiras contam com o time completo e focam no product backlog; as últimas quatro horas são usadas pelo time de desenvolvimento para definir o planejamento do sprint. As principais saídas dessa reunião são: • Seleção do trabalho que será realizado; • Preparação do *sprint backlog* com tempo de cada tarefa; • Identificação de quanto trabalho será provavelmente entregue ao final do *sprint*.
Daily Scrum Meeting	É uma reunião diária de comunicação durante o *sprint* entre a equipe do projeto, dura em torno de quinze minutos e deve acontecer diariamente no mesmo local. As principais saídas dessa reunião são: • Um relatório verbal das atividades do dia anterior; • As atividades a serem desenvolvidas pelo DT no dia.
End Meetings	*Sprint review meeting*: nessa reunião acontece a revisão do trabalho concluído e do trabalho não concluído. *Sprint retrospective*: objetiva promover a reflexão dos membros do time sobre os acontecimentos no *sprint* e quais melhorias serão implementadas no *sprint* seguinte.
Backlog Refinement	Consiste na revisão dos itens do *product backlog*, com o objetivo de definir se estão devidamente priorizados e compreensíveis por toda a equipe.

Fonte: Adaptado de Pereira (2014).

Algumas terminologias são extremamente importantes para a metodologia Scrum, os termos são apresentados por Pereira (2014) como mostrado no Quadro 7:

Quadro 7 Terminologias Scrum (continua)

Abnormal Termination	Em português, Encerramento anormal, significa que o *Product Owner* pode cancelar um *sprint*, se necessário. O *Product Owner* pode fazer isso com o incentivo do time, *Scrum Master* ou da gestão. Por exemplo, a gestão deseja cancelar um Sprint se circunstâncias externas negarem o valor do objetivo do *sprint*. Se um *sprint* é encerrado de forma anormal, o próximo passo é conduzir uma nova reunião de planejamento de *sprint*, onde as razões do encerramento serão revistas.
Definition of Done (DoD)	Em português, Definição de Feito, é um critério de saída para determinar se um item do *Product Backlog* está completo. Em muitos casos, DoD requer que todos os testes de regressão tenham sido bem sucedidos.

Development Team	Em português, Time de Desenvolvimento, consiste em um grupo multifuncional de pessoas responsável por disponibilizar incrementos entregáveis do produto ao final de cada *Sprint*.
Impediment	Em português, Impedimento, consiste em qualquer obstáculo que impeça um membro do time de executar o trabalho da forma mais eficiente possível.
Product Backlog (PBL)	Em português, Histórico do Produto, consiste em uma lista de requisitos priorizados para desenvolvimento.
Product Owner	Em português, Dono do Produto, é a pessoa responsável por manter o *Product Backlog*, representando os interesses dos *stakeholders* e garantindo o valor do trabalho que o time de desenvolvimento executa.
Release Burndown Chart	Em português, Gráfico de Manejo de Entrega, consiste em um gráfico do percentual de itens concluídos do *Product Backlog* em nível de Sprint.
Sashimi	Relatório de que algo está completo. A definição de completo pode variar de time para time, mas precisa ser consistente a todos os integrantes da equipe. Utilizado dentro da realização de um *sprint*.
Scrum But	É uma exceção ao conceito puro da metodologia Scrum, onde o time muda customiza o Scrum para atender as suas próprias necessidades.
Scrum Master	Em português, Mestre Scrum, é a pessoa responsável por manter o processo Scrum, garantindo que ele é utilizado corretamente e seus benefícios são maximizados.
Scrum Team	Em português Time Scrum, é o grupo formado por *Product Owner*, *Scrum Master* e *Development Team*.
Spike	Período fixo de tempo utilizado para pesquisar um conceito e/ou criar um protótipo simples. *Spikes* podem ser planejados para ocorrer entre *sprints* ou, para as equipes maiores, um *Spike* poderia ser aceito como um dos muitos objetivos de entrega do *sprint*. *spikes* são frequentemente lançados antes da entrega dos itens do *Product Backlog* grandes ou complexos, a fim de garantir orçamento, ampliar o conhecimento e/ou produzir uma prova de conceito. A duração e objetivo de um *Spike* serão acordados entre o *Product Owner* e o DT antes do início. Ao contrário de compromissos de *sprint*, *spikes* podem ou não entregar funcionalidades tangíveis, funcionais. Por exemplo, o objetivo de um *spike* pode ser alcançar com sucesso uma decisão sobre um curso de ação.
Sprint	Um período de tempo (tipicamente entre 1 a 4 semanas) onde o desenvolvimento ocorre e um grupo de itens do *Product Backlog* é desenvolvido conforme compromisso do DT.

