UNIVERSIDADE NOVE DE JULHO
PROGRAMA DE MESTRADO PROFISSIONAL EM ADMINISTRAÇÃO
GESTÃO DE PROJETOS

APLICAÇÃO DO *DESIGN THINKING* INTEGRADO COM MÉTODOS ÁGEIS NA GESTÃO DE PROJETOS DE *SOFTWARE*

JULIO CESAR PEREIRA

São Paulo

2018

Julio Cesar Pereira

APLICAÇÃO DO *DESIGN THINKING* INTEGRADO COM MÉTODOS ÁGEIS NA GESTÃO DE PROJETOS DE *SOFTWARE*

DESIGN THINKING APPLIED INTEGRATED IN AGILE SOFTWARE DEVELOPMENT PROJECTS

Dissertação apresentada ao Programa de Mestrado Profissional em Administração: Gestão de Projetos da Universidade Nove de Julho – UNINOVE, como requisito parcial para obtenção do grau de **Mestre em Administração**.

Orientador: Profa. Dra. Rosária de Fátima Segger Macri Russo

São Paulo

2018

Julio Cesar Pereira

APLICAÇÃO DO *DESIGN THINKING* INTEGRADO COM MÉTODOS ÁGEIS NA GESTÃO DE PROJETOS DE *SOFTWARE*

Dissertação apresentada ao Programa de Mestrado Profissional em Administração: Gestão de Projetos da Universidade Nove de Julho – UNINOVE, como requisito parcial para obtenção do grau de **Mestre em Administração**, pela Banca Examinadora, formada por:

São Paulo, 07 de dezembro de 2018

Presidente: Profa. Dra. Rosária de Fátima Segger Macri Russo – Orientadora, UNINOVE

Membro: Profa. Dra. Cristiane Drebes Pedron – UNINOVE

Membro: Profa. Dra. Sabrina dos Santos Marczak – PUCRS

Pereira, Julio Cesar.
Aplicação do design thinking integrado com métodos ágeis na gestão de projetos de software. / Julio Cesar Pereira. 2018.
152 f.
Dissertação (Mestrado) - Universidade Nove de Julho - UNINOVE, São Paulo, 2018.
Orientador (a): Profª. Drª. Rosária de Fátima Segger Macri Russo
1. Sistemas da informação. 2. Software. 3. Métodos ágeis. 4. Design thinking. 5. Gestão de projetos.
I. Russo, Rosária de Fátima Segger Macri. II. Titulo.

CDU 658.012.2

DEDICATÓRIA

Dedico à
minha família que é o maior bem que possuo:
Ana Carolina, Julio Cesar e Ana Helena.
À minha mãe que me ensinou o que é
amor, humildade e gratidão. Sempre.

AGRADECIMENTO

Agradeço a todas as pessoas que direta ou indireta estiverem ao meu lado, apoiando e tornando possível a realização deste trabalho. A minha família estendida que esteve à distância, mas permaneceu desejando meu êxito e compreendendo a minha ausência nos eventos.

Agradeço a todos os professores do programa do Mestrado Profissional em Administração - Gestão de Projetos da Universidade Nove de Julho, que puderam compartilhar o conhecimento adquirido ao longo da jornada do curso. Agradeço em especial a Professora Dra. Rosária de Fátima Macri Segger Russo, que acompanhou de perto todos os passos com apoio e motivação incondicionais.

Agradeço aos colegas do ano de 2017 do programa de Mestrado Profissional em Administração – Gestão de Projetos que talharam horas de estudo e ainda tinham tempo para compartilhar o conhecimento e superar os desafios à medida em que se apresentavam. Agradeço em especial à Lívia Macêdo de Alencar que esteve junto comigo em vários projetos de pesquisa, auxiliando sempre que eu recorria a sua ajuda.

Aos parceiros que tornaram esse trabalho possível: Marcello Dondé Hazan, Fabiano Vieira, Richard Rossi, André Endres, Marcelo Ingarano, André Vidal, Fábio do Carmo Ferreira, Alexandre Enk, Roberto Góes, Rodrigo Muniz e Alexandre Barboza.

E a todos os demais que direta ou indiretamente depositaram confiança no que parecia um desafio intransponível. Não estarem aqui nominalmente não diminui minha gratidão a cada um de vocês: muito obrigado.

RESUMO

A aplicação de um método ágil em projetos de desenvolvimento de *software* promove a entrega rápida de novas funcionalidades por meio de uma melhor gestão de mudanças na priorização das atividades e aumento da produtividade da equipe responsável pelo *software*. Ao longo da execução dos projetos de desenvolvimento do *software* espera-se que as necessidades dos usuários sejam completamente entendidas pelo time, porém a proximidade entre clientes e executores pode ser dificultada por restrições geográficas, cronograma enxuto e restrições orçamentárias. Por sua vez, o *Design Thinking* (DT) é uma abordagem que promove o entendimento das necessidades dos usuários, considerando o que é tecnicamente e economicamente viável simultaneamente. O propósito desta pesquisa é avaliar como o DT pode ser integrado com métodos ágeis em projetos de desenvolvimento de *software*. O método de pesquisa escolhido foi o *Design Science Research* que utilizou entrevistas e grupos focais na coleta de dados, permitindo uma análise qualitativa dos dados. Para identificar os problemas existentes, a pesquisa teve início com entrevistas em uma empresa que utiliza a abordagem de gestão ágil em projetos de *software* e avaliar se o uso de um método que integra DT e métodos ágeis iria solucionar tais problemas. Embora não tenha sido encontrado um método que integra o DT com métodos ágeis, foram encontrados diversos modelos que foram detalhadamente analisados, gerando uma proposta de método para resolver os problemas identificados na empresa que desenvolve *softwares*, considerando a perspectiva do cliente no desenho e validação da solução final. O método que integra a abordagem DT e métodos ágeis foi desenvolvido e recebeu o nome de *Agile Design*. Ele promoverá a aproximação dos clientes, usuários e time do projeto, envolvendo-os em várias etapas dos projetos de desenvolvimento de *softwares* e, assim, habilitará a entrega de produtos com melhor qualidade e maior nível de satisfação dos usuários A contribuição teórica foi prescrever um método a partir dos aspectos encontrados nos modelos já existentes e possibilitar o uso em projetos de desenvolvimento de *softwares* executados por diferentes organizações. Sendo assim, o método proposto consiste em uma contribuição para a prática voltado às organizações que enfrentam problemas nos projetos de desenvolvimento de software para que sejam flexíveis, ágeis e que satisfaçam as necessidades de seus clientes e dos usuários do *software*.

Palavras-chave: Sistemas da Informação; *Software*; Métodos Ágeis; *Design Thinking*; Gestão de Projetos.

ABSTRACT

The use of an agile method in software development projects looks for a faster delivery of new functionalities through a better change management and increasing the productivity of the team responsible for the software. During the project software development life cycle, the development team is expected to fully understand customers' needs, but it is challenged by geographic constraints, narrow project schedule, or short budget. Design Thinking (DT) is an approach that promotes the understanding of customer needs, considering what is technically and economically feasible. The purpose of this research is to evaluate how DT can be integrated with agile method in software development projects. Design Science Research was the research method chosen and interviews and focus groups have been used to collect data, making it possible to develop a qualitative analysis. The research began with interviews in a software development company to identify existing problems when the agile management had been used in software projects and assess whether using a method that integrates DT approach and agile methods would solve them. Although a method that integrates DT with agile methods was not found in the literature, several models were found that were analyzed in detail, generating the proposal of a method to solve the problems investigated in the company which has software development as a main business, considering both contractor and client perspective. Agile Design is the name of the method created in this research. It will promote the approach of customers, users and project team, involving them in various stages of software development projects. In this way, the integrated approach pursues to enhance the quality of the software delivered, and consequently a higher level of user satisfaction. The theoretical contribution is to prescribe a method based on the aspects founded in the existing models and enable the use in software development projects performed by different organizations. Hence, the proposed method consists in a practical contribution aimed to help organizations facing problems in software development projects, directing it to be flexible, agile and to satisfy the needs of its clients and software users.

Keywords: Information Systems; Software; Agile Methodology; Design Thinking; Project Management.

LISTA DE ABREVIATURAS E SIGLAS

API - *Application Programming Interface* (Interface de programação de aplicações)

DevOps – *Development and Operations* (Desenvolvimento e operação)

DSR – *Design Science Research*

DT – *Design Thinking*

HCD – *Human Centered Design* (*Design* centrado no humano)

MVP - *Minimum Viable Product* (Menor produto viável)

PBS – *Product Breakdown Structure* (Estrutura Analítica do Produto)

PO – *Product Owner* (Dono do Produto)

ROI – *Return on investment* (Retorno sobre investimento)

SAFe – *Scaled Agile Framework*

SI – Sistema da Informação

SM – *Scrum Master*

TI – Tecnologia da Informação

UX – *User Experience* (Experiência do usuário)

XP – *Extreme programming* (Programação extrema)

LISTA DE TABELAS

LISTA DE FIGURAS

SUMÁRIO

1 INTRODUÇÃO

As organizações buscam melhorar seus processos, atender seus clientes com maior agilidade e flexibilidade e, assim, serem reconhecidas como provedoras de produtos e serviços de qualidade, resultando em diferencial competitivo em relação aos concorrentes. O gerenciamento da Tecnologia da Informação (TI) necessita de estruturas de comando e controle que se adequem ao rápido mundo digital de hoje (Bossert, Kretzberg, & Laartz, 2018) e que permita implantar novos sistemas da informação para melhor desempenho e eficiência das organizações (Hevner, March, Park, & Ram, 2004). O desenvolvimento de novos sistemas consiste na aplicação da TI como ferramenta estratégica de inovação, apontada como crítica pela maior parte das organizações (Curran, Garrett, & Puthiyamadam, 2017). É notado, em contrapartida, que a ausência de gestão da TI pode levar algumas organizações ao colapso (Dwivedi et al., 2015).

A oferta de produtos e serviços com uso da TI tem ganhado maior relevância pelo fato das pessoas terem acesso instantâneo ao mercado global e diante do aumento da expectativa em relação aos serviços prestados (Sheppard, Edson, & Kouyoumjian, 2017). Deste modo, a estratégia organizacional tem considerado o uso massivo da internet e de sistemas informatizados com ênfase na vantagem competitiva frente aos concorrentes e para o melhor direcionamento dos negócios da organização (Davenport, 1993; Owens & Khazanchi, 2018). Sete em cada 10 executivos informam que a aplicação da TI beneficia o crescimento da receita nas organizações (Curran et al., 2017). Neste contexto, a organização necessita identificar a área de aplicação da TI, aperfeiçoar o modelo de gestão e reconhecer as necessidades dos usuários (Mustonen-Ollila & Lyytinen, 2003), aliando simultaneamente agilidade e qualidade ao implantar um sistema de informação (SI).

Um SI consiste em uma variedade de tecnologias, tais como computadores, *softwares*, bancos de dados, sistemas de comunicação, internet e dispositivos móveis para auxílio às tarefas requeridas por usuários ou organizações (Boell & Cecez-Kecmanovic, 2015). Uma parte importante dessas tecnologias correspondem aos sistemas informatizados, conhecidos como *softwares*, que são projetados e dedicados para auxiliar na execução de trabalhos por seus usuários, tendo propósitos específicos e orientados a tarefas (Smith & Aucella, 1983). Os *softwares* atualmente desempenham um papel fundamental no cotidiano das pessoas e, por esse motivo, empresas de desenvolvimento de *software* nos Estados Unidos investem

anualmente mais de 50 bilhões de dólares em pesquisa e desenvolvimento neste segmento (BSA, 2016).

Associado à sua relevância, no contexto organizacional, a atividade de implantação de *software* possui alta complexidade pela combinação de conhecimentos específicos de diferentes organizações, grupos funcionais, unidades de negócio e subcontratados (Shenhar & Dvir, 2010). O envolvimento direto dos usuários na etapa de modelagem do *software* possibilita a conversão das necessidades apontadas pelos clientes em uma solução integrada (Piccoli & Ives, 2005; Shenhar & Dvir, 2010). As características do ambiente, como, por exemplo, o tamanho do time de projetos, complexidade organizacional, questões regulatórias e distribuição geográfica influenciam no modelo ideal para que a gestão do projeto seja feita adequadamente (PMI, 2017c). Portanto, a complexidade envolvida nos projetos requer que o método de trabalho escolhido melhore a eficiência de como as atividades são executadas e controladas. Sauser, Reilly e Shenhar (2009) mencionam que, em alguns casos, o motivo pelo qual os projetos fracassam não é evidenciado, bem como não é estabelecido o que deveria ser feito em relação à escolha do modelo de gestão de projetos mais adequado. Como as organizações não realizam apenas um projeto, é necessário que seja adaptado o modelo que melhor se adequa a melhor abordagem para o projeto correto (Sauser, Reilly, & Shenhar, 2009). As organizações, como forma de se adaptar ao cenário atual, têm optado por adotar os métodos ágeis em projetos de desenvolvimento de *softwares*, apresentando aumento ao longo dos últimos anos (PMI, 2017b) em virtude de tempo e esforço gastos excessivamente em planejamento não assegurarem um projeto bem-sucedido (Serrador & Pinto, 2015).

A aplicação de métodos ágeis na gestão de projetos permite disponibilizar o *software* para uso por meio de ciclos curtos de entrega visando menor tempo entre a concepção do *software* e o mercado (Steinke, Al-deen, & Labrie, 2017). Adicionalmente, promove testes das entregas parciais e com maior agilidade se comparado com outros métodos clássicos de gestão de projetos pautados em planejamento sofisticado antes de seu início (Steinke et al., 2017). Ao considerar a perspectiva organizacional, os métodos ágeis promovem a colaboração entre as partes interessadas nos projetos de desenvolvimento de *software* com transparência e maior percepção de valor por parte do cliente, além de melhorar a qualidade do produto e reduzir o tempo de resposta no tratamento de possíveis erros (Larson & Chang, 2016; Solinski & Petersen, 2016). Sendo assim, a organização necessita criar mecanismos que aliem as vantagens decorrentes do emprego dos métodos ágeis e melhorar a forma de como as necessidades dos usuários do *software* a ser desenvolvido serão obtidas. Propor soluções a

partir de clientes e usuários finais por meio da empatia é uma das características da abordagem *Design Thinking* (DT).

O DT ou Design Centrado no Humano (HCD, do inglês *Human Centered Design*) é uma abordagem que atua a partir da aplicação do pensamento do *designer* para identificação de problemas e construção de soluções desejáveis para os clientes, tecnologicamente factíveis e viáveis para os negócios da organização (Brown, 2008). Neste contexto, o *designer* pode ser definido como responsável pelo desenho detalhado da solução ou do produto, sendo neste caso, o *software* que será desenvolvido. De acordo com Plattner, 2009, a utilização do DT pode favorecer o entendimento de um novo produto ou serviço desde sua concepção até a entrega final por meio dos processos de (i) entendimento do significado que as pessoas possuem de um assunto por meio da empatia; (ii) definição e contextualização do desafio do design; (iii) captura do que é relevante para quem irá receber o resultado ou objeto daquele projeto por meio da ideação; (iv) prototipação rápida das ideias geradas, e por fim; (v) teste dos protótipos com os usuários e clientes. A aplicação do DT auxilia as organizações na entrega de produtos e serviços e melhora a comunicação com os usuários e clientes finais (Brown, 2008). Adicionalmente, sua aplicação aumenta a satisfação dos usuários em relação à usabilidade do *software* entregue quando o DT passa a ser utilizado de forma integrada com métodos ágeis (Lucena, Braz, Chicoria, & Tizzei, 2016).

1.1 PROBLEMATIZAÇÃO

Os projetos de desenvolvimento de *software*, em função de sua complexidade e incerteza, podem apresentar atrasos ou entregar algo aquém das expectativas das partes interessadas em decorrência da baixa previsibilidade dos requisitos legítimos do projeto ou por não endereçar adequadamente as mudanças quando elas ocorrem (PMI, 2017a). Normalmente os problemas dos *softwares* desenvolvidos estão associados à incapacidade do próprio SI em automatizar alguma atividade e gerar o resultado esperado pelo usuário, visto que a atividade de planejamento e gestão de um projeto de *software* é uma atividade complexa (Shenhar & Dvir, 2010). Deve-se, então, garantir que todas as partes de um SI, desde as pessoas que operacionalizam o *software* até as funcionalidades técnicas que o compõem, estejam adequadamente interligadas para o seu perfeito funcionamento.

Além de o *software* procurar atender as especificações das funcionalidades técnicas, as organizações privadas e públicas bem-sucedidas utilizam a etapa de desenho e especificação do *software* para traduzir a tecnologia em valor para o cliente (Gruber, Leon, George, & Thompson, 2015). Antigamente o sucesso de um *software* estava associado à capacidade do mesmo ser distribuído e chegar ao usuário final, porém hoje é determinante que o *software* atenda às necessidades dos usuários (Lucena et al., 2016) e também do negócio. Portanto, as atividades do projeto de *software* devem estar diretamente relacionadas aos objetivos do projeto e do produto a ser entregue ao cliente. Nos próximos anos, as organizações orientadas a projetos passarão a destinar recursos para (i) melhorar as relações com os clientes e (ii) aumentar sua eficiência operacional, classificando-as como altas prioridades (PMI, 2017b). Sendo assim, as organizações passarão a dar maior relevância ao atendimento das expectativas das partes interessadas, além das preocupações sobre as questões técnicas envolvidas nos projetos.

Porém, nota-se que, ao longo dos anos, as implantações de *softwares* não têm sido bem-sucedidas, inclusive com a percepção dos executivos de que as taxas de insucesso em projetos de TI permanecem nos patamares da última década (Standish Group, 2014). De acordo com a consultoria independente em pesquisas de TI, Standish Group (2014), mais de 30% dos projetos são cancelados antes de seu término e mais da metade consome quase o dobro do investimento estimado inicialmente (Standish Group, 2014). Em complemento a tal informação, metade dos *softwares* criados quase nunca são utilizados e apenas 20% são usados com frequência (Standish Group, 2013), o que aponta falhas na identificação dos requisitos para a entrega de novos *softwares*.

Diante da complexidade do processo de implantação de *softwares* e da integração organizacional, Shenhar e Dvir (2010) sugerem que o desenvolvimento de *softwares* ocorra com: (i) o entendimento e a articulação das perspectivas das partes interessadas; (ii) a avaliação dos *softwares* como um todo (e não como coletânea de subsistemas); (iii) a integração das funcionalidades, e; (iv) o envolvimento do usuário para moldar o *software*. Associado ainda à competitividade entre empresas, as organizações necessitam responder rapidamente às mudanças das necessidades de negócio conforme elas surjam (Farid, Helmy, & Bahloul, 2017), visto que a atividade de implantação de um sistema empresarial quando não se entende as implicações no negócio pode resultar em falhas (Davenport, 1998). Com intuito de reduzir falhas e aumentar o sucesso na implantação, é indicado que o tamanho e a complexidade do projeto de *software* sejam reduzidos em pequenas entregas, aumentando os

benefícios do projeto em 80% e reduzindo o custo incorrido ao longo de sua execução (Standish Group, 2013).

A gestão do projeto, portanto, deve dar maior ênfase na definição e distribuição das atividades aos membros do time do projeto em relação aos aspectos internos de gestão do projeto em si, como orçamento e cronograma. Nota-se que os gerentes, de um modo geral, necessitam desenvolver uma melhor capacidade de comunicação e resolução de problemas, além das competências de planejamento e tomada de decisão (Bossert et al., 2018). Deste modo, é necessário promover o envolvimento das partes interessadas, como membros do time, clientes e fornecedores durante todo o ciclo de vida do projeto por meio de um processo colaborativo e iterativo que descubra as necessidades, proponha soluções e gere rápidos protótipos antes da entrega final (Gruber et al., 2015). Ou seja, durante o desenvolvimento de um *software*, a equipe do projeto deve possibilitar que os usuários avaliem o resultado final por meio de protótipos. Sendo assim, o emprego de métodos ágeis promove rápidas adaptações nas condições do projeto quando ocorrem mudanças em requisitos funcionais e também nas prioridades de negócios (Pieroni, Scarpato, & Scorza, 2018).

Os métodos ágeis que estejam integrados com DT preveem a aproximação do time de desenvolvimento e dos clientes de maneira efetiva ao longo do ciclo do desenvolvimento do *software*. De acordo com o PMI (2017b), a maioria das empresas tem adotado métodos ágeis na condução dos projetos, porém permanecem com questões a serem endereçadas. Uma em cada quatro iniciativas estratégicas falham em função de ausência de clareza nos objetivos (37%) e 19% das empresas apontam problemas na comunicação (PMI, 2017b). Autores defendem que a integração da abordagem DT com métodos ágeis pode gerar melhores resultados. Este fato está associado à capacidade de desenvolver *softwares* que se propõem a solucionar problemas complexos, mesmo quando estejam mal especificados (Newman et al., 2015) e pela proximidade da área de negócio com o time técnico na construção do produto por meio de ciclos rápidos (Góes & Russo, 2018). Além disso, o uso do DT integrado com métodos ágeis busca orientar o time de desenvolvimento para que se concentre nas necessidades dos usuários durante a construção de produtos e serviços que melhor se adequem às necessidades do cliente (Steinke et al., 2017).

Os métodos de desenvolvimento de *software* que integram as abordagens DT e métodos ágeis devem considerar o contexto da cultura organizacional, o ambiente externo e as abordagens utilizadas no desenvolvimento dos sistemas (Darrin & Devereux, 2017). Por este motivo, pesquisas complementares são sugeridas para avaliar de maneira mais precisa o nível de satisfação dos usuários que recebem o *software* que adota um modelo que integrada o DT

com métodos ágeis (Lucena et al., 2016) e a avaliação da agilidade em diferentes contextos de projetos (Conforto, Amaral, da Silva, Di Felippo, & Kamikawachi, 2016). Portanto, diante dos problemas identificados no desenvolvimento de *softwares*, como falhas na comunicação, o não atendimento às expectativas das partes interessadas e pouco envolvimento dos usuários finais, a questão de pesquisa é: **Como o *design thinking* pode ser integrado com métodos ágeis em projetos de desenvolvimento de *softwares*?**

1.2 OBJETIVOS

A entrega de *softwares* com qualidade e agilidade tem sido um desafio para as organizações que buscam melhorar a percepção dos clientes que utilizam os sistemas desenvolvidos. O objetivo principal desse estudo é avaliar como o DT pode ser integrado com métodos ágeis em projetos para o desenvolvimento de *software*.

Os objetivos específicos são:

a. Identificar quais métodos utilizam a abordagem DT integrada com métodos ágeis em projetos de desenvolvimento de *software*.
b. Identificar os problemas existentes no uso exclusivo de métodos ágeis em projetos de desenvolvimento de *software* em uma empresa de TI.
c. Propor um método que integra a abordagem DT e métodos ágeis para uso em projetos de desenvolvimento de *software*.
d. Validar o método proposto que integra a abordagem DT e métodos ágeis no contexto de uma empresa de TI.

1.3 JUSTIFICATIVA

Nos Estados Unidos, são investidos mais de 250 bilhões de dólares por ano no desenvolvimento de *softwares*, distribuídos em aproximadamente 175 mil projetos (Standish Group, 2014). A falha em um *software* implantado pode resultar em perdas financeiras significativas, como ocorreu na implantação de um novo *software* para gerenciar o manuseio das bagagens no aeroporto de Denver, nos Estados Unidos (Standish Group, 2014). Dado o volume expressivo de investimento para este segmento, novas práticas para a gestão de projetos começaram a ser utilizadas. Recentemente, a maior parte das organizações passaram

a utilizar métodos ágeis para a gestão de projetos (PMI, 2017b), sendo que 84% das organizações apresentam baixa maturidade na adoção desses métodos no desenvolvimento de *softwares* (VersionOne, 2018). Deste modo, embora o uso de métodos ágeis possa ser eficaz para a maioria das organizações (VersionOne, 2018), ainda não assegura que a entrega do *software* desenvolvido seja bem-sucedida sob a perspectiva do usuário que o recebe.

Os resultados práticos dos projetos que utilizaram métodos ágeis ao longo do desenvolvimento de *software* nem sempre estão disponíveis para os pesquisadores (Conboy & Fitzgerald, 2010), impedindo avaliar quais atividades da gestão ágil podem ser generalizáveis para outros projetos. Como as pesquisas disponíveis normalmente referenciam práticas ágeis usadas em projetos de forma individual não é possível assegurar que houve redução de erros no *software* entregue, bem como evidencia quais as razões da ineficácia na adoção de métodos ágeis no desenvolvimento de *softwares* (Vallon, da Silva Estácio, Prikladnicki, & Grechenig, 2018). Pela mesma razão, ou seja, pelas pesquisas abordagem o uso de métodos ágeis em projetos exclusivos, não há associação da prática de métodos ágeis com o atingimento dos objetivos da organização no médio e longo prazo (Solinski & Petersen, 2016). Adicionalmente, não é possível garantir que a comunicação entre o time do projeto e os clientes foram corretamente conduzidos nas organizações que utilizaram tal método.

Sob a perspectiva organizacional, os métodos ágeis promovem a colaboração entre as partes interessadas no projeto do *software*, promovendo maior transparência e valor percebido pelo cliente, melhor qualidade no resultado do projeto e redução do tempo para endereçamento de possíveis erros (Larson & Chang, 2016; Solinski & Petersen, 2016). Diante da baixa maturidade na gestão ágil de projetos apontada pela pesquisa da Version One (2018), empresa que desenvolve *softwares* com uso de métodos ágeis, é possível inferir que sua aplicação ainda não resolve questões como a gestão de times distribuídos e nem a redução dos custos do projeto. Buscando resolver estas questões, as organizações devem buscar a criação de mecanismos que associem os desejos e necessidades dos consumidores no processo de desenvolvimento de novos *softwares*.

Além do emprego de métodos ágeis em projetos de desenvolvimento de *software*, nota-se que as organizações necessitam antecipar a identificação das características que devem compor um *software* antes que o projeto seja iniciado. Uma das alternativas para buscar essa antecipação pode ser o uso da abordagem DT, que possui como principais objetivos: (i) explorar ideias no início do projeto antes de dar algum direcionamento; (ii) observação direta para a geração de ideias e; (iii) rápida experimentação por meio de protótipos (Brown, 2008). Adicionalmente, a entrega de resultados inovadores associando

aspectos racionais e cognitivos simultaneamente é resultado de uma comunicação eficaz entre o time do projeto e os usuários finais, uma das principais características da abordagem DT (Martin, 2009).

Portanto, esta pesquisa visa avaliar como o DT pode ser integrado com métodos ágeis em projetos de desenvolvimento de *software*, conforme sugerido por Higuchi & Nakano (2017). O método *Design Science Research* foi escolhido por ser um paradigma de pesquisa no segmento de TI, onde SI está incluso, possibilitando a construção de artefatos tais como sistemas de apoio à decisão, estratégia de governança, métodos para avaliação de SI e intervenções de mudanças em SI (Gregor & Hevner, 2013). Além disso, visa promover a aproximação entre a teoria e a prática, ou seja, academia e sociedade (De Sordi, Meireles, & Sanches, 2011). Ao mesmo tempo que possui procedimentos metodológicos que permitem assegurar a relevância da pesquisa, o método permite que um artefato seja gerado e aplicado em projetos reais. Tais projetos correspondem a gestão de *softwares* desenvolvidos em uma empresa de TI especializada em sistemas.

1.4 ESTRUTURA DO TRABALHO

Este trabalho está estruturado em seis capítulos. O Capítulo 2 contém o referencial teórico e está subdivido em três partes: *Design Thinking*, métodos ágeis para a gestão de projetos e a integração do DT com métodos ágeis em projetos de desenvolvimento de *software*. O Capítulo 3 aborda os procedimentos metodológicos da pesquisa, considerando como a pesquisa será executada, os procedimentos de coleta, de análise dos dados e as limitações. O Capítulo 4 apresenta os resultados por meio das informações coletadas ao longo da pesquisa. Por fim, o Capítulo 5 aborda as contribuições teóricas e práticas da pesquisa e o Capítulo 6 relata as considerações finais da pesquisa, tais como conclusão, limitações e sugestão de pesquisas futuras.

2 REFERENCIAL TEÓRICO

Este capítulo aborda os temas desta pesquisa com base em publicações e estudos realizados anteriormente nas áreas de administração, TI e gestão de projetos. Com o objetivo de suportar o presente estudo, a construção do referencial teórico está subdividido em: *Design Thinking*, Métodos Ágeis e a integração do *Design Thinking* com métodos ágeis em gestão de projetos.

2.1 DESIGN THINKING

A abordagem DT permite que potenciais soluções para o problema estudado sejam definidas, proporcionando melhores produtos e serviços, aumento da produtividade e melhorias operacionais (Gruber et al., 2015). Busca por meio do *design* centrado no humano capturar o que é desejável para as pessoas, tecnicamente possível e viável para os negócios, convertendo algo em benefício para o cliente e ao mesmo tempo de valor para o negócio da organização (Brown, 2008). O conceito do *design* centrado no humano, incorporado na abordagem DT, refere-se à ênfase na observação e na descoberta das necessidades humanas frequentemente tácitas e diferenciadas entre as diferentes pessoas envolvidas no processo de inovação (Gruber et al., 2015).

Um dos pontos chave da abordagem DT é a representação visual para que a ideia em desenvolvimento se torne tangível e aceita, assegurando que os envolvidos reconheçam a saída, potencial futuro resultado, conforme imaginado na etapa de criação (Brown, 2008; Plattner, 2009). A presença de mudanças ao longo de todo o processo é estimulada e permite agregar *feedback* de cada fase para as etapas seguintes do processo de decisão, uma vez que as fases das abordagens DT são cíclicas e iterativas. Estas fases possuem variações entre os modelos propostos por diversos autores (Brown, 2008; Design Council, 2018; DIS, 2010; Google, 2018; Kolko, 2018; Plattner, 2009). Ao longo desta sessão a abordagem DT será relatada em seu contexto para a captura das necessidades por meio da observação, empatia e prototipação para a entrega de produtos e serviços.

Na primeira etapa da abordagem DT busca-se a inspiração ou definição do problema a ser resolvido. Neste momento os participantes buscam entender o problema e as oportunidades que se apresentam, ou seja, a circunstância em que ocorre a questão, o mistério a ser resolvido, com foco nas pessoas e nos seus comportamentos (Giff & Dogan, 2016).

Martin (2009) afirma que uma solução completa só é possível quando se entende verdadeiramente o usuário. Nesta fase inicial são geradas as ideias que podem desencadear a solução para o problema que se deseja resolver (Brown, 2008; Plattner, 2009). A abordagem sugere o engajamento direto com as pessoas na busca pelo real entendimento do que sentem e buscam por meio da empatia. O processo de *design* complementa-se com a geração de uma declaração sintetizada que concentra as ideias e necessidades dessas pessoas

Adicionalmente, Plattner (2009) cita, como exemplo, que os processos de observação e engajamento por meio da empatia são entradas para a etapa de definição, quando toda a coleta de informação resultará nas impressões e informações sobre o seu usuário. Nesta etapa também são avaliados os problemas não estruturados comuns no mundo real (Dorst, 2006) para a proposição e escolha das soluções previstas nas etapas seguintes.

A etapa intermediária da abordagem DT remete a ideia de criação de cenários por meio da criatividade e pensamento integrador, buscando sempre colocar o cliente no centro das ideias (Brown, 2008). Os pontos levantados durante a inspiração são adicionados e a partir deles são geradas novas ideias que, por sua vez, são analisadas, criticadas e incrementadas, com ênfase na solução e na tradução de requisitos em objetivos de modo construtivo (Cross, 1982). Ressalta-se a importância da prototipação na geração de ideias (Brown, 2008), permitindo que os problemas identificados se convertam em soluções para os usuários (Plattner, 2009). A ideia é que, ao final desta etapa, seja obtido o maior número de possíveis soluções para um determinado problema,

A etapa seguinte reserva-se aos testes e implementações das ideias geradas nas etapas anteriores com o apoio dos protótipos. Plattner (2009) estabelece que os testes são oportunidades de apreender sobre a solução e seu cliente. Neste momento é avaliado o enquadramento das ideias coletadas dos clientes por meio da apresentação de uma solução proposta (Plattner, 2009), observando o direcionamento de sua aplicação ao negócio e divulgando-a ao mercado (Brown, 2008). Portanto, ao final desta etapa, a melhor solução dentre as previamente identificadas é escolhida, observando sempre a expectativa do cliente diante desta decisão.

Embora a abordagem do DT considere etapas sequenciais nos modelos disponíveis na literatura, deve-se entender que existem transições entre as etapas de aplicação da abordagem do DT (Plattner, 2009). A Figura 1 ilustra as cinco etapas da abordagem DT promovida pela Escola de Design de Stanford. A etapa da empatia consiste na fase onde a captura do entendimento do problema, sendo entrada para a etapa de definição dos desafios e problemas que serão solucionados. Na etapa de ideação todas as ideias são encorajadas para que o maior

número de possíveis soluções seja estabelecido, para que então sejam rapidamente construídos protótipos, e assim, sejam conduzidos os testes sobre o que realmente funciona sob a perspectiva dos clientes ou usuários.

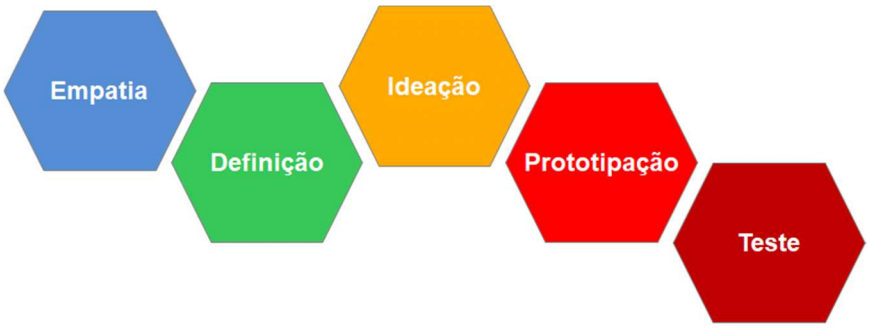

Figura 1. Processo de Design Thinking – Escola de Design de Stanford
Fonte: Adaptado de Plattner (2009, p. 6).

É necessário observar que a abordagem DT é iterativa e pode ser repetida sempre que novos componentes sejam identificados no processo decisório de prosseguimento ou não para atingir o resultado esperado. Durante a fase de empatia, presente na etapa inicial da abordagem DT, é possível que as pessoas gerem protótipos no auxílio ao entendimento do objetivo, com maior profundidade e para criar um melhor resultado (Steinke et al., 2017). Tal iteração é resultado das descobertas que ocorrem durante os ciclos de prototipação, testes e refinamento pelo fato do *design* da solução do problema ter foco nas necessidades do ser humano (Brown, 2008). As necessidades devem ser identificadas pelo esforço para entender o que as pessoas fazem, por que fazem, o que necessitam, o que pensam do mundo e o que é relevante (Plattner, 2009).

Deste modo, os modelos para a aplicação do DT possuem o foco da solução centrada no humano e a criação de uma expectativa de rápida experimentação e prototipagem (Brown, 2008; Design Council, 2018; Google, 2018; Kolko, 2018; Plattner, 2009). A Figura 2 apresenta as variações entre os modelos partindo das fases propostas no modelo de abordagem de Brown (2008). Os modelos propostos pelos diversos autores possuem variações quando avaliamos a nomenclatura e a quantidade de etapas estabelecidas em cada abordagem. Porém é possível afirmar que todos os modelos possuem uma similaridade ao observarmos o sequenciamento das etapas propostas por cada autor: (i) inicialmente busca-se capturar o entendimento do problema dos usuários, (ii) após identificar o problema, elabora-se o desenho e o esboço de possíveis soluções para o problema e, por fim, (iii) define-se como o problema será revolvido considerando prototipagem e testes para assegurar que a solução está aderente

ao problema objeto do *design*. Nota-se que o sequenciamento e a estrutura das etapas conforme os modelos da Dschool, ISO, Google, Ac4d e Design Council estão diretamente relacionados ao conceito de inspiração, ideação e implementação sugeridos por Brown (2008), presente no modelo da IDEO.

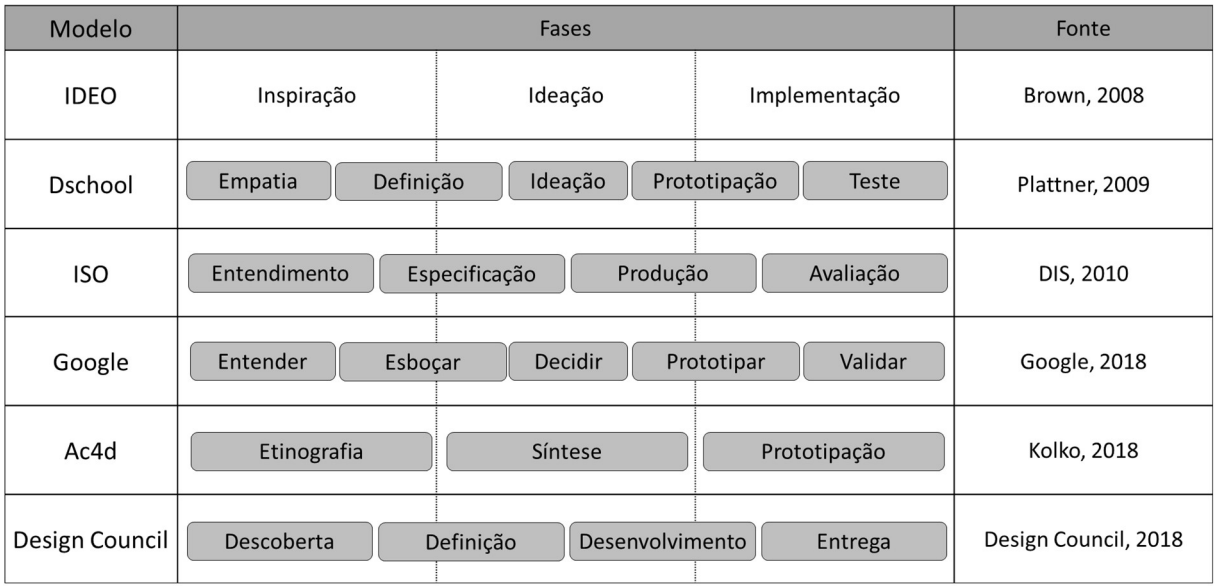

Figura 2. Comparação entre os modelos da abordagem DT

Fonte: Elaborado pelo autor, adaptado de Brown (2008); Plattner (2009); DIS (2010); Google (2018); Kolko (2018); Design Council (2018**)**

O DT reforça o pensamento estruturado no auxílio ao processo criativo de novos produtos que se adaptam às condições de mercado por estarem orientados e relacionados à experiência do cliente (Brown, 2009). Martin (2009) menciona que o pensamento integrativo consiste em não somente comparar duas alternativas opostas e escolher uma, mas que elas sejam analisadas conjuntamente para que obtenha um resultado superior se comparado à escolha de uma ou outra opção original. Deste modo, a criatividade na resolução dos problemas passa a ser estimulada e constantemente explorada.

Conclui-se que a abordagem DT, mesmo diante de algumas variações nos modelos propostos, possui como principais características: (i) o pensamento centrado no humano, (ii) a avaliação de soluções por meio da empatia e colaboração de times multidisciplinares, (iii) a criação de soluções para problemas complexos e não estruturados e (iv) a ênfase nos testes e prototipação (Fleury, Stabile, & Carvalho, 2016). A Tabela 1 apresenta a associação destas características com as ações envolvidas presentes nas diferentes abordagens propostas por diferentes autores. Deste modo, a abordagem DT promove a criatividade partindo das ideias e

necessidades das pessoas e, ao mesmo tempo, concede uma orientação para tornar as ideias possíveis de avaliação por meio da prototipagem.

Tabela 1: Principais características e ações envolvidas na abordagem DT

Principais Características	Ações envolvidas na abordagem
Centrado no humano	Identificar aspectos comportamentais das pessoas e assim convertê-los em benefícios dos clientes e valor ao negócio (Brown, 2008)
Empatia	Ver, pensar e abordar problemas (Pinheiro & Alt, 2011), concentrando as ideias e necessidades das pessoas (Plattner, 2009) , além de promover a configuração de sistemas nos requisitos e recomendações centradas no humano (DIS, 2010)
Criatividade	Gerar constantemente novas ideias com foco na criação de solução e na tradução dos requisitos nos objetivos do projeto (Cross, 2006)
Prototipação	Realizar testes rápidos no intuito de aproximar o que foi desenvolvido com o esperado pelo usuário final (Plattner, 2009)

Fonte: Elaborado pelo autor

2.2 MÉTODOS ÁGEIS

Os métodos ágeis na gestão de projetos possuem ênfase em uma abordagem incremental ao desenvolver um *software* (Steinke et al., 2017), o que representa mais do que seguir fases sequenciais pré-definidas. O método ágil enfatiza (i) os aspectos humanos envolvidos no desenvolvimento de *softwares* mais do que processos e ferramentas, (ii) a execução contínua de testes do *software*, e (iii) a cooperação entre os desenvolvedores e os usuários (Abrahamsson, Salo, Ronkainen, & Warsta, 2002). Highsmith (2002) ressalta que o desenvolvimento ágil não se limita a um conjunto de práticas e técnicas, e sim a capacidade estratégica para a criação e resposta às mudanças, sendo orientada pela flexibilidade, criatividade e inovação de toda a organização, não só do time de desenvolvimento envolvido no projeto.