Fonte: Pereira (2014)

Quadro 8 Terminologias Scrum (conclusão)

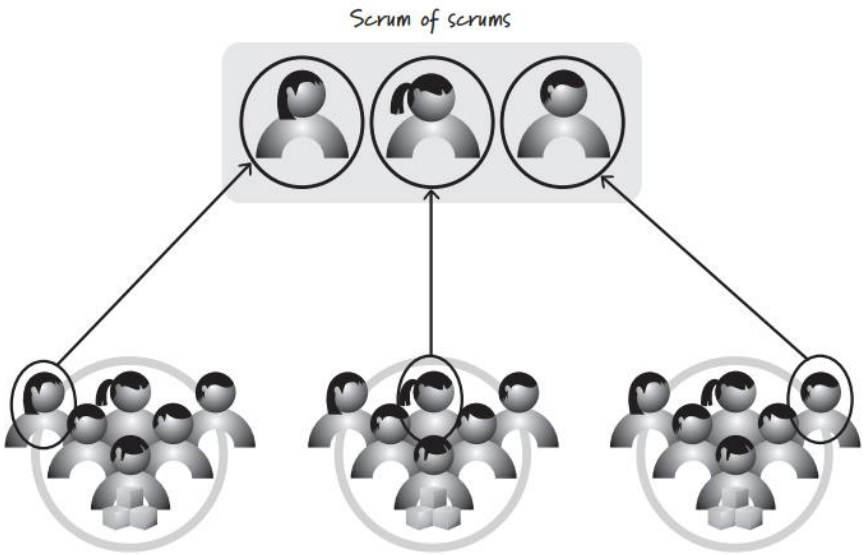
Sprint Backlog (SBL)	**Lista de tarefas priorizadas para serem completadas dentro de um *Sprint*.**
Sprint Burndown Chart	Em português, Gráfico de Manejo do *Sprint*, consiste em um gráfico com o progresso diário ante o que foi planejado para esse *Sprint*.
Tasks	Em português, Tarefas, são itens adicionados ao *Sprint Backlog* no começo de um *Sprint* e são quebradas em horas. Cada tarefa não pode exceder 12 horas (ou dois dias de trabalho).
Velocity	O esforço total que uma equipe é capaz de executar em um *sprint*. O número é obtido pela avaliação do trabalho (normalmente em pontos de Estórias de Usuário) concluído a partir de itens do *backlog* do último *Sprint*. A coleta de dados históricos da velocidade é uma diretriz para

	ajudar a equipe a compreender o quanto de trabalho que eles podem fazer em um *Sprint*.

Fonte: Pereira (2014)

Uma prática aplicada a projetos muito complexos é o *Scrum of Scrums* (SoS), essa prática permite que múltiplos times trabalhem e coordenem o próprio time internamente. O time que forma o SoS é composto por integrantes de todos os times de desenvolvimento. Ou seja, cada equipe define um membro para ser parte do SoS, baseado em quem pode se comunicar com o time. Essa pessoa pode ser alterada de acordo com os problemas do time naquele momento e quem pode representar melhor a equipe (RUBIN, 2012). A Figura 13 representa as equipes de desenvolvimento e o time SoS.

Figura 13 Aplicação de Scrum of Scrums.



Fonte: Layton e Ostermiller (2017)

Quando se trabalha no formato SoS, o Scrum Master deve ser compartilhado por dois ou mais times de desenvolvimento. Quando o projeto é extremamente grande e complexo, o Scrum recomenda que exista um Scrum Master para o time SoS. As reuniões de SoS não costumam acontecer diariamente, mas sim com uma frequência menor, poucas vezes na semana, mas as saídas devem ser similares às das reuniões diárias de cada time (RUBIN, 2012).

4 ANÁLISE E DISCUSSÃO DE RESULTADOS

Este capítulo apresenta as principais discussões sobre os diferentes métodos para gerenciamento de projetos, identificando dentro das particularidades dos projetos de eletrônicos a metodologia mais interessante a ser aplicada na maioria dos projetos desenvolvidos.

Para entender de que maneira cada metodologia se adequa aos projetos de software, foi desenvolvido um quadro mostrando as principais características próprias do desenvolvimento desses sistemas e de que maneira as metodologias atendem ou não à demanda necessária. O levantamento das necessidades foi feito observando-se as particularidades apresentadas na seção 3.1, através dos diferentes tópicos, foram adaptados pequenos termos que referenciassem as particularidades, permitindo a avaliação das diferentes metodologias.

Para a classificação de atendimento, foram classificadas as metodologias avaliadas com nota de 1 a 4, considerando que 1 a metodologia não atende à característica e 4 atende muito bem a característica. A escala com seus respectivos significados pode ser vista a seguir:

1	2	3	4
Não atende à característica	Atende a característica parcialmente	Atende à característica com algumas ressalvas	Atende muito bem à característica

A comparação pode ser vista no Quadro 9:

Quadro 9 Comparação entre as metodologias e as particularidades de desenvolvimento de eletrônicos (continua)

Particularidade	PMBOK/ SXBOK	PRINCE2	Cascata	XP	Scrum
Produto intangível	3	2	3	4	4

Projetos complexos	3	3	4	1	4
Dimensionamento de recursos	2	2	3	4	4
Requisitos mutáveis	1	1	3	4	4
Comunicação clara	1	2	1	4	4
Soluções inovadoras	2	1	1	4	4
Tamanho da equipe	4	4	4	2	4
Velocidade de desenvolvimento	2	3	2	4	4
Testes de uso	1	1	3	4	4
Documentação	4	4	4	2	2
Absorção de ferramentas inovadoras	2	2	2	4	4
Custo de desenvolvimento	1	1	3	4	4
Times colaborativos	2	1	2	4	4
Ciclo de vida adaptável	2	1	2	3	4
	30	28	35	48	54

Fonte: O autor.