Os métodos ágeis no desenvolvimento de *softwares*, adicionalmente, são responsáveis pela habilidade de resposta rápida às constantes mudanças no cenário dos negócios (Baweja & Venugopalan, 2015) com foco na adaptabilidade perante as incertezas e mudanças frequentes (Mishra & Mishra, 2011). Conforto *et al.* (2016) definem 3 implicações para a gestão ágil de projetos: (i) o grau de agilidade do time do projeto; (ii) o desempenho da agilidade pode ser influenciado pela capacidade da organização de gerenciar mudanças, e; (iii) a capacidade da

organização de abordar os impactos causados por fatores internos e externos. Nota-se que as organizações que desejam adotar métodos ágeis no desenvolvimento de *software* necessitam alocar recursos para a gestão da mudança cultural que será promovida, bem como avaliar qual método melhor se adequa à organização (Livermore, 2008).

A utilização dos métodos ágeis em gestão de projetos de *software* busca atingir os resultados de maneira enxuta, direcionando esforços para o resultado final do projeto, principalmente quando é concedida autonomia para o controle de mudanças pelo time de desenvolvimento (Maruping, Venkatesh, & Agarwal, 2009). Existem diferentes abordagens de utilização dos métodos ágeis de acordo com a amplitude de sua utilização e o nível de detalhes requeridos para o desenvolvimento de *software* (PMI, 2017a), conforme ilustrado na Figura 3. De acordo com o PMI (2017a), os métodos ágeis se dividem em (i) abordagem escalável na qual é dada maior ênfase em como os projetos são selecionados previamente à sua execução e (ii) métodos de equipes, orientados ao direcionamento de como os membros do time do projeto são organizados e gerenciados ao longo de sua execução. Os métodos ágeis mais difundidos são: *Extreme programming* (XP), Scrum, *Feature driven development* (FDD), *Crystal Methods*, *Dynamic systems development methods* (DSDM), *Adaptive software development* (ASD) e *Lean development* (Nigam & Gupta, 2017). Nos últimos anos é observado que o Scrum isoladamente e o modelo híbrido de Scrum com XP são as metodologias predominantemente utilizadas (Vallon et al., 2018) no desenvolvimento de *softwares*. Ambos, Scrum e XP, estão classificados pelo PMI (2017a) como métodos de equipe, orientando e coordenando as atividades do projeto com foco na agilidade das entregas. Por sua vez o SAFe figura como o método de abordagem escalável com maior cobertura do ciclo de vida de acordo com a classificação do PMI (2017a).

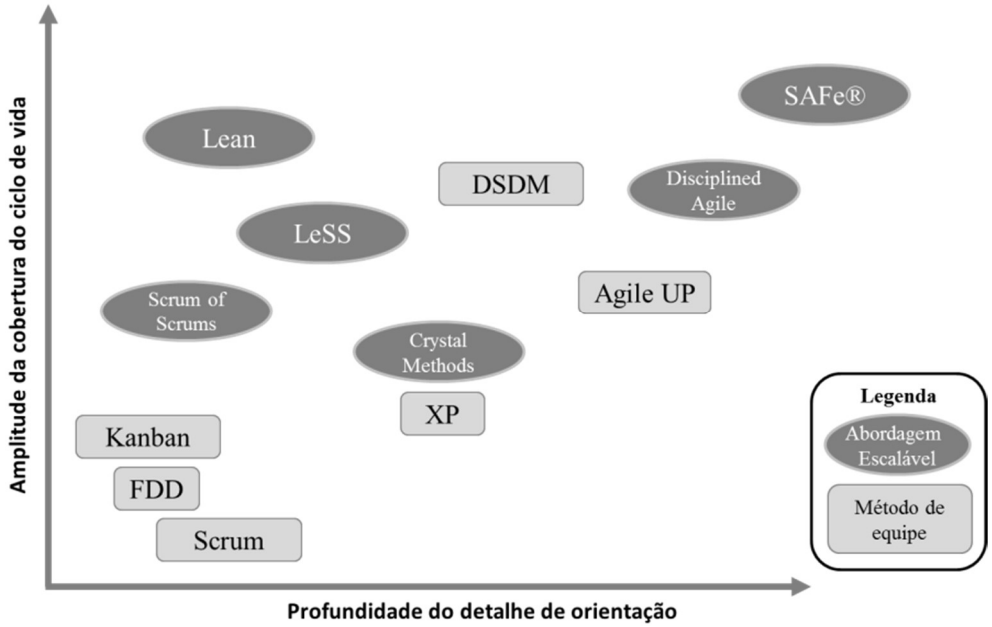

Figura 3. Abordagens ágeis traçadas por amplitude e profundidade
Fonte: PMI, 2017, p. 100.

O *mindset* ágil é definido pelos valores, guiados pelos princípios ágeis e manifestados por meio de práticas ágeis (PMI, 2017a). Este *mindset* é o conjunto de técnicas, modelos e *frameworks* que podem ser utilizados de acordo com as necessidades de cada equipe, projeto ou empresa (Vidal, 2017). Presente em todos os modelos de métodos ágeis propostos, o *mindset* ágil consiste no modo de pensar com foco em ondas sucessivas e rápidas de entrega e possui características colaborativas como resiliência, comprometimento, permissão ao erro, motivação e habilidade para melhoria, aprendizado e adaptação (Senapathi & Drury-Grogan, 2017). Tais características tornam-se relevantes para a organização, proporcionando ampla troca de conhecimento e favorecendo o aprendizado dos colaboradores e gerando maior satisfação, desenvolvimento de habilidades sociais, constantes *feedbacks* e confiança dos profissionais (Solinski & Petersen, 2016).

Pela perspectiva do cliente, em relação aos métodos ágeis, os benefícios mais significativos e visíveis são o valor do produto, melhor relacionamento com o cliente e a qualidade do produto entregue (Solinski & Petersen, 2016). Tal benefício deriva-se da estreita colaboração entre as partes, garantindo requisitos mais claros por meio de melhor compreensão dos dados a serem tratados e responsabilidade mutua pelo projeto, fazendo com que a maior parte do tempo seja dedicada para chegar à solução possível e não só com a coleta de requisitos (Larson & Chang, 2016). Sendo assim, existe uma percepção de valor não só por parte dos membros do time do projeto que desempenham alguma atividade no projeto,

mas também por quem irá se beneficiar pelo produto gerado por um projeto conduzido por um método ágil. Em um projeto de desenvolvimento de um SI, o produto final é o *software*.

O desenvolvimento ágil de *softwares* é orientado por um processo que possui como características: (i) iterações curtas; (ii) testes contínuos; (iii) equipes auto gerenciáveis; (iv) colaboração constante, e; (v) frequente replanejamento com base na realidade atual ao invés de seguir planos potencialmente obsoletos (Highsmith, 2002). Além destas características estarem associadas aos métodos ágeis de forma geral, o Scrum possui mecanismos como auto-organização e urgência, derivados da teoria de controles industriais (Schwaber, 2004). Nota-se, portanto, que há uma orientação ao resultado de fato, ou seja, o projeto tem o enfoque na entrega do *software* funcionando de fato por meio da colaboração dos membros de um time auto gerenciável e também na gestão adequada das mudanças.

Schwaber (2004) estabelece que um projeto de *software* inicia com uma visão geral do sistema que será desenvolvido. Para que este processo seja feito de forma incremental, o Scrum estabelece um fluxo que prevê a realização de uma reunião inicial para definição do backlog com base na visão geral do produto. As responsabilidades da gestão do projeto neste processo são divididas em três papéis: (i) *Product Owner* (PO), responsável pela criação da visão geral dos requisitos, objetivos e plano de entregas; (ii) *Scrum Master* (SM) que é responsável pela condução do projeto e direcionamento para uso do processo do Scrum para resolução dos problemas que surjam ao longo de sua execução, e; (iii) o time do projeto, responsável pelo desenvolvimento das funcionalidades (Schwaber, 2004). A partir da visão geral do projeto, o PO é responsável pela confecção do backlog do produto juntamente com as áreas responsáveis e determinam qual pacote deve ser desenvolvido pelo time do projeto a cada ciclo, denominado iteração. Em cada iteração, o SM e o time do projeto desenvolvem as funcionalidades do *software*, de modo que volte para os usuários com a visão do que foi entregue e o que será priorizado para as próximas iterações (Schwaber, 2004), conforme ilustrado na Figura 4.

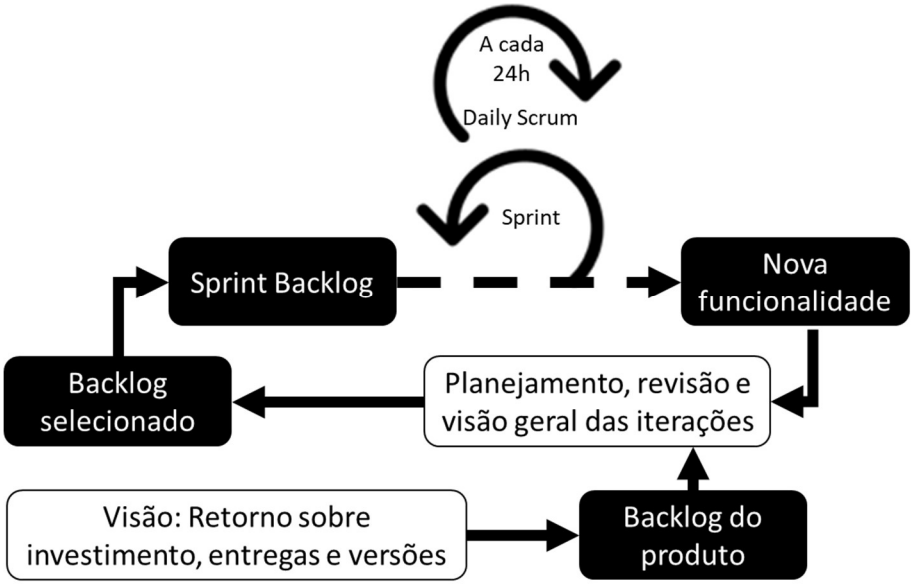

Figura 4. Fluxo do processo Scrum.
Fonte: Adaptado de Schwaber (2004)

Existe a necessidade de que a adoção do método ágil, em todas as suas variações, seja feita de maneira gradual em relação ao modelo rígido de desenvolvimento do *software*, fazendo prevalecer a relevância da qualidade do produto entregue (Solinski & Petersen, 2016). Em função disto, um modelo revisado de uso ágil sustentável sugere que as práticas ágeis considerem os fatores tecnológicos e organizacionais buscando a efetividade no método em termos de produtividade, qualidade e rapidez (Senapathi & Drury-Grogan, 2017), como pode ser observado na Figura 5. Nota-se que a efetividade na entrega do *software* por meio dos métodos ágeis de projetos é obtida quando (i) o time possui competências necessárias para a execução do trabalho, (ii) as ferramentas necessárias estão disponíveis e (iii) a organização oferece estrutura e suporte necessários para a produção do sistema informatizado.

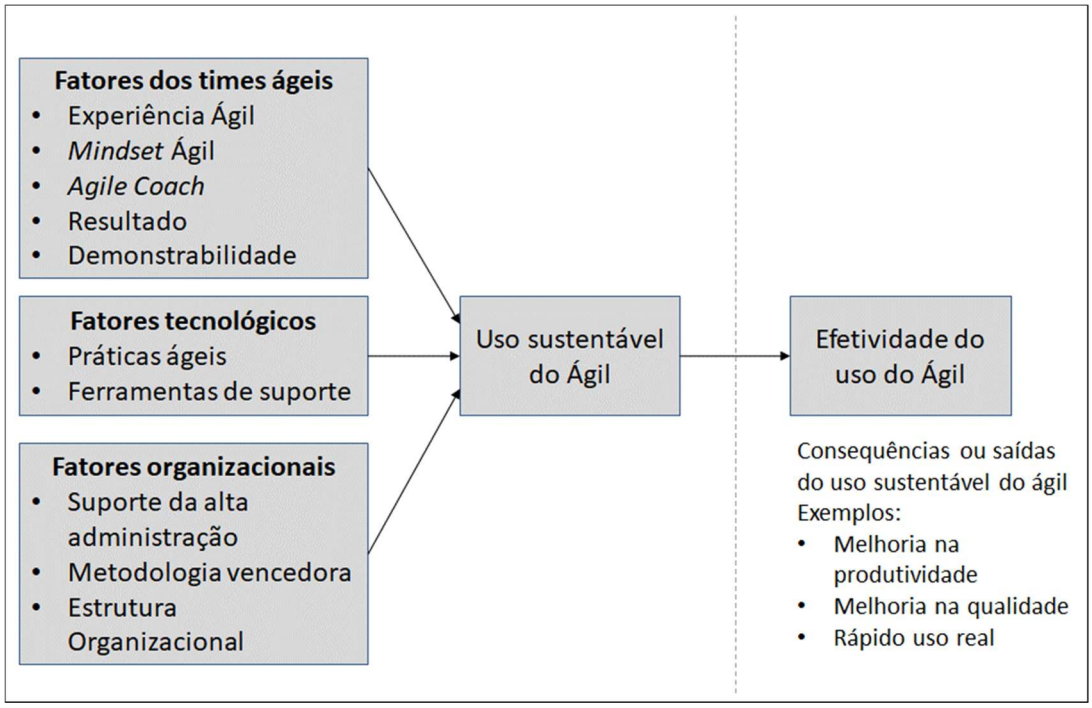

Figura 5. Modelo revisado de uso ágil.

Fonte: Senapathi & Drury-Grogan (2017, p. 312)

Cabe observar que existem variações entre os modelos de gestão de projetos. Algumas organizações aplicam modelos de gestão de projetos sequencial e linear no desenvolvimento de *softwares*, tradicionalmente seguindo um ciclo de vida clássico composto por um planejamento mais sofisticado e rígido, compostos por planos de tarefa, atribuição de responsabilidades, declaração de trabalho e relatório orçamentário (Cohen & Money, 2008). Buscando comparar métodos ágeis e o modelo linear em gestão de projetos de *software*, Ahimbisibwe, Urs e Cavana (2016) avaliaram o desempenho do método ágil e do clássico (sequencial e linear). Ao avaliar os fatores críticos de sucesso sob a perspectiva do produto, os aspectos participação do usuário e mudanças na especificação foram superiores no método clássico em relação aos métodos ágeis (Ahimbisibwe, Urs, & Cavana, 2016). Os métodos ágeis demonstraram-se superiores em relação aos fatores de sucesso do produto do projeto ao considerar aspectos que envolvem o comprometimento e o desenvolvimento do time, a participação ativa dos usuários e a significância da escolha da melhor abordagem para a gestão do projeto de desenvolvimento do *software* (Ahimbisibwe et al., 2016).

2.3 INTEGRAÇÃO DO DESIGN THINKING E MÉTODOS ÁGEIS NA GESTÃO DE PROJETOS DE *SOFTWARE*

A necessidade de entrega de resultados mais rápidos faz com que as organizações busquem líderes ágeis que facilitem a atuação das equipes com confiança e competência mesmo em ambientes com situações ambíguas e com alto índice de mudanças (Baweja & Venugopalan, 2015). A abordagem e metodologia de gerenciamento de projetos devem ser adotadas com critério, sendo necessário considerar o contexto e modalidade de projetos de desenvolvimento de sistemas (Ahimbisibwe, Urs, & Cavana, 2016). Em alguns casos a metodologia de desenvolvimento de *software* escolhida está associada à fidelização dos praticantes em alguma metodologia específica (Ahimbisibwe et al., 2016).

Por sua vez, a metodologia ágil em projetos de *software* recomenda que a alta administração da organização seja atuante, não só fornecendo recursos e direcionamento ao projeto, mas também dando o suporte adequado quando situações inesperadas ocorrem (Ahimbisibwe et al., 2016). Quando tais mudanças são exigidas, as respostas devem ser rápidas e derivadas do desdobramento das práticas, princípios e valores ágeis (Sidky & Smith, 2009). Em relação ao desenvolvimento de *softwares*, Pieroni et al. (2018) apontam prós e contras no uso de métodos ágeis: (i) como pontos positivos o método apresenta melhor comunicação entre os times, aumenta a exatidão na definição de requisitos e facilita a gestão da mudança; (ii) retrabalho e limitação nos testes por parte dos usuários do *software* ainda se apresentam como pontos fracos resultantes do uso de métodos ágeis. Ademais, nota-se uma preocupação em relação às mudanças de escopo causado pelo distanciamento do usuário durante o desenvolvimento do *software*, que acarreta em retrabalho e resulta em aumento do custo do projeto pois as correções nem sempre são realizadas por pessoas que conhecem o sistema (Pinheiro & Alt, 2011).

Diante deste contexto, observa-se que as características do projeto podem influenciar na definição da abordagem mais indicada para sua condução. A identificação preliminar dos métodos a serem adotados, quais objetivos o projeto deve atender, a frequência e a amplitude das modificações que ocorrem durante o ciclo de vida do projeto são elementos que influenciam nesta definição (Turner & Cochrane, 1993), conforme ilustrado na Figura 6.

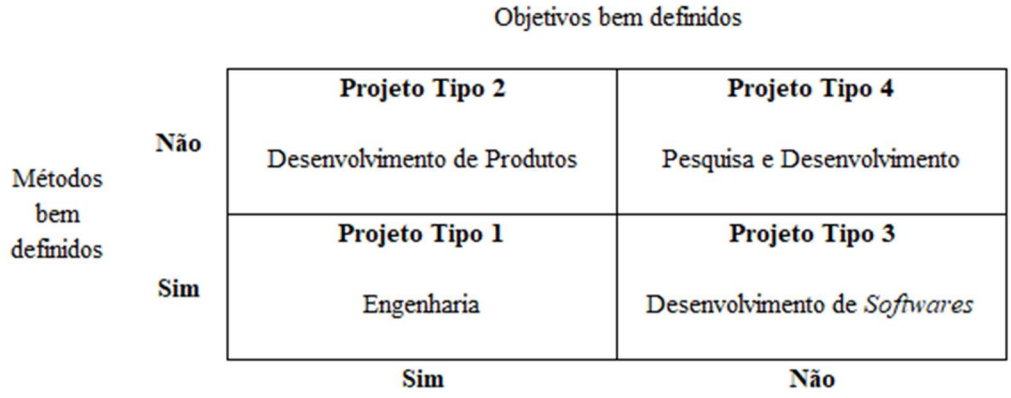

Figura 6. Matriz de objetivos e métodos
Fonte: Adaptado de Turner e Cochrane (1993)

Turner e Cochrane (1993) estabelecem que quando o projeto possui métodos e objetivos bem definidos, como no caso um projeto de engenharia (tipo 1), o aspecto de incerteza tende a ser menor e, portanto, possui maior probabilidade de êxito. Por outro lado, os projetos de desenvolvimento de *software* (tipo 3), como normalmente apresentam pouca ou nenhuma clareza dos objetivos, possuem maior probabilidade de insucesso, mesmo quando há um método bem definido. Deste modo, é necessário que os projetos de *software* usem mecanismos que possibilitem elevar o grau de detalhamento dos objetivos, mesmo que seja ao longo de sua execução.

De acordo com Nigam e Gupta (2017), os métodos ágeis possuem características associadas como (i) modularidade, que permite que os processos sejam quebrados em componentes menores, no caso, atividades; (ii) iteração, que prevê a identificação de erros ao longo do processo antes de chegar ao resultado final, (iii) temporal, ou seja, possui janela de tempo pré-definida para a execução de um trabalho específico; (iv) moderação ao assumir prazos inexequíveis; (v) adaptabilidade, frente à novos riscos identificados; (vi) incremental, recomendando a execução de ciclos curtos para refinamento das entregas; (vii) convergência, enquanto os riscos são tratados o sistema está sendo entregue de forma incremental; (viii) orientado a pessoas, favorecendo pessoas em detrimento à processos e tecnologia, e; (ix) colaborativo, que remete à comunicação entre os membros da equipe. A abrangência que tais características causam na gestão de um projeto de desenvolvimento de *software* pode justificar o aumento da probabilidade de sucesso em projetos no que se refere à eficiência e satisfação das partes interessadas quando métodos ágeis são utilizados (Serrador & Pinto, 2015).

Para que as características relativas à colaboração e orientação à pessoas mencionados por Nigam e Gupta (2017) sejam potencializados, os métodos ágeis em projetos de desenvolvimento de *software* estão sendo integrados com aspectos do *design* centrado no ser humano (Ardito, Baldassarre, Caivano, & Lanzilotti, 2017). Este fato está associado aos pontos comuns que o método ágil e a abordagem DT compartilham, como envolvimento do usuário, importância atribuída às práticas de teste e prototipagem e *design* iterativo (Ardito et al., 2017). Lucena *et al.* (2016) sugerem um modelo que permita uma melhor compreensão dos problemas que precisam ser resolvidos e quais são as melhores soluções para satisfazer as necessidades do usuário. Destaca-se a abordagem para a aproximação do cliente final por meio da colaboração e a repetição dos ciclos de prototipação e testes. Tal afirmação é corroborada com a identificação de aspectos do DT em algumas organizações, tais como (i) a visão centrada no humano, (ii) a criação da solução dos problemas por times multidisciplinares, (iii) o uso dos processos de ideação e experimentação e (iv) a confecção de protótipos para a coleta de percepções dos usuários e mapeamento de requisitos (Góes & Russo, 2018).

Portanto, as mudanças que ocorrem ao longo do ciclo de vida de um projeto de desenvolvimento de *software*, bem como a aproximação do time do projeto com os clientes e usuários finais podem ser conduzidas por meio do uso da abordagem do DT (Steinke et al., 2017), conforme ilustrado na Figura 7. Assim, a criatividade dos indivíduos e dos times de desenvolvimento seria encorajada por meio da utilização dos métodos ágeis para a implantação bem-sucedida do *software* (Mishra & Mishra, 2011), partindo de um desenho de sistema construído sob a perspectiva do cliente (Steinke et al., 2017).

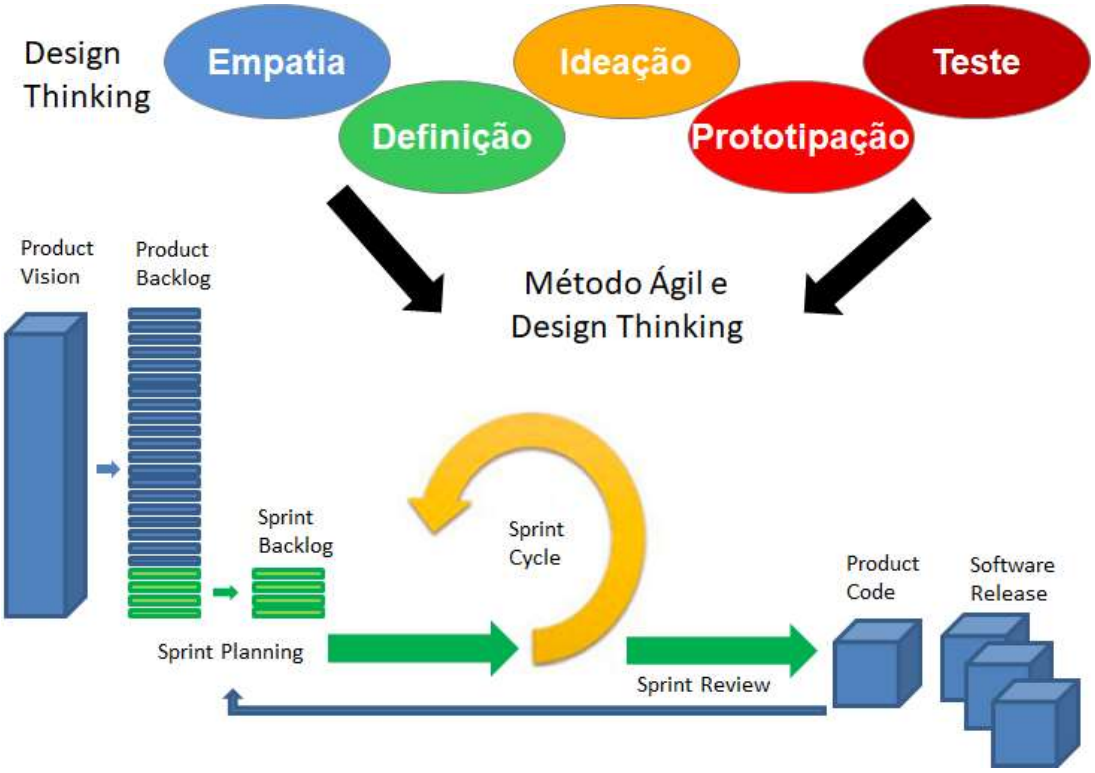

Figura 7. *Design Thinking* e método ágil
Fonte: Steinke et al., 2017, p. 54

Existem variações entre as formas de uso do DT integrado com métodos ágeis entre as organizações que gerenciam projetos de desenvolvimento de *software*. De acordo com Góes e Russo (2018), o uso do modelo de gestão de projeto que integra DT e métodos ágeis pode ocorrer ao longo do ciclo de desenvolvimento do *software*:

- como ondas entre as fases de desenvolvimento, sem nenhum sincronismo;
- durante a etapa de concepção sincronizada com a etapa de desenvolvimento, ou;
- com a integração do *design* e a construção do *software* em todas as etapas de desenvolvimento do *software*.

Sendo assim, é possível afirmar que a aplicação do modelo que integra o DT com métodos ágeis está relacionada às etapas do ciclo de desenvolvimento do *software* que exigem maior interatividade com clientes e usuários, principalmente para que seja obtido um melhor desenho da solução. De acordo com estudo recente que avaliou artigos que mencionam o uso de modelos que integram a abordagem DT e métodos ágeis houve aumento da qualidade do *software* e, consequentemente, maior satisfação de usuários (Pereira & Russo, 2018). De acordo com Pereira e Russo (2018), embora o modelo possa ser aplicado em diferentes estágios do ciclo de vida de desenvolvimento de um *software,* é possível identificar aspectos

relevantes no uso dos modelos que integram a abordagem DT e métodos ágeis, como consta na Tabela 2.

Tabela 2: Benefícios relacionados à aplicação da integração da abordagem DT com métodos ágeis

Ciclo de vida do projeto de desenvolvimento de *software*	Definição e autores
Estágios iniciais (desenho e coleta de requisitos)	(i) observância dos aspectos de interface já nas primeiras ideias do software (Forbrig, 2016a); (ii) avaliação de diferentes alternativas que podem ser desenvolvidas nos primeiros estágios de desenvolvimento (Forbrig & Saurin, 2016; Losada, Urretavizcaya, & De Castro, 2011) e (iii) clareza nos requisitos (Steinke et al., 2017) baseados na opinião e observação dos usuários (Bosch & Bosch-Sijtsema, 2011; Ximenes, Alves, & Araújo, 2015).
Todas as etapas do ciclo de vida	(i) busca não somente capturar as necessidades dos clientes nos estágios iniciais, mas também garantir a usabilidade e adaptabilidade do *software* desenvolvido e implantado (Fischer & Senft, 2016); (ii) favorece a identificação dos problemas que necessitam ser resolvidos (Forbrig, 2016b; Higuchi & Nakano, 2017; Paula & Araujo, 2016); (iii) reflete a satisfação do cliente no *software* que será entregue (Gurusamy, Srinivasaraghavan, & Adikari, 2016; Lucena et al., 2016; Prior, Waller, Black, & Kroll, 2013) para atingir o sucesso do projeto (Hussain et al., 2008); (iv) sugere a preocupação com um melhor alinhamento entre as expectativas do usuário e o time de desenvolvimento (Gurusamy, Srinivasaraghavan, Adikari, et al., 2016) durante todo o ciclo de vida do software (Darrin & Devereux, 2017; Lester, 2011; Newman et al., 2015; Xiong & Wang, 2010); (v) melhor usabilidade (González-González, Toledo-Delgado, & Muñoz-Cruz, 2015; Isa et al., 2014; Sohaib & Khan, 2011); (vi) critérios de aceite do *software* definidos antes da fase de implantação (Lárusdóttir, Cajander, & Gulliksen, 2014).

Fonte: Elaborado pelo autor

Portanto, pode-se concluir que os benefícios no uso de modelos que integram a abordagem DT e métodos ágeis se dão de duas maneiras: (i) nos estágios iniciais do projeto voltado para a concepção do *software* ou (ii) ao longo de todo o ciclo de vida, garantindo que o *software* esteja efetivamente adequado ao uso. Portanto, podemos afirmar que as práticas que integram ambas as ferramentas são aplicadas ao longo da gestão de um projeto para o desenvolvimento de *softwares*. O uso de um método que integra a abordagem DT e métodos

ágeis tem como princípio suportar os projetos de desenvolvimento de *softwares* com o intuito de entrega de um sistema útil e adequado ao uso (Brhel, Meth, Maedche, & Werder, 2015).

3 PROCEDIMENTOS METODOLÓGICOS

Design Science busca introduzir um ciclo de resolução de problemas além de construir teorias, tornando a pesquisa mais relevante para o campo no qual ela é aplicada (Wieringa & Morali, 2012). Como o objetivo principal é avaliar como o DT pode ser integrado com métodos ágeis em projetos de desenvolvimento de *software* gerando uma contribuição prática para o campo de estudo, esta pesquisa de dissertação utilizou o método *Design Science Research*, um dos métodos da *Design Science*. Este método prevê a participação ativa do pesquisador no entendimento e resolução de problemas por meio da criação de soluções (Freitas Jr, Machado, Klein, & Freita, 2015), avaliando simultaneamente o contexto organizacional e a utilidade prática do artefato proposto, por vezes comparando-o com outras possibilidades para soluções na área de TI (Hevner et al., 2004).

O método *Design Science Research* valoriza a produção científica junto à sociedade por sua aplicação, por meio da aproximação entre teoria e prática e entre academia e sociedade (De Sordi et al., 2011). Os itens a seguir apresentam a fundamentação teórica que foi utilizada nesta pesquisa.

3.1 DESIGN SCIENCE

A *Design Science* é a ciência do estudo dos artefatos dentro de um contexto, orientando o novo conhecimento para a resolução de problemas (De Sordi et al., 2011). A área de TI, mais precisamente o segmento de desenvolvimento de SI, exige a adição de um ciclo de resoluções dos problemas além de um ciclo de construção teórica (Wieringa & Morali, 2012). A proposta da *Design Science* é gerar uma base de conhecimento a partir da aplicação de um método para resolução de um problema (Wieringa & Morali, 2012). No que se refere a *Design Science* no campo de SI, Hevner *et al.* (2004) sugerem um *framework* conceitual do entendimento, execução e avaliação da pesquisa combinando a ciência comportamental e o paradigma da *Design Science*. Este *framework* sugere que as necessidades de negócio sejam derivadas do ambiente (pessoas, organizações e tecnologia) para a construção da teoria e que os artefatos gerados sejam avaliados em relação à aplicação da base do conhecimento gerado (Hevner et al., 2004).

March & Smith (1995) propõem a aplicação do *Design Science* dividida em duas dimensões: (i) a primeira, e usada no presente estudo, está relacionada às saídas esperadas e aos artefatos, compostos por construtos, modelos, métodos e instâncias; (ii) a segunda está relacionada às atividades de pesquisa, categorizadas em construção, avaliação, teorização e justificativa. Como a *Design Science* é a ciência que procura a consolidação de conhecimentos sobre o projeto e o desenvolvimento de melhorias dos sistemas existentes (Dresch, Lacerda, & Antunes Jr, 2015), ela possui diferentes métodos de pesquisa. Dentre os métodos da *Design Science* estão a pesquisa-ação, estudo de caso e um método de pesquisa denominado *Design Science Research* (Dresch et al., 2015).

O método *Design Science Research* promove a geração do conhecimento em formato de prescrição para apoiar soluções de problemas reais (Dresch et al., 2015) e busca gerar conhecimento para profissionais no campo da pesquisa (Van Aken, 2004). Por esta razão será o método de pesquisa escolhido para esse projeto de dissertação em função de avaliar e desenvolver soluções dentro de uma organização para a resolução de um problema real.

3.1.1 Design Science Research

O método *Design Science Research* prevê que o conhecimento gerado seja prescritivo ao invés de apenas descrever os processos e o contexto (Lacerda, Dresch, Proença, & Antunes Junior, 2013). É necessário considerar que o método de pesquisa escolhido apresente clareza e detalhe suficientes para definir a teoria da utilidade em três vertentes: (i) o espaço do problema, que representa o entendimento do pesquisador sobre o problema; (ii) o espaço da solução, que promove o entendimento dos conceitos que descrevem a solução, e; (iii) a utilidade que as interligam (Venable, 2006). O método *Design Science Research* valoriza a produção científica junto à sociedade por sua aplicação, por meio da aproximação entre teoria e prática e entre academia e sociedade (De Sordi et al., 2011).

Para facilitar o entendimento de aplicação do *Design Science Research* em projetos de SI, Hevner *et al.* (2004) estabelecem diretrizes para o projeto e o desenvolvimento da pesquisa, como pode ser visto na Tabela 3. A escolha do método assegura que pesquisadores definam como aplicar o artefato em outros projetos no campo de TI no desenvolvimento de sistemas (Hevner et al., 2004), dando o rigor metodológico necessário para que o artefato seja generalizado para outras empresas que desenvolvem *softwares*. Neste contexto, as diretrizes servem para guiar: (i) como o artefato será definido, (ii) como será identificada a relevância

do problema, (ii) como o projeto de pesquisa será avaliado, (iv) estabelecimento da contribuição esperada, (v) quais os componentes que irão trazer o rigor esperado, (vi) como o processo de busca se dará e (vii) como a pesquisa será comunicada.

Tabela 3: Diretrizes para aplicação do método *Design Science Research*

Diretriz	**Descrição**
Artefato como objeto do estudo	Produzir um artefato viável no formato de um construto, um modelo, um método ou uma instanciação.
Relevância do Problema	Desenvolver soluções baseadas em tecnologia tais como sistemas da informação para a resolução de problemas relevantes do negócio.
Avaliação do projeto	A utilidade, a qualidade e a eficácia de um artefato devem ser rigorosamente demonstradas por meio de métodos de avaliação executados adequadamente.
Contribuição para a área de conhecimento do artefato	Pesquisas científicas eficazes de *design* devem fornecer contribuições claras e verificáveis nas áreas do artefato, bases de projeto e das metodologias de pesquisa.
Pesquisa Rigorosa	O método *Design Science Research* tem como base a aplicação de métodos rigorosos para a construção e avaliação do artefato da pesquisa.
Design como um processo de busca	A busca para a efetividade de um artefato requer o uso de meios disponíveis para atingir sua finalidade e que satisfaça as questões no ambiente do problema.
Comunicação da Pesquisa	A pesquisa deve ser apresentada para os públicos orientados à tecnologia e à gestão.

Fonte: Hevner *et al.*, 2004, p. 83.

De modo a complementar o suporte metodológico do presente estudo algumas atividades propostas por Gill e Hevner (2013) foram adotadas ao longo desta pesquisa. Para tal, foi adotado um protocolo que contém as seguintes características: (i) contribuição da pesquisa: a efetividade do *Design Science Research* é derivado do impacto causado onde o projeto é realizado por meio das contribuições em termos do artefato, adequação, fundamentos, teorias e / ou métodos de *design*; (ii) rigor da pesquisa: o *Design Science Research* exige que a construção e a avaliação de artefatos de *design* sejam investigados empregando um nível de rigor apropriado à natureza e ao estágio do projeto; e (iii) comunicação da pesquisa: a pesquisa de *design* deve ser comunicada à comunidade de desenvolvimento de *software* (Gill & Hevner, 2013).

3.1.2 Classes de Problemas

A classe de problemas é definida como "a organização de um conjunto de problemas, práticos ou teóricos, que contenha artefatos avaliados, ou não, úteis para a ação nas organizações" (Lacerda *et al*., 2013, p. 747). O objetivo da classe de problemas é possibilitar a generalização para outras organizações, "orientando a trajetória do desenvolvimento do conhecimento no âmbito da *Design Science*" (Dresch, Lacerda, & Antunes Jr, 2015, p. 103).

Sendo assim, a construção da classe de problemas possui uma lógica compreendida em quatro passos: (i) conscientização: levantamento dos problemas, práticos ou teóricos; (ii) revisão sistemática da literatura, composta pelas bases de dados científicas e técnicas; (iii) identificação dos artefatos que encaminham soluções ao problema em questão, e; (iv) configuração da classe de problemas considerando o agrupamento dos artefatos (Lacerda et al., 2013).

3.1.3 Artefatos

O artefato é a organização dos componentes do ambiente interno para atingir objetivos em um determinado ambiente externo (Simon, 1996). A realização da pesquisa a partir da decomposição em partes menores simplifica um sistema complexo facilitando a compreensão de como as informações foram utilizadas na pesquisa, tanto em seu desenvolvimento quanto em sua reprodução (Simon, 1996). De acordo com Simon (1996), a ciência do artificial, que consiste em tudo o que é feito derivado do comportamento humano, requer a combinação de três aspectos: o propósito do artefato em si, a característica do artefato e o ambiente onde o artefato interage. Os artefatos podem ser classificados em cinco tipos: constructos, modelos, métodos, instanciação e *design proposition* (Dresch et al., 2015).

Os construtos consistem nos conceitos utilizados na descrição dos problemas de um domínio e na especificação da solução, sendo relevante tanto para a ciência tradicional ou para a *Design Science Research* (March & Smith, 1995).

Os modelos correspondem ao conjunto de proposições ou definições que expressam os relacionamentos entre os construtos e podem ser interpretados como as representações de como as coisas são (March & Smith, 1995).

Os métodos consistem no conjunto de etapas (algoritmo ou guia de atividades) usado para o desenvolvimento de uma atividade (March & Smith, 1995). Os métodos de desenvolvimento de sistemas facilitam a construção da representação da necessidade dos usuários, divulgados em problemas, decisões, fatores críticos de sucesso e fatores sociais, técnicos e de implementação (March & Smith, 1995) e favorecem a transformação dos

sistemas em busca de sua melhoria, típico das pesquisas fundamentadas em *design science* (Dresch et al., 2015).

A instanciação corresponde à implantação de um artefato em seu ambiente real (March & Smith, 1995) que operacionaliza e orientam a utilização dos artefatos (constructos, modelos e métodos) em seu ambiente (Dresch et al., 2015).

Design proposition se refere às contribuições teóricas que podem ser feitas por meio da aplicação do método *Design Science Research* (Dresch et al., 2015). Desta forma, o artefato que gerar uma contribuição teórica originária do método em questão torna-se um conhecimento que pode ser aplicado em diversas situações similares em função de generalizar a solução para uma determinada classe de problemas (Dresch et al., 2015).

Existem diferentes métodos para a avaliação dos artefatos: (i) observacional, que pode ser estudo de caso ou estudo de campo; (ii) analítico, caracterizado pela análise estatística, análise da arquitetura, otimização ou análise dinâmica; (iii) experimental, sendo classificado em experimento controlado ou simulação; (iv) teste, dividido em teste funcional (*white box*) ou teste estrutural (*black box*), e; (v) descritivo, classificado em argumento informado ou cenários detalhados em torno do artefato (Hevner et al., 2004).

Quando ocorrem mudanças nas necessidades ao longo do desenvolvimento da pesquisa é preciso que os artefatos sejam modificados para atender seu propósito (March & Smith, 1995). A avaliação final do artefato deve ser feita em cada etapa do *Design Science Research*, sendo possível identificar os resultados parciais e assegurando que a pesquisa está aderente aos objetivos propostos (Lacerda et al., 2013).

3.2 UNIDADE DE ANÁLISE

A Keep IT Simple é uma empresa de TI que atua no desenvolvimento de *softwares* e fornece mão de obra qualificada para empresas cujo negócio principal não seja tecnologia. A empresa está presente em dois escritórios, ambos localizados na região metropolitana da cidade de São Paulo, no Brasil e possui mais de 50 colaboradores. A Keep IT Simple possui em sua carteira de serviços consultoria em transformação digital, inteligência competitiva e *business intelligence* e pacotes de soluções para automatização de processos e monitoramento de concorrentes. Atualmente grande parte dos projetos de desenvolvimento de *softwares* da Keep IT Simple visa atender a demanda de sistemas voltados para um cliente do segmento de varejo. A partir da análise dos problemas identificados nos projetos de desenvolvimento de *softwares* da empresa Keep IT Simple foi definido o artefato, no caso um método de gestão de

projetos que seja capaz de solucionar tais problemas, sejam eles de ordem técnica ou que causam limitações no uso do *software* desenvolvido quando entregue aos clientes. A análise foi feita a partir das informações coletadas em entrevistas com representantes da Keep IT Simple e com representantes da empresa cliente e recebedora do *software.*

A presente pesquisa realizou, a partir de um modelo, o desenvolvimento de um método que integra o DT com métodos ágeis, avaliando seus benefícios e resolução dos problemas identificados ao longo do ciclo de vida de um projeto de desenvolvimento de *software*. O uso do método *Design Science Research* prevê o desenvolvimento, a aplicação, avaliação e generalização do artefato em projetos que desenvolvem *softwares*. Portanto, a unidade de análise da presente pesquisa corresponde aos projetos de desenvolvimento de *software* executados pela Keep IT Simple e que suportaram o desenvolvimento do método de gestão de projetos que integra a abordagem DT e métodos ágeis, mas possível de ser generalizado para outras organizações que desenvolvem *softwares*.

3.3 ETAPAS DE PESQUISA

A aplicação do método *Design Science Research* segue um conjunto de etapas com o objetivo de estabelecer um fluxo que possibilite o entendimento do problema (Dresch et al., 2015), como pode ser visto na Figura 8. Este fluxo foi adotado para o desenvolvimento do artefato capaz de avaliar o problema existente na empresa Keep IT Simple e que tenha potencial para mudar o contexto ao ser aplicado. As características e abordagens de cada uma das etapas da pesquisa serão detalhadas nas seções 3.3.1 a 3.3.9.

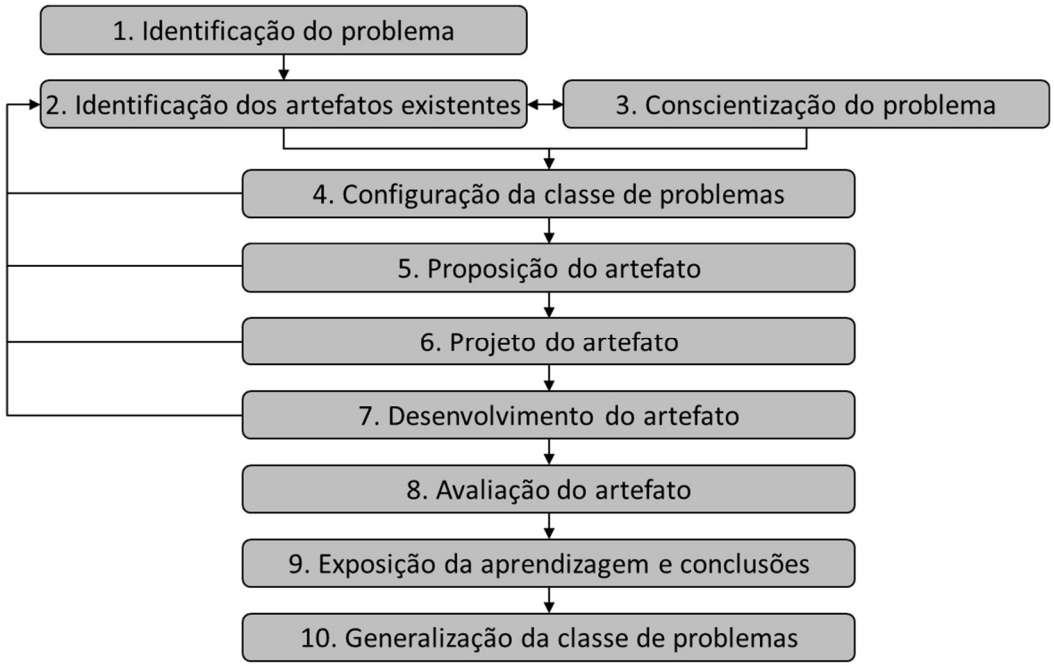

Figura 8. Fluxo e etapas para a condução do *Design Science Research*

Fonte: Baseado em Dresch, Lacerda, e Antunes Jr (2015, p.125)

Cada uma das etapas de pesquisa previstas no presente estudo quando desenvolvidas foram compostas de atividades de coleta e análise de dados. De modo a favorecer o entendimento de como o presente estudo foi desenvolvido, a Tabela 4 contém o relacionamento da etapa de pesquisa e a respectiva atividade. Dentre as atividades de coletas de dados foram realizadas entrevistas e grupos focais, além de uma revisão sistemática da literatura para a verificação de um artefato de pesquisa. As atividades finais consistiram na consolidação e descrição dos resultados e conclusões do presente estudo.

Tabela 4: Etapas da *Design Science Research* e atividades realizadas no presente estudo.

Etapa da *Design Science Research*	Atividades de coleta e análise de dados
1. Identificação do problema	Realização de três entrevistas e documentação das atas de reunião
2. Identificação dos artefatos	Revisão sistemática da literatura
3. Conscientização do problema	Análise dos problemas identificados na Keep Simple por meio de três entrevistas adicionais e com a validação de especialistas em DT e métodos ágeis
4. Configuração da classe de problemas	Com base nos problemas, o método integrado de DT com métodos ágeis foi estabelecido como o artefato
5. Proposição do artefato	Validação da aderência do modelo com os problemas mapeados na Keep IT Simple
6. Projeto do artefato	Entrevistas e características do projeto do artefato
7. Desenvolvimento do artefato	Grupo focal exploratório para o desenho do artefato
8. Avaliação do artefato	Grupo focal confirmatório e validação das características do artefato estabelecidas por Gill e Hevner (2013)

Etapa da *Design Science Research*	Atividades de coleta e análise de dados
9. Exposição da aprendizagem e 10. Generalização para uma Classe de Problemas	Descrição das contribuições teóricas e para a prática e descrição das considerações finais

Fonte: Autor

3.3.1 Identificação do Problema

A primeira etapa do método *Design Science Research* possui atividades para identificar o problema e justificar a relevância para seu estudo, além de considerar como saída desta etapa a definição da questão de pesquisa (Dresch et al., 2015). A identificação do problema estudado ocorreu por meio de três entrevistas, cujo roteiro está presente no Apêndice A. A primeira entrevista foi feita com o representante da alta administração da Keep IT Simple, considerando os aspectos negativos ou problemáticos do processo atual de desenvolvimento de *softwares*, que consistem nos produtos comercializados pela organização. Além da entrevista com a alta administração da Keep IT Simple, também foram consultados dois representantes da empresa de varejo, cliente da Keep IT Simple, buscando capturar a percepção sob a visão da contratante. Deste modo, foi possível identificar o nível de satisfação dos clientes e dos usuários finais do *software* desenvolvido. O resultado desta etapa permitiu a formulação da questão da presente pesquisa. As características dos entrevistados estão relacionadas na Tabela 5.

Tabela 5: Características dos entrevistados da etapa de identificação dos problemas

Entrevistado	Experiência em TI (em anos)	Experiência em gestão de projetos	Cargo	Característica da empresa em relação ao *software* desenvolvido	Grau de instrução	Vivência na organização
E1	19 anos	7 anos	CEO	Contratada	Mestrado	Especialista em sistemas da informação
E2	10 anos	3 anos	Analista de Negócios	Contratante	Pós-graduado (em curso)	Especialista em sistemas da informação
E3	12 anos	9 anos	Gerente de desenvolvimento	Contratante	Pós-graduado	Gerente técnico de sistemas

Fonte: Autor

3.3.2. Identificação dos artefatos existentes

Esta etapa consiste na identificação dos artefatos existentes a partir da literatura disponível, feito por meio de uma revisão sistemática da literatura, que busca encontrar conteúdo relevante ao estudo desejado (Ferreira, 2013). Tal etapa permite o uso do conhecimento por meio da consulta de outros estudos com foco no mesmo problema ou problemas similares (Dresch et al., 2015). Deste modo, o resultado desta etapa permitiu a identificação de artefatos que possam ser utilizados para a solução dos problemas da Keep IT Simple.

Com o intuito de identificar na literatura a existência de artigos que exploram o uso integrado da abordagem DT com métodos ágeis no desenvolvimento de software, foi realizada pesquisa nas bases de dados *Web of Science* (WoS) e *Scopus*. O objetivo é avaliar e fazer com que seja dado um sentido a um grande volume de informação para que sejam identificadas as características sobre um determinado tópico (Petticrew & Roberts, 2006).

A revisão sistemática da literatura considerou os onze passos estabelecidos por Petticrew e Roberts (2006). Os onze passos estão distribuídos em três fases: (i) a fase de definições iniciais, onde a questão, protocolo e o objetivo são definidos; (ii) a fase de pesquisa, onde a busca pela informação disponível na literatura é feita, e, por fim; (iii) a fase de avaliação, onde as informações são consolidadas e os achados são expostos. A Figura 9 apresenta o detalhe das fases e passos em uma disposição gráfica, permitindo a identificação do fluxo no qual as mesmas foram dispostas, bem como avaliar como o subconjunto dos artigos foram sendo obtidos ao longo do processo da revisão sistemática da literatura.

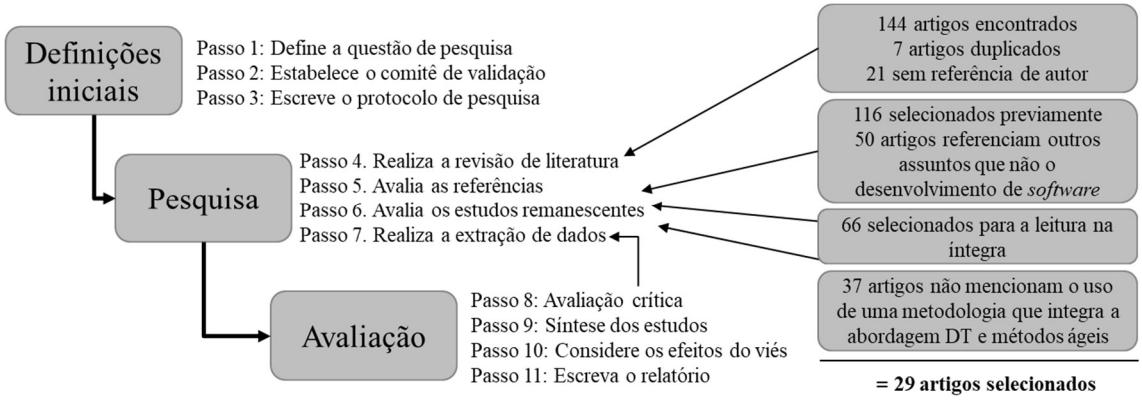

Figura 9. Fases e etapas para a condução de uma revisão sistemática da literatura

Fonte: Baseado em Patticrew e Roberts (2006)

A revisão sistemática da literatura avaliou os artigos existentes que abordam a aplicação integrada da abordagem DT com métodos ágeis na gestão de projetos de

desenvolvimento de *softwares* em conformidade com a questão de pesquisa. A avaliação teve início da fase de **definições iniciais**. Ainda nesta fase, foram realizadas (i) a consulta de especialistas e (ii) a definição do protocolo que orientou a pesquisa. O protocolo de pesquisa inclui, dentre outras características, a definição das bases de dados utilizadas na busca dos artigos, os critérios de inclusão e de exclusão da pesquisa. Após a obtenção das informações gerais provenientes da busca inicial nas bases de dados, foi iniciado o trabalho de pesquisa com maior nível de profundidade, presente na fase dois, denominada como **pesquisa**. Nesta fase ocorreu a seleção de artigos mais relacionados à área de projetos de desenvolvimento de *software*, que teve como princípio avaliar a literatura disponível no segmento desejado. Na terceira e última fase, denominada como **avaliação**, a pesquisa foi concluída por meio da sintetização dos artigos encontrados e onde foram descritas as limitações, vieses e o relatório final. As etapas e os passos estão descritos detalhadamente na Tabela 6. Cabe ressaltar que no presente estudo, as etapas de identificação dos artefatos existentes por meio da revisão sistemática da literatura e de conscientização do problema foram desenvolvidas concomitantemente.

Tabela 6: Fases e etapas da revisão sistemática da literatura

Fase	Passos	Descrição
Definições iniciais	1	A questão de pesquisa e objetivo foram definidos
	2	Um comitê de validação foi estabelecido, composto por dois pesquisadores, um especialista em DT e outro em metodologias de gerenciamento ágil de projetos
	3	Foi elaborado um protocolo de pesquisa, contendo: - Ferramentas para acessar informações nas bases de dados WoS e *Scopus*; - Palavras pesquisadas ("*design thinking*" ou "*human cent * design*" e "*agile*" e "*information system*" ou "*software*" e "*project **" no título, resumo ou palavras-chave); - Áreas de escopo (Ciência da Computação, Engenharia, Matemática, Negócios, Gestão e Contabilidade, Ciências da Decisão, Ciências Sociais, Ciência dos Materiais, Energia e Economia, Econometria e Finanças); - Informações que serão capturadas de cada artigo, como exemplo, autor, ano de publicação, periódico, resumo, palavras-chave; - Critérios de inclusão: o artigo deve combinar DT e pelo menos um método ágil; deve ter sido publicado nos últimos dez anos; e detalhar as características da integração deles; - Critérios de exclusão; - Idioma dos artigos selecionados; - Critérios de qualidade da literatura a ser pesquisada.
Pesquisa	4	Na pesquisa bibliográfica, foram retirados alguns registros, pois haviam estudos publicados em conferências sem referências de autores e alguns artigos foram identificados em ambas as bases de dados. Após essa etapa, selecionamos artigos para a

Fase	Passos	Descrição
		leitura do resumo.
	5	Nesta etapa, foram removidos artigos referentes a diferentes assuntos que não retratavam desenvolvimento de *software*, resultando em um subconjunto de 66 artigos selecionados. Dentre os assuntos não considerados na revisão sistemática estão: métodos educacionais, estrutura organizacional, segurança da informação, computação em nuvem, arquitetura de referência e comportamento profissional. Todos esses aspectos foram incluídos nos critérios de exclusão da revisão sistemática da literatura.
	6	Artigos que não faziam referência ao uso de uma metodologia que integrava a abordagem DT com métodos ágeis durante o ciclo de vida de desenvolvimento de *software*. Deste modo, 37 artigos foram removidos, resultando na lista de 29 artigos finalmente selecionados.
	7	A extração de dados feita neste passo incluiu: os autores, a referida revista, conferência ou livro, o ano de publicação, o objeto do estudo, o resumo, o tipo de problema estudado na pesquisa, o tipo de *software* desenvolvido, a abordagem DT utilizada, o método ágil aplicado, se o artigo sugere um novo modelo, as fases de tipo de estudo (avaliação prática, conceitual ou de modelo) e quando a abordagem foi utilizada ao longo do ciclo de vida de desenvolvimento de *software*, referenciando em quais fases o modelo foi aplicado.
Avaliação	8	Nesta etapa, ao rever criticamente os artigos primários, foram necessárias mais informações para verificar os métodos ágeis aplicados simultaneamente com DT durante um ciclo de vida de desenvolvimento de *software*;
	9	A sintetização dos casos selecionados e a complementação com base na literatura disponível para fundamentar as conclusões foram as atividades realizadas neste passo
	10 e 11	Foram escritas, nestes passos, as limitações, vieses e um relatório final da revisão sistemática da literatura

Fonte: Baseado em Patticrew e Roberts (2006)

3.3.3. Conscientização do Problema

A etapa de conscientização do problema prevê que o pesquisador busque o maior número de informações possível, possibilitando a compreensão das características, causas e contexto, formalizando as faces do problema identificado e a compreensão dos requisitos necessários para que o artefato solucione o problema em questão (Dresch et al., 2015). O objetivo nesta fase foi entender as necessidades gerenciais da Keep IT Simple na produção de *softwares* e as dificuldades do cliente em relação ao recebimento do resultado final do projeto.

Sendo assim, nesta fase, as informações coletadas nas entrevistas feitas para a identificação do problema, seção 3.3.1, foram classificadas e complementadas pelo conteúdo de três entrevistas com os responsáveis pelos projetos de *software*, no caso, profissionais que desempenham o papel de SM na Keep IT Simple, conforme ilustrado na Tabela 6. O roteiro é igual ao das entrevistas anteriores e está detalhado no Apêndice A. O conteúdo foi categorizado para identificar as necessidades, aprofundar os problemas a serem solucionados,

bem como os fatores positivos do processo atual de desenvolvimento de *software* que utiliza exclusivamente métodos ágeis. Além da coleta de dados obtida por meio de entrevistas foram consultados dois especialistas que auxiliaram na identificação de quais problemas poderiam ser solucionados por meio da aplicação da abordagem DT combinada com métodos ágeis. O E7 possui experiência em gestão de projetos de TI e é especialista no uso de métodos ágeis. Já o E8 é um consultor de gestão e inovação e possui especialização em abordagem DT. A presença de ambos os especialistas, também relacionados na Tabela 7 permitiu o aprofundamento nas características e complementariedades da integração entre essas abordagens.

Tabela 7: Características dos entrevistados da etapa de identificação dos problemas

Entrevistado	Experiência em TI (em anos)	Experiência em gestão de projetos	Cargo	Característica da empresa em relação ao *software* desenvolvido	Grau de instrução	Vivência na organização
E4	21 anos	4 anos	SM	Contratada	Graduação	Especialista em gestão de projetos
E5	20 anos	13 anos	SM	Contratada	Pós-graduado	Especialista em gestão de projetos
E6	12 anos	1 anos	SM	Contratada	Pós-graduado (em curso)	Especialista em gestão de projetos
E7	10 anos	18 anos	Consultor em gestão de projetos	Não se aplica	Mestrado	Especialista em gestão de projetos
E8	15 anos	17 anos	Consultor de gestão e inovação	Não se aplica	Pós-graduado	Especialista em abordagens DT

Fonte: Autor

3.3.4. Configuração da classe de problemas

Nesta etapa são identificados os artefatos (constructos, modelos, métodos, instanciações ou *design propositions*) que auxiliarão a pesquisa na proposta de desenvolvimento de novos artefatos (Dresch et al., 2015). A correta configuração da classe de problemas permite (i) assegurar que a pesquisa ofereça uma contribuição relevante e (ii) definir o alcance das contribuições do artefato (Dresch et al., 2015).

Ao realizar a identificação dos artefatos existentes por meio da revisão sistemática da literatura, segunda etapa do método de pesquisa, foi possível identificar se havia ou não um artefato que possibilitasse a resolução dos problemas identificados nos *softwares* desenvolvidos pela Keep IT Simple. A classe de problemas nesta pesquisa, portanto, é definida como **um método de gestão de projetos que integre DT e método ágil para o desenvolvimento de *softwares***.

3.3.5. Proposição do Artefato

Durante a etapa que constitui a proposição do artefato para a resolução do problema o pesquisador reflete sobre em que situação o problema ocorre e quais seriam as soluções que alteram ou aperfeiçoam a situação atual por meio da proposta de soluções (Dresch et al., 2015). Nesta fase, portanto, com base nos problemas identificados nas etapas de identificação e conscientização dos problemas, os modelos existentes foram avaliados de acordo com os problemas coletados e categorizados nas etapas de identificação e conscientização do problema, seções 3.3.1 e 3.3.3, respectivamente.

3.3.6. Projeto do Artefato

O projeto do artefato deve considerar as características internas e o contexto no qual a pesquisa ocorrerá, considerando seus componentes, relações internas de funcionamento, limites e relações externas (Dresch et al., 2015). Sendo assim, o projeto do artefato deve avaliar as soluções formalizadas na etapa anterior e que conduzam satisfatoriamente o estudo do problema (Dresch et al., 2015). Portanto, o projeto do artefato requer uma base teórica adequada, sem enfatizar apenas questões técnicas, mas que seja igualmente útil nas configurações das organizações que pertencem ao campo de estudo (Hevner et al., 2004). O artefato construído nesta pesquisa deve possuir as características definidas por Gill e Hevner (2013) para que a eficácia possa ser repetida e a utilidade da *Design Science Research* esteja alinhada com o problema a ser solucionado:

- Aplicabilidade: o artefato deve ser possível de ser aplicado no contexto para o qual o problema foi identificado, ou seja, para projetos de desenvolvimento de *software*, sendo capaz de resolvê-lo sem maiores adaptações e com alta expectativa de vida.

- Decomposição: o artefato deve possibilitar sua decomposição em partes menores, permitindo que seja aplicado parcialmente.

- Flexibilidade: o artefato deve permitir sua adaptação para que seu uso seja o mais efetivo possível diante do contexto de uma organização.

- Reutilização: os componentes do artefato devem ter a capacidade de serem inspecionados, modificados e possível de reutilização no contexto em que os mesmos são aplicados.

- Evolução: o artefato deve ser capaz de ter novos componentes incorporados ao seu modelo prévio para que seja rapidamente adaptado.

- Inovação: o artefato requer a capacidade de ser reinventado a partir do contexto em que o mesmo está sendo aplicado, levando em conta as características do ambiente

- Interesse: o artefato é gerado para explorar e demonstrar um propósito, sendo que por vezes ressalte o interessante para outros pesquisadores para subsequente verificação.

- Elegância: o artefato deve possuir como características a compactação, simplicidade, transparência de uso e comportamento e clareza em sua representação.

Com o objetivo de validar as características que deveriam estar presentes no artefato, ou seja, no método que integra a abordagem DT e métodos ágeis a partir de um modelo foram realizadas duas entrevistas de modo a entender como esse método poderia ser usado nos projetos de desenvolvimento de *software*. Tais entrevistas foram conduzidas por um roteiro previamente elaborado, presente no Apêndice B, de modo a capturar as percepções de resolução dos problemas encontrados nos projetos de desenvolvimento de *software* por meio deste método. Este roteiro continha as informações disponíveis até o presente momento da pesquisa, que compreendiam: (i) o mapeamento dos problemas mapeados ao longo dos projetos de desenvolvimento de softwares na Keep IT Simple; (ii) o modelo de gestão de projetos integrando DT e métodos ágeis, bem como sua associação com os problemas, e; (iii) a apresentação preliminar do método integrando DT e métodos ágeis contendo: processos, entradas, ferramentas e técnicas, participantes e saídas. Nesta etapa os entrevistados, relacionados na Tabela 8, foram selecionados considerando que houvesse pelo menos um representante da contratada para o desenvolvimento do *software* (E9) e um outro representante da contratante (E10), cliente do *software*.

Tabela 8: Características dos entrevistados para a validação do projeto do artefato

Entrevistado	Experiência em TI (em anos)	Experiência em gestão de projetos (em anos)	Cargo	Característica da empresa em relação ao software desenvolvido	Grau de instrução	Vivência na organização
E9	19 anos	7 anos	CEO	Contratada	Mestrado	Especialista em sistemas da informação
E10	12 anos	9 anos	Gerente de desenvolvimento	Contratante	Pós-graduado	Gerente técnico de sistemas

Fonte: Autor

3.3.7. Desenvolvimento do Artefato

Os resultados desta etapa compreendem o artefato em seu estado funcional e também a heurística da construção, principais saídas apontadas por Dresch, Lacerda e Antunes Jr (2015). Ao definir a heurística da construção do artefato, o pesquisador define os requisitos necessários para o seu desenvolvimento considerando os aspectos internos e externos à organização, permitindo que novos artefatos sejam projetados ou para melhorias no artefato construído (Dresch et al., 2015). O método integrado da abordagem DT com métodos ágeis, que compreende o artefato desta pesquisa, teve sua primeira versão desenvolvida a partir do modelo proposto por Bosch e Bosch-Sijtsema (2011). Este modelo foi utilizado em função da aderência entre os problemas relatados, mapeados e categorizados na Keep IT Simple e o relato dos resultados obtidos após a aplicação desse modelo pelos autores Bosch e Bosch-Sijtsema (2011) em uma empresa de desenvolvimento de *software*.

Grupos focais são métodos eficazes para avaliação de artefatos provenientes da *Design Science Research*, uma vez que permitem a avaliação da utilidade, qualidade e eficácia do artefato (Gibson & Arnott, 2007). Adicionalmente, os grupos focais também são reconhecidos como uma abordagem relevante e rigorosa para o refinamento e validação dos artefatos (Tremblay, Hevner, & Berndt, 2010). Deste modo, com o objetivo de assegurar que o desenvolvimento do método considerasse os aspectos necessários para a resolução dos problemas, foi realizado, nesta etapa, o primeiro grupo focal, de caráter exploratório que possui aspectos de melhorias incrementais na composição do projeto do artefato (Tremblay et al., 2010). O grupo focal exploratório possui como principais objetivos (i) obter *feedbacks* que serão utilizados na mudança do projeto do artefato e (ii) o refinamento do roteiro e da identificação dos construtos que serão utilizados em outros grupos focais (Tremblay et al., 2010). Sendo assim, os pesquisadores desenvolveram uma proposta preliminar do método para servir de orientação aos entrevistados e, deste modo, oferecer subsídios para a coleta dos

feedbacks dos participantes do grupo focal exploratório permitindo que esse método fosse refinado e tivesse os construtos aperfeiçoados.

O primeiro grupo focal realizado permitiu que os participantes pudessem expor as necessidades e características do método, buscando a identificação dos aspectos necessários ao método para a resolução dos problemas identificados nas etapas anteriores da pesquisa. O grupo focal exploratório teve duração de uma hora e 50 minutos, seu conteúdo foi gravado e a transcrição do mesmo gerou um documento de 58 páginas. Este grupo focal teve a participação de oito profissionais da área de TI, sendo quatro representantes da empresa desenvolvedora de *software* e quatro representantes de clientes desta empresa, conforme pode ser visto na Tabela 9. O critério de seleção dos participantes dos grupos focais considerou a relação dos entrevistados com os aspectos do artefato (Tremblay et al., 2010) que foi desenvolvido no presente estudo.

Tabela 9: Características dos participantes do Grupo Focal 1

Entrevistado	Experiência em TI (em anos)	Experiência em gestão de projetos (em anos)	Cargo	Característica da empresa em relação ao software desenvolvido	Grau de instrução	Vivência na organização
GF1.1	10 anos	18 anos	Consultor em gestão de projetos	Contratante	Mestrado	Especialista em gestão de projetos
GF1.2	24 anos	11 anos	*Product Owner*	Contratante	Pós-graduado	Especialista em sistemas da informação
GF1.3	21 anos	1 ano	*Product Owner*	Contratante	Pós-graduado	Especialista em sistemas da informação
GF1.4	14 anos	5 anos	Arquiteto de soluções	Contratada	Pós-graduado	Especialista em sistemas da informação
GF1.5	19 anos	7 anos	CEO	Contratada	Mestrado	Especialista em sistemas da informação
GF1.6	12 anos	1 ano	*Scrum Master*	Contratada	Pós-graduado (em curso)	Especialista em gestão de projetos
GF1.7	12 anos	9 anos	Gerente de desenvolvimento	Contratante	Pós-graduado	Gerente técnico de sistemas
GF1.8	21 anos	4 anos	*Scrum Master*	Contratada	Graduação	Especialista em gestão de projetos

Fonte: Autor

3.3.8. Avaliação do Artefato

Na fase de avaliação do artefato são verificados os resultados dos projetos submetidos ao uso da abordagem integrada, por meio de indicadores definidos anteriormente. Sendo assim, a avaliação do artefato desta pesquisa foi feita por meio da realização de um segundo grupo focal, de caráter confirmatório. Conforme Tremblay *et al.* (2010) o grupo focal confirmatório promove a demonstração da utilidade do artefato, embora este possua métodos de análise de dados similares em relação ao grupo focal exploratório. A Figura 10 ilustra a relação do ambiente com a *Design Science Research*, bem como o relacionamento entre os aspectos que envolvem a execução dos grupos focais exploratório e confirmatório. Nesta ilustração é possível identificar a relação da *Design Science Research*, na qual ocorre a construção do projeto do artefato por meio da avaliação do artefato por um grupo focal exploratório, e o ambiente, onde é realizado o grupo focal confirmatório no campo de teste do artefato gerado.

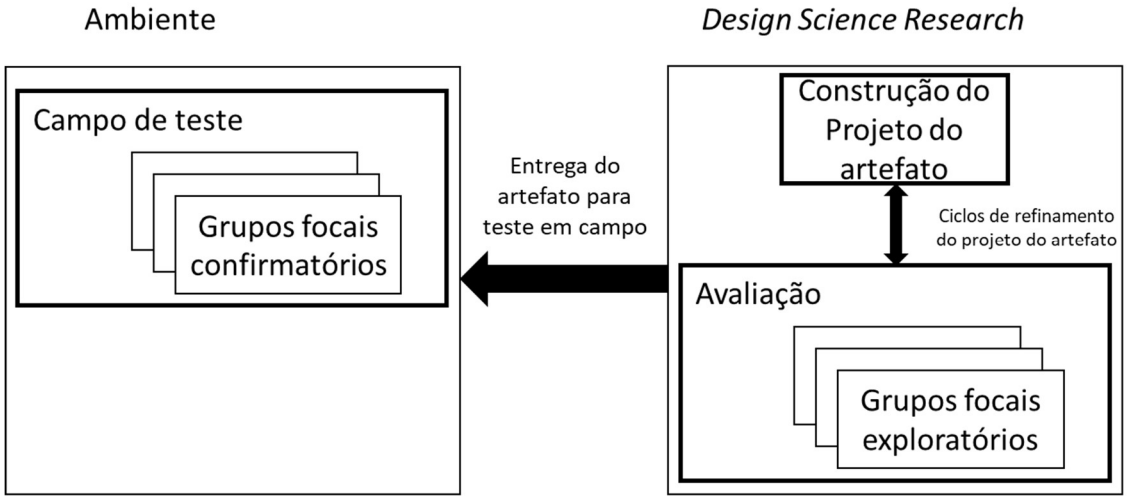

Figura 10. Grupos focais na *Design Science Research*
Fonte: Tremblay et al., 2010, p. 603

O segundo grupo focal teve como objetivo principal referendar as informações previamente estabelecidas e, assim, confirmar a composição do modelo integrado com a abordagem DT e métodos ágeis na gestão de projetos de *software*. Embora o segundo grupo focal tenha contado com cinco participantes, todos estiveram presentes no primeiro grupo focal. O segundo grupo focal confirmatório teve duração de uma hora e 27 minutos e sua gravação foi transcrita em um documento de 41 páginas.

A participação dos cinco componentes no segundo grupo focal assegurou a presença de representantes de ambas as empresas (contratada e contratante), sendo três representantes da contratante (cliente do *software*) e dois representantes da contratada, no caso a Keep IT Simple. Portanto, assim como ocorreu no grupo focal exploratório, dentre os participantes existiam representantes da empresa que desenvolve o *software*, no caso a Keep IT Simple, e a empresa cliente, no caso, definida também como a contratante do *software*. Na Tabela 10 é possível visualizar que todos os cinco participantes do segundo grupo focal, de caráter confirmatório, que estiveram presentes no grupo focal exploratório. Visto que o desenho final do artefato é o objetivo do segundo grupo focal (Tremblay et al., 2010), nota-se a importância de contar com os mesmos participantes do primeiro grupo focal.

Tabela 10: Relacionamento entre os participantes dos dois grupos focais

Entrevistado Grupo Focal Exploratório	Entrevistado Grupo Focal Confirmatório
GF1.1	GF2.1
GF1.2	GF2.2
GF1.3	GF2.3
GF1.4	GF2.4
GF1.5	GF2.5

Fonte: Autor

3.3.9. Exposição da Aprendizagem e Conclusões e Generalização para uma Classe de Problemas

Os resultados em si são descritos nesta fase em termos práticos e teóricos para a geração de conhecimento, assim como foram descritas as limitações da pesquisa. O objetivo destas etapas é assegurar que a pesquisa pode servir como referência e como subsídio para a geração do conhecimento (Dresch et al., 2015). Deste modo, empresas ou áreas responsáveis pelo desenvolvimento de *software* poderão ser capazes de utilizar o método estabelecido.

A generalização da classe de problemas consiste na possibilidade de aplicar o método desenvolvido nesse trabalho em outras organizações na gestão de projetos de desenvolvimento de *software*. Deste modo, o conhecimento gerado nas fases anteriores será generalizado para a classe de problema possibilitando sua aplicação em situações similares (Dresch et al., 2015). Os resultados serão divulgados tanto em periódicos acadêmicos como em meios voltados para a aplicação prática.

3.4 ANÁLISE DE DADOS

De modo a permitir a descrição do problema pesquisado, é necessário documentar e descrever o fenômeno de interesse, permitindo que as ações, eventos, comportamentos, atitudes dos participantes do processo sejam mapeados (Marshall & Rossman, 2014). Em função disto, foram utilizadas as fases sugeridas por Marshall e Rossman (2014) na análise das entrevistas e grupos focais:

- Os dados foram organizados em planilha Excel;
- As entrevistas foram transcritas e documentadas pelo autor;
- Os grupos focais exploratório e confirmatório foram gravados, transcritos e documentados pelo autor;
- O autor analisou os dados, de modo a identificar os destaques informados por cada respondente;
- Os dados foram classificados com base nas categorias de problemas que ocorrem ao longo da execução do projeto e de problemas que impactam na qualidade do produto, no caso o *software*;
- As categorias foram interpretadas e, de acordo com o padrão, foram reorganizadas de modo a permitir a identificação dos problemas e suas respectivas alternativas de soluções com base nas assertivas coletadas nas entrevistas e grupos focais;
- Por fim, as categorias foram comparadas com a teoria para gerar os resultados e as conclusões do presente estudo.

4 RESULTADOS

Este capítulo apresenta a análise das informações obtidas ao longo da execução de cada etapa da pesquisa.

4.1 IDENTIFICAÇÃO DO PROBLEMA NA EMPRESA DE DESENVOLVIMENTO DE SOFTWARE

Na etapa de pesquisa identificação do problema, descrita na seção 3.3.1, foram realizadas três entrevistas com um representante da Keep IT Simple e com dois representantes da empresa demandante do *software*, respectivamente, fornecedora e cliente. Deste modo, as entrevistas permitiram mapear os problemas existentes durante a execução dos projetos de desenvolvimento de *software* sob a perspectiva do contratante e da contratada. Nesta primeira etapa foram identificados oito problemas: (i) complexidade das funcionalidades e componentes dos sistemas envolvidos na solução; (ii) usuário final não participa da captura dos requisitos; (iii) complexidade nos objetivos de negócio; (iv) estórias incompletas; (v) baixa maturidade do *software* implantado em ambiente de produção; (vi) qualidade do software; (vii) gestão de projetos, relacionado ao ciclo de vida do desenvolvimento do *software*, e; (viii) time do projeto, considerando os profissionais envolvidos no desenvolvimento do *software*. Estes oito problemas, então, foram agrupados em cinco categorias: (i) engenharia de *software*; (ii) escopo; (iii) planejamento; (iv) qualidade, e; (v) organizacional.

A partir deste mapeamento, foi possível realizar a identificação dos problemas que seriam ou não solucionados por meio do uso de um método de gestão de projetos que integre a abordagem DT com métodos ágeis para o desenvolvimento de *software*. A Tabela 11 contém a relação dos problemas, categorias e assertivas dos entrevistados ao longo da identificação dos problemas na Keep IT Simple. Os problemas e aspectos positivos do uso do método ágil atualmente empregado no desenvolvimento de *software* constam na íntegra no Apêndice C deste documento.

Tabela 11: Relato dos entrevistados na etapa de identificação dos problemas

Problema	Categoria	Assertiva dos entrevistados
Complexidade das funcionalidades e componentes dos sistemas envolvidos na solução	Engenharia de software	E1: "existem diferentes plataformas a serem conectadas: Banco de Dados (BD), Cobol (Alta Plataforma), APIs (Java) e FrontEnd (Angular)." E2: "Ponto crítico do projeto é acoplar as novas funcionalidades em sistema de alta plataforma, que possui algumas restrições. O modelo de solução do sistema previsto para o projeto teve que ser moldado ao longo do desenvolvimento. Determinadas comunicações e tratamento de dados precisaram ser isolados para que não houvesse interrupção dos demais sistemas organizacionais que permaneceriam inalterados." E2: "Arquitetura da empresa hoje não está preparada para avaliar novas opções tecnológicas e acaba atravancando potenciais aceleradores." E2: "Maturidade da área está aquém do que é necessário e o time do projeto, de uma maneira geral, não estava capacitado." E2: "Conceito de DevOps (*development and operations*, relação ágil entre desenvolvimento e sistemas em produção) está sendo implantado na organização agora, mas ainda embrionário." E3: "... é difícil integrar todos as áreas de negócios e áreas de TI dispersas..." E3: "A junção das entregas de cada um desses times para que o sistema funcionasse de forma integrada foi apontada como um dos grandes problemas do projeto."
Usuário final não participa da captura dos requisitos	Escopo	E1: "a necessidade vem pré-moldada dos *Business Partners* (BP's) que cuidam da captura das necessidades (integrante do time de TI que atende o negócio)...."lacuna entre o que está sendo pedido e o que o cliente realmente quer" E1: "não há clareza nos requisitos do *software*" E2: "Diversas vezes os usuários do sistema exigem que todas as funcionalidades já existentes (*as is*) devem ser incorporadas ao projeto, sendo que algumas podem ter menor relevância. E2: "a área demandante exigiu que o pagamento em cheque fosse uma funcionalidade mandatória, sendo que na visão do (PO) conceder ao sistema o critério de multicanal teria maior valor ao negócio (venda iniciada no site e terminada na loja)." E3: "Não houve a participação dos usuários finais na definição dos requisitos do software que futuramente seriam utilizados por vendedores, analistas de crédito, operadores de caixa, estoquistas e pessoal administrativo. Por esta razão, atualmente o sistema está implantado em aproximadamente 10% das lojas com pouca aderência de uso, visto que o sistema antigo ainda não foi continuado."
Complexidade nos objetivos de negócio	Escopo	E1: "Desafio está em integrar diferentes plataformas para atendimento dos desejos do negócio: *user experience* , áreas de TI, áreas corporativas e time de desenvolvimento" E2: "a atuação área de operações de vendas, demandante e patrocinadora do projeto não favorece a elaboração de um backlog estruturado e que o MVP [*minimum viable product*, menor produto viável] muitas vezes se torna inviável pela complexidade da funcionalidade do sistema que deveria ser entregue. De acordo com o entrevistado, os representantes da área demandante querem o estado da arte já na primeira entrega."
Estórias incompletas	Planejamento	E1: "Estórias estão hoje incompletas. *Inception* [planejamento] não conta com a participação nem do BP, nem de representante do negócio"..."houveram situações de "vai e volta", por exemplo, na definição do crediário." E2: "Cultura organizacional atual reflete a uma postura dos usuários e clientes do sistema como recebedores de algo, sem características de parceria". E3: "...distância entre o que o cliente desejava e o que tecnicamente era possível de ser feito e no curto prazo."

Problema	Categoria	Assertiva dos entrevistados
Baixa maturidade do *software* implantado em ambiente de produção	Qualidade	E2: "Processo de desenvolvimento e *deploy* [publicação] de programas em produção estão sendo realizados sem um modelo pré-estabelecido." E2: "...programas que ainda estariam em testes subiram para produção indevidamente. " E3: " Os problemas caracterizavam no aumento do tempo para os testes, grande número de inconformidades e até erros que eram colocados após a implantação do projeto para os usuários finais." E3: "Foi adotado um procedimento de revisão de código fonte entre os integrantes do time de desenvolvimento, buscando melhoria na qualidade do software."
Qualidade do *software*	Qualidade	E1: "Solicitações sendo feitas em ondas, sem tempo para planejamento prévio e além do escopo originalmente acordado." E2: "Em alguns casos, houve disputa sobre a responsabilidade dos erros ou itens em não conformidade identificados durante o projeto, sem a indicação de quem deveria arcar com o ônus das horas da correção." E3: "...grande número de reclamações associadas ao não uso correto do sistema (30% dos *feedbacks*) e de dúvidas (60%), o que pode indicar um problema de falta de treinamento do sistema, além da não participação dos usuários finais ao longo do processo de desenvolvimento do novo sistema."
Gestão do projeto	Planejamento	E1: "Dificuldade em gerenciar diferentes times (*squads*) dentro de um mesmo projeto" E2: "...a ausência de um processo estruturado que estabeleça as fases do ciclo de vida de um *software* (*inception, design, development, tests, user acceptance and production*) dificulta o alinhamento entre as diversas frentes do projeto." E3: "cada time de desenvolvimento possuía um especialista em *user experience* (UX) sendo que depois os componentes não possuíam um padrão, ao tentar integrá-los"
Time do projeto	Organizacional	E2: "É fundamental que o time do projeto contenha profissionais que conheçam o negócio da empresa. Caso não possua esse conhecimento dentro do time, será necessário contratar uma consultoria que traga boas práticas de mercado (nesse caso, varejo e comércio eletrônico)."

Fonte: Autor

Em relação à complexidade dos componentes e funcionalidades do *software* desenvolvido pela Keep IT Simple, os entrevistados relataram de forma similar problemas relacionados a quantidade de sistemas e tecnologias envolvidos na solução. O E1 informou que existem diversas plataformas a serem integradas para que uma solução de *software* seja entregue. O E2 concordou afirmando que “Determinadas comunicações e tratamento de dados precisaram ser isolados para que não houvesse interrupção dos demais sistemas organizacionais que permaneceriam inalterados”. O E3 endossou essa afirmação dizendo que “A junção das entregas de cada um dos times que integravam o time do projeto para que o sistema funcionasse de forma integrada foi apontada como um dos grandes problemas do

projeto". Tais problemas foram classificados na categoria "Complexidade das funcionalidades e componentes dos sistemas envolvidos na solução" presente na Tabela 11.

Sob e perspectiva da contratada para desenvolver o *software*, o entrevistado E1 disse que a falta de **clareza dos requisitos do *software*** é um dos principais desafios, visto que essa atividade exige a participação de profissionais da área de TI e de pessoas que representam os usuários. Os usuários propriamente ditos raramente participam das discussões que ocorrem no desenvolvimento de novos sistemas na empresa na qual o estudo está sendo realizado. Sob a perspectiva do cliente, contratante do desenvolvedor do *software*, o principal problema é a **ausência de alinhamento da priorização e integração das necessidades** entre as áreas demandantes e entre estas e os usuários finais.

Outro ponto relatado na entrevista está relacionado ao **desalinhamento entre a entrega do *software* e a expectativa existente do usuário final**. O entrevistado E1 informou que o processo de definição das histórias, que traduzem as funcionalidades do software, frequentemente apresentam **retrocesso de itens anteriormente já definidos** ("situações de 'vai e volta'"). Deste modo, o tempo que deveria ser designado para o desenho e desenvolvimento de novas funcionalidades do *software* é substituído por retrabalho nas entregas já realizadas.