Os produtos de software são intangíveis, assim, essa particularidade ajuda a classificar, a adaptação da metodologia a uma entrega final com essa característica. Considera-se que a metodologia se adequa bem a projetos nos quais não serão necessariamente desenvolvidas partes físicas, como o hardware. Nesse aspecto, o PMBOK recebeu classificação 3 por considerar no SXBOK a possibilidade desse desenvolvimento, mas pela dificuldade para lidar com o gerenciamento e medição desse desenvolvimento. O PRINCE2 foi classificado com nota 2 por ser uma metodologia fortemente adaptável, mas principalmente direcionada para produtos físicos e com ciclo de vida padrão e sequencial. O Modelo Cascata foi classificado como 3 por ser uma metodologia específica para software, o principal aspecto pelo qual a metodologia consegue atender parcialmente a mudanças no desenvolvimento estão relacionados às variantes criadas, principalmente o modelo iterativo. Os modelos XP e Scrum foram feitos para captar as principais mudanças entre produtos de ciclo de vida padrão e produtos de ciclo de vida dinâmico, permitindo o gerenciamento de entregas intangíveis com clareza, por isso podem ser classificados como 4.

Quanto à complexidade dos projetos, o PMBOK e PRINCE2 são metodologias fortemente adaptáveis, que se adequam a complexidade e às principais características de desenvolvimento, mas por realizarem todo o planejamento no começo do processo de desenvolvimento, podem perder algum ponto do processo, por isso, foram classificadas como 3. O Modelo Cascata foi considerado como 4 pois se adapta a qualquer modelo de projeto de

software, apesar de as entregas serem realizadas com mais lentidão, devido à característica sequencial do processo. Quando se consideram as variantes do processo, essa característica é reduzida. O modelo XP é feito para projetos mais simples, por isso foi considerado como não atende à característica. O Scrum é uma metodologia que consegue se adaptar a todas as complexidades de projetos, por dividir o processo priorizando os requisitos mais importantes para o cliente, assim, essa metodologia é altamente aderente à característica, sendo classificada como 4.

Quando o dimensionamento de recursos é considerado, são pensados os aspectos do momento da definição e da precisão com a qual os recursos são dimensionados para o projeto. Assim, as três primeiras metodologias (PMBOK, PRINCE2 e Modelo Cascata) realizam o dimensionamento no início do projeto, o que pode levar a falhas substanciais de acordo com problemas que podem surgir durante o desenvolvimento. A diferença que faz com que a nota do Modelo Cascata seja 3, em comparação com o 2 dos outros dois modelos é a natureza das variantes desse procedimento, que permitem uma compreensão relativamente melhor dos recursos necessários. Os modelos XP e Scrum por dimensionarem os recursos no início de cada entrega sprint ou ciclo de implementação conseguem se adaptar melhor e serem mais precisos nesse quesito, por isso foram avaliados com nota 4.

A análise de requisitos mutáveis considera que os requisitos nem sempre são fechados no início do desenvolvimento ou podem sofrer alterações, variando o escopo do produto. As metodologias PMBOK e PRINCE2 trabalham com a hipótese de que os requisitos não serão alterados durante o processo e que o desenvolvimento já possui os riscos conhecidos, por isso, não são metodologias aderentes a característica de requisitos mutáveis de desenvolvimento. O Modelo Cascata em suas variações consegue absorver relativamente bem as mudanças de requisito, por isso foi classificado com nota 3. Já as metodologias XP e Scrum trabalham por pequenas etapas, permitindo facilmente a adição de características ao sistema ou a remoção de itens que não fazem mais sentido. Um dos princípios das metodologias ágeis é justamente a abertura a mudanças, mesmo ao final do desenvolvimento.

A comunicação clara no desenvolvimento de software relaciona-se com o atendimento das necessidades do cliente e à comunicação interna da equipe. Nas metodologias tradicionais, a busca pelas características do sistema a ser desenvolvido e pelas necessidades do cliente é dada no início do projeto, em muitos casos, o cliente ainda não tem clareza da sua necessidade, o que acarreta em erros de desenvolvimento e possível insatisfação na entrega final. Enquanto isso, nas metodologias ágeis o cliente está constantemente envolvido no desenvolvimento,

podendo, de acordo com a maturidade do projeto, definir melhor suas necessidades e a compreensão completa dos requisitos.

Quanto à comunicação interna da equipe, nas metodologias tradicionais é importante destacar que a metodologia PRINCE2, por trabalhar com pacotes de trabalho permite que o time compreenda melhor o desenvolvimento, enquanto que nas metodologias PMBOK e Modelo Cascata a comunicação dentro da equipe de desenvolvimento pode resultar em características e compreensões equivocadas do projeto. As metodologias ágeis por considerarem reuniões curtas diárias permitem a maior integração do time e melhor compreensão do que deve ser abordado no desenvolvimento.

A adaptabilidade das metodologias permite ou não o surgimento de soluções inovadoras. Nesse aspecto as metodologias ágeis se destacam por encorajarem a auto-gestão e a constante integração do time de desenvolvimento, evidenciando que todos na equipe de desenvolvimento tem autoridade para propor soluções que se adequem mais ao cliente. Já nas metodologias tradicionais, as formas pelas quais os problemas são resolvidos normalmente são com os mesmos métodos utilizados historicamente, o que muitas vezes inibe o surgimento de soluções criativas e inovadoras na equipe.

Quanto ao tamanho da equipe, as metodologias tradicionais se adaptam muito bem, uma vez que seu processo estruturado permite que o projeto seja desenvolvido independente da complexidade e número de colaboradores. Já o modelo XP, por ser direcionado para projetos simples não se adapta muito bem a equipes grandes, sendo avaliado como 2. O Scrum, por meio do mecanismo SoS pode ser aplicado a qualquer tamanho de equipe, aderindo completamente a particularidade.

Na avaliação da velocidade de desenvolvimento, é importante considerar que retrabalhos nas metodologias tradicionais levam muito mais tempo do que nas metodologias ágeis, já que nessas, cada implementação é testada, o que leva a identificação e solução de erros mais rápida. Nesse aspecto, a variante iterativa do Modelo Cascata permite acelerar o desenvolvimento, mas a necessidade de aprovação da documentação ao final da etapa reduz a velocidade do processo, por isso a metodologia foi considerada 3, enquanto as outras metodologias tradicionais não conseguem absorver a demanda de software.

A avaliação dos testes nas metodologias considerou o momento nos quais os produtos desenvolvidos são testados. Nas metodologias tradicionais, todas as características são desenvolvidas e em seguida o sistema completo passa por uma série de testes, sendo necessário alterar aquilo que não estiver adequado. A diferença da metodologia Cascata é a possibilidade de desenvolvimento iterativo, que permite o teste das funcionalidades de acordo com o

desenvolvimento. Já nas metodologias ágeis, cada implementação é testada e entregue ao cliente para compreender se nenhuma alteração será necessária, assim, o desenvolvimento fica mais adequado a outras características de software, como a velocidade de desenvolvimento e o entendimento dos requisitos.

Quando a documentação é avaliada, as pontuações se invertem um pouco das observadas anteriormente. Esse aspecto está relacionado com a valorização de documentações claras e detalhadas pelas metodologias tradicionais, uma vez que só é permitido o início de uma nova etapa quando a anterior já teve sua documentação revisada e aprovada. Já as metodologias ágeis, consideram que a documentação é secundária, priorizando o desenvolvimento e compreensão dos requisitos, absorvendo apenas parcialmente a necessidade de documentação para aprendizados futuros.

Considerando que os desenvolvedores de sistemas eletrônicos estão sempre trabalhando com ferramentas novas e em constante atualização, é importante entender de que maneira as metodologias absorvem as ferramentas inovadoras. O PMBOK, o PRINCE2 e o Modelo Cascata, por considerarem que o desenvolvimento é fixo em características pré-definidas, não conseguem atender com clareza esse requisito, já que incentivam o desenvolvimento de maneiras já conhecidas para que o gerenciamento de risco não seja alterado. Enquanto isso, o XP e o Scrum absorvem isso ao início de cada nova etapa de desenvolvimento, permitindo que o time se auto gerencie e defina a forma mais eficaz de entregar os resultados.

O custo de desenvolvimento é bastante relacionado à capacidade das metodologias de dimensionarem os recursos, mas nesse aspecto considera principalmente o conhecimento de horas necessárias para a criação das características que o cliente precisa. As metodologias tradicionais acabam falhando nessa definição por considerarem que os riscos de falha são conhecidos no início do processo. Ao monitorar esses riscos específicos, outros aspectos podem ser esquecidos. Outra consideração importante na falha dos modelos tradicionais relaciona-se a incapacidade de captar e implementar mudanças enquanto o projeto está em desenvolvimento, fazendo que os custos sejam substancialmente maiores ao ser necessário retornar de uma etapa de instalação para a implementação de alterações. Os modelos PMBOK e PRINCE2, portanto, não conseguem aderir a característica de custo de desenvolvimento de sistemas eletrônicos. O Modelo Cascata, por ser direcionado para software traduz com mais propriedade os custos envolvidos, principalmente se a variação de prototipação é utilizada, por isso é classificado como 2. Já os modelos XP e Scrum consideram o custo de cada implementação, sendo bastante precisas quanto ao valor total do desenvolvimento

A característica de times colaborativos é principalmente a adaptação dos desenvolvedores a diferentes papéis em diferentes fases do desenvolvimento, de acordo com a necessidade do projeto. O PMBOK considera os papéis bem definidos no início do desenvolvimento, apresentando certa resistência à mudança. A metodologia foi classificada com nota 2 por considerar que é necessário trabalhar com times mistos de especialistas e papéis colaborativos, absorvendo parcialmente a demanda de software. Já os métodos PRINCE2 e Modelo Cascata consideram os papéis especificados no início e não alterados durante o desenvolvimento, o que pode dificultar a maleabilidade do desenvolvimento e também ser um obstáculo ao surgimento de soluções inovadoras.