O entrevistado E2 informou que a cultura organizacional atual reflete uma postura dos **usuários e clientes como demandantes e recebedores apenas**, ao invés de serem parte da solução buscada em parceria com a área de TI, representada pelo time de desenvolvimento de *softwares*. Adicionalmente, informou que a **atuação da área de operações de vendas**, demandante e patrocinadora do projeto e que representa o usuário final **não favorece a elaboração de um backlog estruturado** e que o **MVP muitas vezes se torna inviável pela complexidade da funcionalidade do sistema que deveria ser entregue.** De acordo com o entrevistado, os representantes da área demandante querem o sistema completo ("estado da arte") já na primeira entrega. Estes problemas foram categorizados como "Complexidade nos objetivos de negócio" presente na Tabela 11.

Adicionalmente, o entrevistado E2 informou que os benefícios mais significativos do projeto de desenvolvimento do novo *software* foram atingidos por entregas rápidas que possibilitaram testes de algumas funcionalidades em seu ambiente real, mesmo quando a solução ainda exigia intervenções operacionais para serem efetivas por ter baixa sofisticação. As entregas rápidas, embora não possuíssem integralmente todas as funcionalidades, foi possível diante de um cenário que permitia a convivência de um sistema de vendas precursor do sistema que estava sendo desenvolvido e que seria de fato implantado no futuro: "diversas

vezes os usuários do sistema exigem que todas as funcionalidades já existentes no *software* atual devem ser incorporadas ao projeto do novo sistema, sendo que algumas podem ter menor relevância", disse o entrevistado E2.

O entrevistado E2 cita, adicionalmente, que a área exigiu que a funcionalidade de pagamento em cheque, meio de pagamento em declínio em relação aos demais, fosse priorizada no novo sistema. Na visão do *Product Owner* (PO), a funcionalidade que permita os clientes prosseguirem com a compra de um produto na loja física após buscarem um produto no *website* da empresa, traria uma melhor experiência ao cliente, ao invés de conceder prioridade na habilitação do meio de pagamento em cheque nas lojas. Tal funcionalidade está relacionada ao termo multicanal, que permite que o cliente utilize diferentes canais de venda para realizar sua compra, estabelecendo como conceitos principais: troca de canal de compra, fidelidade e precificação, tornando a experiência do cliente integrada e mais atraente (Lazaris & Vrechopoulos, 2014). Deste modo, o cliente pode iniciar sua compra com a escolha do produto desejado ainda em sua residência no site da empresa de varejo e se dirigir à loja fisicamente para efetivar o pagamento e retirar o produto.

O entrevistado E3 aponta que os problemas estão, em parte, relacionados com a **distância entre o que o cliente desejava e o que tecnicamente era possível de ser feito e no curto prazo**. Como os requisitos eram definidos para TI por intermédio dos membros que compõem a área de operações de vendas, nomeados como clientes do *software*, não houve a participação dos usuários finais na definição das funcionalidades do sistema. Os usuários finais do *software* compreendem os vendedores, analistas de crédito, operadores de caixa, estoquistas e pessoal administrativo.

De acordo com o entrevistado E3, o sistema que estava implantado no momento da entrevista em aproximadamente 10% das lojas, tem com **pouca aderência ao uso**, visto que o sistema antigo ainda não foi descontinuado. Como a estratégia da organização é manter a operação funcionando e em função do novo sistema ainda estar em processo de melhorias e identificação de erros (ou *bugs*, como são conhecidas as falhas no *software*), o uso do novo sistema ainda não é mandatório. Nota-se, portanto, que um sistema com novas funcionalidades pode ser inferior a outro *software*, mesmo que obsoleto, por não atender as reais necessidades dos usuários finais. Tal informação foi classificada como "Qualidade de *software*" presente na Tabela 11.

Como meio para identificar se a baixa aderência poderia estar associada aos erros do *software*, a equipe de projetos contou com um grupo de recursos que visa receber *feedbacks* dos usuários da loja, seja pelo sistema ou por aplicativo de mensagens para que sejam

identificados, classificados e encaminhados ao time de desenvolvimento. Dependendo da severidade do erro, foi estabelecida a estratégia de implantação quase diária de *hot fixes* (reparos urgentes) quando se é identificado um erro de alta severidade. É notado também que há muitas reclamações associadas ao não uso correto do sistema (30% dos *feedbacks*) e de dúvidas (60%), o que pode indicar um problema de **falta de treinamento para uso do sistema**, além da não participação dos usuários finais ao longo do processo de desenvolvimento do novo sistema. Estes problemas foram classificados como "Qualidade do *software*" na Tabela 11.

Os desafios apontados pelos entrevistados podem ser endereçados por meio de maior profundidade na etapa do *design* do *software*, fazendo com que as decisões sejam tomadas quando uma visão mais ampla do desenho do software é obtida por todas as partes interessadas (Riis, 2012). A etapa de prototipação permite que os usuários finais possam avaliar o *software* no contexto real de uso por meio de exemplos (Ardito, Buono, Caivano, Costabile, & Lanzilotti, 2014), facilitando assim a identificação prévia de problemas. Portanto, a adoção de um modelo que integra DT com métodos ágeis permitirá que tais desafios sejam superados com a observação não só dos aspectos técnicos, mas com foco nas pessoas e na cultura organizacional, permitindo que significante valor seja fornecido à organização com os *softwares* desenvolvidos (O'Driscoll, 2016).

4.2 IDENTIFICAÇÃO DOS ARTEFATOS QUE INTEGRAM DESIGN THINKING E MÉTODOS ÁGEIS

4.2.1. Artigos selecionados

Inicialmente a busca ocorreu para que fossem encontrados métodos que integrassem a abordagem DT com métodos ágeis. Porém, na revisão sistemática da literatura, foram identificados modelos, e não métodos, que utilizavam a abordagem DT de forma integrada com métodos ágeis em projetos de desenvolvimento de *software*. Inicialmente foram encontrados 33 artigos na WoS e 111 na base de dados *Scopus*. 21 artigos foram descartados por serem artigos publicados em conferências sem mencionar autores, além de sete artigos terem sido identificadas em ambas as bases. Como resultado preliminar, 116 artigos foram selecionados para a leitura do resumo. Conforme o protocolo de pesquisa mencionado na seção 3.3.2, no passo cinco, 50 artigos foram removidos por não ter como tema principal o

desenvolvimento de *softwares*, resultando em 66 artigos. Deste total, 37 artigos não faziam referência ao uso de um modelo ou método que integrasse DT e métodos ágeis ao longo de um projeto de desenvolvimento de *software*, resultando em 29 artigos. Todos os artigos estão catalogados na Tabela 12, contendo: (i) autor(es) da pesquisa, (ii) ano em que o estudo foi publicado em periódico ou congresso, (iii) qual a abordagem DT referenciada na pesquisa, (iv) qual o método ágil referenciado na pesquisa, (v) qual o tipo de *software* submetido ao uso do modelo referenciado na pesquisa, (vi) se o resultado da pesquisa sugere um modelo, (vii) qual o tipo de estudo, classificado em caso prático, conceitual ou avaliação de modelo, e (vii) fases do ciclo de vida do *software* no qual o modelo foi utilizado: ao longo de todo o ciclo de desenvolvimento do *software*, aplicado somente nas fases de desenho e requisitos do software, aplicado apenas nas fases de aceitação e testes de implantação ou não especificado.

Tabela 12: Modelos que integram a abordagem DT com métodos ágeis

Autor	Ano	Abordagem DT	Método ágil	Tipo de *software*	Modelo sugerido	Tipo de estudo	Ciclo de vida do *software*
(Higuchi & Nakano, 2017)	2017	IDEO	Scrum	Aplicativo de jogos	Sim	1	2
(Lucena et al., 2016)	2017	DSchool	Scrum	Portal de vendas e outras aplicações	Sim	1	1
(Darrin & Devereux, 2017)	2017	DSchool	Não especificado	Não especificado	Sim	2	2
(Gamble, 2016)	2016	Não especificado	SAFe	Não especificado	Sim	2	2
(Paula & Araujo, 2016)	2016	IDEO	Lean	Mobile game	Sim	1	1
(Newman et al., 2015)	2015	Não especificado	Não especificado	Aplicação *web based*	Não	1	1
(Bosch & Bosch-Sijtsema, 2011)	2011	Não especificado	Não especificado	Solução empresarial integrada	Sim	1	1
(Forbrig, 2016b)	2016	ISO	Scrum	Não especificado	Sim	2	1
(Schön, Winter, Uhlenbrok, Escalona, & Thomaschewski, 2016)	2016	ISO	Kanban	Internet Portal	Não	1	1
(Forbrig & Saurin, 2016)	2016	ISO	Scrum	Aplicação *web based*	Sim	1	2
(Fischer & Senft, 2016)	2016	ISO	AMDD	Software de planejamento	Sim	1	1
(Steinke et al., 2017)	2017	DSchool	Scrum	Não especificado	Sim	2	2
(Gurusamy, Srinivasaraghavan, & Adikari, 2016)	2016	DSchool	Não especificado	Não especificado	Sim	2	1
(Ardito et al., 2017)	2017	Não especificado	Scrum	Portal (*web based*)	Sim	1	1
(Sebok, Walters, & Plott,	2017	ISO	Similar ao ágil e	Sistema de	Sim	1	1

Autor	Ano	Abordagem DT	Método ágil	Tipo de *software*	Modelo sugerido	Tipo de estudo	Ciclo de vida do *software*
2017)			SAFe	segurança e missão crítica			
(Forbrig, 2016a)	2016	ISO	Scrum	Não especificado	Sim	2	2
(Ximenes et al., 2015)	2015	DSchool	XP, Lean	Aplicativo de armazenagem de dados	Não	1	1
(Cano, González, Collazos, Arteaga, & Zapata, 2015)	2015	ISO	XP	Aplicativo de jogos	Não	2	1
(González-González et al., 2015)	2015	ISO	Scrum e Lean	Sistema educacional *web based*	Não	1	1
(Isa et al., 2014)	2014	Não especificado	XP	Sistema *web based*	Não	1	1
(Lárusdóttir et al., 2014)	2014	ISO	Scrum	Não especificado	Não	3	4
(Prior et al., 2013)	2013	Não especificado	Scrum	Dispositivo de assistência tecnológica	Não	1	1
(Sohaib & Khan, 2011)	2011	Não especificado	XP	Não especificado	Sim	2	3
(Lester, 2011)	2011	ISO	XP	*Website*	Não	1	2
(Losada et al., 2011)	2011	Não especificado	AMDD	Não especificado	Sim	2	1
(Bourimi, Barth, Haake, Ueberschär, & Kesdogan, 2010)	2010	Não especificado	Scrum	Sistema educacional web	Sim	2	4
(Xiong & Wang, 2010)	2010	Não especificado	Scrum	Aplicativo *mobile*	Sim	2	1
(Adikari, McDonald, & Campbell, 2009)	2009	Não especificado	Não especificado	Não especificado	Sim	2	2
(Hussain et al., 2008)	2008	Não especificado	XP	Aplicativo de gestão de conteúdo	Sim	1	1

Nota1: Tipo de Estudo 1 é caso prático, 2 é conceitual e 3 é uma avaliação de modelo.

Nota2: Como a abordagem DT é integrada com métodos ágeis durante o ciclo de vida do *software*: 1 - ao longo de todo o ciclo de desenvolvimento do software, 2 - aplica-se somente nas fases de desenho e requisitos do software, 3 - aplicado apenas nas fases de aceitação e testes de implantação, 4 - Não especificado.

Fonte: Autor.

Dentre os artigos encontrados, o modelo da abordagem DT proposto pela ISO apresentou maior incidência, sendo identificada em dez ocorrências, seguida pelos modelos propostos pela DSchool e IDEO, com cinco e dois artigos, respectivamente, como pode ser notado na Figura 17. Este dado está presente na coluna “Abordagem DT” da Tabela 12. Em 12 artigos a abordagem DT não foi especificada pelo fato de os autores realizarem referência a uma abordagem centrada no usuário do *software*, sem necessariamente mencionar a

abordagem DT de forma explícita. Os resultados refletem a variedade da aplicação dos modelos que integram a abordagem DT e métodos ágeis no segmento de SI. A abordagem DT "promove requisitos e recomendações para os princípios e atividades para o desenvolvimento de sistemas computadorizados com atividades que buscam a centralização nas necessidades humanas ao longo do ciclo de vida" (DIS, 2010).

Assim como estabelecido na coluna "Método ágil" presente na Tabela 12, a revisão sistemática da literatura mostrou que o Scrum é o método ágil de gestão de projetos mais frequentemente integrado com a abordagem DT no desenvolvimento de *software*, sendo referenciado em 11 artigos (38%), como consta na Figura 11. Adicionalmente a maioria dos artigos que utilizam o Scrum como método ágil integrado com a abordagem DT expõe sua aplicação na prática, na condução de um projeto de desenvolvimento de *software*, ao longo de todo o ciclo de vida, não só nas fases iniciais de especificação e requisitos do *software*.

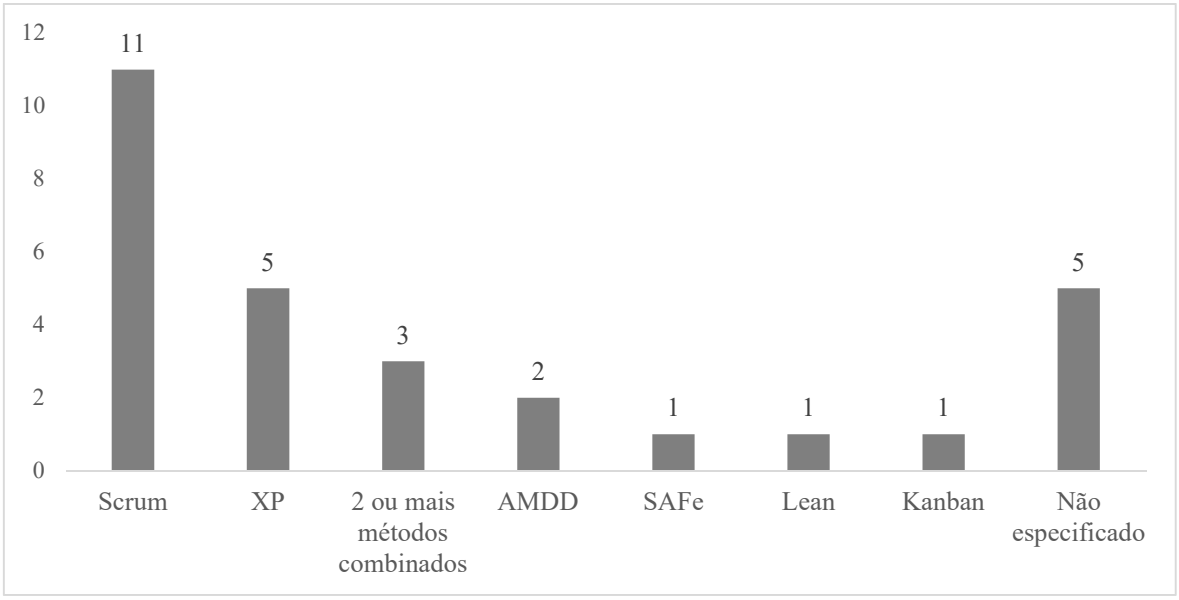

Figura 11. Distribuição dos métodos ágeis aplicados de forma integrada com DT
Fonte: Autor.

Ao desconsiderar os artigos que não explicitam a abordagem DT usada de forma integrada com métodos ágeis, a abordagem DT proposta pela ISO com o Scrum é a que aparece com maior frequência dentre os artigos pesquisados (cinco ocorrências), sendo que um dos artigos ainda mencionam o uso simultâneo com Lean. Outros dois artigos mencionam o uso da abordagem DT da ISO com o método ágil XP, a mesma quantidade de ocorrências do modelo integrado com uso da abordagem DT da DSchool integrada com Scrum. A Figura

12 mostra a distribuição da combinação da abordagem DT com métodos ágeis dos artigos encontrados na revisão sistemática da literatura.

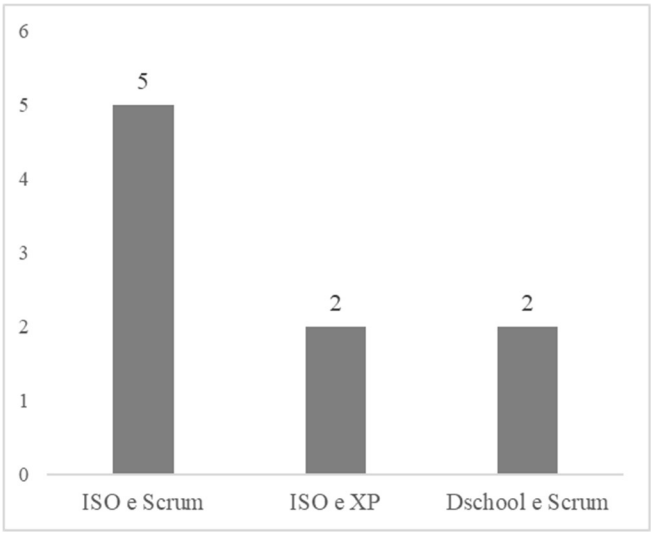

Figura 12. Artigos que referenciam modelos combinando a abordagem DT e métodos ágeis
Fonte: Autor.

4.2.2. Características dos modelos integrados da abordagem DT com métodos ágeis

Embora tenha havido variações entre os tipos de *softwares* desenvolvidos referenciando a integração da abordagem DT e métodos ágeis, incluindo aplicativos para celular, *softwares* organizacionais e sistemas *web based*, por exemplo, diversos autores expuseram o modelo aplicado. Os usuários dos *softwares* desenvolvidos ficaram satisfeitos e suas necessidades foram atendidas (Schön et al., 2016) como resultado de produtos e serviços mais bem ajustados às necessidades dos clientes (Bourimi et al., 2010; Isa et al., 2014; Steinke et al., 2017). A qualidade do *software* entregue foi aumentada pelo fato das mudanças terem sido gerenciadas apropriadamente quando os desafios e requisitos foram descobertos (Ardito et al., 2017; Gamble, 2016), seguido pelos frequentes testes para ver se com as funcionalidades do *software* estavam sendo implementadas (Sebok et al., 2017). Usuários também acharam o *software* mais fácil de ser usado e, portanto, menor nível de suporte foi requerido (Adikari et al., 2009). O modelo da abordagem DT integrada com métodos ágeis, portanto, apresentou benefícios sem estar associado a nenhum tipo específico de *software*.

O uso da abordagem DT promove a comunicação entre as equipes de desenvolvimento de *software* e os clientes ao longo de todo ciclo de vida do projeto. Os aspectos da abordagem

DT são caracterizados por empatia, definição e prototipagem rápida, porém apresentam variações. Higuchi e Nakano (2017), Percival et al. (2016) e Gamble (2016) estabelecem, em seus artigos, modelos de desenvolvimento de *software* que integram a abordagem DT e métodos ágeis. Diante das variações que se apresentam, é possível notar que os modelos se alternam (i) na variação do uso da abordagem DT, utilizando as disponíveis na literatura; (ii) no método ágil utilizado como subsídio para a aplicação da gestão do projeto de *software*; (iii) no ciclo de vida do *software* no qual a abordagem é aplicada e (iv) na combinação entre outras abordagens, tais como Lean e TOGAF, que não necessariamente são abordagens utilizadas para o desenvolvimento de *software*.

O modelo proposto por Forbrig (2016b) que utiliza a abordagem DT divulgada pela ISO (DIS, 2010) e Scrum (Schwaber, 2004), como método ágil, estabelece inúmeros ciclos para o desenvolvimento do *software*, definindo as responsabilidades dos atores: analista, desenvolvedor e PO. Este ciclo inicia com a visão e as necessidades que devem ser obtidas junto aos usuários e fazem com que os aspectos humanos sejam mais importantes, tornando a usabilidade do sistema e a experiência do usuário fatores chave de sucesso ou falha do *software* desenvolvido. É estabelecido então um modelo iterativo, ou seja, considera ciclos únicos ou que podem ser executados diversas vezes para gerar o produto do projeto em conformidade com os requisitos, podendo ser expandido para o contexto inovador, não se limitando ao desenvolvimento de *software* (Forbrig, 2016b). Gurusamy et al. (2016) estabelecem um *framework* que integra DT com métodos ágeis para o desenvolvimento de *software*. Nesta pesquisa, os autores combinam as características da abordagem DT e métodos ágeis com o intuito de elaborar e revisar o desenho da solução e requisitos por todo o ciclo de desenvolvimento de *software*, incluindo a etapa de desenvolvimento e testes. Deste modo, os conceitos de desenho e requisitos do *software* e avaliação da abordagem DT são incorporados às etapas de desenho e de requisitos previstos no uso de métodos ágeis para o desenvolvimento de SI (Gurusamy, Srinivasaraghavan, & Adikari, 2016).

4.3 CONSCIENTIZAÇÃO DO PROBLEMA

O diagnóstico sobre o problema do *software* desenvolvido considerou os aspectos informados pelos entrevistados na etapa de identificação do problema, sob a perspectiva da contratada (fornecedor e desenvolvedor do *software*) e da contratante (cliente e recebedora do *software*). Para que os problemas previamente identificados fossem confirmados como

problemas reais, foram realizadas três entrevistas adicionais com os profissionais que desempenham o papel de SM na Keep IT Simple, conforme diretrizes da etapa de pesquisa descrita na seção 3.3.3. As assertivas coletadas por meio das entrevistas realizadas nessa etapa, assim como as referenciadas na seção 4.1, foram classificadas em seis categorias: escopo, planejamento, qualidade, testes, ambiente organizacional e engenharia de *software*.

Das categorias utilizadas para a classificação dos aspectos positivos e problemas identificados, quatro estão relacionadas ao método de gestão de projetos que será utilizado, portanto, serão escolhidas para delinear o artefato necessário para resolução do problema: (i) escopo, que consiste nas características do *software*; (ii) planejamento, referente aos aspectos de estrutura e sequenciamento das atividades para a entrega do *software*; (iii) qualidade, relativo aos critérios de aceitação do *software*; (iv) testes, que aborda as atividades de validação e entrega do *software* em conformidade com o que foi desenhado; (v) organizacional, sendo a categoria referente ao ambiente organizacional e que remete aos aspectos de cultura, comportamento, metas e gestão, e por fim; (vi) engenharia de *software*, que corresponde aos aspectos relacionados à infraestrutura de TI e operação dos sistemas. Portanto, a Tabela 13 contém os aspectos positivos derivados do uso exclusivo de método ágil nos sistemas desenvolvidos pela Keep IT Simple sob a perspectiva da contratada e do contratante do *software*, caracterizados como fornecedor e cliente, respectivamente. Os aspectos positivos, assim como ocorreu com os problemas foram caracterizados em: escopo, ambiente organizacional, qualidade, planejamento e engenharia de *software*.

Tabela 13: Aspectos positivos do uso do método ágil

Perspectiva	Categoria	Assertiva
Contratada	Escopo	Equipes distintas (squads) trabalhando no mesmo produto com uso de método ágil (Scrum)
Contratada	Escopo	Conhecimento do ambiente (organizacional) permite a oferta de soluções
Contratada	Escopo	A identificação do escopo do projeto tem sido elaborada pelo PO com sucesso por meio das reuniões previstas no modelo ágil utilizado (cerimônias do Scrum como grooming e refinamento).
Contratada	Planejamento	Scrum é bem flexível, permite acomodar as necessidades do processo de desenvolvimento de sistemas. Aderente a mudanças, impactos são vistos ao longo do processo, não na entrega final.
Contratada	Planejamento	Demandas são identificadas pelo PO (Product Owner) e pelos gestores do projeto da TI do cliente. O desenvolvimento do software atual possui como etapas: - *Sprint backlog* - *Grooming* - *Inception* - *Solution Design* - *Sprint* (*Planning, timebox, sprint* de 2 semanas) - Retrospectiva (volta para o *sprint backlog*)

Perspectiva	Categoria	Assertiva
Contratada	Qualidade	Papel do QA permite a identificação de uma necessidade de negócio que o PO não identificou
Contratada	Qualidade	Prototipagem junto com a área de negócio (usuários finais) e refinamento ao longo do processo
Contratante	Engenharia de software	Foi adotado um procedimento de revisão de código fonte entre os integrantes do time de desenvolvimento, buscando melhoria na qualidade do software.
Contratante	Escopo	O entrevistado informou que os benefícios mais significativos do projeto foram atingidos por entregas rápidas que possibilitou testes, mesmo quando a solução ainda exigia intervenções operacionais por ter baixa sofisticação.
Contratante	Escopo	Configuração dos aspectos de usabilidade do sistema de forma centralizada
Contratante	Organizacional	É fundamental que o time do projeto contenha profissionais que conheçam o negócio da empresa. Caso não possua esse conhecimento dentro do time, será necessário contratar uma consultoria que traga boas práticas de mercado (nesse caso, varejo e comércio eletrônico).

Fonte: Autor.

Após a categorização das informações coletadas junto aos entrevistados, a análise prosseguiu para a identificação de como os problemas identificados poderiam ser solucionados pelo emprego de um método que integra a abordagem DT com métodos ágeis em projetos de desenvolvimento de *software*. Todos os problemas relatados pelos entrevistados, classificados em escopo, planejamento, qualidade, testes, organizacional e engenharia de *software* foram analisados e receberam um atributo sinalizando se: (i) os métodos ágeis possuem características que oferecem solução para o problema em questão; (ii) se a abordagem DT oferece solução para o problema relatado, e; (iii) se a abordagem DT integrada com métodos ágeis soluciona o problema informado pelo entrevistado. Tal categorização e atribuição de resolução de cada problema relacionado aos projetos de desenvolvimento de *software* foi validada posteriormente com especialistas em métodos ágeis de gestão de projetos e abordagens visuais, dentre elas, a abordagem DT. Esse atributo foi definido para cada uma das categorias, presentes na Tabela 14, e receberam como valores possíveis (i) “sim”, quando a abordagem ou o método em questão atende ao problema ou (ii) “não”, quando a abordagem ou método não soluciona o problema mapeado na Keep IT Simple.

Tabela 14: Atributo do método ou abordagem para a solução do problema

Indicação do atributo na tabela	Descrição
Ágil atende?	A aplicação exclusiva do método ágil soluciona o problema?
DT atende?	A aplicação exclusiva da abordagem DT soluciona o problema?
DT + Ágil atende?	O uso de um método que integra a abordagem DT e métodos ágeis soluciona o problema?

Fonte: Autor.

Quanto aos problemas identificados, os que estão relacionados ao escopo apresentam a maioria das ocorrências de acordo com a categorização dos problemas reportados, considerando ambas as perspectivas, de contratante e contratada responsável pelo desenvolvimento de software, como pode ser visto na Tabela 15. Das 41 ocorrências de problemas levantados nesta etapa da pesquisa, 13 assertivas se referem a problemas de escopo, o que corresponde à 32% dos problemas mapeados. As categorias qualidade e testes foram as que apresentaram menor incidência de problemas, com 2 ocorrências cada. As assertivas coletadas nas entrevistas estão relacionadas no Apêndice C.

Tabela 15: Categorização dos problemas apontados pelos entrevistados

Classificação	Categoria	Ocorrências	%
Problema	Escopo	13	32%
	Engenharia de software	9	22%
	Planejamento	8	20%
	Organizacional	7	17%
	Qualidade	2	5%
	Teste	2	5%

Fonte: Autor.

Dentre os problemas relacionados ao **escopo** e reportados pelos entrevistados está a afirmação de que a expectativa dos usuários excede a capacidade de execução em termos de tecnologia e restrição de prazo para entrega. Tais afirmações foram corroboradas pelos entrevistados E4, E5 e E6, que desempenham a função de SM na Keep IT Simple. Além disso, a presença de usuários reais na elaboração das estórias, que servem de insumo para o desenho da solução do *software*, nem sempre ocorre. Tal afirmação foi mencionada por pelos entrevistados E1, E2 e E3 na etapa de identificação do problema e pelos entrevistados E4, E5 e E6 na etapa de conscientização do problema, caracterizando que este é um problema sinalizado por representantes da contratada e da contratante. A Tabela 16 consolida os problemas apontados pelos entrevistados e categorizados como escopo.

Tabela 16: Problemas de Escopo apontados em relação ao uso do método ágil

Perspectiva	Assertiva	Ágil atende?	DT atende?	DT + Ágil atende

Perspectiva	Assertiva	Ágil atende?	DT atende?	DT + Ágil atende
Contratada	PO pode passar uma diretriz que não é o que o usuário final quer	Não	Sim	Sim
Contratada	MVP: perguntar ao usuário o que ele precisa para hoje (exemplo de resposta: preciso me locomover). Não perguntar ao usuário o que ele quer (arrisca ele dizer que quer uma BMW)	Não	Sim	Sim
Contratada	A necessidade vem pré-moldada dos BP's que cuidam da captura das necessidades (integrante do time de TI que atende o negócio). Lacuna entre o que está sendo pedido e o que o cliente realmente quer	Não	Sim	Sim
Contratante	Distância entre o que o cliente desejava e o que tecnicamente era possível de ser feito e no curto prazo.	Não	Sim	Sim
Contratante	Não houve a participação dos usuários finais na definição dos requisitos do software que futuramente seriam utilizados por vendedores, analistas de crédito, operadores de caixa, estoquistas e pessoal administrativo. Por esta razão, atualmente o sistema está implantado em aproximadamente 10% das lojas com pouca aderência de uso, visto que o sistema antigo ainda não foi continuado.	Não	Sim	Sim
Contratante	Sistema com várias novas funcionalidades pode ser inferior a outro software tido como obsoleto por não atender as reais necessidades dos usuários finais.	Não	Sim	Sim
Contratante	Grande número de reclamações associadas ao não uso correto do sistema (30% dos *feedbacks*) e de dúvidas (60%), o que pode indicar um problema de falta de treinamento do sistema, além da não participação dos usuários finais ao longo do processo de desenvolvimento do novo sistema.	Não	Sim	Sim
Contratada	Subjetividade do que é o valor para o negócio	Sim	Sim	Sim
Contratada	Entrega de um produto durante um prazo pré-estabelecido pela iteração pode não trazer valor, de fato, ao negócio	Sim	Sim	Sim
Contratada	Organização é mais importante que o skill técnico do desenvolvedor. PO / SM devem saber o que passar para o desenvolvedor	Sim	Sim	Sim
Contratada	PO representa o negócio e pode priorizar. Algumas determinações vêm pela área de negócio e isso afeta o processo, independentemente da etapa do ciclo de vida do projeto (*planning* ou homologação de *iterações*).	Sim	Sim	Sim
Contratante	a atuação área de operações de vendas, demandante e patrocinadora do projeto não favorece a elaboração de um backlog estruturado e que o MVP muitas vezes se torna inviável pela complexidade da funcionalidade do sistema que deveria ser entregue. De acordo com o entrevistado, os representantes da área demandante querem o estado da arte já na primeira entrega.	Sim	Sim	Sim
Contratante	Diversas vezes os usuários do sistema exigem que todas as funcionalidades já existentes (*as is*) devem ser incorporadas ao projeto, sendo que algumas podem ter menor relevância. O entrevistado citou que a área exigiu que o pagamento em cheque fosse uma funcionalidade mandatória, sendo que na visão do (PO) conceder ao sistema o critério de multicanal teria maior valor ao negócio (venda iniciada no site e terminada na loja).	Sim	Sim	Sim

Fonte: Autor.

A necessidade de construção do escopo contendo as características e funcionalidades do *software* esperados pelos usuários requer que haja a proximidade dos usuários e do time de desenvolvimento, confirmada pelos entrevistados E7 e E8, especialistas em métodos ágeis e abordagens DT, respectivamente. Dos 13 problemas listados, sete não poderiam ser atendidos exclusivamente com a aplicação de métodos ágeis, mas poderiam ser atendidos com a introdução de práticas da abordagem DT, bem como conduzidos pela aplicação da abordagem DT com métodos ágeis. Esses problemas estão relacionados com a (i) não atuação dos usuários finais no processo de desenvolvimento do software; (ii) ausência de necessidades dos usuários do *software* no produto entregue, e; (iii) expectativa do usuário maior do que a percepção adquirida no uso do *software*.

A resolução dos problemas apontados pelos entrevistados em relação ao **escopo** do projeto de desenvolvimento de *software* com o uso da abordagem DT foi confirmada pelo E8 nesta etapa da pesquisa. O entrevistado E8 pontuou que o DT possui inúmeras ferramentas que promovem a captura do real valor esperado pelo cliente, no caso, o usuário do *software*, quando as estórias são bem escritas, o nível de subjetividade do que se espera como valor do produto, no caso, o software, tente a diminuir. Em se tratando da composição do escopo do projeto, o entrevistado informou que a abordagem DT possui inúmeras ferramentas que permitem o entendimento da real necessidade do usuário levando em conta o seu ponto de vista por meio da empatia. Busca-se o entendimento real do desafio a ser perseguido na etapa denominada inspiração, na abordagem DT. Portanto, o DT permite que o objetivo do projeto esteja associado à clareza do desafio a ser perseguido para entregar o valor ao cliente, sendo parte de um processo pela busca ao conhecimento daquilo que é necessário.

Ao considerarmos uma outra etapa presente na abordagem DT, a prototipação deve estar diretamente alinhada com a necessidade identificada na etapa de inspiração. Deste modo, esta etapa permite que os usuários tenham contato com a solução e validem cada uma das ideias por meio de um protótipo, podendo ser algo simples ou sofisticado que permita a percepção visual de quem irá receber o *software*. Em alguns casos, a etapa de prototipação, ao buscar sua simplicidade, pode ser desenvolvida em uma atividade de até uma hora de duração. Na etapa de prototipação deve-se considerar os vários tipos de protótipos que devem ser elaborados. Um protótipo pode validar uma ideia geral, um MVP ou qualquer outro objetivo do projeto.

Com relação ao tema de promover a atuação dos demandantes ou usuários, o entrevistado afirmou que o projeto deve considerar a presença de times multidisciplinares,

observando que o time que participa da fase de criação pode não ser o mesmo time que irá atuar ao longo das outras etapas do projeto, observando o ciclo de vida do desenvolvimento de um *software*. Para que isso ocorra, o real problema do projeto deve ser entendido para que o desafio seja estruturado e permita que as atividades para atender este problema sejam executadas. Um projeto pode ser idealizado para que 15 ou mais problemas sejam solucionados. Por exemplo, cada conjunto de problemas ou até um problema específico deve ser reduzido, enquadrado e validado junto ao cliente para que um time voltado para a organização e distribuição das iterações e entregas seja feito de forma adequada. Outra ferramenta de DT que poderia ser utilizada é a de usuários extremos. Essa técnica permite que o escopo do projeto não esteja embasado apenas na solicitação de uma pessoa. Com isso, o entendimento do problema passa a ser mais abrangente. Deve-se considerar inclusive os representantes de todas as áreas envolvidas ou afetadas pelo *software*. Normalmente, nesta técnica, devem ser designadas de três a cinco usuários para elaborar, idealizar, validar e testar as funcionalidades que o software deverá possuir.

Quanto ao tema relacionado ao estabelecimento de um tempo pré-determinado para cada iteração, o entrevistado mencionou que deve ser realizada uma redefinição sob o ponto de vista de quem está solicitando o software, quando o mesmo passa pela decomposição em várias entregas. Como em um primeiro momento o SM pode ter apenas uma visão inicial do que o software irá ter como funcionalidades, essa visão deve ser novamente validada a partir do aprofundamento do nível de detalhe do projeto. Ou seja, um projeto que se inicie como requisitos superficiais pode apresentar problemas. Sendo assim, aproximar o time de desenvolvimento e os usuários e clientes para o entendimento do negócio aumenta a compreensão sobre os principais problemas dos usuários e das partes interessadas (Vidal, 2017). Deste modo, os usuários e demandantes do software poderão ter a visão de quando cada problema será resolvido, sendo que ele será tratado em ondas (iterações), que devem ser igualmente avaliadas e validadas pelos usuários.

Os problemas relacionados ao **planejamento** e que foram mencionados nas entrevistas estão relacionados com a condução das atividades de desenvolvimento do *software*. As etapas que sucedem a elaboração do escopo, como o desenvolvimento do *software* em si, podem ser prejudicadas pelo nível de detalhamento exigido para a elaboração das corretas funcionalidades a serem entregues. Como pode ser notado na Tabela 17, dos oito problemas mencionados, cinco são atendidos pelo uso de métodos ágeis, conforme opinião do especialista em métodos ágeis, E7. Cabe ressaltar que, pelo fato desses cinco problemas terem sido relatados pelos entrevistados, o uso de métodos ágeis exclusivamente não tem impedido

que problemas como estes ocorram nos projetos de desenvolvimento de *software* na Keep IT Simple. Diante deste fato, podemos inferir que a aplicação combinada da abordagem DT e métodos ágeis poderiam suprir melhor tais deficiências em virtude da proposta de aproximação dos usuários e do time de desenvolvimento.

Tabela 17: Problemas de Planejamento apontados em relação ao uso do método ágil

Perspectiva	Assertiva	Ágil atende?	DT atende?	DT + Ágil atende
Contratante	Em alguns casos, houve disputa sobre a responsabilidade dos erros ou itens em não conformidade identificados durante o projeto, sem a indicação de quem deveria arcar com o ônus das horas da correção. Essa disputa normalmente possuía duas alternativas: se a falha ocorreu no desenho ou especificação e deveria ser arcada pelo contratante ou se a falha foi derivada de erro técnico e deveria ser realizado sem pagamento.	Não	Sim	Sim
Contratante	Representante da contratada também citou o desafio de integrar diferentes plataformas, decorrente de desejos do negócio. Integrar negócio + UX + TI Corporativo + time de desenvolvimento.	Não	Sim	Sim
Contratada	Mudanças de prioridade, iteração sem fechar o planejamento, direcionando a equipe para resolução dos bugs. Mudança de prioridade das atividades em execução por solicitação do cliente	Não	Sim	Sim
Contratada	Deixar o negócio mais próximo pode trazer problemas como o aumento da exigência. Itens no backlog é responsabilidade do PO. Cliente atuando próxima pode afetar o desempenho como o dono do time e isso afetar o desempenho da equipe.	Sim	Sim	Sim
Contratada	Nível de detalhe do backlog para o início do desenvolvimento. Ao longo da iteração o time de desenvolvimento deve montar o entendimento com a profundidade necessária. PO não consegue atuar previamente nesta atividade.	Sim	Sim	Sim
Contratada	O projeto usa as estórias de usuário para documentar as demandas, porém no geral as histórias no projeto possuem visão em alto nível e sem regras de negócio.	Sim	Sim	Sim
Contratada	Planejamento de 2 semanas (curtíssimo prazo) ocorre em função da falta de visão de negócios, metas, etc. pelo PO. TI normalmente é reativa, não sabe o que de fato o negócio precisa.	Sim	Sim	Sim
Contratada	Estórias estão hoje incompletas.	Sim	Sim	Sim

Fonte: Autor.

Três dos oito problemas, relacionados à integração de usuários e de outros times de TI que deveriam ser envolvidos na etapa de desenho do *software* a ser desenvolvido, foram classificados como possíveis de serem atendidos com a integração da abordagem DT e métodos ágeis. Uma delas, relacionada com a mudança de prioridades durante a execução das iterações, ficou categorizada como possível de ser atendida com uso do DT exclusivamente ou combinado com métodos ágeis. De acordo com o especialista em DT, para se obter um melhor planejamento das estórias, várias técnicas de DT podem ser utilizadas.

Uma das ferramentas da abordagem DT que pode ser adotada, de acordo com o E8, é o mapa das partes interessadas, no qual pode-se gerar um cruzamento das características do

software a ser desenvolvido versus quais as áreas que o utilizam. As partes interessadas do projeto devem ser classificadas, como por exemplo, em usuários finais, setores internos e gestores. Conforme o detalhamento das necessidades do *software* forem avançando, deve-se checar junto aos usuários, o que poderá ser feito por meio da técnica de usuários extremos. Deste modo, é esperado que seja feito um cruzamento entre os usuários de cada um dos setores mapeados e seja realizado um filtro de todos os interesses de cada um deles. E assim, conforme relatado pelo E8, ter a decomposição do problema com os interesses de cada um dos grupos e a prioridade de cada um deles.

O E8, especialista em DT, relatou adicionalmente que o problema de mudanças na priorização durante a execução da iteração é comum e não é solucionado por meio do uso de métodos ágeis. Para que a inclusão de novas atividades impeça a interrupção de uma iteração em andamento, o entrevistado sugere o estabelecimento de um indicador, não só para o time, mas para toda a área (considerando também a área demandante). Uma vez que a demanda é dividida em várias iterações de mesma duração, a área demandante passa a ter a visão clara de quanto tempo ela levará para ser concluída, mesmo que ao final de cada iteração seja entregue parte da resolução de um problema maior. Tal afirmação foi reiterada pelo E7. Portanto, deve-se combinar com a área demandante o prazo em que cada uma delas será entregue e demonstrar quais seriam os impactos decorrentes de uma nova priorização. Embora isso não esteja diretamente alinhado com o uso da abordagem DT, o E8 mencionou que esse problema pode ser atendido pela abordagem em função da transparência com a área de negócio ser decorrente do processo de empatia e entendimento do real desafio. Deste modo, o DT permitirá que ao longo da etapa de inspiração sejam capturados os reais desafios e, portanto, entrará para ser desenvolvido aquilo que é relevante e desejável pelos usuários, reduzindo e até mesmo eliminando possíveis mudanças de priorização durante a execução de uma iteração.

Ambos os problemas relacionados à **qualidade** do *software* são atendidos pelo emprego dos métodos ágeis, desde que corretamente. A etapa de homologação, referenciada nos métodos ágeis deve permitir a automatização dos testes para que o *software* seja entregue aos usuários finais com a qualidade requerida, sem que a entrega da iteração em execução cause impacto negativo nas funcionalidades entregues nas iterações anteriores. Isso requer que o time de projetos identifique formas de assegurar que a homologação ocorra adequadamente, conforme relato feito pelo E7.