As considerações de ciclo de vida adaptável relacionam-se principalmente à capacidade da metodologia de alterar fases e o desenvolvimento para absorver as características de software e principalmente o seu ciclo de vida dinâmico. As metodologias PMBOK, PRINCE2 e Cascata são fundamentalmente sequenciais, o que dificulta as mudanças no ciclo de vida. As diferenças entre essas metodologias são a possibilidade de desenvolver o projeto com fases parcialmente simultâneas no PMBOK e a possibilidade de trabalho em iterações no Modelo Cascata, enquanto o PRINCE2 se mantém sequencial, não absorvendo a característica de software. Já os modelos ágeis são desenvolvidos para ciclos de vida dinâmicos compreendendo-os com muita clareza.

As metodologias tradicionais apresentadas (PMBOK, PRINCE2, Modelo Cascata, Modelo de Prototipação e Modelo Iterativo) tanto direcionadas a software quanto gerais se apresentam com características mais fixas. Essas metodologias são direcionadas a processos e trabalham com a premissa de que alterações de requisitos e papéis não serão necessárias durante o desenvolvimento do projeto. Além disso, considera-se que os riscos são conhecidos desde o princípio dos projetos, o que pode dificultar a adaptação do desenvolvimento em caso de novos riscos emergirem.

Entre as metodologias direcionadas para software e gerais é possível ver que as gerais são mais abrangentes e adaptativas, ao passo que as direcionadas se baseiam nos mesmos princípios e já foram refinadas para atender especificamente a área de TI e dos projetos de eletrônicos, nos quais o produto não é necessariamente tangível.

Layton e Ostermiller (2017) consideram que trabalhar com as metodologias tradicionais em software é uma tentativa de implementar metodologias utilizadas para projetos com ciclos de vida tradicionais em projetos com ciclo de vida dinâmico e contemporâneo. Para os autores, instituir uma metodologia ágil faz com que não seja necessário definir cronogramas e prazos extremamente apertados para promover o foco da equipe, os problemas são resolvidos

na prática, dinamicamente e na ordem correta priorizada para garantir que os problemas críticos sejam solucionados primeiro.

As principais dificuldades dos modelos tradicionais, como o Modelo Cascata estão na fixação de alguns parâmetros que não são necessariamente conhecidos. A estimativa dos requisitos totais é complexa e por isso difícil de estimar no início do projeto. Os clientes e *stakeholders* não se envolvem no desenvolvimento, isso pode acarretar em requisitos mal compreendidos pelo time de desenvolvimento ou ainda em características que não serão utilizadas após o lançamento do produto.

Nos desenvolvimentos tradicionais, o trabalho é racionado no início e as tarefas e funções são bem conhecidas desde o princípio. Pessoas especializadas desenvolvem o produto, esperando-se que não sejam necessários retrabalhos. Em contrapartida, nas metodologias ágeis, o desenvolvimento é feito por times altamente adaptativos, o que faz surgir a necessidade de trabalho com times muito dinâmicos e que consigam desenvolver diferentes características ao longo do projeto.

Somente 11% dos projetos de software desenvolvidos com metodologias tradicionais obtiveram sucesso no lançamento dos seus produtos, em uma pesquisa de 2015, apresentada por Layton e Ostermiller (2017). Esses projetos foram entregues na data correta e com o orçamento correto. Dos projetos pesquisados, 29% foram cancelados antes do lançamento e não apresentaram ganhos para a empresa que os lançou e 60% foram finalizados, mas lançaram após a data esperada, com características ultrapassadas ou, ainda, com a qualidade abaixo do esperado. Em metodologias ágeis, apenas 9% dos projetos são cancelados antes do lançamento.

Outro dado alarmante trazido por Layton e Ostermiller (2017) é que projetos desenvolvidos com metodologias tradicionais apresentam ao final 64% de características que são raramente ou nunca utilizadas. Apenas 7% das características entregadas são empregadas frequentemente pelo usuário. Esses dados levantados indicam a tendência ao uso das metodologias ágeis. O Quadro 10 apresenta as diferenças principais do gerenciamento de projeto tradicional e ágil.

A implementação de metodologias ágeis para desenvolvimento de software e hardware é mais indicada por trabalhar melhor com questões de alterações rápidas, as quais são constantes em eletrônicos e projetos de tecnologia. A característica fundamentalmente sequencial das metodologias tradicionais é a principal dificuldade na utilização das mesmas em desenvolvimentos de eletrônicos, uma vez que ao finalizar uma etapa os gastos são enormes para retornar e alterar características, isso porque é necessário realizar a etapa novamente.

Quadro 10 Comparação entre metodologias ágeis e tradicionais.

Gerenciamento de tarefas em metodologias tradicionais	**Gerenciamento de tarefas em metodologias ágeis**
Criação de um documento completo e detalhado dos requisitos no início do projeto. Tenta controlar mudanças de requisito no desenvolvimento do projeto.	Criação do *product backlog* – uma lista simples dos requisitos organizada por prioridades. Rapidamente altera a lista em casos de alterações de requisitos.
Conduz semanalmente longas reuniões com todos os *stakeholders* e desenvolvedores. Envia a ata e os relatórios de status após todas as reuniões.	O time de desenvolvimento se reúne diariamente em torno de 15 minutos para coordenar e sincronizar o andamento do projeto. Podem atualizar rapidamente o *burdown chart*.
Cria um cronograma detalhado no início do projeto com todas as tarefas necessárias e tenta manter as tarefas dentro das datas definidas e atualiza o cronograma regularmente.	Trabalha com *sprints* e identifica somente as tarefas necessárias para o desenvolvimento daquela fase.
Determina e direciona as tarefas para o time de desenvolvimento	Suporta o time de desenvolvimento removendo distrações e impedimentos. O time de desenvolvimento define suas próprias tarefas.

Fonte: Layton e Ostermiller (2017)

Conforme observado no Quadro 9, a metodologia que mais se adequa é o Scrum, mas há pontos que são mais fortes em outras metodologias, como a documentação. A utilização de metodologias de gestão ágeis causa uma mudança de cultura no ambiente de trabalho, devendo-se observar na implementação a melhor forma de direcionar os times para o novo *mindset*.

5 CONSIDERAÇÕES FINAIS

A verificação da metodologia mais adequada para o desenvolvimento de eletrônicos seguiu a sequência de identificar as particularidades de projetos de sistemas eletrônicos, verificando as semelhanças entre desenvolvimento de software e hardware; conhecimento e descrição das principais metodologias de gestão de projetos, conhecidas através de análise bibliométrica; e, finalmente através da comparação entre as metodologias, considerando as particularidades do desenvolvimento de sistemas eletrônicos. Portanto, esse trabalho concluiu todos os objetivos propostos de maneira clara.

A comparação entre as diferentes metodologias de gestão de desenvolvimento de projetos permitiu concluir que a metodologia mais adequada para projetos de sistemas eletrônicos é o Scrum, que alcançou 54 pontos, por se adaptar melhor aos diferentes tamanhos de equipe, complexidade de projeto e mudanças de requisitos e papéis que podem ser necessárias durante o processo. Para casos nos quais o desenvolvimento é longo e complexo é válido considerar a combinação de metodologias para otimizar e aperfeiçoar o desenvolvimento e documentação do projeto.

Na sequência das metodologias ágeis, a metodologia tradicional que mais se adapta aos desenvolvimentos de eletrônicos é o Modelo Cascata, com 35 pontos, devido aos seus direcionamentos para software e às variantes que permitem inibir as maiores limitações do modelo original. A metodologia PMBOK, com 30 pontos, é pouco melhor que a PRINCE2, que teve 28 pontos, principalmente por ter a extensão SXBOK, que permite a adequação de algumas particularidades de software. Para ilustrar com mais prioridade as diferenças descritas, o Quadro 11 mostra as distinções principais entre as três metodologias tradicionais:

Quadro 11 Principais diferenças entre metodologias tradicionais.

PMBOK	PRINCE2	Cascata
Controla o projeto em dez frentes: Integração, Escopo, Tempo, Custo, Qualidade, Recursos Humanos, Comunicações, Riscos, Aquisições e Partes interessadas.	Tem o foco em seis aspectos do projeto: Escopo, Tempo, Custo, Qualidade, Riscos e Benefícios.	Considera uma análise de viabilidade prévia ao início do projeto. Criada para o controle de entregas e documentações de software
É abrangente e adaptável a qualquer projeto, para eletrônicos, considera seis fases específicas: Análise, Arquitetura, Design, Construção, Integração e Testes. As fases podem ser parcialmente sobrepostas.	Não possui uma diferenciação para software/hardware.	Criada com a intenção de desenvolvimento de software, considera as fases de: Estudo de viabilidade, Análise de requisitos, Projeto, Codificação, Testes, Instalação e Manutenção. Permite adaptação das fases para validação dos requisitos.
Para que a próxima fase seja iniciada, a anterior precisa ter fornecido dados suficientes, permitindo fases parcialmente simultâneas.	Possui um check list a ser realizado antes do início de uma nova fase, depende da conclusão da fase anterior para iniciar.	Fundamentalmente é sequencial, possui variações que permitem a validação da solução proposta e desenvolvimento incremental.

Fonte: A autora.

Para complementar esse trabalho, é sugerida a validação desse trabalho por especialistas com certificação e experiência em projetos para garantir que a avaliação realizada é condizente com o que acontece na prática. São sugeridas também pesquisas aplicadas da implementação de cada uma das metodologias, sendo essa, uma maneira de garantir que a conclusão alcançada está correta. Outra sugestão é a aplicação direta da metodologia Scrum em um projeto de sistema eletrônico, confirmando se o modelo é o mais adequado para projetos de eletrônicos.

O desenvolvimento dessa pesquisa reforça a necessidade de estudo na área de gestão adequada para a área de projetos de sistemas eletrônicos, de forma que são garantidas entregas mais precisas e rápidas com maior satisfação do cliente e equipe de desenvolvimento mais motivada.

REFERÊNCIAS

ALMEIDA, Luís Fernando Magnanini de. **UM MODELO PARA APOIAR A GESTÃO DO CONHECIMENTO NO GERENCIAMENTO ÁGIL DE PROJETOS DE SOFTWARE.** 2016. 322 f. Tese (Doutorado) - Curso de Engenharia de Produção, Centro de Ciências Exatas e de Tecnologia, Universidade Federal de São Carlos, São Paulo, 2016.

ARAÚJO, Carlos Alberto. BIBLIOMETRIA: Evolução Histórica E Questões Atuais. **Em questão**, Porto Alegre, RS, v. 12, n. 1, p. 11-32, 2006.