Por outro lado, a qualidade do *software* entregue e a percepção do usuário pode ser melhorada com o emprego de práticas de prototipação e testes considerando a atuação dos

usuários finais. Cabe reforçar que tais práticas integram a abordagem DT. A Tabela 18 mostra o detalhe das duas ocorrências de problemas relacionados à qualidade. O especialista DT reforça que o ciclo de um projeto de desenvolvimento software deve considerar o sequenciamento das atividades quer permitem (i) a identificação do problema, (ii) qual o impacto que isso causa em todas as áreas afetadas e/ou clientes, (iii) qual a solução que será dada para resolução desse problema e, (iv) a validação desta solução por todos as partes interessadas. Uma vez que isso ocorre, tendo como ponto de partida o entendimento do real desafio e considerando os pontos de vista das partes interessadas adequadamente mapeadas, a qualidade do *software* e a probabilidade de êxito no projeto em termos do nível de satisfação dos usuários tende a aumentar, de acordo com o depoimento do E8.

Tabela 18: Problemas de Qualidade apontados em relação ao uso do método ágil

Perspectiva	Assertiva	Ágil atende?	DT atende?	DT + Ágil atende
Contratada	Homologação com o maior número de usuários possível. Muitas vezes a área que irá receber o software não tem um bom relacionamento com a demandante dos ajustes no sistema. PO começa a ter papel fundamental no sentido de permitir que o sistema seja utilizado.	Sim	Sim	Sim
Contratada	PO / SM são os papéis que asseguram a qualidade do software	Sim	Sim	Sim

Fonte: Autor.

A categoria dos problemas relacionados ao **teste** do *software* contém dois problemas mencionados pelos entrevistados, somente sob a perspectiva da contratada. A Tabela 19 ilustra os problemas mencionados pelos representantes da empresa desenvolvedora do *software*. De acordo com o E8, especialista em DT, os problemas decorrentes da etapa de testes nos projetos de desenvolvimento de software podem ser solucionados pelo emprego da abordagem DT, seja ela isoladamente ou combinada com métodos ágeis.

Tabela 19: Problemas de Testes apontados em relação ao uso do método ágil

Perspectiva	Assertiva	Ágil atende?	DT atende?	DT + Ágil atende
Contratada	No método ágil, não há um modelo preditivo que possibilite avaliar o que foi construído de fato com o que foi prometido, visto que o plano é feito ao longo da execução.	Não	Sim	Sim
Contratada	Acompanhamento ineficaz (não tem modelo para acompanhamento)	Sim	Sim	Sim

Fonte: Autor.

Também de acordo com o E8, é na etapa dos testes que ocorre a verificação de tudo aquilo que foi idealizado e solicitado pelas áreas demandantes. Como o *software* é idealizado a partir do real entendimento das necessidades dos usuários, a etapa de testes e de validação do produto entregue ocorrerá dentro do previsto desde que o entendimento e mapeamento dos participantes seja feito corretamente. Por esse motivo, o DT também está classificado como item que atenderá à resolução do problema dessa categoria no desenvolvimento de *softwares* pela Keep IT Simple.

Além dos aspectos que estão relacionados ao modelo de gestão de projetos de desenvolvimento de *software*, também foram mencionados problemas que não estão diretamente ligados à condução do projeto, tais como a engenharia de *software* e ambiente organizacional. Os problemas apontados e categorizados como engenharia de *software* estão vinculados aos aspectos de infraestrutura de TI e operação dos sistemas (disponibilidade e escalabilidade, por exemplo). Muitos desses aspectos não estão diretamente relacionados ao processo de desenvolvimento do *software* em si, portanto, não estão vinculados a gestão de como o projeto está sendo conduzido.

Os nove problemas relativos à **engenharia de *software*** estão listados na Tabela 20, sendo seis problemas mencionados pela contratada e três afirmações concedidas pelos representantes da contratante, isto é, cliente do *software*. Isso aponta que esses aspectos, embora desvinculados da gestão do projeto em si, tem maior impacto para o fornecedor do *software*.

Tabela 20: Problemas de Engenharia de *Software* apontados em relação ao uso do método ágil

Perspectiva	Assertiva	Ágil atende?	DT atende?	DT + Ágil atende
Contratada	Problemas são normalmente identificados na etapa de QA do ciclo de desenvolvimento de software. Tarefas e atividades complexas. Coloca muita coisa em produção com erro. Uma subida para corrigir algo, afeta o que já está em produção.	Não	Não	Não
Contratada	Problemas de ambiente causa atrasos de dias. *DevOps* seria uma alternativa para os problemas de arquitetura. Corrige bug, sobe para produção, mas na semana seguinte o bug ressurge por problema de versionamento. Time acaba atuando em retrabalho, mesmo quando o item já foi resolvido	Não	Não	Não
Contratada	Problemas de infraestrutura (VPN) para o trabalho da equipe	Não	Não	Não

Perspectiva	Assertiva	Ágil atende?	DT atende?	DT + Ágil atende
Contratada	Interdependência entre frentes: vários projetos entrando simultaneamente exigindo um cuidado com o versionamento do *software* (versionamento)	Não	Não	Não
Contratada	Melhor planejamento para a entrega quando há concorrência de códigos-fonte entre projetos (*squads* ou artefatos). Em alguns casos, a subida em produção ocorre sem alguma outra funcionalidade que já havia sido implantada por outro projeto.	Não	Não	Não
Contratada	Existem diferentes plataformas a serem conectadas: Banco de Dados (BD), Cobol (Alta Plataforma), *Application Programming Interface* (API) desenvolvidas em Java e Front-end (Angular).	Não	Não	Não
Contratante	Arquitetura da empresa hoje não está preparada para avaliar novas opções tecnológicas e acaba atravancando potenciais aceleradores. Maturidade da área está aquém do que é necessário e o time do projeto, de uma maneira geral, não estava capacitado. Conceito de DevOps (*development and operations*), relação ágil entre desenvolvimento e sistemas em produção) está sendo implantado na organização agora, mas ainda embrionário.	Não	Não	Não
Contratante	Processo de desenvolvimento e *deploy* (publicação) de programas em produção estão sendo realizados sem um modelo pré-estabelecido. Ele citou que alguns programas que ainda estariam em testes subiram para produção indevidamente. Ele informou que a ausência de um processo estruturado que estabeleça as fases do ciclo de vida de um software *(inception, design, development, tests, user acceptance and production*) dificulta o alinhamento entre as diversas frentes do projeto.	Não	Não	Não
Contratante	Ponto crítico do projeto é acoplar as novas funcionalidades em sistema de alta plataforma, que possui algumas restrições. O modelo de solução do sistema previsto para o projeto teve que ser moldado ao longo do desenvolvimento. Determinadas comunicações e tratamento de dados precisaram ser isolados para que não houvesse interrupção dos demais sistemas organizacionais que permaneceriam inalterados.	Não	Não	Não

Fonte: Autor.

Os entrevistados mencionaram problemas categorizados como aspectos do **ambiente organizacional**, tais como cultura, comportamento, metas e gestão do processo de desenvolvimento de *software*. As assertivas categorizadas deste modo totalizaram seis ocorrências, sendo a maioria derivada das observações sob a perspectiva da contratada, como observado na Tabela 21. Nota-se que o aspecto comportamental, que expõe o problema do time de desenvolvimento e os usuários do *software* serem geridos por estruturas distintas e com expectativas diferentes, apareceu na opinião de representantes da contratada e da contratante.

Tabela 21: Problemas de Ambiente Organizacional apontados em relação ao uso do método ágil

Perspectiva	Assertiva	Ágil atende?	DT atende?	DT + Ágil atende
Contratada	Metas e alta gestão ajudam na organização das atividades	Não	Não	Não
Contratada	Comportamento das pessoas: as atividades são derivadas da pessoalidade e do interesse das pessoas. A adesão ao método e desempenho da equipe estão associadas aos benefícios esperados.	Não	Não	Não
Contratada	Durante a retrospectiva são identificadas como as atividades foram feitas, não quais atividades deveriam ser feitas. Nesta etapa, denominada como "*check*" são validados os aspectos que envolvem ferramenta, processos e pessoas, apontando itens que demandam ações (ex. dificuldades de acessos, perfil dos recursos do time, atividades novas não planejadas, processos de clientes que são modificados sem prévio alinhamento).	Não	Não	Não
Contratada	Houve uma antecipação no desenvolvimento de algumas funcionalidades, porém a contratante optou por não implantar	Não	Não	Não
Contratante	Cultura organizacional atual reflete a uma postura dos usuários e clientes do sistema como recebedores de algo, sem características de parceria.	Não	Não	Não
Contratante	É fundamental que o time do projeto contenha profissionais que conheçam o negócio da empresa. Caso não possua esse conhecimento dentro do time, será necessário contratar uma consultoria que traga boas práticas de mercado.	Não	Não	Não

Fonte: Autor.

4.4 CONFIGURAÇÃO DA CLASSE DE PROBLEMAS

A necessidade de resolver os problemas identificados ao longo do ciclo de vida do projeto de desenvolvimento de *software* na Keep IT Simple faz com que a avaliação do método integrado da abordagem DT com métodos ágeis seja avaliada com potencial para a resolução destes. A Tabela 22 contém o embasamento teórico para a aplicação de cada método, abordagem e modelo, relacionando-os a cada uma das categorias definidas para os problemas coletados na Keep IT Simple. A classe de problemas da presente pesquisa foi definida como um método de gestão de projetos que integre DT e método ágil para o desenvolvimento de *softwares*.

Tabela 22: Configuração da classe de problemas e referências bibliográficas

Modelo, abordagem ou método	Categoria	Referência
Métodos Ágeis	Escopo	Requerimentos são mais precisos em função de redução de escopo, sendo igualmente mais fáceis de serem estimados; além disso, o escopo pode ser reduzido para dar ênfase apenas aos requisitos relevantes (Petersen & Wohlin, 2009). O escopo, portanto, não é apenas um aspecto de eficiência do projeto, mas também impacta o nível de satisfação do usuário (Serrador &

Modelo, abordagem ou método	Categoria	Referência
		Pinto, 2015).
	Qualidade	A probabilidade de sucesso no projeto gerenciado por métodos ágeis aumenta em função da medição nas múltiplas perspectivas do projeto, tais como eficiência e satisfação das partes interessadas (Serrador & Pinto, 2015)
Abordagem DT	Escopo	A abordagem DT, caracterizada pelo aspecto da observação direta e centrada no humano busca a captura do entendimento mesmo que inesperados para produção de inovações que melhor reflitam a expectativa dos usuários (Brown, 2008)
	Planejamento	Os praticantes da abordagem DT devem navegar entregas etapas de inspiração para a ideação e, assim, para a implementação de forma rápida, permitindo a experimentação do ciclo completo e a criação de um melhor julgamento buscando benefícios mais relevantes à longo prazo para a organização (Brown, 2008)
	Qualidade	Prototipação é a etapa da abordagem DT que visa responder perguntas que o aproximam da sua solução final, sendo desempenhada de forma gradativa na qual o usuário possa interagir e validar aquilo que está sendo entregue (Plattner, 2009). A etapa de teste, por sua vez, possibilita uma nova oportunidade para entender seu usuário, enquadrando o problema e testando os protótipos criados (Plattner, 2009). Ambas as etapas, prototipação e testes, necessitam ser executadas de modo que seja permitido transitar entre elas considerando o que e como será validado e testado (Plattner, 2009).
Modelo Integrado DT e métodos ágeis	Escopo	Artigos exploraram a abordagem DT apenas nos estágios iniciais, que compreendem nas fases de desenho e coleta de requisitos, indicando como benefícios: (i) observância dos aspectos de interface já nas primeiras ideias do software (Forbrig, 2016b); (ii) avaliação de diferentes alternativas que podem ser desenvolvidas nos primeiros estágios de desenvolvimento (Forbrig & Saurin, 2016; Losada et al., 2011) e clareza nos requisites (Steinke et al., 2017) baseados na opinião e observação dos usuários (Bosch & Bosch-Sijtsema, 2011; Ximenes et al., 2015).
	Planejamento	Artigos mencionam o uso da abordagem DT integrada com métodos ágeis durante todo o ciclo de vida do software, buscando não somente capturar as necessidades dos clientes nos estágios iniciais, mas garantindo a usabilidade e adaptabilidade do software desenvolvido e implantado (Fischer & Senft, 2016). O modelo integrado contribui durante todo o ciclo de vida do projeto (Forbrig, 2016b; Higuchi & Nakano, 2017; Paula & Araujo, 2016), favorecendo a identificação dos problemas que necessitam ser resolvidos e que reflitam a satisfação do cliente no software que será entregue (Gurusamy, Srinivasaraghavan, & Adikari, 2016; Lucena et al., 2016; Prior et al., 2013). O uso da abordagem DT integrada com métodos ágeis sugere a preocupação com um melhor alinhamento entre as expectativas do usuário e o time de desenvolvimento (Darrin & Devereux, 2017; Lester, 2011; Newman et al., 2015; Xiong & Wang, 2010).
	Qualidade	Necessidades dos usuários dos softwares foram atendidas (Schön et al., 2016) como resultado de produtos e serviços mais bem ajustados às necessidades dos clientes (Bourimi et al., 2010; Isa et al., 2014; Steinke et al., 2017) por meio do uso da abordagem DT integrada com métodos ágeis. Há a percepção de melhor qualidade do software entregue quando as mudanças são gerenciadas apropriadamente (Ardito et al., 2017; Gamble, 2016), seguido pelos frequentes testes para ver se com as funcionalidades do software estavam sendo implementadas (Sebok et al., 2017). Usuários também acharam o software mais fácil de ser usado e requeriam menor nível de suporte para que fossem utilizados (Adikari et al., 2009).

Fonte: Autor.

De acordo com Gregor e Hevner (2013) um método pode ser entendido como um algoritmo, práticas ou receitas para realizar uma tarefa. Diante da necessidade de solucionar os problemas encontrados nos projetos de desenvolvimento de *softwares* na Keep Simple, o presente estudo buscou estabelecer um método para suporte à gestão de projetos de desenvolvimento de *software* integrando a abordagem DT e métodos ágeis. O resultado da presente pesquisa, portanto, gerou um conhecimento prescritivo por meio da teoria do *design* (Gregor & Hevner, 2013).

4.5 PROPOSIÇÃO DO ARTEFATO

Conforme mencionado na seção 4.2.1, em virtude de um método que integra a abordagem DT e métodos ágeis não ter sido encontrado na literatura, o artefato foi elaborado a partir de um modelo que estabelecia tal combinação. A escolha do modelo foi realizada por meio da comparação entre os 25 problemas identificados na execução dos projetos de desenvolvimento de *software* da Keep IT Simple e dos benefícios relatados pelos autores em relação ao emprego dos métodos, quando os mesmos foram utilizados de forma prática. Dos 29 artigos que estabeleciam modelos integrados da abordagem DT e métodos ágeis, sete artigos foram escolhidos para a proposição do método, pois sugeriam um modelo a partir de um caso prático e propunham o uso do modelo integrado por todo o ciclo de vida de desenvolvimento do *software*. Na Tabela 23 é possível verificar a relação dos sete artigos e a quantidade de problemas identificados na Keep IT Simple que foram encontrados como benefícios da aplicação do modelo.

Tabela 23: Modelos que integram a abordagem DT com métodos ágeis

Autor	Tipo de *software*	Modelo sugerido	Tipo de estudo	Ciclo de vida do *software*	Quantidade de problemas apontados como benefício
(Bosch & Bosch-Sijtsema, 2011)	Solução empresarial integrada	Sim	Caso Prático	Completo	25
(Hussain et al., 2008)	Aplicativo de gestão de conteúdo	Sim	Caso Prático	Completo	17
(Ardito et al., 2017)	Portal (*web based*)	Sim	Caso Prático	Completo	16
(Paula & Araujo, 2016)	Mobile game	Sim	Caso Prático	Completo	13
(Sebok et al., 2017)	Sistema de segurança e missão crítica	Sim	Caso Prático	Completo	11
(Lucena et al., 2016)	Portal de vendas e outras	Sim	Caso Prático	Completo	9

Autor	Tipo de *software*	Modelo sugerido	Tipo de estudo	Ciclo de vida do *software*	Quantidade de problemas apontados como benefício
(Fischer & Senft, 2016)	aplicações Software de planejamento	Sim	Caso Prático	Completo	7

Fonte: Autor.

Sendo assim, de todos os modelos encontrados na literatura e que traziam um estudo de caso com aplicação prática, o modelo que teve maior aderência na resolução dos problemas da Keep IT Simple foi o proposto por Bosch e Bosch-Sijtsema (2011). A relação detalhada entre os problemas identificados e o modelo escolhido está no Apêndice D.

O modelo proposto por Bosch e Bosch-Sijtsema (2011) combina as características da abordagem DT com métodos ágeis para o desenvolvimento de *softwares*, embora no artigo não tenha sido possível identificar, de forma explícita, qual abordagem DT ou método ágil que foram utilizados em conjunto. A abordagem proposta por Bosch e Bosch-Sijtsema (2011) combina elementos dos métodos ágeis e da abordagem DT, além de estabelecer que os times sejam auto selecionados, direcionados e gerenciados para atingir um desenvolvimento orientado ao cliente por times pequenos e independentes. De acordo com Bosch e Bosch-Sijtsema (2011)tal abordagem é composta por dois processos interativos: (i) funcionalidades identificadas como importantes pelos clientes são desenvolvidas num processo interativo, e, (ii) processo de entrega do produto no qual as contribuições dos vários times são combinadas, testadas e liberadas para os clientes (Bosch & Bosch-Sijtsema, 2011). O modelo é composto por quatro fases principais: imersão na solução, imersão no código; primeira iteração, demais iterações de desenvolvimento e validação da solução.

Nesta mesma pesquisa, Bosch e Bosch-Sijtsema (2011) estabelecem quatro elementos essenciais para a aplicação da abordagem integrada: (i) equipes auto gerenciáveis, sendo organizadas em times menores que buscam moldar a solução partindo das necessidades dos usuários; (ii) envolvimento contínuo do cliente ao longo de todo o processo de desenvolvimento, não só no final; (iii) solução e codificação sob imersão dos membros do time, buscando a conversão de uma ideia em um conceito que seja real, que permita sua validação e que explore as complexidades de se implantar no sistema legado, e; (iv) geração de uma versão "alfa", que consiste na capacidade de apresentar ao cliente uma versão estável do software, sendo possível que o cliente conceda o *feedback* sobre aquilo que já foi entregue (Bosch & Bosch-Sijtsema, 2011). Na Figura 13 é possível identificar as etapas, durações e atores de cada uma das etapas do processo de desenvolvimento de sistemas, proposto pelo

modelo de Bosch e Bosch-Sijtsema (2011). Deste modo, nota-se que a integração entre usuários e time de desenvolvimento permeiam por todo o ciclo de vida de um projeto de desenvolvimento de *software*.

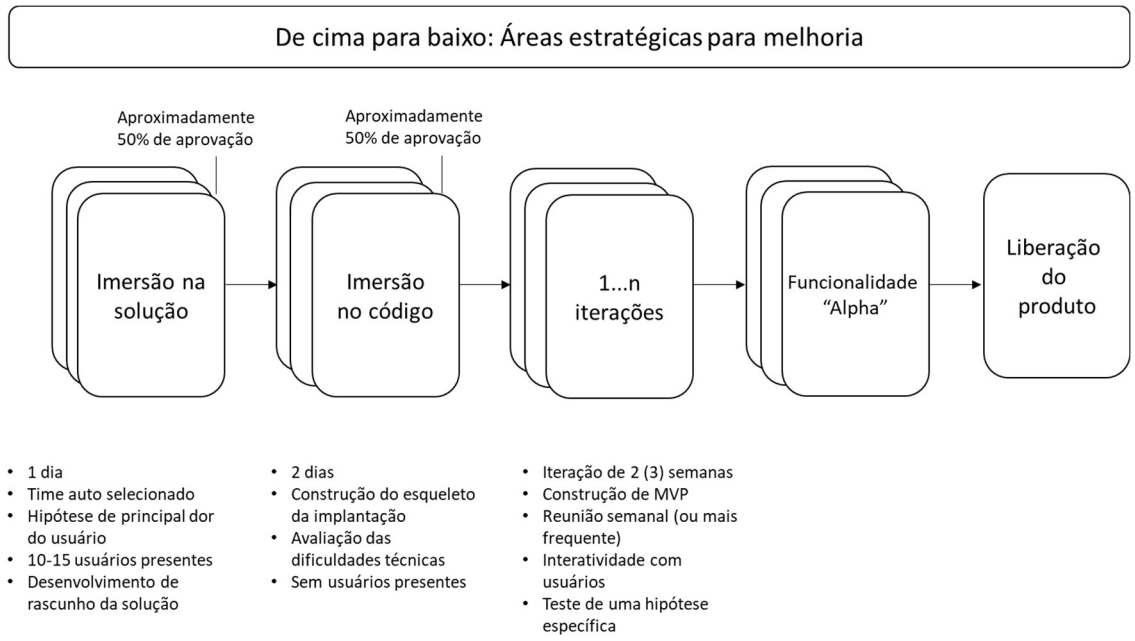

Figura 13. Modelo integrado da abordagem DT com Método Ágil
Fonte: (Bosch & Bosch-Sijtsema, 2011), p. 877

- **Imersão na solução**: nesta etapa o time realiza a atribuição de peso para as "dores" que os clientes informam e iniciam o desenho da solução por meio de rascunhos e complementados por comentários. Um primeiro esboço é criado e apresentado para os usuários presentes concederem *feedback*. O *feedback* dos usuários do *software* no estágio inicial assegura que o foco do time de desenvolvimento será dado aos aspectos corretos e importantes da solução. Durante todo o dia o time realiza várias iterações recebendo informações e *feedbacks* dos usuários e apresentam um painel com os conceitos mais importantes, sendo que normalmente 50% dos componentes passam para o próximo estágio.
- **Imersão no código**: normalmente realizada na mesma semana da fase anterior, quando o time de desenvolvimento constrói o esqueleto da implementação da solução com base nos conceitos aceitos. Ao invés de considerar as crenças da organização, a imersão de código valida a complexidade técnica por meio de um esqueleto de implantação. O esqueleto traz uma implementação parcial das funcionalidades mais básicas com ênfase na avaliação das maiores áreas de risco da etapa de implantação, traduzidas nos maiores riscos. Esta fase termina com um painel mostrando a avaliação do valor estimado de cada um dos conceitos

bem como a estimativa de custo e complexidade de implantação e integração. Normalmente 50% dos conceitos são selecionados, com base no maior retorno sobre o investimento (ROI, do inglês, *return on investment*).

- **Primeira Iteração de desenvolvimento**: a primeira iteração voltada para o início do desenvolvimento da implementação da solução, com duração de duas ou três semanas. Nesta iteração, é esperado o desenho que confirma a hipótese sobre o valor e usabilidade do conceito. O *feedback* do usuário normalmente é obtido por meio de ligações ao usuário e o compartilhamento de tela com o usuário para mostrar o que será implementado. Ao final desta etapa, quando a implementação dos conceitos é apresentada para a gestão de produtos, a decisão pode ser: (i) seguir para as próximas iterações; (ii) interromper a implantação caso o resultado ou o *feedback* do usuário não seja satisfatório, ou; (iii) incorporar a implementação no funcionamento do software, embora isso não seja comum.
- **Iterações de desenvolvimento subsequentes**: em cada iteração, o desenvolvimento é realizado e a hipótese relatada pelos conceitos informados pelos clientes. Ao final de cada iteração os resultados são apresentados para a gestão de produtos. Para projetos maiores e mais complexos o mecanismo chamado de funcionalidade "alfa" é gerado para capturar o *feedback* dos usuários. A funcionalidade alfa consiste na implementação de um agrupamento de soluções do *software* incorporado ao sistema já existente.
- **Processo de liberação do produto:** é dividido em duas categorias: *bottom-up*, no qual os times identificam as novas funcionalidades e *top-down*, quando a gestão do produto identifica grandes funcionalidades que precisam ser desenvolvidas em adição às funcionalidades identificadas como parte do processo *bottom-up*, mesmo que o *feedback* dos usuários também faça parte nos estágios de desenvolvimento. Para que esse *feedback* seja obtido, são criados produtos "alfas", que são similares às funcionalidades "alfas" geradas, exceto pelo fato de combinar inúmeras funcionalidades e conceitos em um release do software para ser distribuído aos clientes selecionados. O produto "alfa" possui várias novas funcionalidades, portanto não pode ser utilizado pelos clientes de forma operacional, visto ser liberado em uma máquina virtual e com uso de uma base de dados copiada de modo a não causar impacto nos dados reais.

Os benefícios da abordagem sugerida por Bosch e Bosch-Sijtsema (2011) no desenvolvimento de *software* são: (i) constantes *feedbacks* dos clientes ao longo de todo o desenvolvimento, desde os estágios iniciais, participando ativamente no amadurecimento das novas funcionalidades; (ii) redução do *overhead* no processo de desenvolvimento derivado da redução da interdependência entres os diferentes times e aumentando a autonomia dos times

alto-gerenciáveis; (iii) uso eficiente dos recursos, uma vez que as etapas de desenvolvimento, QA e integrações ocorrem de forma paralela e não mais sequencialmente, permitindo que um modelo de implantação contínuo seja implementado, e; (iv) aumento do engajamento dos membros do time, derivado do maior e mais frequente contato com os clientes do *software*, aumentando o engajamento e motivação dos componentes e pela responsabilidade pelas funcionalidades que lhes são atribuídas.

A elaboração do artefato, que compreende o método que integra a abordagem DT e métodos ágeis para a gestão de projetos de desenvolvimento de *software* foi realizada em três etapas: (i) a avaliação preliminar do artefato, relacionada com o projeto do artefato; (ii) a exploração do método, onde ocorreu o desenvolvimento do artefato, e; (iii) a confirmação do método que está relacionada com a avaliação do artefato.

4.6 PROJETO DO ARTEFATO

A etapa de projeto do artefato iniciou com a escolha de um modelo para uso na elaboração do método de gestão de projetos de desenvolvimento de software. A seleção do artefato, portanto teve como primeiro passo a realização de duas entrevistas, sendo a primeira entrevista com um representante da empresa contratada (Keep IT Simple) e a segunda com o representante da contratante do *software*. O detalhe dos entrevistados está presente na seção 3.3.6. A Figura 14 apresenta o método preliminar gerado para que as primeiras percepções sobre como as cinco etapas do modelo deveriam ser desdobradas em processos. Deste modo, os processos da versão preliminar do método foram decompostos em objetivos, facilitadores, participantes e ferramentas visando a resolução dos problemas identificados nos projetos de desenvolvimento de *software* na Keep IT Simple e categorizados em escopo, planejamento, qualidade e testes.

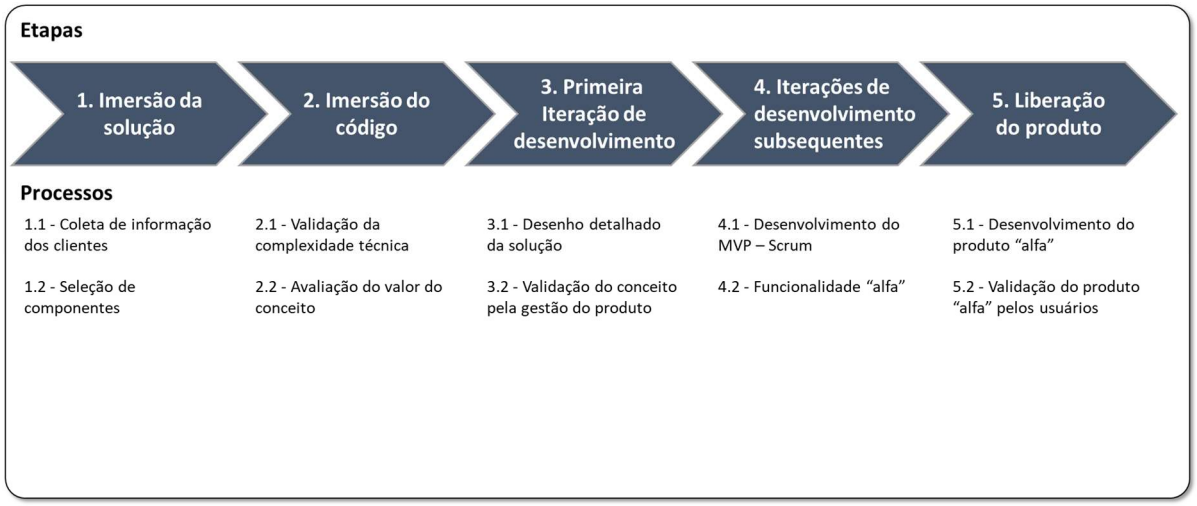

Figura 14. Etapas e processos do método preliminar de gestão de projetos de desenvolvimento de software: DT e métodos ágeis

Fonte: Autor

A primeira versão do método continha os processos, entradas, ferramentas e técnicas, participantes e saídas. Esta primeira versão do método possuía como pontos fracos um sequenciamento dos processos que não permitia identificar como os processos se complementariam a medida em que o *software* fosse sendo desenvolvido de forma incremental. Além disso, o entrevistado E9 informou que a abordagem, da forma como estava sendo apresentada na Figura 14, não tornava o método visualmente possível de ser interpretado e entendido. Deste modo, nesta etapa da pesquisa, o método já apresentava a necessidade de melhorias em sua composição.

4.7 DESENVOLVIMENTO DO ARTEFATO

O primeiro grupo focal, de caráter exploratório, foi realizado nesta etapa e contou com a participação de oito integrantes, sendo quatro entrevistados da empresa desenvolvedora do *software,* Keep IT Simple, e outros quatro representantes da empresa cliente ou demandante do *software*, identificados na seção 3.3.7. Todos os participantes do grupo focal assinaram o termo de consentimento de pesquisa, presente no Apêndice E deste documento. Durante a apresentação foram realizadas inúmeras sugestões de melhorias que foram incorporadas ao método que foi gerado com base no modelo de Bosch e Bosch-Sijtsema (2011). As principais modificações realizadas após a coleta de informações do primeiro grupo focal, considerando as questões presentes no Apêndice F, foram:

- Incluir um processo específico para a identificação dos participantes na etapa 1, antes da coleta de informações e requisitos do *software*
- Características de um modelo de gestão em cascata com entrada e saídas não favorece o princípio ágil
- Considerar a jornada do cliente para o mapeamento das necessidades
- *Blueprint* denota a ideia de ser algo completo, o que normalmente ocorre ao longo do projeto. *Blueprint* preliminar na fase 1 requer a definição da visão, cliente, objetivos de negócio por exemplo. Já na etapa 3, o blueprint detalhado se dará por meio da definição das estórias e critérios de composição do sistema.
- Premissas foram incluídas no método, tais como: (i) investimento necessário ao projeto deve ter sido aprovado, associando-o ao ROI estabelecido preliminarmente; (ii) definição de *capacity planning* do time de desenvolvimento para limitar a seleção do que será colocado para o desenvolvimento a cada ciclo; (iii) time contratado para a realização das iterações (interno ou fábrica).

Como resultado destas contribuições, o modelo foi revisto e ficou com o formato de processos cíclicos, conforme pode ser visto na Figura 15. Embora as etapas tenham permanecido as mesmas em relação ao modelo-base, o método passou a contar com 12 processos ao invés de 10, que haviam sido preliminarmente definidos. Além desse aspecto, as etapas foram divididas em: (i) conceitual, onde as três primeiras etapas (1 - Imersão da solução, 2 - Imersão do código e 3 - Primeira iteração de desenvolvimento) estavam voltadas para a elaboração do conceito de solução, e; (ii) produto, onde as últimas duas etapas (4 - Iterações de desenvolvimento subsequentes e 5 - Liberação do produto) se encarregavam de desenvolver o produto, no caso, o *software*.

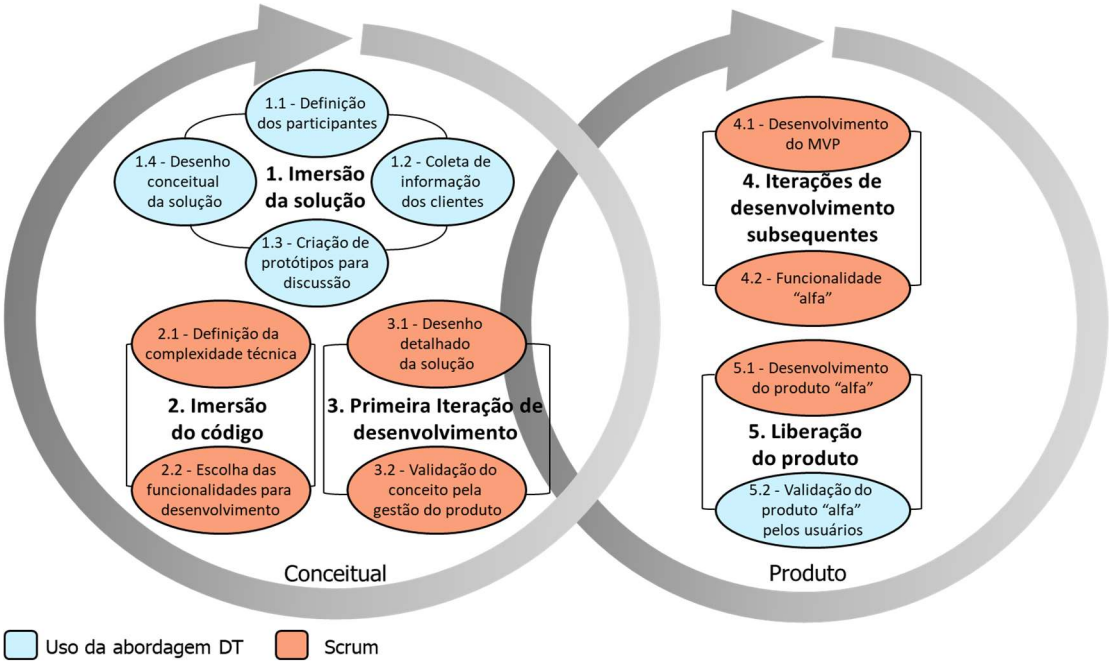

Figura 15. Etapas e processos do método preliminar de gestão de projetos de desenvolvimento de *software* após grupo focal 1

Fonte: Autor

4.8 VALIDAÇÃO DO ARTEFATO

A terceira e última etapa da validação do método ocorreu por meio de um segundo grupo focal e teve ao seu término a avaliação dos critérios de validação do artefato a partir da coleta de informações junto aos participantes do grupo focal confirmatório, que contou com representantes da empresa desenvolvedora de *software* e do cliente. As questões submetidas aos participantes presentes no segundo grupo focal, de caráter confirmatório, para a avaliação dos critérios de validação do artefato estão presentes no Apêndice G deste documento.

O segundo grupo focal, de caráter confirmatório contou com a participação de dois representantes da empresa desenvolvedora de software e três representantes da empresa cliente. Todos os cinco entrevistados presentes no grupo focal confirmatório também participaram do primeiro grupo focal, exploratório. Nesta ocasião, o método estava mais robusto e os participantes puderam avaliar com maior nível de detalhe e com maior critério a composição do mesmo como uma ferramenta para a resolução de problemas em projetos de desenvolvimento de *software*. Dentre as solicitações de mudança que ocorreram ao longo do grupo focal, destacam-se as seguintes modificações realizadas e incorporadas na versão final do método, presente na íntegra no Apêndice H deste documento:

- O método embora esteja disposto em etapas e processos não precisa necessariamente ser executado de forma sequencial e linear. A informação foi adicionada na introdução do método;
- Destaque aos objetivos de cada processo estabelecido no método;
- Inclusão do analista de testes como participante em processos das etapas 3, 4 e 5;
- A etapa 3 foi adaptada para planejamento apenas, ficando o desenvolvimento exclusivo para a etapa 4, onde as iterações de desenvolvimento serão desempenhadas;
- Os dois processos existente na etapa 3 foram consolidados em um único, voltado para o desenho detalhado da solução;
- Inclusão do processo 5.3. Garantia do produto na etapa 5. Liberação do produto;
- Escolha do nome do método: *Agile Design.*

Ao término do segundo grupo focal foi possível realizar as modificações no método *Agile Design* dando origem ao modelo final composto pelas cinco etapas e 12 processos. Na Figura 16 é possível avaliar as etapas e processos que compõe o método, seis processos possuem aspectos da abordagem DT pelo fato de estarem relacionados à interatividade com clientes e usuários e os outros seis processos são voltados para a elaboração da solução com os aspectos do Scrum, um dos mais difundidos métodos ágeis para o desenvolvimento de *softwares*.

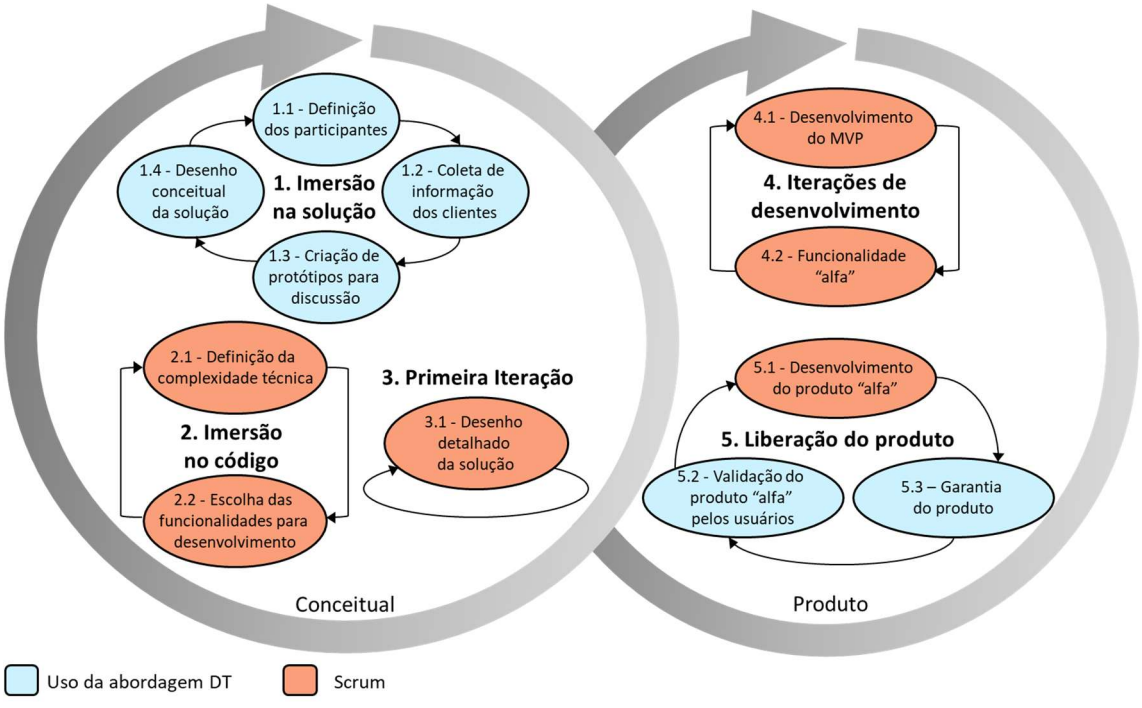

Figura 16. Etapas e processos do método *Agile Design*
Fonte: Autor

Com o intuito de aferir que o método *Agile Design*, resultante deste trabalho é possível de uso por parte da comunidade de TI responsável pelo desenvolvimento de *softwares*, bem como generalizar o mesmo para a aplicação em outras empresas, foram validados os critérios estabelecidos por Gill e Hevner (2013) e descritos na seção 3.3.6. Estes critérios foram validados ao término do grupo focal confirmatório, diante de um roteiro com perguntas sobre as características do método no qual todos os participantes opinaram sobre os mesmos. A Tabela 24 traz a relação das características propostas por Gill e Hevner (2013) e a assertiva de cada entrevistado em relação ao nível de aderência do artefato em relação aos aspectos funcionais e de uso do método proposto. Para que o artefato gerado seja generalizável para outras organizações, Gill e Hevner (2013) propõem características associadas (i) ao seu uso (aplicável às situações reais; aderência à resolução do problema; eficiência, fácil utilização), (ii) à sua modificação (adaptável; flexível, verificável e reutilizável) e (iii) à sua promoção (evolução; simples, claro, inovador, interessante e elegante).

Tabela 24: Características do método *Agile Design* sob a perspectiva de generalização.

Característica a ser validada	Nível de aderência de cada característica
- Aplicável às situações reais	GF2.1: "Sim. Existem aspectos deste modelo já utilizados em projetos da unidade de negócio online da contratante do software"
- Aderência à resolução do problema - eficiência	GF2.4: Acho que ele vai ajudar principalmente na identificação dos requisitos. Entender o que realmente precisa ser feito e diagnosticar as pessoas que precisam nos contar quais são as necessidades. Às vezes metade das necessidades são obtidas e a outra metade nem sabemos quem seriam as pessoas que falariam sobre". GF2.1: "A partir da identificação de que existe algo errado, ou não aderente à necessidade do usuário, possibilitar que qualquer etapa deste método seja revisitada"
- Fácil utilização	GF2.4: "É possível ver o fluxo completo e o detalhe da primeira etapa. Então você já consegue imaginar como o método de comporta até a entrega do produto. Fica fácil de acompanhar"
- Adaptável	GF2.1: "Método pode ser flexibilizado até determinado ponto." GF2.5: "Método deve ser aplicado como ele foi concebido, porém o tempo e esforço de cada etapa, isso sim pode ser flexibilizado." GF2.2: "O tempo e esforço (*time box*) de cada etapa podem ser distintos de acordo com a complexidade do projeto." GF2.4: "Você pode passar mais rápido e mais raso ou demorar mais para dar uma solução em cada uma das etapas. Quanto mais flexível o método, melhor. Se o projeto possui uma necessidade pequena, você pode ir mais rápido e mais profundo."