BARACAT, C. C. **GERENCIAMENTO DE PROJETOS: UM CONFRONTO ENTRE METODOLOGIAS ÁGEIS E TRADICIONAIS**Juiz de Fora, 2016. Disponível em: <http://www.ufjf.br/engenhariadeproducao/files/2016/12/cesarcarneirobaracat.pdf>. Acesso em: 14 out. 2018

BARCELOS, A. **SIMILARIDADE DOS PROCESSOS QUE COMPÕE OS PRINCIPAIS MODELOS DE REFERÊNCIA UTILIZADOS NA GESTÃO DE SERVIÇOS TERCEIRIZADOS DE TI: UMA SURVEY COM FORNECEDORES E CLIENTES**. [s.l: s.n.].BECK, K. **Extreme Programming Explained**. Pearson Education, Inc, 1999a.

BECK, K. **EXTREME PROGRAMMING EXPLAINED**. Addison Wesley, Reading, 1999.

BUZZETTO, Fabiano Alberto. **IMPLANTAÇÃO DE UM NOVO MÉTODO DE PROJETOS EM UMA EMPRESA DE COMPONENTES ELETRÔNICOS**. 2008. 156 p. Dissertação (Mestrado) – Pós-Graduação em Engenharia de Produção, Universidade Federal do Rio Grande do Sul. Porto Alegre, 2008.

Campbell, D. et al. **BIBLIOMETRICS AS A PERFORMANCE MEASUREMENT TOOL FOR RESEARCH EVALUATION**: The case of research funded by the National Cancer Institute of Canada. American Journal of Evaluation 31(1): 66-83, 2010.

COLIN MCINTOSH (Reino Unido). **CAMBRIDGE ADVANCED LEARNER'S DICTIONARY**. 4. ed. Cambridge: Cambridge University Press, 2013.

CORDEIRO, Jose Henrique Dell'osso. **AMBIDESTRIA EM EMPRESAS DESENVOLVEDORAS DE SOFTWARE:** Barreiras para Adoção de Metodologias Ágeis e seu Impacto na Escolha do Modelo Organizacional. 2014. 61 f. Dissertação (Mestrado) - Curso de Mestrado em Engenharia de Produção, Universidade de São Paulo, São Paulo, 2014.

Costa, R., **A COMUNICAÇÃO ELETRÔNICA E A ALTERAÇÃO DE TEMPO E ESPAÇO NA PRODUÇÃO DO CONHECIMENTO CIENTÍFICO**. Ciência da Informação 36: 7-15, 2010.

CYRENO, Manoel Filipe Pereira Silva. **RELATO DA MELHORIA DO PLANEJAMENTO E ESTIMATIVAS EM PROJETOS DE DESENVOLVIMENTO DE SOFTWARE UTILIZANDO SCRUM: um estudo de caso.** 2014. 19 f. Dissertação (Mestrado) - Curso de Mestrado Profissional em Engenharia de Software, Centro de Estudos e Sistemas Educacionais do Recife – C.e.s.a.r, Recife, 2014.

DELLA BRUNA JUNIOR, E.; ENSSLIN, L.; ENSSLIN, S. R. **SELEÇÃO E ANÁLISE DE UM PORTFÓLIO DE ARTIGOS SOBRE AVALIAÇÃO DE DESEMPENHO NA CADEIA DE SUPRIMENTOS**. GEPROS. Gestão da Produção, Operações e Sistemas, Ano 7, nº 1, jan-mar/2012, p. 113-125

DRECHSLER, Rolf; BREITER, Andreas. **HARDWARE PROJECT MANAGEMENT-WHAT WE CAN LEARN FROM THE SOFTWARE DEVELOPMENT PROCESS FOR HARDWARE DESIGN?.** In: ICSOFT (SE). 2007. p. 409-416.

FERNANDEZ, Gabriela de Rezende. **DIRETRIZES PARA CONSTRUÇÃO DE UMA METODOLOGIA DE GERENCIAMENTO DE PROJETOS DE PESQUISA, DESENVOLVIMENTO E INOVAÇÃO TECNOLÓGICA:** CASO DO ESCRITÓRIO DE GERENCIAMENTO DE PROJETOS DO INT NA EMBRAPII. 2016. 144 f. Dissertação (Mestrado) - Curso de Sistemas de Gestão, Departamento de Engenharia de Produção, Universidade Federal Fluminense, Niterói, 2016. Cap. 2.

GONÇALVES, Elton. **ANÁLISE DAS PRÁTICAS DE GERENCIAMENTO DE PROJETOS EM DESENVOLVIMENTO DE SOFTWARE.** 2016. 112 f. Dissertação (Mestrado) - Curso de Sistemas de Gestão, Universidade Federal Fluminense, Niterói, 2016.

LACERDA; ENSSLIN, L.; ENSSLIN, S R. UMA ANÁLISE BIBLIOMÉTRICA DA LITERATURA SOBRE ESTRATÉGIA E AVALIAÇÃO DE DESEMPENHO. **Gestão & Produção**, v. 19, n. 1, 2012.

LAYTON, Mark C; OSTERMILLER, Steven J. **AGILE PROJECT MANAGEMENT FOR DUMMIES.** 2. ed. New Jersey: John Wiley & Sons, Inc., 2017. 435 p.