Característica a ser validada	Nível de aderência de cada característica
- Flexibilidade	GF2.1: "Sim, inclusive existem movimentos para que o ágil seja aplicado em situações não relacionadas à TI"
- Verificação - Reutilização	GF2.1: "No método detalhado, cada etapa deve ter um parágrafo de introdução que sinalize seu objetivo para que isso fique claro para outras pessoas que não participaram deste debate e discussão." GF2.4: "O que se espera de cada processo e etapa deve estar claro, e não só o objetivo de cada processo."
Flexibilidade	GF2.4: "Sim, independe da tecnologia"
Evolução	GF2.1: "Nenhuma etapa pode ser pulada: você não vai do levantamento preliminar (primeira *inception,* no caso, iteração) já para o time de desenvolvimento" GF2.4: "Na última etapa de liberação do produto, deve constar uma etapa de garantia, relacionada a questão do projeto."
Simplicidade Clareza	GF2.5: "Sim, o método pode ser utilizado facilmente"
Inovação	GF2.1: "DT já é utilizado hoje na prática pelos times de desenvolvimento, mas não seu uso não está formalizado e nem todos esses processos estão cadenciados desta forma. Não tem nada de novo, mas o método buscou estruturar todos os processos para que a coisa funcione". GF2.5: "A inovação está na visão clara de como os processos estão integrados, de como você faz, e em que momento cada processo deve ser realizado. Você está usando o benefício de um (abordagem DT) e aplicado em outro (método ágil)." GF2.4: "DT e ágil não são novidades. A junção, encadeamento, incorporação e sequenciamento dos dois, isso é novidade. Usar o DT no ágil é novidade"
Interesse Elegância	GF2.3: "Eficaz. Acho que funciona. Está estruturado em etapas, e na clareza entre elas..." GF2.1: "Está organizado e estruturado em um processo que as pessoas fazem aleatoriamente." GF2.2: "Eu acho que a organização é a palavra-chave: porque você sabe onde você está. E muitas vezes o cara está desenvolvendo e você tá ainda coletando ideias, ainda absorvendo, e aí vê se o *capacity* consegue. É lógico que a gente tem toda a dinâmica aqui, mas aqui você sabe onde realmente você está." GF2.5: "Você sabe o porquê de cada coisa. A organização que dá ter o método, a facilidade de implementar isso e em que ordem implementar isso, eu acho que é a grande sacada. Ficou muito legal." GF2.4: "A evolução do primeiro para cá ficou muito bacana. Até a forma de apresentar, ficou bem fácil de entender, com o objetivo claro não sentimos falta. Acho que tem começo, meio e fim, mesmo que não seja cascata, mas ele permite várias vezes recomeçar o processo..."

Fonte: Autor.

Em relação a possibilidade de o método ser adaptável, o mesmo não foi confirmado pelos participantes em função do método prescrever como gerenciar um projeto de desenvolvimento de *software* integrando a abordagem DT e métodos ágeis. Na opinião dos participantes a necessidade de seguir um sequenciamento das atividades é o que determinará a obtenção de um resultado positivo na entrega de um *software* contendo as necessidades dos

usuários finais e os requisitos dos clientes. O método proposto, por sua vez, poderia ser adaptável em relação ao tempo de duração das atividades e nos processos derivados da complexidade do *software* a ser desenvolvido, de acordo com o consenso dos participantes do grupo focal, tanto na perspectiva do contratante quanto da contratada. Ou seja, caso o *software* seja complexo e possua várias funcionalidades e diversos usuários extremos, é capaz que a etapa de imersão no código seja realizada durante um prazo maior se comparado com um sistema de baixa complexidade e com poucos usuários e clientes envolvidos. Todos os demais aspectos apontados por Gill e Hevner (2013) foram identificados pelos participantes do segundo grupo focal como inclusos no método *Agile Design*.

5 CONTRIBUIÇÕES TEÓRICAS E PARA A PRÁTICA

O presente estudo apresentou a relevância de um método para gestão de projetos de desenvolvimento de *software* que integra o uso da abordagem DT com métodos ágeis promovendo a entrega de *softwares* com maior aderência às necessidades dos usuários, com maior qualidade e em menor prazo. Em função de seu caráter prescritivo, o presente trabalho avaliou os problemas existentes no seu contexto original, no caso, em uma empresa de desenvolvimento de *softwares* considerando concomitantemente a perspectiva dos clientes do sistema.

Além disso, os resultados desta pesquisa contribui teoricamente com a heurística da construção de um método, que não foi encontrado na literatura, a partir de um modelo. A revisão sistemática da literatura, parte integrante desta pesquisa para a identificação de um método/modelo, sinaliza a existência recente da integração da abordagem DT e métodos ágeis na gestão de projetos de *software*. No que se refere às contribuições para prática o método promove um roteiro para uso em projetos de desenvolvimento de *software* que pela validação dos envolvidos pode ser generalizável.

5.1 CONTRIBUIÇÕES TEÓRICAS

O método de pesquisa escolhido, *Design Science Research*, permitiu que os problemas fossem identificados, categorizados e tratados por meio da prescrição de um método, definido como o artefato do presente trabalho. Em função da aplicação do *Design Science Research* em pesquisas que envolvam SI ter como objetivo servir de auxílio aos pesquisadores no consumo e produção do conhecimento (Gregor & Hevner, 2013), cabe ressaltar a heurística da construção do presente estudo, que compreende como o método foi construído. Ou seja, a partir das etapas de pesquisa estabelecidas por Dresch *et al.* (2015), a presente pesquisa desenvolveu um método a partir de um modelo existente, em função do mesmo não ter sido encontrado.

A presente pesquisa também contribui teoricamente ao cobrir uma lacuna com a extensão de um modelo proposto para a geração de um método para a gestão de projetos de desenvolvimento de *softwares* com uso da abordagem DT integrada com métodos ágeis. O desenvolvimento de um artefato é reconhecido como uma contribuição de alta utilidade para o conhecimento (Gregor & Hevner, 2013). De acordo com Gregor e Hevner (2013) existem

diferentes níveis de contribuição quando o método *Design Science Research* é aplicado. Em virtude do presente estudo ter desenvolvido o método *Agile Design* como o artefato da *Design Science Research* é possível afirmar que foi feita uma contribuição para promover o conhecimento por meio de uma arquitetura e princípios operacionais (Gregor & Hevner, 2013).

O presente estudo considerou a validação conceitual do método por meio de grupos focais compostos por praticantes no segmento de SI. De acordo com Gregor e Hevner (2013), existem três níveis de contribuição da *Design Science Research*: (i) o nível 1 corresponde a contribuição específica por meio da implementação do artefato denominada instanciação; (ii) o nível 2 corresponde ao conhecimento por meio de princípios operacionais, tais como modelos e métodos, e; (iii) o nível 3 consiste em um nível maior de abstração em virtude de um conhecimento mais maduro. Sendo assim, em função de ter sido gerado um método, o presente estudo está no nível 2 de contribuição, em virtude da teoria estar em desenvolvimento. Isso significa que embora já exista literatura sobre abordagem DT e métodos ágeis, a proposta de integração de ambos em um método com passo a passo gerou uma contribuição para que, a partir dele, novas interações para garantir a efetiva contribuição do método e validar todos os processos, técnicas e ferramentas sugeridas sejam realizados. Ademais, cabe ressaltar que o método *Agile Design* necessita ser validado por meio de instanciação em outros contextos, visto que o desenvolvimento da teoria necessita que o artefato seja avaliado com medidas quantitativas de efetividade (Gregor & Hevner, 2013). Portanto, o artefato desenvolvido poderá ser validado por meio da instanciação em pesquisas futuras conforme exposto no Capítulo 6, apontando esta como a principal contribuição teórica do presente estudo.

5.2 CONTRIBUIÇÕES PARA A PRÁTICA

A presente pesquisa propôs para a área de TI, mais especificamente para o segmento de SI, um método de gestão de projetos para auxiliar as empresas no desenvolvimento de *softwares*. Por meio de uma prescrição, o *Agile Design* busca ser um método, com processos detalhados para a gestão de um projeto de desenvolvimento de *software*, sendo composto por 5 etapas e 12 processos, cada um com ferramentas e técnicas sugeridas para serem utilizados pelos profissionais da área de TI. Cabe ressaltar que o roteiro que compõe o método presente no Apêndice H deste documento, foi elaborado para que as ferramentas e técnicas sejam

adotadas ao longo do ciclo de vida do desenvolvimento de um *software*, considerando, em sua composição, a integração da abordagem DT com métodos ágeis.

Esse método prevê a identificação das necessidades dos usuários e clientes do *software* logo no início do projeto e promove, simultaneamente, a entrega do software em ondas sucessivas. Por esse motivo, o *Agile Design* é um método concebido a partir do modelo que integra a abordagem DT e métodos ágeis de Bosch & Bosch-Sijtsema (2011). Tal modelo foi escolhido em função da aderência aos problemas identificados ao longo dos projetos de desenvolvimento de *software* mapeados em uma empresa de tecnologia provedora de soluções em sistemas.

Em virtude do presente estudo utilizar a combinação da abordagem DT e de métodos ágeis, sendo este último usado por 7 entre 10 organizações (PMI, 2017b), buscou-se identificar quais os potenciais benefícios derivados da combinação de ambos. Embora autores defendam a combinação da abordagem DT e métodos ágeis (Newman et al., 2015; Steinke et al., 2017), outros autores mencionam sua similaridade (Fischer & Senft, 2016; Higuchi & Nakano, 2017). Como não foi possível identificar a existência de um método que integra a abordagem DT e métodos ágeis na busca realizada na literatura, o *Agile Design* foi proposto, desenvolvido e validado.

Deste modo, foi possível identificar aspectos presentes na metodologia ágil e os benefícios da aplicação da abordagem DT quando usados conjuntamente, definindo-se assim as características do método *Agile Design*, desenvolvido e validado por representantes da Keep IT Simple e por representantes do cliente de um dos *softwares* desenvolvido por ela. A Tabela 25 relaciona (i) os benefícios da aplicação dos métodos ágeis de acordo com Nigam e Gupta (2017), (ii) os benefícios obtidos a partir do uso da abordagem DT na opinião de diversos autores e (iii) as ferramentas e características incorporadas ao *Agile Design*, validados pelos grupos focais. Este relacionamento ocorreu para seis aspectos considerados como relevantes na execução do projeto: comunicação, empatia, dinamismo, eficiência, flexibilidade e qualidade. Deste modo é possível inferir que existe uma complementaridade entre os métodos ágeis e a abordagem DT. Conclui-se que suas características estão refletidas no *Agile Design*, que por sua vez poderá ser aplicado em projetos de desenvolvimento de *software* de modo a obter de forma potencializada os benefícios esperados pela aplicação do método ágil e da abordagem DT, em comparação ao uso individual de cada um.

Tabela 25: Métodos ágeis, abordagem DT e características do *Agile Design*

Aspectos relevantes para o projeto	Benefícios da aplicação de métodos ágeis (Nigam e Gupta, 2017)	Benefícios obtidos com o uso da abordagem DT	Características incorporadas ao método *Agile Design*
Comunicação	Comunicação eficaz entre os membros da equipe auto gerenciáveis	Identificar aspectos comportamentais das pessoas e assim convertê-los em benefícios dos clientes e valor ao negócio (Brown, 2008)	Identificação de personas e usuários extremos permitem que a comunicação entre time do projeto e os clientes reais seja encorajada desde os estágios iniciais do projeto
Empatia	Não se aplica	Ver, pensar e abordar problemas (Pinheiro & Alt, 2011), concentrando as ideias e necessidades das pessoas (Plattner, 2009)	A coleta das informações dos clientes é executada por meio das ferramentas de coleta das hipóteses de solução, mapeamento da jornada do cliente e desenho da solução
Dinamismo	Fruto da integração contínua entre time do projeto e usuários, os desenvolvedores lançam as *iterações* do produto em poucas semanas	Não se aplica	Limitação das iterações em até 3 semanas para promover entregas com menor duração e com maior frequência.
Eficiência	Iterações curtas com *feedback* do cliente promovem agilidade e eficiência na resposta às mudanças nos requisitos	Realizar testes rápidos por meio da prototipação no intuito de aproximar o que foi desenvolvido com o esperado pelo usuário final (Plattner, 2009)	Participação de usuários-chave e usuários-extremos na maioria dos processos previstos no método, de forma em que atuem conjuntamente com o time do projeto
Flexibilidade	Quebra de processos de desenvolvimento de *software* rígidos	Não se aplica	Durante a realização do desenho detalhado da solução é possível que os praticantes do método voltem à etapa de identificação dos participantes ou do desenho preliminar caso seja identificada lacunas nos requisitos nos estágios mais avançados do projeto
Qualidade	Monitoramento do progresso em tempo real, tornando o processo dinâmico e com menor sobrecarga	Gerar constantemente novas ideias com foco na solução e tradução dos requisitos em objetivos do negócio (Cross, 2006)	Processo de garantia do produto incorporado ao processo para coletar informações relacionadas ao produto após a implantação em tempo de projeto e realizar o suporte ao software

Fonte: Autor.

6 CONSIDERAÇÕES FINAIS

Este trabalho apresentou o modelo *Agile Design* desenvolvido com base na integração de características presentes na abordagem DT e também nos métodos ágeis para o desenvolvimento de *software*. Este método, presente no Apêndice H e ilustrado na Figura 16, apresenta características que podem ser benéficas para a resolução dos problemas que ocorrem ao longo do ciclo de vida existente no desenvolvimento de *softwares*. Por esta razão, os resultados permitem a resposta da seguinte questão de pesquisa: **Como o *design thinking* pode ser integrado com métodos ágeis em projetos de desenvolvimento de *softwares*?** O método *Agile Design* é caracterizado pelo aspecto inovador no sentido de unificar uma abordagem e um método em projetos de desenvolvimento de *softwares*, caracterizado por aspectos de incerteza e intangibilidade, visto que normalmente a concepção de uma solução tecnológica nem sempre surge de objetivos claros e já solidificados. O método *Agile Design* dá ênfase aos aspectos relevantes da abordagem DT e métodos ágeis, auxiliando no desenvolvimento de *softwares* com qualidade e, assim, atender ao objetivo principal da presente pesquisa que é **como o DT pode ser integrado com métodos ágeis em projetos para o desenvolvimento de *software***.

Para atender ao objetivo específico, identificar quais métodos utilizam a abordagem DT integrada com métodos ágeis em projetos de desenvolvimento de *software*, foi desenvolvida uma revisão sistemática da literatura que permitiu identificar 29 artigos que estabeleciam modelos integrando a abordagem DT com métodos ágeis. Com base nesta revisão, pode-se constatar que não existia método concebido para a gestão de projetos de desenvolvimento de software e por esse motivo, um método foi concebido.

Por meio de entrevistas realizadas em conformidade com as etapas da pesquisa de identificação e conscientização do problema, foram identificados 25 problemas nos projetos de desenvolvimento de software da Keep IT Simple que poderiam ser solucionados com a aplicação da abordagem DT integrada com métodos ágeis. Atendendo assim o objetivo específico de identificar os problemas existentes no uso exclusivo de métodos ágeis em projetos de desenvolvimento de *software* em uma empresa de TI. Estes problemas, além de serem documentados, categorizados e agrupados foram submetidos à análise de especialistas em métodos ágeis e abordagens DT de modo que houvesse a verificação de que tais problemas seriam solucionados pela aplicação do método integrado de gestão de projetos.

O método *Agile Design* que integra a abordagem DT e métodos ágeis, criado no presente estudo, atende ao terceiro objetivo secundário. Esse método é composto por cinco etapas e 12 processos, foi derivado do esforço conjunto do pesquisador, dos membros das equipes da empresa desenvolvedora do *software*, Keep IT Simple e de representantes da empresa contratante. Sendo assim, o método *Agile Design* incorpora características que beneficiam não só o fornecedor do *software*, mas também aumenta a possibilidade de os clientes terem a sua disposição um produto que atenda às suas necessidades em virtude de o método incorporar as perspectivas da contratante e da contratada. A composição do método Agile Design foi derivada da validação de representantes da empresa desenvolvedora e da empresa cliente do *software*, assegurando que o objetivo específico de validar o método proposto que integra a abordagem DT e métodos ágeis no contexto de uma empresa de TI fosse atendido.

A busca pela melhoria no desempenho das atividades em TI tem justificativa por ser uma área onde normalmente o grau de clareza dos requisitos de um *software* poucas vezes é conhecido no início de um projeto. Além disso, frequentemente os próprios clientes e usuários possuem um problema, mas têm dificuldade de externar quais são as suas principais dores ou necessidades.

A limitação principal deste trabalho caracteriza-se pelo fato de o método não ter sido instanciado, ou seja, não foi aplicado durante a realização de um projeto de desenvolvimento de *software* de fato. A partir da instanciação é possível que eventuais ajustes no método *Agile Design* sejam identificados a partir de sua aplicação prática. Diante do uso em situações reais será possível avaliar se os processos e ferramentas propostos irão suportar a concepção, o desenvolvimento, e a entrega de *softwares* com melhor qualidade e aumentando a satisfação dos clientes e usuários.

Outra limitação deste trabalho é o fato de ter sido utilizada uma empresa de pequeno porte do segmento de TI como unidade de análise para o desenvolvimento do artefato. Em função da Keep IT Simple possuir o histórico de poucos projetos concluídos até o momento em que a pesquisa foi realizada, os problemas identificados podem estar associados a questões específicas, que podem não ser as mesmas existentes em grandes organizações. Dependendo da variação da complexidade envolvida no contexto organizacional, variações no método propostos podem ser exigidas, como por exemplo, aplicar etapas e processos específicos em determinadas etapas do ciclo de vida de um projeto de desenvolvimento de *software*.

Algumas questões permanecem pendentes de resolução, que poderão ser solucionadas por meio da realização de pesquisas futuras. Como exemplo de pesquisas futuras, desenvolver

um estudo para a construção de uma plataforma com *templates* e padrões de cada ferramenta proposta pelo método *Agile Design* para que sirva como roteiro de uso do método em um formato de guia completo para a execução. Por fim, também é possível que seja realizado um estudo que avalie o desempenho dos *softwares* desenvolvidos por meio do método *Agile Design*, que integra a abordagem DT e métodos ágeis de modo a aferir seus benefícios em comparação com outros métodos de gestão de projetos no desenvolvimento de *softwares*. A proposta de estudo futuro com a instanciação do *Agile Design* reitera a característica de generalização do método, resultado do presente estudo, para outras organizações de TI para o desenvolvimento de SI.

REFERÊNCIAS

Abrahamsson, P., Salo, O., Ronkainen, J., & Warsta, J. (2002). Agile software development methods: Review and analysis. *Espoo, Finland: Technical Research Centre of Finland, VTT Publications*, 478. http://doi.org/10.1076/csed.12.3.167.8613

Adikari, S., McDonald, C., & Campbell, J. (2009). Little design up-front: A design science approach to integrating usability into agile requirements engineering. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *5610 LNCS*(PART 1), 549–558. http://doi.org/10.1007/978-3-642-02574-7_62

Ahimbisibwe, A., Urs, D., & Cavana, R. Y. (2016). Empirical comparison of traditional plan-based and agile methodologies: Critical success factors for outsourced software development projects from vendors' perspective. *Journal of Enterprise Information Management*, *30*(3), 400–453.

Ardito, C., Baldassarre, M. T., Caivano, D., & Lanzilotti, R. (2017). Integrating a SCRUM-Based Process with Human Centred Design: An Experience from an Action Research Study. *2017 IEEE/ACM 5th International Workshop on Conducting Empirical Studies in Industry (CESI)*, 2–8. http://doi.org/10.1109/CESI.2017.7

Ardito, C., Buono, P., Caivano, D., Costabile, M. F., & Lanzilotti, R. (2014). Investigating and promoting UX practice in industry: An experimental study. *International Journal of Human-Computer Studies*, *72*(6), 542–551.

Baweja, S., & Venugopalan, N. (2015). Agility in Project Management. *PM World Journal Agility in Project Management*, *IV*(X), 1–14. Retrieved from www.pmworldlibrary.net

Boell, S. K., & Cecez-Kecmanovic, D. (2015). What is an Information System? In *2015 48th Hawaii International Conference on System Sciences* (pp. 4959–4968). HICSS. http://doi.org/10.1109/HICSS.2015.587

Bosch, J., & Bosch-Sijtsema, P. M. (2011). Introducing agile customer-centered development in a legacy software product line. *Software - Practice and Experience*, *41*, 871–882.

Bossert, O., Kretzberg, A., & Laartz, J. (2018). Unleashing the power of small, independent teams. Retrieved August 11, 2018, from https://www.mckinsey.com/business-functions/organization/our-insights/unleashing-the-power-of-small-independent-teams?cid=other-eml-alt-mkq-mck-oth-1807&hlkid=f203710029004b82912999bd5cd56279&hctky=2833613&hdpid=e99581b0-56c3-4b7e-bb17-d578ed705304

Bourimi, M., Barth, T., Haake, J. M., Ueberschär, B., & Kesdogan, D. (2010). *AFFINE for enforcing earlier consideration of NFRs and human factors when building socio-technical systems following agile methodologies*. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* (Vol. 6409 LNCS). http://doi.org/10.1007/978-3-642-16488-0_15

Brhel, M., Meth, H., Maedche, A., & Werder, K. (2015). Exploring principles of user-centered agile software development: A literature review. *Information and Software Technology*, *61*, 163–181.

Brown, T. (2008). Design Thinking. *Harvard Business Review*.

Brown, T. (2009). *Change by Design: How Design Thinking Transforms Organizations and Inspires Innovation*. New York: Harper Collins.

Brown, T., & Katz, B. (2011). Change by design. *Journal of Product Innovation Management*, *28*, 381–383.

BSA. (2016). *Economic Impact of Software*.

Cano, S. P., González, C. S., Collazos, C. A., Arteaga, J. M., & Zapata, S. (2015). Agile

Software Development Process Applied to the Serious Games Development for Children from 7 to 10 Years Old. *International Journal of Information Technologies and Systems Approach*, *8*(2), 64–79. http://doi.org/10.4018/IJITSA.2015070105

Cohen, S. J., & Money, W. H. (2008). Bridge methods: Complementary steps integrating Agile development tools & methods with formal process methodologies. *Proceedings of the Annual Hawaii International Conference on System Sciences*, 1–10. http://doi.org/10.1109/HICSS.2008.76

Conboy, K., & Fitzgerald, B. (2010). Method and developer characteristics for effective agile method tailoring. *ACM Transactions on Software Engineering and Methodology*, *20*(1), 1–30. http://doi.org/10.1145/1767751.1767753

Conforto, E. C., Amaral, D. C., da Silva, S. L., Di Felippo, A., & Kamikawachi, D. S. L. (2016). The agility construct on project management theory. *International Journal of Project Management*, *34*(4), 660–674. http://doi.org/10.1016/j.ijproman.2016.01.007

Cross, N. (1982). Designerly ways of knowing. *Design Studies*, *3*(4), 221–227. http://doi.org/10.1016/0142-694X(82)90040-0

Cross, N. (2006). *Designerly ways of Knowing*. Berlin: Springer.

Curran, C., Garrett, D., & Puthiyamadam, T. (2017). *A decade of digital - Keeping pace with transformation*. *PwC*. Retrieved from http://www.pwc.com/us/en/advisory-services/digital-iq/assets/pwc-digital-iq-report.pdf

Darrin, M. A. G., & Devereux, W. S. (2017). The Agile Manifesto, design thinking and systems engineering. In *11th Annual IEEE International Systems Conference (SysCon)*. http://doi.org/10.1109/SYSCON.2017.7934765

Davenport, T. H. (1993). *Reengineering Work through Information Technology*. http://doi.org/10.5465/AME.1993.9411302338

Davenport, T. H. (1998). Putting the enterprise into the enterprise system. *Harvard Business Review*, *76*(4), 121–131. http://doi.org/Technical Report

De Sordi, J. O., Meireles, M., & Sanches, C. (2011). Applied Design Science To the Business Management Researches:Reflections Starting From the Recent Historical of International Publications. *Review of Administration and Innovation - RAI*, *8*(1). http://doi.org/10.5773/rai.v8i1.770

Design Council. (2007). *Eleven lessons: managing design in eleven global companies. Desk research report*. *Engineering* (Vol. 44). Retrieved from http://www.designcouncil.org.uk/Documents/Documents/Publications/Eleven Lessons/ElevenLessons_DeskResearchReport.pdf

Design Council. (2018). The Design Process: What is the Double Diamond? Retrieved April 22, 2018, from https://www.designcouncil.org.uk/news-opinion/design-process-what-double-diamond

DIS, I. (2010). *9241-210: 2010. Ergonomics of human system interaction-Part 210: Human-centred design for interactive systems. International Standardization Organization (ISO)*. *International Organization for Standardization*. Switzerland. Retrieved from https://www.iso.org/standard/52075.html

Dorst, C. H. (2006). Design Problems and Design Paradoxes. *Design Issues*, *22*(3), 4–17. http://doi.org/10.1162/desi.2006.22.3.4

Dresch, A., Lacerda, D. P., & Antunes Jr, J. A. V. (2015). *Design Science Research*. (B. E. Ltda., Ed.). São Paulo: Springer International Publishing.

Dwivedi, Y. K., Wastell, D., Laumer, S., Henriksen, H. Z., Myers, M. D., Bunker, D., … Srivastava, S. C. (2015). Research on information systems failures and successes: status update and future directions. *Information Systems Frontiers*, *17*(1), 143–157. http://doi.org/http://dx.doi.org/10.1007/s10796-014-9500-y

Farid, A. B., Helmy, Y. M., & Bahloul, M. M. (2017). Enhancing Lean Software

Development by using DevOps Practices. *International Journal of Advanced Computer Science and Applications*, *8*(7), 267–277.
Ferreira, M. A. S. V. P. (2013). A pesquisa e a estruturação do artigo acadêmico em administração. *Revista Ibero-Americana de Estratégia - RIAE*, *12*(2), 1–11.
Fischer, H., & Senft, B. (2016). Human-Centered Software Engineering as a Chance to Ensure Software Quality Within the Digitization of Human Workflows. In *Human-Centered and Error-Resilient Systems Development* (pp. 30–41). Springer, Cham.
Fleury, A. L., Stabile, H., & Carvalho, M. M. (2016). An Overview of the Literature on Design Thinking: Trends and Contributions. *International Journal of Engineering Education*, *32*(4), 1704–1718.
Forbrig, P. (2016a). Continuous software engineering with special emphasis on continuous business-process modeling and human-centered design. In *ACM International Conference Proceeding Series* (Vol. 07–08–Apri). http://doi.org/10.1145/2882879.2882895
Forbrig, P. (2016b). When do Projects End? – The Role of Continuous Software Engineering. In *International Conference on Business Informatics Research* (pp. 107–121). Springer, Cham.
Forbrig, P., & Saurin, M. (2016). Supporting the HCI Aspect of Agile Software Development by Tool Support for UI-Pattern Transformations. In *Human-Centered and Error-Resilient Systems Development* (pp. 17–29). Springer, Cham.
Freitas Jr, J. C. da S., Machado, L., Klein, A. Z., & Freita, A. S. de. (2015). Design Research: Aplicações práticas e lições aprendidas. *Revista de Administração FACES Journal*, *1*(14), 95–116.
Gamble, M. T. (2016). Can Metamodels Link Development to Design Intent? In *1st International Workshop on Bringing Architectural Design Thinking Into Developers' Daily Activities* (pp. 14–17).
Gibson, M., & Arnott, D. (2007). The use of focus groups in design science research. In *ACIS 2007 Proceedings, 14* (Vol. 14). http://doi.org/10.1007/978-1-4419-5653-8
Giff, S., & Dogan, H. (2016). UX Research is Dead. Long live UX Research. In *British HCI* (pp. 1–3). Bournemouth: BCS Learning and Development Ltd. http://doi.org/http://dx.doi.org/10.14236/ewic/HCI2016.61
Gill, T. G., & Hevner, A. R. (2013). A Fitness-Utility Model for Design Science Research. *ACM Transactions on Management Information Systems*, *4*(2), 1–24. http://doi.org/10.1145/2499962.2499963
Góes, R. s., & Russo, R. de F. S. M. (2018). Integração do Design Thinking com Métodos Ágeis em projetos de desenvolvimento de software. *Revista Mundo PM, 78*, 48–58.
González-González, C. S., Toledo-Delgado, P., & Muñoz-Cruz, V. (2015). Agile human centered methodologies to develop educational software | Metodologías ágiles centradas en personas para desarrollar software educativo. *DYNA (Colombia)*, *82*(193), 187–194. http://doi.org/10.15446/dyna.v82n193.53495
Google. (2018). Design sprint kit. Retrieved April 22, 2018, from https://designsprintkit.withgoogle.com/methods/
Gregor, S., & Hevner, A. R. (2013). Positioning and presenting design science research for maximum impact. *MIS Quarterly*, *37*(2), 337–355.
Gruber, M., Leon, N. de, George, G., & Thompson, P. (2015). Managing by design. *Academy of Management Journal*, *58*, 1–7. http://doi.org/http://dx.doi.org/10.5465/amj.2015.4001
Gurusamy, K., Srinivasaraghavan, N., & Adikari, S. (2016). An Integrated Framework for Design Thinking and Agile Methods for Digital Transformation. In *International Conference of Design, User Experience, and Usability* (pp. 34–42). Springer, Cham.
Gurusamy, K., Srinivasaraghavan, N., Adikari, S., B, H. F., Forbrig, P., Saurin, M., … Labrie,

R. C. (2016). Human-Centered and Error-Resilient Systems Development. *The Design Journal*, *8*(sup1), 1–28. http://doi.org/10.1007/978-3-319-44902-9

Hevner, A. R., March, B. S. T., Park, J., & Ram, S. (2004). Design science in information systems research. *MIS Quaterly*, *28*(1), 75–105.

Highsmith, J. A. (2002). What Is Agile Software Development? *The Journal of Defense Software Engineering*, *15*(10), 4–9. http://doi.org/10.1109/MC.2007.73

Higuchi, M. M., & Nakano, D. N. (2017). Agile Design: A Combined Model Based on Design Thinking and Agile Methodologies for Digital Games Projects. *Revista de Gestão e Projetos*, *8*(2), 109–126. http://doi.org/10.5585/10.5585

Hussain, Z., Lechner, M., Milchrahm, H., Shahzad, S., Slany, W., Umgeher, M., & Wolkerstorfer, P. (2008). *Agile user-centered design applied to a mobile multimedia streaming application*. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* (Vol. 5298 LNCS). http://doi.org/10.1007/978-3-540-89350-9-22

Isa, W. A. R. W. M., Lokman, A. M., Aris, S. R. S., Aziz, M. A., Taslim, J., Manaf, M., & Sulaiman, R. (2014). Engineering rural informatics using agile user centered design. *2014 2nd International Conference on Information and Communication Technology, ICoICT 2014*, 367–372. http://doi.org/10.1109/ICoICT.2014.6914093

Kolko, J. (2018). Introduction to Design Strategy. http://doi.org/10.1016/B978-012064155-0/50001-X

Lacerda, D. P., Dresch, A., Proença, A., & Antunes Junior, J. A. V. (2013). Design Science Research: Método de pesquisa para a engenharia de produção. *Gestao e Producao*, *20*(4), 1–21.

Larson, D., & Chang, V. (2016). A review and future direction of agile, business intelligence, analytics and data science. *International Journal of Information Management*, *36*(5), 700–710. http://doi.org/10.1016/j.ijinfomgt.2016.04.013

Lárusdóttir, M., Cajander, Å., & Gulliksen, J. (2014). Informal feedback rather than performance measurements - User-centred evaluation in Scrum projects. *Behaviour and Information Technology*, *33*(11), 1118–1135. http://doi.org/10.1080/0144929X.2013.857430

Lazaris, C., & Vrechopoulos, A. (2014). From multi-channel to "omnichannel" retailing: review of the literature and calls for research. In *2nd International Conference on Contemporary Marketing Issues* (pp. 18–20). ICCMI. http://doi.org/10.13140/2.1.1802.4967

Lester, C. Y. (2011). Combining agile methods and user-centered design to create a unique user experience: An empirical inquiry. *ACHI 2011 - 4th International Conference on Advances in Computer-Human Interactions*, (c), 16–21. Retrieved from https://www.scopus.com/inward/record.uri?eid=2-s2.0-84883132112&partnerID=40&md5=a0f3dc55aa7b9242a96ba3c8be4427af

Livermore, J. A. (2008). Factors that significantly impact the implementation of an agile software development methodology. *Journal of Software*, *3*(4), 31–36. http://doi.org/10.4304/jsw.3.4.31-36

Losada, B., Urretavizcaya, M., & De Castro, I. F. (2011). An integrated approach to develop interactive software. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *6949 LNCS*(PART 4), 470–474. http://doi.org/10.1007/978-3-642-23768-3_60

Lucena, Braz, A., Chicoria, A., & Tizzei, L. (2016). IBM Design Thinking Software Development Framework. In *Brazilian Workshop on Agile Methods* (pp. 98–109). Springer, Cham. http://doi.org/10.1007/978-3-319-55907-0

March, S. T., & Smith, G. F. (1995). Design and natural science research on information

technology. *Decision Support Systems*, *15*(4), 251–266. http://doi.org/10.1016/0167-9236(94)00041-2

Marshall, C., & Rossman, G. B. (2014). *Designing qualitative research*. Sage publications.

Martin, R. L. (2009). *The design of business - why design thinking is the next competitive advantage*. Boston: Harvard Business Press.

Maruping, L. M., Venkatesh, V., & Agarwal, R. (2009). A control theory perspective on agile methodology use and changing user requirements. *Information Systems Research*, *20*(3), 377–399. http://doi.org/10.1287/isre.1090.0238

Massari, V. L., & Vidal, A. (2018). *Gestão Ágil de Produtos com Agile Think Business Framework: Guia para certificação EXIN Agile Scrum Product Owner*. Brasport.

Mishra, D., & Mishra, A. (2011). Complex software project development : agile methods adoption. *Journal of Software Maintenance and Evolution*, *23*(January), 549–564. http://doi.org/10.1002/smr

Mustonen-Ollila, E., & Lyytinen, K. (2003). Why organizations adopt information system process innovations: a longitudinal study using Diffusion of Innovation theory. *Info Systems J*, *13*, 275–297. http://doi.org/10.1046/j.1365-2575.2003.00141.x

Nakao, Y., Moriguchi, M., & Noda, H. (2014). Using agile software development methods to support human-centered design. *NEC Technical Journal*, *8*(3), 37–40.

Newman, P., Ferrarioy, M. A., Simm, W., Forshawz, S., Friday, A., & Whittle, J. (2015). The Role of Design Thinking and Physical Prototyping in Social Software Engineering. In *37th International Conference on Software Engineering* (Vol. 2, pp. 487–496). IEEE Press.

Nigam, C., & Gupta, S. (2017). Agile Methodology for Software Development. *Institute of Innovation in Technology & Management - Journal of Information Technology*, *3*, 54–64.

O'Driscoll, K. (2016). The agile data modelling & design thinking approach to information system requirements analysis. *Journal of Decision Systems*, *25*(sup1), 632–638.

Osterwalder, A., & Pigneur, Y. (2010). *Business model generation: a handbook for visionaries, game changers, and challengers*. John Wiley & Sons.

Owens, D., & Khazanchi, D. (2018). From Strategic Intent to Implementation: How Information Technology initiatives take shape in organizations. In *51st Hawaii International Conference on System Sciences* (pp. 4783–4792).

Paula, D. F. O. De, & Araujo, C. C. (2016). *Pet Empires: Combining Design Thinking, Lean Startup and Agile to Learn from Failure and Develop a Successful Game in an Undergraduate Environment*. *International Conference on Human-Computer Interaction*. Cham: Springer. http://doi.org/10.1007/978-3-319-40548-3

Pereira, J. C., & Russo, R. de F. S. M. (2018). Design Thinking Integrated in Agile Software Development: A Systematic Literature Review. In *Procedia Computer Science* (Vol. 138, pp. 775–782). Elsevier B.V. http://doi.org/10.1016/j.procs.2018.10.101

Petersen, K., & Wohlin, C. (2009). A comparison of issues and advantages in agile and incremental development between state of the art and an industrial case. *Journal of Systems and Software*, *82*(9), 1479–1490. http://doi.org/10.1016/j.jss.2009.03.036

Petticrew, M., & Roberts, H. (2006). *Systematic reviews in the social sciences: A practical guide*. Oxford, England: Blackwell Publishing.

Piccoli, & Ives. (2005). Review: IT-Dependent Strategic Initiatives and Sustained Competitive Advantage: A Review and Synthesis of the Literature. *MIS Quarterly*, *29*(4), 747. http://doi.org/10.2307/25148708

Pieroni, A., Scarpato, N., & Scorza, M. (2018). Affective agile design a proposal for a new software development model. *Journal of Theoretical and Applied Information Technology*, *96*(1).

Pinheiro, T., & Alt, L. (2011). *Design thinking brasil*. Rio de Janeiro: Campus.

Plattner, H. (2009). *An introduction to design thinking: Process guide*. *Stanford Institute of Design*. Palo Alto, CA: Stanford Institute of Design. http://doi.org/10.1007/978-1-4302-6182-7_1

PMI. (2017a). *Agile Practice Guide*.

PMI. (2017b). *Success Rates Rise 2017 9th Global Project Management Survey*. *PMI's Pulse of the Profession*. Retrieved from http://www.pmi.org/-/media/pmi/documents/public/pdf/learning/thought-leadership/pulse/pulse-of-the-profession-2017.pdf

PMI. (2017c). *Um guia do conhecimento em gerenciamento de projetos - Guia PMBOK* (6a Edição).

Prior, S., Waller, A., Black, R., & Kroll, T. (2013). Use of an agile bridge in the development of assistive technology. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems - CHI '13*, 1579. http://doi.org/10.1145/2470654.2466210

Riis, J. O. (2012). *Design of enterprise information systems: Roots, nature and new approaches*. *Lecture Notes in Business Information Processing* (Vol. 105 LNBIP). http://doi.org/10.1007/978-3-642-28827-2_1

Sauser, B. J., Reilly, R. R., & Shenhar, A. J. (2009). Why projects fail? How contingency theory can provide new insights - A comparative analysis of NASA's Mars Climate Orbiter loss. *International Journal of Project Management*, *27*(7), 665–679. http://doi.org/10.1016/j.ijproman.2009.01.004

Schön, E.-M., Winter, D., Uhlenbrok, J., Escalona, M. J., & Thomaschewski, J. (2016). Enterprise Experience into the Integration of Human-Centered Design and Kanban. In *11th International Joint Conference on Software technologies* (pp. 133–140).

Schwaber, K. (2004). *Agile Project Management with Scrum*. *Microsoft Press* (Vol. 7). http://doi.org/10.1201/9781420084191-c2

Sebok, A., Walters, B., & Plott, C. (2017). Integrating Human-Centered Design and the Agile Development Process for Safety and Mission Critical System Development, 1086–1090. http://doi.org/10.1177/1541931213601876

Senapathi, M., & Drury-Grogan, M. L. (2017). Refining a model for sustained usage of agile methodologies. *Journal of Systems and Software*, *132*, 298–316. http://doi.org/10.1016/j.jss.2017.07.010

Serrador, P., & Pinto, J. K. (2015). Does Agile work? - A quantitative analysis of agile project success. *International Journal of Project Management*, *33*(5), 1040–1051. http://doi.org/10.1016/j.ijproman.2015.01.006

Shenhar, A. J., & Dvir. (2010). *Reinventando o gerenciamento de projetos: a abordagem diamante ao crescimento e inovação bem sucedidos*. *São Paulo: MBrooks* (1st ed.). São Paulo: M. Books.

Sheppard, B., Edson, J., & Kouyoumjian, G. (2017). More than a feeling: Ten design practices to deliver business value. Retrieved March 20, 2018, from https://www.mckinsey.com/business-functions/mckinsey-design/our-insights/more-than-a-feeling-ten-design-practices-to-deliver-business-value

Sidky, A., & Smith, G. (2009). *Becoming Agile in an imperfect World*. Greenwich: Manning Publication Co.

Simon, H. A. (1996). *The sciences of the artificial* (3rd Ed.). Cambridge: MIT Press.

Smith, S. L., & Aucella, A. F. (1983). *Design guidelines for the user interface to computer-based information systems* (No. MTR-88). MITRE CORP BEDFORD MA.

Sohaib, O., & Khan, K. (2011). Incorporating discount usability in extreme programming. *International Journal of Software Engineering and Its Applications*, *5*(1), 51–62.

Solinski, A., & Petersen, K. (2016). *Prioritizing agile benefits and limitations in relation to*

practice usage. *Software Quality Journal* (Vol. 24). Springer US. http://doi.org/10.1007/s11219-014-9253-3

Standish Group, I. (2013). *The Chaos Manifesto*. *Chaos Manifesto* (Vol. 2).

Standish Group, I. (2014). *The Chaos Manifesto*.

Steinke, G. H., Al-deen, M. S., & Labrie, R. C. (2017). Innovating Information System Development Methodologies with Design Thinking. In *5th International Conference on Applied Innovations in IT* (pp. 51–55).

Tremblay, M. C., Hevner, A. R., & Berndt, D. J. (2010). Focus Groups for Artifact Refinement and Evaluation in Design Research. *Communications of the Association for Information Systems*, *26*, 599–618. http://doi.org/10.1007/978-1-4419-5653-8_10

Turner, J. R., & Cochrane, R. (1993). Goals-and-methods matrix: coping with projects with ill defined goals and/or methods of achieving them. *International Journal of Project Management*, *11*(2), 93–102.

Vallon, R., da Silva Estácio, B. J., Prikladnicki, R., & Grechenig, T. (2018). Systematic literature review on agile practices in global software development. *Information and Software Technology*, *96*, 161–180. http://doi.org/10.1016/j.infsof.2017.12.004

Van Aken, J. E. (2004). Management research on the basis of the design paradigm: The quest for field-tested and grounded technological rules. *Journal of Management Studies*, *41*(2), 219–246.

Venable, J. R. (2006). The Role of Theory and Theorising in Design Science Research. In *1st International Conference on Design Science in Information Systems and Technology - DESRIST* (pp. 1–18).

VersionOne. (2018). 12th Annual State of Agile Report.

Vianna, M., Vianna, Y., Adler, I. K., Brenda, L., & Russo, B. (2012). *Design Thinking: Inovação em negócios*. *MVJ Press* (1st ed., Vol. 28). Rio de Janeiro: MVJ Press.

Vidal, A. (2017). *Agile Thinking Canvas* (1a ed.). São Paulo: Brasport.

Wieringa, R. J., & Morali, A. (2012). Technical Action Research as a Validation Method in Information Systems Design Science. *Design Science Research in Information Systems. Advances in Theory and Practice 7th International Conference, DESRIST 2012, Las Vegas, USA*, *7286*, 220–238. http://doi.org/10.1007/978-3-642-29863-9_17

Ximenes, B. H., Alves, I. N., & Araújo, C. C. (2015). *Software project management combining Agile, Lean startup and design thinking*. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* (Vol. 9186). http://doi.org/10.1007/978-3-319-20886-2_34

Xiong, Y., & Wang, A. (2010). A new combined method for UCD and software development and case study. In *2nd International Conference on Information Science and Engineering, ICISE2010 - Proceedings*. http://doi.org/10.1109/ICISE.2010.5690032

APÊNDICE A: ROTEIRO DAS ENTREVISTAS DE IDENTIFICAÇÃO E CONSCIENTIZAÇÃO DO PROBLEMA

O roteiro preliminar das entrevistas das etapas 3.3.1 e 3.3.3 foi composto por 12 questões:

Questões demográficas:

1. Nome do entrevistado
2. Empresa onde o entrevistado trabalha atualmente
3. Quantidade de anos de experiência do entrevistado em TI (em anos)
4. Quantidade de anos de experiência do entrevistado em gestão de projetos
5. Cargo que o entrevistado ocupava no momento em que a entrevista foi realizada
6. Grau de instrução do entrevistado no momento da entrevista
7. Qual a vivência na organização

Questões práticas em relação à gestão de projetos para o desenvolvimento de *software*:

8. Como funciona a gestão de projetos atualmente empregado pela Keep Simple?
9. Quais os problemas encontrados? Com que nível de intensidade e recorrência esses problemas ocorrem?
10. Em que etapas do ciclo de vida do software os problemas ocorrem?
11. Em que o método ágil não atende sob a ótica de time de desenvolvimento de software?
12. Quais seriam as recomendações de melhorias para a resolução dos problemas que hoje vocês enfrentam?

APÊNDICE B: ROTEIRO DAS ENTREVISTAS NA ETAPA DE PROJETO DO ARTEFATO

O roteiro das duas reuniões realizadas na etapa de projeto do artefato desta pesquisa teve a seguinte composição:

1. Identificação e conscientização dos problemas da Keep IT Simple

   Neste item foram apresentados os principais problemas identificados nas entrevistas anteriores com representantes da contratante e da contratada do *software*, informando a categoria do problema, a perspectiva de cliente ou desenvolvedor do *software* e as assertivas coletadas.
2. Modelo integrado da abordagem DT com métodos ágeis (proposição), apresentando os conceitos principais e as etapas propostas no modelo: imersão na solução, imersão no código, primeira iteração, demais iterações e liberação do produto.
3. Método da abordagem DT integrado com métodos ágeis, compreendendo as características propostas por meio de:
   - Processos
   - Entradas
   - Ferramentas e técnicas
   - Participantes
   - Saídas

APÊNDICE C: IDENTIFICAÇÃO E CATEGORIZAÇÃO DOS PROBLEMAS E BENEFÍCIOS DO USO DE MÉTODOS ÁGEIS

Tabela 26: Problemas e aspectos positivos da aplicação dos métodos ágeis

Entrevistado	Perspectiva	Assertiva	Categoria	Classificação
E1	Contratada	Diferentes plataformas a serem conectadas: Banco de Dados (BD), Cobol (Alta Plataforma), APIs (Java) e Front-End (Angular).	Engenharia de software	Problema
E1	Contratada	Equipes distintas (*squads*) trabalhando de forma simultânea com uso de método ágil (Scrum)	Escopo	Aspecto positivo
E1	Contratada	a necessidade vem pré-moldada dos *Business Partners* (BP's) que cuidam da captura das necessidades (integrante do time de TI que atende o negócio). Lacuna entre o que está sendo pedido e o que o cliente realmente quer	Escopo	Problema
E1	Contratada	Desafio está em Integrar diferentes plataformas, decorrente de desejos do negócio. Integrar negócio + UX + TI Corporativo + time de desenvolvimento.	Planejamento	Problema
E1	Contratada	Estórias estão hoje incompletas. *Inception* não conta com a participação nem do BP, nem de representante do negócio. Houveram situações de "vai e volta", por exemplo, na definição do crediário.	Planejamento	Problema
E2	Contratante	Arquitetura da empresa hoje não está preparada para avaliar novas opções tecnológicas e acaba atravancando potenciais aceleradores. Maturidade da área está aquém do que é necessário e o time do projeto, de uma maneira geral, não estava capacitado. Conceito de DevOps (*development and operations*, relação ágil entre desenvolvimento e sistemas em produção) está sendo implantado na organização agora, mas ainda embrionário.	Engenharia de software	Problema
E2	Contratante	Processo de desenvolvimento e *deploy* (publicação) de programas em produção estão sendo realizados sem um modelo pré-estabelecido. Ele citou que alguns programas que ainda estariam em testes subiram para produção indevidamente. Ele informou que a ausência de um processo estruturado que estabeleça as fases do ciclo de vida de um software (*inception, design, development, tests, user acceptance and production*) dificulta o alinhamento entre as diversas frentes do projeto.	Engenharia de software	Problema
E2	Contratante	Ponto crítico do projeto é acoplar as novas funcionalidades em sistema de alta plataforma, que possui algumas restrições. O modelo de solução do sistema previsto para o projeto teve que ser moldado ao longo do desenvolvimento. Determinadas comunicações e tratamento de dados precisaram ser isolados para que não houvesse interrupção dos demais sistemas organizacionais que permaneceriam inalterados.	Engenharia de software	Problema

Entrevistado	Perspectiva	Assertiva	Categoria	Classificação
E2	Contratante	O entrevistado informou que os benefícios mais significativos do projeto foram atingidos por entregas rápidas que possibilitou testes, mesmo quando a solução ainda exigia intervenções operacionais por ter baixa sofisticação.	Escopo	Aspecto positivo
E2	Contratante	a atuação área de operações de vendas, demandante e patrocinadora do projeto não favorece a elaboração de um backlog estruturado e que o MVP muitas vezes se torna inviável pela complexidade da funcionalidade do sistema que deveria ser entregue. De acordo com o entrevistado, os representantes da área demandante querem o estado da arte já na primeira entrega.	Escopo	Problema
E2	Contratante	Diversas vezes os usuários do sistema exigem que todas as funcionalidades já existentes deveriam ser incorporadas ao projeto, sendo que algumas podem ter menor relevância. O entrevistado citou que a área exigiu que o pagamento em cheque fosse uma funcionalidade mandatória, sendo que na visão do (PO) conceder ao sistema o critério de multicanal teria maior valor ao negócio (venda iniciada no site e terminada na loja).	Escopo	Problema
E2	Contratante	É fundamental que o time do projeto contenha profissionais que conheçam o negócio da empresa. Caso não possua esse conhecimento dentro do time, será necessário contratar uma consultoria que traga boas práticas de mercado (nesse caso, varejo e comércio eletrônico).	Organizacional	Aspecto positivo
E2	Contratante	cultura organizacional atual reflete a uma postura dos usuários e clientes do sistema como recebedores de algo, sem características de parceria.	Organizacional	Problema
E2	Contratante	É fundamental que o time do projeto contenha profissionais que conheçam o negócio da empresa. Caso não possua esse conhecimento dentro do time, será necessário contratar uma consultoria que traga boas práticas de mercado (nesse caso, varejo e comércio eletrônico).	Organizacional	Problema
E2	Contratante	Em alguns casos, houve disputa sobre a responsabilidade dos erros ou itens em não conformidade identificados durante o projeto, sem a indicação de quem deveria arcar com o ônus das horas da correção. Essa disputa normalmente possuía duas alternativas: se a falha ocorreu no desenho ou especificação e deveria ser arcada pelo contratante ou se a falha foi derivada de erro técnico e deveria ser realizado sem pagamento.	Planejamento	Problema
E3	Contratante	Foi adotado um procedimento de revisão de código fonte entre os integrantes do time de desenvolvimento, buscando melhoria na qualidade do software.	Engenharia de software	Aspecto positivo
E3	Contratante	Configuração dos aspectos de usabilidade do sistema de forma centralizada	Escopo	Aspecto positivo
E3	Contratante	distância entre o que o cliente desejava e o que tecnicamente era possível de ser feito e no curto prazo.	Escopo	Problema

Entrevistado	Perspectiva	Assertiva	Categoria	Classificação
E3	Contratante	não houve a participação dos usuários finais na definição dos requisitos do software que futuramente seriam utilizados por vendedores, analistas de crédito, operadores de caixa, estoquistas e pessoal administrativo. Por esta razão, atualmente o sistema está implantado em aproximadamente 10% das lojas com pouca aderência de uso, visto que o sistema antigo ainda não foi continuado.	Escopo	Problema
E3	Contratante	sistema com várias novas funcionalidades pode ser inferior a outro software tido como obsoleto por não atender as reais necessidades dos usuários finais.	Escopo	Problema
E3	Contratante	grande número de reclamações associadas ao não uso correto do sistema (30% dos *feedbacks*) e de dúvidas (60%), o que pode indicar um problema de falta de treinamento do sistema, além da não participação dos usuários finais ao longo do processo de desenvolvimento do novo sistema.	Escopo	Problema
E3	Contratante	Representante da contratada também citou o desafio de integrar diferentes plataformas, decorrente de desejos do negócio. Integrar negócio + UX + TI Corporativo + time de desenvolvimento.	Planejamento	Problema
E4	Contratada	Problemas são normalmente identificados na etapa de QA do ciclo de desenvolvimento de software. Tarefas e atividades complexas. Coloca muita coisa em produção com erro. Uma subida para corrigir algo, afeta o que já está em produção.	Engenharia de software	Problema
E4	Contratada	Conhecimento do ambiente (organizacional) permite a oferta de soluções	Escopo	Aspecto positivo
E4	Contratada	Entrega de um produto durante um prazo pré-estabelecido pela iteração pode não trazer valor, de fato, ao negócio	Escopo	Problema
E4	Contratada	Subjetividade do que é o valor para o negócio	Escopo	Problema
E4	Contratada	Organização é mais importante que o skill técnico do desenvolvedor. PO / SM devem saber o que passar para o desenvolvedor	Escopo	Problema
E4	Contratada	PO pode passar uma diretriz que não é o que o usuário final quer	Escopo	Problema
E4	Contratada	MVP: perguntar ao usuário o que ele precisa para hoje (exemplo de resposta: preciso me locomover). Não perguntar ao usuário o que ele quer (arrisca ele dizer que quer uma BMW)	Escopo	Problema
E4	Contratada	Papel do QA permite a identificação de uma necessidade de negócio que o PO não identificou	Qualidade	Aspecto positivo
E4	Contratada	Prototipagem junto com a área de negócio (usuários finais) e refinamento ao longo do processo	Qualidade	Aspecto positivo
E4	Contratada	PO / SM são os papéis que asseguram a qualidade do software	Qualidade	Problema
E4	Contratada	Homologação com o maior número de usuários possível. Muitas vezes a área que irá receber o software não tem um bom relacionamento com a demandante dos ajustes no sistema. PO começa a ter papel fundamental no sentido de permitir que o sistema seja utilizado.	Qualidade	Problema

Entrevistado	Perspectiva	Assertiva	Categoria	Classificação
E5	Contratada	Problemas de ambiente. Causa atrasos de dias. DevOps seria uma alternativa para os problemas de arquitetura. Corrige bug, sobe para produção, mas na semana seguinte o bug ressurge por problema de versionamento. Time acaba atuando em retrabalho, mesmo quando o item já foi resolvido	Engenharia de software	Problema
E5	Contratada	Problemas de infraestrutura (VPN) para o trabalho da equipe	Engenharia de software	Problema
E5	Contratada	Interdependência entre frentes (ex. carrinho de compra): vários projetos entrando simultaneamente. Como a gestão de software (versionamento) deve ser acomodada pelo método de desenvolvimento de software distribuído?	Engenharia de software	Problema
E5	Contratada	A identificação do escopo do projeto tem sido elaborada pelo PO com sucesso por meio das reuniões previstas no modelo ágil utilizado (cerimônias do Scrum como *grooming* e refinamento).	Escopo	Aspecto positivo
E5	Contratada	PO representa o negócio e pode priorizar. Algumas determinações vêm pela área de negócio e isso afeta o processo, independentemente da etapa do ciclo de vida do projeto (*planning* ou homologação de *iterações*).	Escopo	Problema
E5	Contratada	Metas e alta gestão ajudam na organização das atividades	Organizacional	Problema
E5	Contratada	Comportamento das pessoas: as atividades são derivadas da pessoalidade e do interesse das pessoas. A adesão ao método e desempenho da equipe estão associadas aos benefícios esperados.	Organizacional	Problema
E5	Contratada	Scrum é bem flexível, permite acomodar as necessidades do processo de desenvolvimento de sistemas. Aderente a mudanças, impactos são vistos ao longo do processo, não na entrega final.	Planejamento	Aspecto positivo
E5	Contratada	Mudanças de prioridade, Iteração sem fechar o planejamento, direcionando a equipe para resolução dos bugs. Nova priorização das atividades por solicitação do cliente	Planejamento	Problema
E5	Contratada	Deixar o negócio mais próximo pode trazer problemas como o aumento da exigência. Itens no backlog é responsabilidade do PO. Cliente atuando próxima pode afetar o desempenho como o dono do time e isso afetar o desempenho da equipe.	Planejamento	Problema
E6	Contratada	Melhor planejamento para a entrega quando há concorrência de códigos-fonte entre projetos (*squads* ou artefatos). Em alguns casos, a subida em produção ocorre sem alguma outra funcionalidade que já havia sido implantada por outro projeto.	Engenharia de software	Problema
E6	Contratada	Durante a retrospectiva são identificadas como as atividades foram feitas, não quais atividades deveriam ser feitas. Nesta etapa, denominada como "*check*" são validados os aspectos que envolvem ferramenta, processos e pessoas, apontando itens que demandam ações (ex. dificuldades de acessos, perfil dos recursos do time, atividades novas não planejadas, processos de clientes que são modificados sem prévio alinhamento).	Organizacional	Problema

Entrevistado	Perspectiva	Assertiva	Categoria	Classificação
E6	Contratada	Houve uma antecipação no desenvolvimento de algumas funcionalidades, porém a contratante optou por não implantar	Organizacional	Problema
E6	Contratada	Demandas são identificadas pelo PO e pelos gestores do projeto da TI do cliente *Sprint backlog* *Grooming* *Inception* *Solution Design* *Sprint (Planning, timebox,* iteração de 2 semanas) Retrospectiva (volta para o *sprint backlog*)	Planejamento	Aspecto positivo
E6	Contratada	O projeto usa as estórias de usuário para documentar as demandas, porém no geral as histórias no projeto possuem visão em alto nível e sem regras de negócio.	Planejamento	Problema
E6	Contratada	Nível de detalhe do backlog para o início do desenvolvimento. Ao longo da iteração o time de desenvolvimento deve montar o entendimento com a profundidade necessária. PO não consegue atuar previamente nesta atividade.	Planejamento	Problema
E6	Contratada	Planejamento de 2 semanas (curtíssimo prazo) ocorre em função da falta de visão de negócios, metas, etc. pelo PO. TI normalmente é reativa, não sabe o que de fato o negócio precisa.	Planejamento	Problema
E6	Contratada	Acompanhamento ineficaz (não tem modelo para acompanhamento)	Teste	Problema
E6	Contratada	No método ágil, não há um modelo preditivo que possibilite avaliar o que foi construído de fato com o que foi prometido, visto que o plano é feito ao longo da execução.	Teste	Problema

Fonte: Autor

APÊNDICE D: RELACIONAMENTO PROBLEMA IDENTIFICADO E MODELO ESCOLHIDO

Tabela 27: Relacionamento entre problema e características da integração da abordagem DT e métodos ágeis

Categoria	Perspectiva	Problema	Bosch e Bosch-Sijtsema 2011
Escopo	Contratada	PO pode passar uma diretriz que não é o que o usuário final quer	(ii) envolvimento contínuo do cliente ao longo de todo o processo de desenvolvimento, não só no final
Escopo	Contratada	MVP: perguntar ao usuário o que ele precisa para hoje (exemplo de resposta: preciso me locomover). Não perguntar ao usuário o que ele quer (arrisca ele dizer que quer uma BMW)	(ii) envolvimento contínuo do cliente ao longo de todo o processo de desenvolvimento, não só no final.
Escopo	Contratada	A necessidade vem pré-moldada dos BP's que cuidam da captura das necessidades (integrante do time de TI que atende o negócio). Lacuna entre o que está sendo pedido e o que o cliente realmente quer	(iii) solução e codificação sob imersão dos membros do time, buscando a conversão de uma ideia em um conceito que seja real, que permita sua validação e que explore as complexidades de se implantar no sistema legado.
Escopo	Contratante	Distância entre o que o cliente desejava e o que tecnicamente era possível de ser feito e no curto prazo.	(ii) envolvimento contínuo do cliente ao longo de todo o processo de desenvolvimento, não só no final.
Escopo	Contratante	não houve a participação dos usuários finais na definição dos requisitos do software que futuramente seriam utilizados por vendedores, analistas de crédito, operadores de caixa, estoquistas e pessoal administrativo. Por esta razão, atualmente o sistema está implantado em aproximadamente 10% das lojas com pouca aderência de uso, visto que o sistema antigo ainda não foi continuado.	(ii) envolvimento contínuo do cliente ao longo de todo o processo de desenvolvimento, não só no final.
Escopo	Contratante	sistema com várias novas funcionalidades pode ser inferior a outro software tido como obsoleto por não atender as reais necessidades dos usuários finais.	(iii) solução e codificação sob imersão dos membros do time, buscando a conversão de uma ideia em um conceito que seja real, que permita sua validação e que explore as complexidades de se implantar no sistema legado
Escopo	Contratante	grande número de reclamações associadas ao não uso correto do sistema (30% dos *feedbacks*) e de dúvidas (60%), o que pode indicar um problema de falta de treinamento do sistema, além da não participação dos usuários finais ao longo do processo de desenvolvimento do novo sistema.	(iv) geração de uma versão "alfa", que consiste na capacidade de apresentar ao cliente uma versão estável do software, sendo possível que o cliente conceda o *feedback* sobre aquilo que já foi entregue
Escopo	Contratada	Subjetividade do que é o valor para o negócio	(iii) solução e codificação sob imersão dos membros do time, buscando a conversão de uma ideia em um conceito que seja real, que permita sua validação e que explore as complexidades de se implantar no sistema legado
Escopo	Contratada	Entrega de um produto durante um prazo pré-estabelecido pela iteração pode não trazer valor, de fato, ao negócio	(iii) solução e codificação sob imersão dos membros do time, buscando a conversão de uma ideia em um conceito que seja real, que permita sua validação e que explore as complexidades de se implantar no sistema legado
Escopo	Contratada	Organização é mais importante que o skill técnico do desenvolvedor. PO / SM devem saber o que passar para o desenvolvedor	(i) equipes auto gerenciáveis, sendo organizadas em times menores que buscam moldar a solução partindo das necessidades dos usuários
Escopo	Contratada	PO representa o negócio e pode priorizar. Algumas determinações vêm pela área de negócio e isso afeta o processo, independentemente da etapa do ciclo de vida do projeto (planning ou homologação de iterações).	(ii) envolvimento contínuo do cliente ao longo de todo o processo de desenvolvimento, não só no final
Escopo	Contratante	a atuação área de operações de vendas, demandante e patrocinadora do projeto não favorece a elaboração de um backlog estruturado e que o MVP muitas vezes se torna inviável pela complexidade da funcionalidade do sistema que deveria ser entregue. De acordo com o entrevistado, os representantes da área demandante querem o estado da arte já na primeira entrega.	(iii) solução e codificação sob imersão dos membros do time, buscando a conversão de uma ideia em um conceito que seja real, que permita sua validação e que explore as complexidades de se implantar no sistema legado *coded base*.
Escopo	Contratante	Diversas vezes os usuários do sistema exigem que todas as funcionalidades já existentes (*as is*) devem ser incorporadas ao projeto, sendo que algumas podem ter menor relevância. O entrevistado citou que a área exigiu que o pagamento em cheque fosse uma funcionalidade mandatória, sendo que na visão do PO conceder ao sistema o critério de multicanal teria maior	(iv) geração de uma versão "alfa", que consiste na capacidade de apresentar ao cliente uma versão estável do software, sendo possível que o cliente conceda o *feedback* sobre aquilo que já foi entregue

Categoria	Perspectiva	Problema	Bosch e Bosch-Sijtsema 2011
		valor ao negócio (venda iniciada no site e terminada na loja).	
Planejamento	Contratante	Em alguns casos, houve disputa sobre a responsabilidade dos erros ou itens em não conformidade identificados durante o projeto, sem a indicação de quem deveria arcar com o ônus das horas da correção. Essa disputa normalmente possuía duas alternativas: se a falha ocorreu no desenho ou especificação e deveria ser arcada pelo contratante ou se a falha foi derivada de erro técnico e deveria ser realizado sem pagamento.	(i) equipes auto gerenciáveis, sendo organizadas em times menores que buscam moldar a solução partindo das necessidades dos usuários
Planejamento	Contratante	Representante da contratada também citou o desafio de integrar diferentes plataformas, decorrente de desejos do negócio. Integrar negócio + UX + TI Corporativo + time de desenvolvimento.	(ii) envolvimento contínuo do cliente ao longo de todo o processo de desenvolvimento, não só no final.
Planejamento	Contratada	Mudanças de prioridade, Iteração sem fechar o planejamento, direcionando a equipe para resolução dos bugs. Nova priorização das atividades por solicitação do cliente	(i) equipes auto gerenciáveis, sendo organizadas em times menores que buscam moldar a solução partindo das necessidades dos usuários
Planejamento	Contratada	Deixar o negócio mais próximo pode trazer problemas como o aumento da exigência. Itens no backlog é responsabilidade do PO. Cliente atuando próxima pode afetar o desempenho como o dono do time e isso afetar o desempenho da equipe.	(i) equipes auto gerenciáveis, sendo organizadas em times menores que buscam moldar a solução partindo das necessidades dos usuários
Planejamento	Contratada	Nível de detalhe do backlog para o início do desenvolvimento. Ao longo da iteração o time de desenvolvimento deve montar o entendimento com a profundidade necessária. PO não consegue atuar previamente nesta atividade.	(iii) solução e codificação sob imersão dos membros do time, buscando a conversão de uma ideia em um conceito que seja real, que permita sua validação e que explore as complexidades de se implantar no sistema legado.
Planejamento	Contratada	O projeto usa as estórias de usuário para documentar as demandas, porém no geral as histórias no projeto possuem visão em alto nível e sem regras de negócio.	(iii) solução e codificação sob imersão dos membros do time, buscando a conversão de uma ideia em um conceito que seja real, que permita sua validação e que explore as complexidades de se implantar no sistema legado.
Planejamento	Contratada	Planejamento de 2 semanas (curtíssimo prazo) ocorre em função da falta de visão de negócios, metas, etc pelos PO's. TI normalmente é reativa, não sabe o que de fato o negócio precisa.	(ii) envolvimento contínuo do cliente ao longo de todo o processo de desenvolvimento, não só no final.
Planejamento	Contratada	Estórias estão hoje incompletas.	(ii) envolvimento contínuo do cliente ao longo de todo o processo de desenvolvimento, não só no final.
Qualidade	Contratada	Homologação com o maior número de usuários possível. Muitas vezes a área que irá receber o software não tem um bom relacionamento com a demandante dos ajustes no sistema. PO começa a ter papel fundamental no sentido de permitir que o sistema seja utilizado.	(iv) geração de uma versão "alfa", que consiste na capacidade de apresentar ao cliente uma versão estável do software, sendo possível que o cliente conceda o *feedback* sobre aquilo que já foi entregue.
Qualidade	Contratada	PO / SM são os papéis que asseguram a qualidade do software	(i) equipes auto gerenciáveis, sendo organizadas em times menores que buscam moldar a solução partindo das necessidades dos usuários
Teste	Contratada	No método ágil, não há um modelo preditivo que possibilite avaliar o que foi construído de fato com o que foi prometido, visto que o plano é feito ao longo da execução.	(iv) geração de uma versão "alfa", que consiste na capacidade de apresentar ao cliente uma versão estável do software, sendo possível que o cliente conceda o *feedback* sobre aquilo que já foi entregue.
Teste	Contratada	Acompanhamento ineficaz (não tem modelo para acompanhamento)	(iv) geração de uma versão "alfa", que consiste na capacidade de apresentar ao cliente uma versão estável do software, sendo possível que o cliente conceda o *feedback* sobre aquilo que já foi entregue.

Fonte: Autor

Tabela 28: Relacionamento entre problema e artigos selecionados na literatura

ID	Problema	Lucena et al., 2016	Paula & Araujo, 2016	Bosch & Bosch-Sijtsema, 2011	Fischer & Senft, 2016	Ardito et al., 2017	Sebok et al., 2017	Hussain et al., 2008
1	PO pode passar uma diretriz que não é o que o usuário final quer	Sim	Sim	Sim	Sim	Sim	Sim	Sim
2	MVP: perguntar ao usuário o que ele precisa para hoje (exemplo de resposta: preciso me locomover). Não perguntar ao usuário o que ele quer (arrisca ele dizer que quer uma BMW)	Sim	Sim	Sim	Sim	Sim	Sim	Sim
3	A necessidade vem pré-moldada dos Business Partners (BP's) que cuidam da captura das necessidades (integrante do time de TI que atende o negócio). Lacuna entre o que está sendo pedido e o que o cliente realmente quer	Sim	Sim	Sim	Sim	Sim	Sim	Sim
4	Distância entre o que o cliente desejava e o que tecnicamente era possível de ser feito e no curto prazo.	Sim	Sim	Sim	Sim	Sim	Sim	Sim
5	não houve a participação dos usuários finais na definição dos requisitos do software que futuramente seriam utilizados por vendedores, analistas de crédito, operadores de caixa, estoquistas e pessoal administrativo. Por esta razão, atualmente o sistema está implantado em aproximadamente 10% das lojas com pouca aderência de uso, visto que o sistema antigo ainda não foi continuado.	Sim	Sim	Sim	Sim	Sim	Sim	Sim
6	sistema com várias novas funcionalidades pode ser inferior a outro software tido como obsoleto por não atender as reais necessidades dos usuários finais.	Não	Sim	Sim	Sim	Não	Não	Não
7	grande número de reclamações associadas ao não uso correto do sistema (30% dos *feedbacks*) e de dúvidas (60%), o que pode indicar um problema de falta de treinamento do sistema, além da não participação dos usuários finais ao longo do processo de desenvolvimento do novo sistema.	Sim	Não	Sim	Sim	Sim	Não	Sim
8	Subjetividade do que é o valor para o negócio	Não	Não	Sim	Não	Sim	Não	Não
9	Entrega de um produto durante um prazo pré-estabelecido pela sprint pode não trazer valor, de fato, ao negócio	Não	Não	Sim	Não	Sim	Não	Sim
10	Organização é mais importante que o skill técnico do desenvolvedor. PO / SM devem saber o que passar para o desenvolvedor	Não	Não	Sim	Não	Não	Não	Não
11	PO representa o negócio e pode priorizar. Algumas determinações vem pela área de negócio e isso afeta o processo, independentemente da etapa do ciclo de vida do projeto (planning ou homologação de sprints).	Não	Sim	Sim	Não	Não	Sim	Sim
12	a atuação área de operações de vendas, demandante e patrocinadora do projeto não favorece a elaboração de um backlog estruturado e que o minimum viable product (MVP, menor produto viável) muitas vezes se torna inviável pela complexidade da funcionalidade do sistema que deveria ser entregue. De acordo com o entrevistado, os representantes da área demandante querem o estado da arte já na primeira entrega.	Não	Não	Sim	Não	Não	Sim	Sim

ID	Problema	Lucena et al., 2016	Paula & Araujo, 2016	Bosch & Bosch-Sijtsema, 2011	Fischer & Senft, 2016	Ardito et al., 2017	Sebok et al., 2017	Hussain et al., 2008
13	Diversas vezes os usuários do sistema exigem que todas as funcionalidades já existentes (as is) devem ser incorporadas ao projeto, sendo que algumas podem ter menor relevância. O entrevistado citou que a área exigiu que o pagamento em cheque fosse uma funcionalidade mandatória, sendo que na visão do (PO) conceder ao sistema o critério de multicanal teria maior valor ao negócio (venda iniciada no site e terminada na loja).	Não	Não	Sim	Não	Não	Não	Sim
14	Em alguns casos, houve disputa sobre a responsabilidade dos erros ou itens em não conformidade identificados durante o projeto, sem a indicação de quem deveria arcar com o ônus das horas da correção. Essa disputa normalmente possuía duas alternativas: se a falha ocorreu no desenho ou especificação e deveria ser arcada pelo contratante ou se a falha foi derivada de erro técnico e deveria ser realizado sem pagamento.	Não	Não	Sim	Não	Não	Não	Não
15	Representante da contratada também citou o desafio de integrar diferentes plataformas, decorrente de desejos do negócio. Integrar negócio + UX + TI Corporativo + time de desenvolvimento.	Não	Sim	Sim	Não	Não	Não	Não
16	Mudanças de prioridade, Sprint sem fechar o planejamento, direcionando a equipe para resolução dos bugs. Novas priorizações das atividades por solicitação do cliente	Sim	Não	Sim	Não	Sim	Não	Sim
17	Deixar o negócio mais próximo pode trazer problemas como o aumento da exigência. Itens no backlog é responsabilidade do PO. Cliente atuando próxima pode afetar o desempenho como o dono do time e isso afetar o desempenho da equipe.	Não	Não	Sim	Não	Sim	Não	Sim
18	Nível de detalhe do backlog para o início do desenvolvimento. Ao longo da sprint o time de desenvolvimento deve montar o entendimento com a profundidade necessária. PO não consegue atuar previamente nesta atividade.	Não	Sim	Sim	Não	Sim	Sim	Sim
19	O projeto usa as estórias de usuário para documentar as demandas, porém no geral as histórias no projeto possuem visão em alto nível e sem regras de negócio.	Sim	Sim	Sim	Não	Sim	Sim	Sim
20	Planejamento de 2 semanas (curtíssimo prazo) ocorre em função da falta de visão de negócios, metas, etc pelos PO's. TI normalmente é reativa, não sabe o que de fato o negócio precisa.	Sim	Não	Sim	Não	Sim	Não	Não
21	Estórias estão hoje incompletas.	Não	Sim	Sim	Não	Sim	Sim	Sim
22	Homologação com o maior número de usuários possível. Muitas vezes a área que irá receber o software não tem um bom relacionamento com a demandante dos ajustes no sistema. PO começa a ter papel fundamental no sentido de permitir que o sistema seja utilizado.	Não	Sim	Sim	Não	Não	Não	Não
23	PO / SM são os papéis que asseguram a qualidade do software	Não	Não	Sim	Não	Não	Sim	Sim
24	No método ágil, não há um modelo preditivo que possibilite avaliar o que foi construído de fato com o que foi prometido, visto que o plano é feito ao longo da execução.	Não	Não	Sim	Não	Sim	Não	Não
25	Acompanhamento ineficaz (não tem modelo para acompanhamento)	Não	Sim	Sim	Não	Sim	Não	Sim

APÊNDICE E: TERMO DE CONSENTIMENTO E PARTICIPAÇÃO EM PESQUISA

Título da Pesquisa: Aplicação do *Design Thinking* integrado com métodos ágeis na gestão de projetos de *software*

Pesquisadores: Rosária de Fátima Segger Macri Russo; Julio Cesar Pereira

1.Natureza da pesquisa: Você está sendo convidado a participar de uma pesquisa que objetiva a definição de um método para gestão de projetos de desenvolvimento de *software*.

2.Participantes selecionados: Pessoas com conhecimento em projetos de desenvolvimento de *software*.

3.Envolvimento na pesquisa: Quando você participar deste estudo, você permitirá que o pesquisador Julio Cesar Pereira use os dados coletados durante a reunião de *focus group* para esta pesquisa. Você está livre para recusar sua participação a qualquer momento, antes ou depois da ocorrência deste *focus group*. Quando você desejar, você pode solicitar mais informações sobre esta pesquisa pelos telefones dos pesquisadores.

4.Sobre a entrevista: A entrevista será conduzida através de uma reunião de *focus group*.

5.Riscos e desconfortos: A participação nesta pesquisa não traz complicações legais.

6.Confidencialidade: Todas as informações coletadas neste estudo são estritamente confidenciais. Somente o pesquisador e o orientador possuem acesso a esses dados.

7.Benefícios: Com a participação deste estudo você não terá benefícios diretos, porém, nós esperamos que este estudo possa promover informações importantes para a gestão ágil de projetos no desenvolvimento de *softwares*.

8.Pagamento: Você não terá nenhuma despesa em participar desta pesquisa, e nada será pago por sua participação.

Após estas clarificações, requisitamos seu livre acordo para participar desta pesquisa. Em caso positivo, por favor, preencha os itens a seguir. Não assine este termo se você ainda possui dúvidas sobre a pesquisa.

Declaração de Informação e Livre Consentimento

De acordo com os itens apresentados, eu, livre e informado, manifesto meu consentimento em participar desta pesquisa. Eu declaro que recebi uma cópia desta declaração, e autorizo a pesquisa e a disseminação dos dados obtidos neste estudo.

_____
Nome do participante

_____
Assinatura do participante

_____
Assinatura do pesquisador

_____
Assinatura do orientador

APÊNDICE F: QUESTÕES LEVANTADAS NO PRIMEIRO GRUPO FOCAL (EXPLORATÓRIO)

Ao final da apresentação das etapas e a coleta de *feedback* junto aos participantes do grupo focal exploratório, o facilitador aplicou o questionário com as três perguntas a seguir para os oito representantes presentes:

1. Este método é possível de ser aplicado nesta organização?
2. Existe alguma etapa que não deveria ser utilizada neste método?
3. É possível que as etapas sejam ajustadas dependendo da fase em que o projeto está?

APÊNDICE G: VALIDAÇÃO DOS CRITÉRIOS NO SEGUNDO GRUPO FOCAL

Tabela 29: Critérios de validação do artefato (Gill e Hevner, 2013)

ID	Questão	Característica a ser validada
1	Este método é possível de ser aplicado nesta organização?	- Aplicável às situações reais
2	Tal método é possível de ser utilizado em projetos já em andamento?	- Expectativa de vida
3	Ao ser aplicado nesta organização, no que este método poderia ajudar? Este método pode solucionar os problemas desta organização?	- Aderência à resolução do problema - eficiência
4	Este método pode ser facilmente utilizado? Estão claras as atividades que devem ser feitas em cada etapa?	- Fácil utilização
5	É possível que as etapas do método sejam ajustadas dependendo da fase em que o projeto está?	- Adaptável
6	Este método pode ser aplicado em outros mercados, diferentes do segmento desta organização, em projetos de desenvolvimento de software de qualquer tecnologia de programação? É aplicado a outros tipos de clientes?	- Flexibilidade
7	Quanto ao método, existe alguma etapa que não seja possível identificar seu retorno?	- Verificação - Reutilização
8	Este método pode ser aplicado em outros projetos de desenvolvimento de software, com diferentes tecnologias?	- Flexibilidade
9	Com relação ao método, existe alguma etapa que não deve ser feita? Existe alguma etapa que deveria ser dividida ou que esteja faltando e que deve ser adicionada?	- Evolução
10	Este método está simples e apresenta clareza suficiente para ser utilizado por um Product Owner recém incorporado à organização?	- Simplicidade - clareza
11	Este método possui aspectos inovadores em relação ao uso dos métodos ágeis convencionais?	- Inovação
12	De uma maneira geral, o que acharam do método?	- Interesse - elegância

Fonte: Autor

APÊNDICE H: MÉTODO GERADO: *AGILE DESIGN*

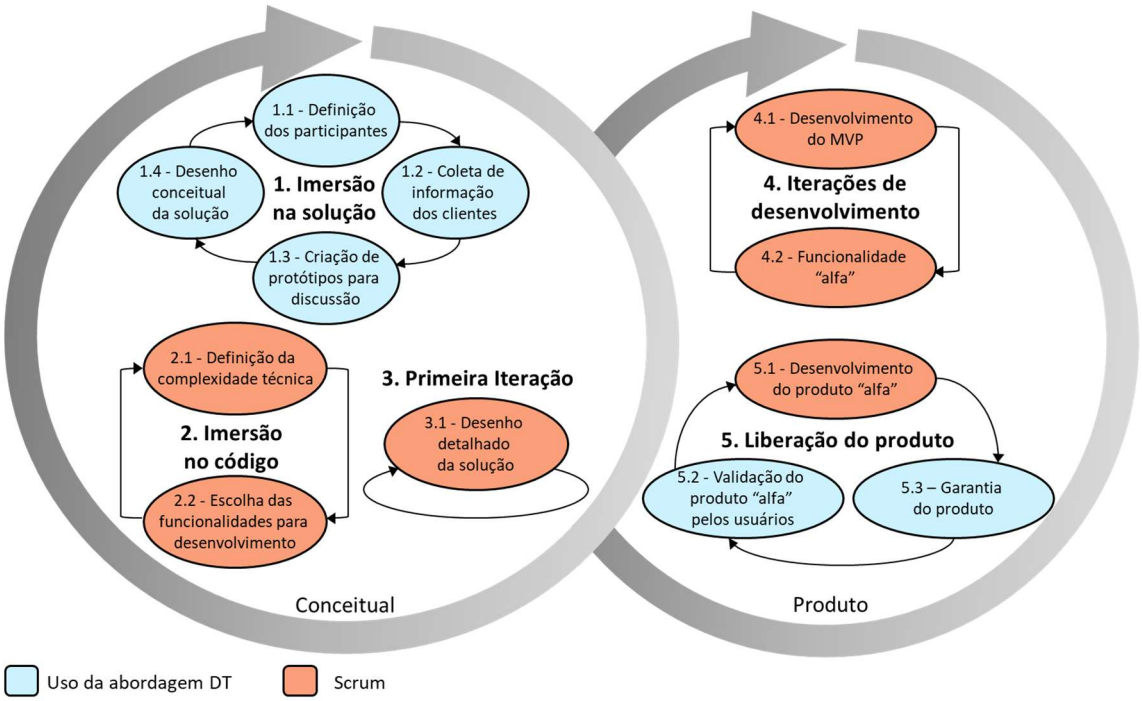

Figura 17. Esquema do *Agile Design* – integração da Abordagem DT e métodos ágeis
Fonte: Autor

Etapa 1 - Imersão na solução

O objetivo da Etapa 1 - Imersão na solução é realizar a identificação dos clientes e usuários do *software*, bem como entender as "dores" e necessidades e atribuir o peso ou importância de cada uma delas, gerando hipóteses para a solução. O resultado final desta etapa é o entendimento das necessidades do cliente e usuário do *software*. Para que isso ocorra, as informações são capturadas a partir das pessoas envolvidas no processo buscando desenhar a solução por meio de rascunhos e complementados por comentários pelos próprios usuários e clientes do *software*. A Figura 18 traz o resumo dos processos que compõem a Etapa 1, incluindo objetivos, facilitadores, participantes e ferramentas de cada processo.