LUDVIG, Diogo; REINERT, Jonatas Davson. **ESTUDO DO USO DE METODOLOGIAS ÁGEIS NO DESENVOLVIMENTO DE UMA APLICAÇÃO DE GOVERNO ELETRÔNICO.** 2007. 79 f. TCC (Graduação) - Curso de Sistemas de Informação, Departamento de Informática e Estatística, Universidade Federal de Santa Catarina, Florianópolis, 2007.

MARCONDES, Aníbal. **UMA BREVE HISTÓRIA DO GERENCIAMENTO DE PROJETOS.** 2017. Disponível em: <http://pmimt.org.br/site/index.php/artigo/vis/4>. Acesso em: 02 nov. 2018.

MORAES, A. J. M. DE. **CUSTOMIZATION OF PROJECT MANAGEMENT TECHNIQUES FOR THE CONSTRUCTION OF IT - INFORMATION TECHNOLOGY SYSTEMS WITH THE DEVELOPMENT METHODOLOGIES KNOWN AS AGILE PROCESSES** (C. N.C. et al., Eds.)20th World Multi-Conference on Systemics, Cybernetics and Informatics, WMSCI 2016. **Anais**...Brasília, DF 73252-200, Brazil: International Institute of Informatics and Systemics, IIIS, 2016

PATAH, L.; PATAH, L. A.; CARVALHO, M. M. DE. MÉTODOS DE GESTÃO DE PROJETOS E SUCESSO DOS PROJETOS: Um Estudo Quantitativo do Relacionamento entre estes Conceitos. **Revista de Gestão e Projetos - GeP**, v. 3, n. 2, p. 178–206, 1 ago. 2012.

PEREIRA, Carlos Diego Cavalcanti. **X-PRO (EXTREME SOFTWARE PROCESS):** Um framework para desenvolvimento eficiente de software baseado em metodologias ágeis. 2014. 167 f. Dissertação (Mestrado) - Curso de Ciência da Computação, Centro de Informática,

Universidade Federal de Pernambuco, Recife, 2014.

PROJECT MANAGEMENT INSTITUTE. **UM GUIA DO CONHECIMENTO EM GERENCIAMENTO DE PROJETOS**. Guia PMBOK® 5nd. ed. Estados Unidos: Project Management Institute, 2013.

PROJECT MANAGEMENT INSTITUTE, T. **SOFTWARE EXTENSION TO THE PMBOK® GUIDE**. 2013.

RISING, L.; JANOFF, N. S. THE SCRUM SOFTWARE DEVELOPMENT PROCESS FOR SMALL TEAMS. **IEEE Software**, v. 17, n. 4, p. 26-32, 2000.

RUBIN, Kenneth S. **ESSENTIAL SCRUM:** A Practical Guide to the most popular agile process. Ann Arbor: Pearson Education, 2012. 495 p.

SANTOS, Danillo Moura. **PROJETO DE SISTEMAS EMBARCADOS**: Um estudo de caso baseado em microcontrolador e seguindo AOSD. 2002. 54 f. TCC (Graduação) - Curso de Ciências da Computação, Centro Tecnológico, Universidade Federal de Santa Catarina, Florianópolis, 2006. Cap. 1

SILVEIRA, J. K. DA. **PROPOSTA DE METODOLOGIA DE GESTÃO DE PROJETOS DE NOVOS PRODUTOS PARA UMA EMPRESA DESENVOLVEDORA DE SOLUÇÕES TECNOLÓGICAS**. [s.l: s.n.].

VARGAS, Ricardo. **GERENCIAMENTO DE PROJETOS.** 6. ed. Rio de Janeiro: Brasport, 2005.

VASCONCELOS, Alexandre Meira de. **DIMENSÕES, COMPONENTES E ITENS DE AVALIAÇÃO DA GESTÃO DA QUALIDADE EM SERVIÇOS TURÍSTICOS.**2014. 195 f. Tese (Doutorado) - Curso de Engenharia de Produção, Centro Tecnológico, Universidade Federal de Santa Catarina, Florianópolis, 2014.

VAVASSORI, Fabiane Barreto. **METODOLOGIA PARA O GERENCIAMENTO DISTRIBUÍDO DE PROJETOS E MÉTRICA DE SOFTWARE.** 2002. 106 f. Tese (Doutorado) - Curso de Pós-graduação em Engenharia de Produção, Universidade Federal de Santa Catarina, Florianópolis, 2002.

ZAVAN, André Ricardo. **PRODES – UM PROCESSO DE DESENVOLVIMENTO DE SOFTWARE BASEADO NO SCRUM E FDD.** 2013. 140 f. Dissertação (Mestrado) - Curso de Engenharia de Software, Centro de Estudos e Sistemas Avançados do Recife, Recife, 2013.

ZUFFO, João Antonio et al. **UMA BREVE HISTÓRIA DOS SISTEMAS ELETRÔNICOS.** São Paulo: -, 2018. 12 p. PSI-2222 Práticas de Eletricidade e Eletrônica II.

YOSHIDA, Nelson D. **ANÁLISE BIBLIOMÉTRICA**: um estudo aplicado à previsão tecnológica. Future Studies Research Journal, v. 2, n. 1, p. 52-84, 2010