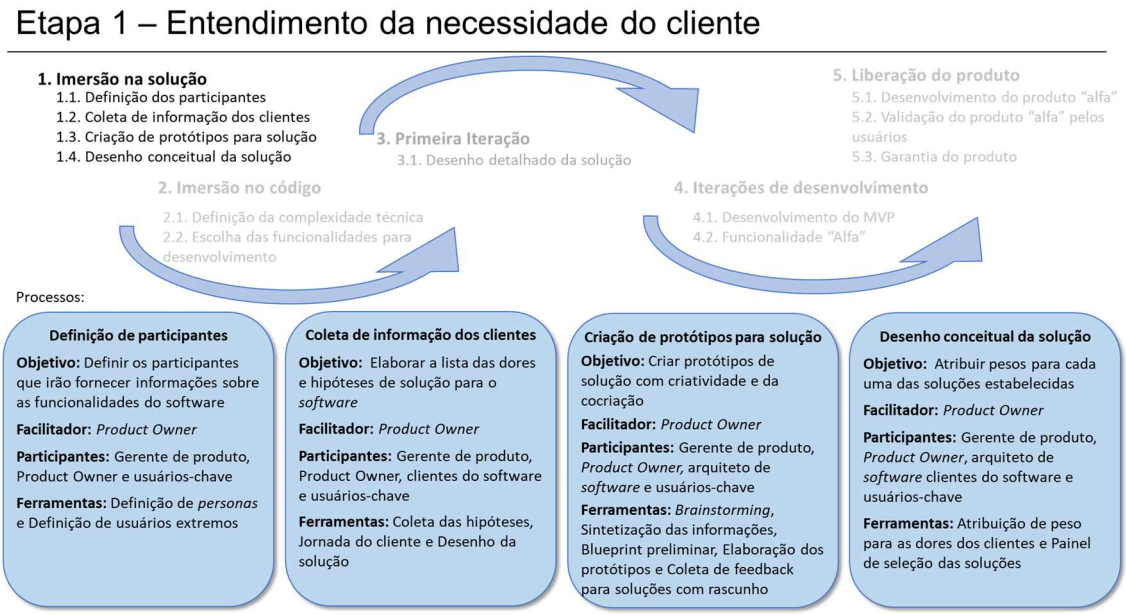

Figura 18. Processos e ferramentas da Etapa 1: Imersão na solução

Fonte: Autor

Um primeiro esboço é criado e apresentado para que os usuários e clientes presentes concedem *feedback* sobre o escopo preliminar da solução dos problemas apontados. O *feedback* dos usuários no estágio inicial assegura que o foco do time de desenvolvimento seja dado aos aspectos corretos e importantes da solução. Durante todo o dia reservado para estas atividades, o time realiza várias iterações recebendo informações e *feedbacks* dos usuários e apresentam um painel com os conceitos mais importantes, sendo que normalmente 50% dos componentes passam para a próxima etapa. A etapa 1 é subdividida em quatro processos, como pode ser visto na Tabela 30.

Tabela 30: Processos da etapa 1. Imersão na solução

Processo	***Descrição***
1.1	Definição dos participantes
1.2	Coleta de informação dos clientes
1.3	Criação de protótipos para discussão
1.4	Desenho conceitual da solução

Fonte: Elaborado pelo autor

Cada um dos processos acima será detalhado nas sessões a seguir, incluindo objetivos, responsável (facilitador), participantes, ferramentas e técnicas.

1.1. Definição dos participantes

O objetivo do processo 1.1. Definição dos participantes é definir os participantes que irão fornecer informações sobre as funcionalidades do *software*. A Tabela 31 apresenta o facilitador, responsável pela execução do processo, e os atores participantes.

Tabela 31: Facilitador e participantes do processo 1.1. Definição dos participantes

Papel	Atores
Facilitador	*Product Owner*
Participantes	Gerente de produto, Product Owner e usuários-chave

Fonte: Elaborado pelo autor

As novas solicitações de funcionalidades do *software* podem surgir de duas origens distintas: demanda estratégica ou demanda operacional (Design Council, 2007). Novos produtos ou serviços normalmente são expostos a maior quantidade de riscos e exigirão a participação de maior número de pessoas em relação à evolução de *softwares* já existentes.

Como pré-requisito deverá haver o direcionamento das ações necessárias, desdobradas da estratégia da organização, bem como orçamento financeiro para a alocação do time do projeto para a realização da imersão na solução. As ferramentas e técnicas deste processo são: definição de *personas* e definição de usuários extremos. Além dessas ferramentas e técnicas, recomenda-se a busca de informações históricas da organização contendo dados sobre as áreas e pessoas afetadas por projetos anteriores, tais como ata de reuniões, documentos do projetos e lições aprendidas.

O objetivo desse processo é obter a relação dos usuários e clientes do *software* que farão parte das etapas subsequentes, incluindo (i) a definição das informações necessárias para o desenvolvimento do *software* em si; (ii) os critérios de validação da solução, e; (iii) o aceite formal de entrega do produto.

1.1.1. Definição de *personas*

A técnica de definição de *personas* tem como objetivo realizar a identificação de usuários específicos e elaboração do mapa de empatia, considerando os aspectos que compõe o objetivo, desejo e limitação de cada usuário. Devem ser considerados seu nome, informações relativas às questões demográficas (tais como idade e região), necessidade, desejos e tarefas. Além disso considerar o que escuta, sente e pensa, vê, diz e faz, quais dores e ganhos esperados por cada indivíduo (Osterwalder & Pigneur, 2010).

A definição de *personas* deve permitir a identificação do comportamento, necessidades e preocupações de cada indivíduo que serão elementos para a definição da solução. A importância de definição das personas habilita que os membros do time tenham

consciência dos requisitos dos usuários (Nakao, Moriguchi, & Noda, 2014). Todas as *personas* envolvidas no processo, sejam usuários do *software* (ex. vendedor que utiliza o *software* em um processo de venda) ou pessoas que serão beneficiadas pelo processo suportado pelo sistema (ex. clientes que estão sendo atendidos pelo vendedor que opera o *software*).

1.1.2. Definição de usuários extremos

Promover a identificação dos usuários que estejam nas extremidades, sendo representados por pessoas que possuam características, necessidades e pensamentos divergentes, considerando também que o uso do *software* pode ser feito de forma bem diferente entre eles (T. Brown & Katz, 2011). Deste modo, deve-se considerar pessoas que sejam usuários do *software*, clientes que terão o processo de compra suportado pelo *software* a ser desenvolvido e representantes das áreas de BackOffice, tais como logística, área financeira e contabilidade. As extremidades de comportamento são benéficas para que o mapeamento de novas funcionalidades de um *software* seja mais eficaz, auxiliando na definição dos limites que deverão ser atendidos pela oferta de um novo produto (Pinheiro & Alt, 2011). Entende-se que, ao invés de selecionar dez diferentes usuários, capturar dois deles que estejam nas extremidades seja suficiente para entender que ao atender a necessidade de ambos, a necessidade dos outros oito usuários será igualmente atendida.

1.2. Coleta de informação dos clientes

O objetivo do processo 1.2. Coleta de informação dos clientes é elaborar a lista das dores e hipóteses de soluções para o *software* com base no depoimento dos clientes e usuários, selecionados no processo anterior, sobre o que é necessário incorporar ao *software*. A Tabela 32 apresenta o facilitador, responsável pela execução do processo, e os atores participantes.

Tabela 32: Facilitador e participantes do processo 1.2. Coleta de informação dos clientes

Papel	Atores
Facilitador	*Product Owner*
Participantes	Gerente de produto, *Product Owner*, clientes do *software* e usuários-chave

Fonte: Elaborado pelo autor

O processo visa obter as necessidades do *software*, seja um novo produto ou a melhoria derivada de uma nova funcionalidade identificada como necessária. Dentre as

técnicas utilizadas neste processo estão: (i) coleta das hipóteses; (ii) jornada do cliente, e; (iii) desenho da solução conceitual. Este processo prevê a geração da lista das soluções desejadas pelo cliente para o *software* a ser desenvolvido ou melhorado. Caso a solução ainda não tenha chegado ao nível de maturidade suficiente para desenvolver o protótipo, se faz necessário identificar quais outras pessoas deveriam ser adicionadas na etapa 1.1. Definição dos participantes. Cabe ressaltar que as dores e hipóteses geradas neste processo serão detalhados ao longo dos próximos processos e etapas, e deverão ser utilizados como fontes de informação para a composição dos requisitos do *software*.

1.2.1. Coleta das hipóteses

A coleta das hipóteses pretende realizar a captura das principais "dores" e necessidades de todos os participantes por meio de entrevistas, sessões generativas, cadernos de sensibilização etc., permitindo o mapeamento de todo o contexto de interação com o *software* a ser explorado pelo projeto (Vianna, Vianna, Adler, Brenda, & Russo, 2012). Deste modo, a discussão das hipóteses sobre como o problema poderá ser solucionado são amplamente discutidas e poderão dar origem a solução mais adequada para as "dores" e necessidades identificadas.

1.2.2. Jornada do cliente

O mapeamento da jornada do cliente é uma ferramenta que permite a identificação do comportamento de quem opera ou irá se beneficiar pelo uso do *software*, promovendo a visualização e tornando tangível para o time técnico toda a informação coletada junto aos usuários e clientes (Pinheiro & Alt, 2011). Entender como os usuários irão interagir com o *software* e como os clientes serão impactados pelos processos suportados pelo *software* permitirá um desenho adequado para a solução. O entendimento do comportamento do cliente deve ser obtido por meio das seguintes perguntas (Vidal, 2017):

- O que pensa o cliente ao utilizar um produto com as características apresentadas?
- Qual o *feedback* de cada cliente?
- Como o *software* será utilizado?
- Qual seria o processo utilizado pelo cliente ao usar o *software*?
- Qual a sensação causada em cada cliente ao utilizar o *software*?

- Qual a motivação para o cliente utilizar o *software*?

Cabe ao *Product Owner* conduzir as discussões para que as corretas informações sejam coletadas com todos os envolvidos na solução do *software*.

1.2.3. Desenho da solução

O desenho da solução deve considerar como o usuário pensa e realiza suas escolhas, com foco na definição da jornada do usuário do *software*. Dentre os aspectos do modelo mental do cliente ou usuário estão: (i) a definição de uma tarefa relacionada ao perfil do usuário; (ii) o entendimento das escolhas feitas pelo usuário; (iii) o entendimento dos efeitos de cada tarefa executada pelo usuário, e; (iv) entender a escolha da estratégia adotada pelo usuário (Vidal, 2017). Sendo assim, o desenho da solução deve estar diretamente relacionado à jornada de cada usuário ou cliente envolvido no processo de concepção e projeto do *software*.

1.3.Criação de protótipos para discussão

O objetivo do processo 1.3. Criação de protótipos para discussão é criar protótipos de solução por meio de atividades que promovam a criatividade entre o time do projeto, os usuários e os clientes do *software* com o intuito de elaborar soluções por meio da cocriação. A Tabela 33 apresenta o facilitador, responsável pela execução do processo, e os atores participantes.

Tabela 33: Facilitador e participantes do processo 1.3. Criação de protótipos para discussão

Papel	Atores
Facilitador	*Product Owner*
Participantes	Gerente de produto, *Product Owner*, arquiteto de *software*, clientes do *software* e usuários-chave

Fonte: Elaborado pelo autor

Este processo parte das informações obtidas no processo 1.2. Coleta de informações dos clientes. Dentre as ferramentas e técnicas a serem utilizadas nesse processo estão: (i) *brainstorming*, (ii) sintetização das soluções, (iii) *blueprint* preliminar; (iv) elaboração dos protótipos, e; (v) coleta de *feedback* para as soluções com uso de rascunhos. Todas elas serão detalhadas nas sessões a seguir.

Este processo tem como saída esperada desenvolver descritivo da solução preliminar contendo: objetivos, usuários e clientes impactados, requisitos funcionais, requisitos não-funcionais, características mandatórias da solução. Todas essas informações devem ser

estruturadas possibilitando que o nível de detalhe seja construído de forma incremental nas etapas subsequentes previstas neste método.

1.3.1. Brainstorming

O *brainstorming* é uma ferramenta que permite aos participantes realizarem a identificação do maior número de ideias a serem exploradas para a resolução dos problemas, identificando alternativas que possibilitem a chegar à solução, desde que sejam utilizados alguns preceitos (Vianna et al., 2012):

- Quantidade de ideias é importante para se chegar na qualidade, ou seja, quanto maior o número de ideias maior será a possibilidade de produzir uma solução inovadora e funcional
- Julgamento de ideias deve ser evitado, impulsionando o processo criativo com novas ideias e aprimorando às já existentes
- Ideias ousadas são bem-vindas, visto que opiniões em novas perspectivas podem gerar soluções inovadoras
- Combinação, adaptação, transformação e desmembramento de ideias deve ser encorajada para que todos na equipe possam participar.

O *brainstorming* é indicado quando existe a necessidade de obter um grande volume de ideias. No processo da ideação, o *brainstorming* possibilita uma abordagem rica para gerar ideias em cima de questões relevantes que nasceram durante as fases de imersão e de análise provenientes da abordagem DT (Vianna, Vianna, Adler, Brenda, & Russo, 2012).

1.3.2. Sintetização das soluções

Esta ferramenta auxilia na estruturação das ideias geradas na etapa 1.3.1. *Brainstorming*. Deste modo, é possível expor a lista de cada uma das ideias geradas considerando seus comentários, eventuais desdobramentos e oportunidades de negócio (Vianna et al., 2012). A organização das ideias se faz necessário para que seja possível a estruturação dos dados, de modo a definir um padrão que auxilie na geração de soluções.

De acordo com Vidal (2017) uma das formas de auxiliar na organização das informações coletadas é realizar o preenchimento de um cartão contendo a ideia, o insight obtido, a informação em si e a classificação do conhecimento, subdividido em visível, processual, tácito e aprendido.

1.3.3. *Blueprint* preliminar

Tal ferramenta consiste no processo de definição da solução considerando as atividades necessárias para implementação do *software*. O *blueprint* é indicado para mapear a abordagem e os pontos de contato do cliente (Design Council, 2007). Tal ferramenta é capaz de projetar novos serviços, experiências e jornadas dos usuários, permitindo que a solução final seja detalhada de modo a possibilitar melhorias ao longo de sua elaboração visando aperfeiçoar a solução final (Design Council, 2007) por meio do entendimento das características do *software*.

O melhor entendimento do produto, no caso o *software*, pode ser facilitado por meio da estruturação das informações que permitam melhor alinhamento do conhecimento dos requisitos com as necessidades obtidas junto aos clientes e usuários. Vidal (2017) propõe que o melhor entendimento do produto inclui os seguintes requisitos:

- Visão
- Cliente-alvo
- Necessidades
- Produto
- Objetivos de negócio

1.3.4. Elaboração dos protótipos

Desenvolver os protótipos de solução de modo a possibilitar que os usuários tenham uma percepção visual daquilo que será desenvolvido no futuro. Com base no protótipo é possível avaliar se as “dores” ou necessidades dos usuários serão atendidas, ainda na fase preliminar do projeto. Sendo assim, caso seja identificada que a “dor” ou necessidade do usuário não será solucionada, é possível, ainda na fase de concepção, retomar a discussão para corrigir o curso do projeto de desenvolvimento de *software*.

1.3.5. Coleta de *feedback* para as soluções com rascunho

Identificar e relacionar as características derivadas das necessidades e desejos dos usuários respeitando as questões levantadas pelas diferentes perspectivas de clientes e outras pessoas que utilizam ou que serão afetadas pelo uso do *software*. Para que haja efetividade na captura das ações, o facilitador deve (i) estabelecer uma comunicação eficiente para expor como os problemas serão resolvidos; (ii) assumir a responsabilidade pela solução do problema em si; (iii) ser responsivo, permitindo expor as ideias e quais serão os próximos passos e; (iv)

ter empatia, ou seja, se colocar no lugar do cliente ou usuário do *software* (Pinheiro & Alt, 2011).

1.4.Desenho conceitual da solução

O processo 1.4. Desenho conceitual da solução tem como objetivo desenvolver o modelo conceitual das soluções para o *software*, alinhado com as expectativas das áreas usuárias e clientes obtidos no processo 1.3. Criação de protótipos para discussão. A Tabela 34 apresenta o facilitador, responsável pela execução do processo e os atores participantes.

Tabela 34: Facilitador e participantes do processo 1.4. Desenho conceitual da solução

Papel	Atores
Facilitador	*Product Owner*
Participantes	Gerente de produto, *Product Owner*, arquiteto de *software*, clientes do *software* e usuários-chave

Fonte: Elaborado pelo autor

Para que o processo atenda seu objetivo, é necessário que sejam definidas as características da solução por meio da atribuição de peso para as "dores" ou necessidades dos usuários e clientes do *software* e um painel que possibilite a escolha das soluções. Sendo assim, este processo prevê a geração de uma lista com todas as soluções aprovadas pelos usuários e clientes participantes da etapa de imersão na solução.

As soluções aprovadas normalmente serão àquelas que possuírem maior relevância na opinião dos clientes, bem como às de maior importância para o negócio. A relevância do cliente pode representar a necessidade de atendimento para uma ou mais "dores" principais ou necessidade de inovação do *software*. Cada solução deve conter: (i) o problema a ser solucionado; (ii) a descrição da funcionalidade; (iii) os requisitos principais do *software*, e; (iv) quais serão os benefícios gerados.

1.4.1. Atribuição de peso para as dores dos clientes

Esta técnica permite que a atribuição de pesos para cada uma das soluções estabelecidas a partir do mapeamento das dores dos clientes seja realizada. Este peso pode ser atribuído por meio de uma classificação de pontos derivada da relevância da "dor" ou necessidade apontada pelos usuários e clientes do software. Os pontos são definidos de forma quantitativa, como exemplo: definir a necessidade em uma escala de um até cinco, sendo: 1 – menor impacto e 5 – maior impacto. Dependendo da quantidade de "dores" e necessidades a serem avaliadas, pode-se adotar critérios mais sofisticados.

Massari e Vidal (2018) sugerem algumas técnicas que podem ser utilizadas neste processo:

- Técnica MoSCoW difundida pelo *Dynamic Systems Development Method* (DSDM) que permite classificar os objetivos do cliente e relacionando-os aos objetivos do negócio
- Análise de Kano que tem por objetivo classificar e ordenar os itens de um produto como necessários ou que aumentam a satisfação do cliente
- *Theme screening*, onde são definidos os critérios de avaliação para pontuação e ordenação
- Gravidade, Urgência e Tendência (GUT) que prevê a pontuação e priorização dos valores do negócio
- WSJF (*Weighted Shortest Job First*, ou "o menor trabalho primeiro") que considera a técnica de priorização quer considera o custo do atraso, favorecendo a escolha de itens com maior valor e de menor tempo de construção.

1.4.2. Painel de seleção das soluções

Elaborar o painel contendo a classificação de todas as soluções mapeadas e o respectivo grau de importância e urgência para deliberação dos participantes da sessão. Como um dos exemplos do processo anterior, a realização da classificação das soluções pode ser feita por meio da técnica MoSCoW difundida pelo *Dynamic Systems Development Method* (DSDM), conforme ilustrado na Figura 19. Cada uma das soluções pode ser avaliada sob a perspectiva de relevância para o cliente e sobre a importância para o negócio. Soluções que atendam a ambos os critérios deveriam ser obrigatórias (direcionadas ao quadrante must). Ao passo que demandas com baixa relevância ao cliente e de importância baixa para o negócio seriam posicionadas no quadrante "*won't*", ou seja, que não devem ser feitas neste momento. Para as necessidades que tenham sido classificadas como baixa em algum dos aspectos, para o cliente ou para o negócio, estas serão preteridas em função de demandas que possuam maior relevância.

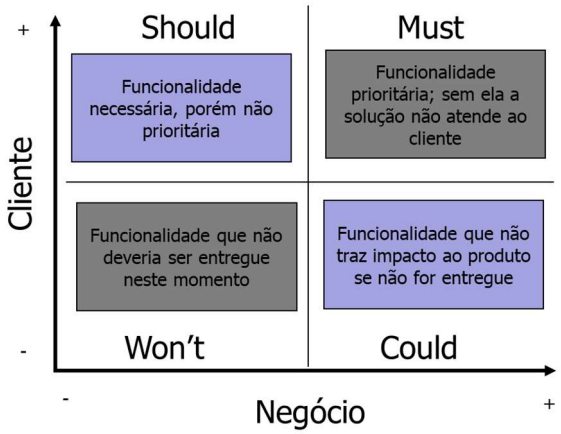

Figura 19. Exemplo de categorização de painel para a seleção das soluções com base na técnica MoSCoW do Dynamic Systems Development Method (DSDM)

Fonte: Massari e Vidal (2018)**,** p. 171.

Em função de possíveis restrições quanto à capacidade de produção de novas funcionalidades no *software*, é preciso que neste processo seja definido o limite de demandas que serão aprovadas para a fase de imersão no código, onde o detalhamento das funcionalidades e complexidade técnica serão realizadas. Dependendo da orientação da organização, seja por produto ou por projeto, existem variações da forma como a organização realiza a definição de sua capacidade para a construção do *software*.

Em virtude de o método auxiliar no processo de gestão do projeto de desenvolvimento do *software*, considera-se que a aprovação preliminar do projeto em si já tenha ocorrido, assim como mencionado nas premissas na seção de introdução deste documento. Portanto, deve-se considerar que o esforço de horas de desenvolvimento estimado inicialmente, dentre outros aspectos, já tenha sido definido e aprovado pelo patrocinador do projeto de desenvolvimento de *software*, embora na etapa 2 o ROI do projeto poderá ser revisado mediante a avaliação de complexidade da solução detalhada na etapa 1 – Imersão na solução.

Etapa 2 - Imersão no código

A Etapa 2 - Imersão no código tem como objetivo o desenvolvimento do esqueleto da implementação da solução com base nos conceitos aceitos pelos clientes e usuários do *software*. Normalmente esta etapa é realizada na mesma semana da etapa anterior, sendo executada pelo time de desenvolvimento e pretende definir a complexidade técnica por meio de um esqueleto de implantação. A Figura 20 traz o resumo dos processos que compõem a Etapa 2, incluindo objetivos, facilitadores, participantes e ferramentas de cada processo.

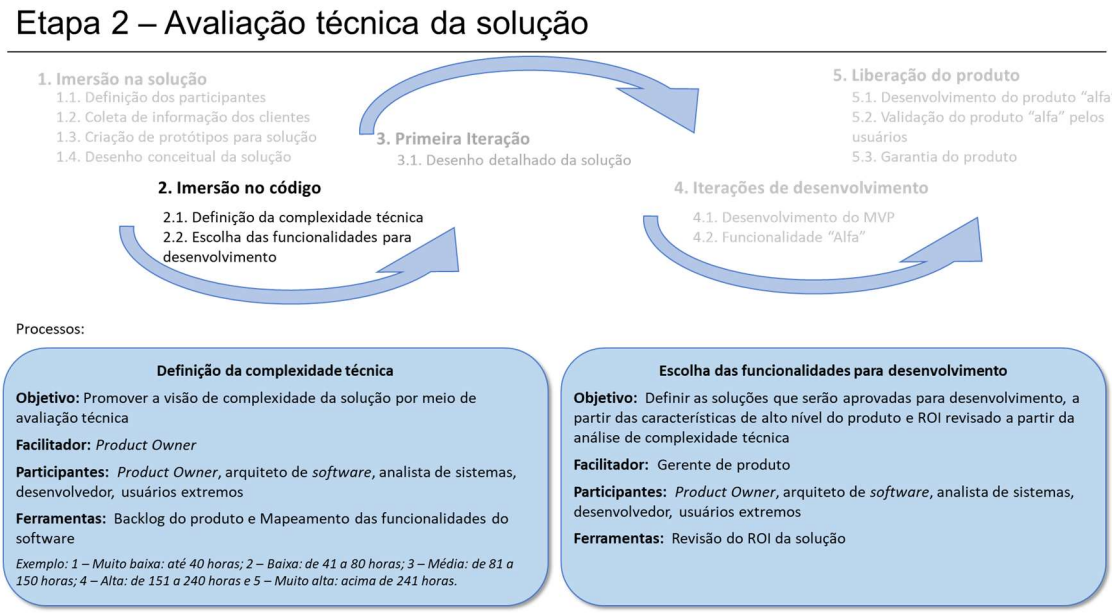

Figura 20. Processos e ferramentas da Etapa 2: Imersão no código

Fonte: Autor.

A Etapa 2 - Imersão de código termina com um painel mostrando a avaliação do valor estimado de cada solução, considerando a estimativa de custo e complexidade de implantação e integração das novas funcionalidades do *software*. Normalmente 50% das funcionalidades avaliadas nesta etapa são selecionadas, considerando que quanto maior é o retorno sobre o investimento (ROI, do inglês, *return on investment*) maior sua probabilidade de aprovação. A Etapa 2 é subdividida em dois processos, como pode ser visto na Tabela 35.

Tabela 35: Processos da etapa 2. Imersão no código

Processo	***Descrição***
2.1	Definição da complexidade técnica
2.2	Escolha das funcionalidades para desenvolvimento

Fonte: Elaborado pelo autor

Cada um dos processos acima será detalhado nas sessões a seguir, incluindo objetivos, responsável (facilitador), participantes, ferramentas e técnicas.

2.1.Definição da complexidade técnica

O processo 2.1. Definição da complexidade técnica tem como objetivo promover a visão de complexidade da solução por meio de avaliação técnica, determinando a classificação por meio da atribuição de pontos para cada pacote de solução avaliado. A Tabela 36 apresenta o facilitador, responsável pela execução do processo, e os atores participantes.

Tabela 36: Facilitador e participantes do processo 2.1. Definição da complexidade técnica

Papel	Atores
Facilitador	*Product Owner*
Participantes	*Product Owner*, arquiteto de *softwares*, analista de sistemas, desenvolvedor, usuários extremos

Fonte: Elaborado pelo autor

Como pré-requisito, o desenho conceitual da solução deve ter sido aprovado no processo 1.4. Desenho conceitual da solução, presente na Etapa 1. Além das características conceituais da solução, neste processo também devem ser obtidas as informações de orçamento preliminar aprovado. Deste modo, recursos técnicos poderão desempenhar as atividades de (i) estimativa da complexidade técnica; (ii) concepção das alternativas de solução, e; (iii) avaliação da capacidade de realização do desenvolvimento, que ocorrerá nas etapas subsequentes deste método.

O processo prevê o uso de ferramentas e técnicas, tais como opinião de especialistas técnicos, análise de riscos, análise da demanda pelo time técnico, definição do esforço e complexidade para a solução. A complexidade deve ser estabelecida com uma escala de horas necessárias para desenvolvimento, sendo que a escala de complexidade e horas pode variar de acordo com a complexidade do sistema e interdependência entre diferentes soluções e tecnologias. Como orientação e exemplo prático, a escala de complexidade da solução pode ter a atribuição de um número de 1 a 5, podendo ser categorizada em:

- 1 – Muito baixa: até 40 horas
- 2 – Baixa: de 41 a 80 horas
- 3 – Média: de 81 a 150 horas
- 4 – Alta: de 151 a 240 horas
- 5 – Muito alta: acima de 241 horas.

Como resultado, a lista das soluções e as respectivas escalas de complexidade deve ser elaborada, considerando: (i) a identificação das soluções; (ii) os objetivos, (iii) a complexidade da solução; (iv) a relevância para o negócio, e; (v) principais riscos. A Tabela 37 ilustra um exemplo de lista de soluções e suas principais características:

Tabela 37: Características da solução e respectiva complexidade

Descrição	Objetivo	Complexidade	Relevância	Principais riscos
Habilitar nova modalidade de pagamento no site de venda de produtos próprios	Aumentar o volume de vendas por meio da flexibilidade de pagamento pelos clientes	Alta	Alta	Concorrência lançar modalidade antes
Habilitar nova modalidade de pagamento para os parceiros	Oferecer diferencial para parceiros que utilizam a plataforma de vendas da	Média	Média	Descontinuação de algum parceiro

Descrição	Objetivo	Complexidade	Relevância	Principais riscos
	empresa			
Relatório da contabilidade	Providenciar a lista dos recebíveis da nova modalidade para receber os valores da nova forma de pagamento	Baixa	Média	Necessidade de um novo *headcount* na área contábil

Fonte: Elaborado pelo autor

2.1.1. *Backlog* do produto

Realizar a construção do *backlog* de um produto que busca satisfazer as necessidades dos clientes e usuários do *software* preliminarmente identificadas. O *backlog* do produto é construído a partir de uma visão e deve permitir que uma lista de necessidades seja priorizada e estimada para determinar o planejamento de como o produto será desenvolvido e entregue ao cliente (Massari & Vidal, 2018).

Neste processo é necessário que as informações relacionadas ao produto a ser entregue possibilitem a clareza sobre a real ideia e entendimento do que será desenvolvido ao longo do projeto. De acordo com Massari e Vidal (2018), a definição das informações necessárias do produto devem responder as seguintes questões:

- Para quais clientes e usuários o *software* será entregue
- Qual o problema que será solucionado pelo *software*
- Qual função do *software* resolverá o problema do cliente ou usuário
- Qual(is) benefício(s) serão gerados
- O que muda em relação ao que é feito hoje
- Característica principal do *software*

2.1.2. Mapeamento das funcionalidades do software

Realizar a decomposição do produto em partes menores, possibilitando a organização das metas e capacidades do produto por meio da decomposição em componentes do *software* (Massari & Vidal, 2018). Deste modo, ao detalhar os componentes, é possível que sejam definidas características do produto e que podem ser utilizadas como fonte de informação para a avaliação técnica das funcionalidades do *software*. A Figura 21 ilustra uma ferramenta possível de utilização neste processo, que consiste na construção da Estrutura Analítica do Produto, ou PBS (*Product Breakdown Structure*), que, a partir do produto, decompõe-se em partes menores gerando uma visão hierárquica (Massari & Vidal, 2018).

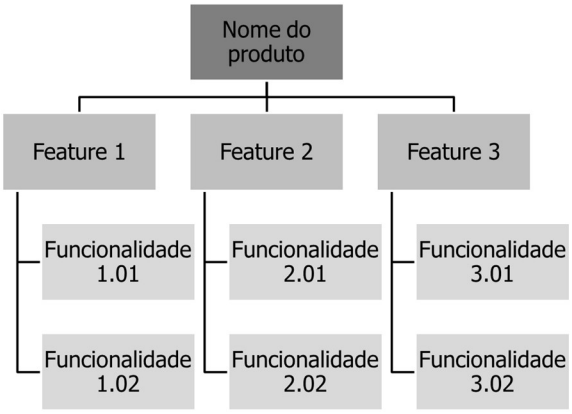

Figura 21. Exemplo de uma Estrutura Analítica do Produto (PBS)

Fonte: Massari e Vidal (2018)**,** p. 163

Ao avaliar a complexidade do produto deve-se identificar todas as possíveis interdependências do mesmo com outros *softwares* ou módulos que poderão ser necessários para o correto funcionamento do produto. Neste processo, portanto, a avaliação técnica da abrangência do que será necessário para prover a solução desejada deve ser feita para que o esforço não seja subestimado.

2.2. Escolha das funcionalidades para desenvolvimento

O processo 2.2. Escolha das funcionalidades para desenvolvimento tem como objetivo definir as soluções que serão aprovadas para desenvolvimento do *software* com base nos pontos pré-aprovados pela organização no item 1.4. Desenho conceitual da solução aprovado. A Tabela 38 apresenta o facilitador, responsável pela execução do processo, e os atores participantes.

Tabela 38: Facilitador e participantes do processo 2.2. Escolha das funcionalidades para desenvolvimento

Papel	Atores
Facilitador	Gerente de produto
Participantes	*Product Owner*, arquiteto de *software*, analista de sistemas, desenvolvedor, usuários extremos

Fonte: Elaborado pelo autor

A capacidade de desenvolvimento de novas funcionalidades do *software* deve ser considerada neste processo. A delimitação desta capacidade pode estar relacionada com os recursos financeiros ou humanos disponíveis para a atuação no grupo de iniciativas aprovadas na Etapa 2 - Imersão no código. Deste modo, é possível realizar a identificação das funcionalidades que podem prosseguir para a próxima etapa, observando a delimitação do que poderá ser executado pelo time de desenvolvimento disponível ou que será contratado.

Como base para a escolha, este processo prevê que o *backlog* do produto e a lista detalhada de funcionalidades tenham sido gerados anteriormente. O *backlog* do produto e a lista de funcionalidades são saídas do processo 2.1. Definição da complexidade técnica. A saída do processo 2.2. Escolha das funcionalidades para desenvolvimento considera a aprovação das funcionalidades a partir das informações detalhadas e revisadas após o detalhamento técnico. Dentre as informações necessárias para definir quais funcionalidades do *software* serão escolhidas estão:

- Relação das características do produto (alto nível)
- Detalhe conceitual da funcionalidade do *software*
- ROI revisado com base na estimativa técnica elaborada pelo time
- Alinhamento com as expectativas dos usuários e do patrocinador.

A partir deste processo, será dado início à elaboração do desenho detalhado das funcionalidades que forem escolhidas, conforme previsto na Etapa 3 - Primeira iteração.

2.2.1. Revisão do ROI da solução

A técnica de revisão do ROI da solução pretende reavaliar os benefícios esperados com base nas características necessárias do *software* relatadas pelos consumidores e a necessidade de investimento derivado da complexidade técnica determinada no processo anterior. Para que esta etapa seja realizada, é necessário que todas as funcionalidades do *software* previamente identificadas contenham a descrição, o objetivo e os benefícios esperados, sejam eles qualitativos ou quantitativos.

Para que as iniciativas sejam selecionadas neste processo, é necessário que o retorno esperado de cada funcionalidade tenha sido revisado conforme previsto no processo anterior. Esta revisão deriva-se da avaliação da complexidade técnica, visto que o ROI está diretamente relacionado ao contexto do serviço ou produto que será entregue ao longo do projeto, se aprovado. Ou seja, o impacto que a funcionalidade poderá gerar após sua implantação e o resultado esperado ou desejado que se previa antes do projeto ter sido iniciado (Pinheiro & Alt, 2011) estão diretamente relacionados aos esforços necessários para sua implantação. Nenhuma iniciativa que tenha um esforço superior ao seu benefício, seja ele tangível ou intangível, deverá prosseguir para as próximas etapas.

De posse das informações coletadas junto aos clientes e usuários do *software*, a definição da complexidade e do benefício esperado, é montado o painel para a escolha das

funcionalidades que seguirão para a etapa de desenho detalhado. A Tabela 39 contém exemplos de produtos apontados como necessários pelos clientes, funcionalidades desejadas em alto nível, complexidade derivada da análise técnica, e o benefício esperado que justifique o desenvolvimento das novas funcionalidades no *software*. A última coluna da tabela é reservada para que seja atribuído o critério de demanda aprovada ou não para prosseguir para a Etapa 3 - Primeira Iteração.

Tabela 39: Painel com as funcionalidades aprovadas na etapa de imersão no código

Produto Id	Funcionalidade	Complexidade	Benefício esperado	Risco	Aprovado
1	Habilitar nova modalidade de pagamento no site de venda de produtos próprios	Alta	Aumento na receita em 10%	Concorrência lançar modalidade antes	Sim
2	Habilitar nova modalidade de pagamento para os parceiros	Média	Aumento na receita em 0,5%	Descontinuação de algum parceiro	Não
3	Relatório da contabilidade	Baixa	Redução do pagamento de dividendos em 2 dias	Necessidade de um novo *headcount* na área	Não

Fonte: Elaborado pelo autor

Etapa 3 - Primeira iteração

A Etapa 3 – Primeira iteração visa elaborar o desenho que confirma a hipótese sobre o valor e usabilidade do conceito que dará origem a proposta de solução a ser desenvolvida no *software*. A primeira iteração, que precede o início do desenvolvimento, possui duração de duas ou três semanas. O *feedback* do usuário deve ser obtido por interações diretas que podem ser realizadas por meio de ligações e compartilhamento de tela com o usuário para mostrar o que será implementado no *software*. A Figura 22 traz o resumo do processo da Etapa 3, incluindo objetivos, facilitadores, participantes e ferramentas.

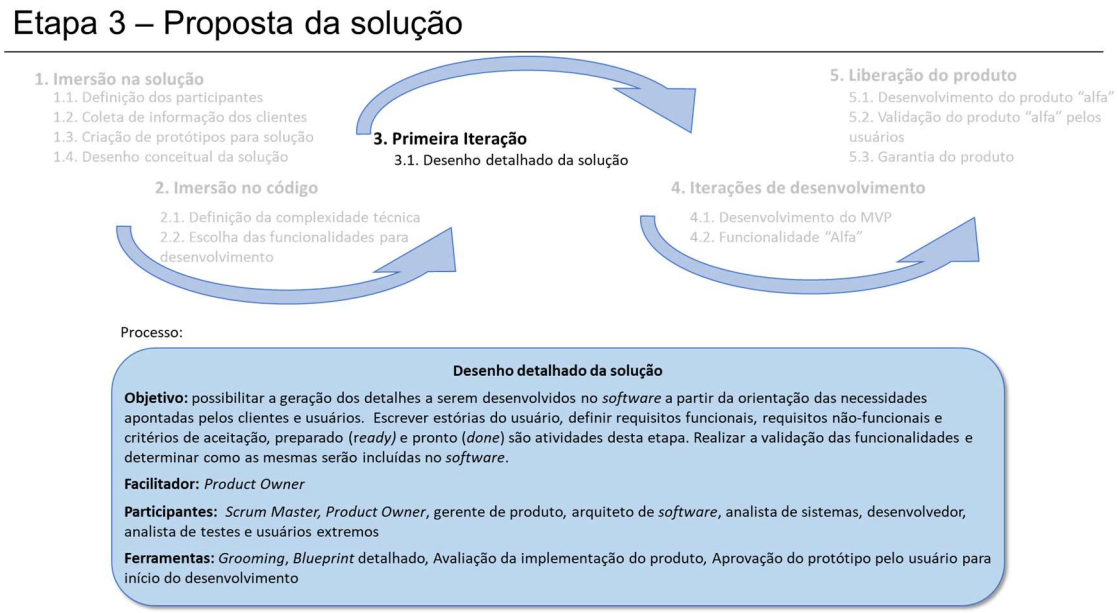

Figura 22. Processo e ferramentas da Etapa 3: Primeira iteração

Fonte: Autor

Diante da necessidade de obter o detalhamento da solução e do que será necessário incorporar ao *software*, é necessário que sejam adotadas ferramentas que permitam a estruturação detalhada, incluindo o *grooming* e *blueprint* detalhado. A estruturação de como as informações serão detalhadas é um passo importante, visto que se trata da definição do que será exigido do time de desenvolvimento e servirá como base para que o cliente possa dar como validado o resultado gerado pelo desenvolvimento (Massari & Vidal, 2018).

Ao final desta etapa, quando a implementação dos conceitos é apresentada para a gestão de produtos, a decisão pode ser: (i) seguir para as próximas iterações ou (ii) interromper a implantação caso o resultado ou o *feedback* do usuário não seja satisfatório. Embora no modelo original de Bosch e Bosch-Sijtsema (2011) a Etapa 3 - Primeira iteração cite ser possível que funcionalidades de pequena complexidade sejam desenvolvidas e entregues nesta etapa, esta característica não foi migrada para o método por não ser algo comum. Portanto, a Etapa 3 possui um único processo voltado para o desenho e planejamento detalhado da solução, como pode ser visto na Tabela 40.

Tabela 40: Processos da etapa 3. Primeira iteração de desenvolvimento

Processo	***Descrição***
3.1	Desenho detalhado da solução

Fonte: Elaborado pelo autor

O processo acima será detalhado nas sessões a seguir, incluindo objetivos, responsável (facilitador), participantes, ferramentas e técnicas.

3.1. Desenho detalhado da solução

O processo 3.1. Desenho detalhado da solução tem como objetivo possibilitar a geração dos detalhes a serem desenvolvidos no *software* a partir da orientação das necessidades apontadas pelos clientes e usuários. A Tabela 41 apresenta o facilitador, responsável pela execução do processo, e os atores participantes.

Tabela 41: Facilitador e participantes do processo 3.1. Desenho detalhado da solução

Papel	Atores
Facilitador	*Product Owner*
Participantes	*Scrum Master, Product Owner,* gerente de produto, arquiteto de *software,* analista de sistemas, desenvolvedor, analista de testes e usuários extremos

Fonte: Elaborado pelo autor

Ao longo da execução deste processo é esperado que o *Product Owner* obtenha o detalhamento da solução a partir da visão do produto e a identificação das características do *software* a serem implementadas durante a primeira iteração de desenvolvimento. A primeira iteração possui como principal característica fornecer, em um maior nível de detalhe, o conceito do que será entregue ao time de desenvolvimento. É a primeira oportunidade de demonstrar sob a perspectiva funcional como o *software* irá se comportar. Critérios de usabilidade, visualização das funcionalidades e fluxo de informações são inicialmente promovidas para que os usuários tenham um primeiro contato com o *software* e possam dar os primeiros *feedbacks* sobrea aderência do produto em relação aos requisitos. O objetivo final neste processo é ter a definição das estórias do usuário e categorizá-las na forma de requisitos (Vidal, 2017), que podem ser realizados por meio de ferramentas como o *grooming* e *blueprint* detalhado.

Este processo finaliza com a validação das funcionalidades e determinação como as mesmas serão incluídas no *software*. O nível de detalhe gerado nesta etapa deve permitir que o usuário avalie como as funcionalidades serão implementadas, permitindo identificar e formalizar o alinhamento de cada uma delas com as expectativas destes usuários com *software* a ser recebido ao término do desenvolvimento. Esta validação ocorre em dois momentos: no primeiro momento é realizada a avaliação da implementação do produto para que num segundo momento seja realizada a aprovação do protótipo pelo usuário. A partir da aprovação formal do protótipo da funcionalidade do *software* por parte dos usuários, o desenvolvimento deve ser iniciado conforme previsto na Etapa 4.

3.1.1. Grooming

As reuniões de *grooming* entre os membros do time do projeto e produtos no início de cada iteração são realizadas para criar o entendimento sobre o escopo (Vidal, 2017). As regras de negócio devem ser estabelecidas nesta fase para que seja criado o modelo de medição das características do *software* a ser desenvolvido, bem como seus critérios de aceite que irão nortear a entrega bem-sucedida da solução.

3.1.2. *Blueprint* detalhado

As características da solução detalhada devem permitir a identificação de como a demanda está atendida em virtude da implementação de funcionalidades no *software*. Massari e Vidal (2018) estabelecem as seguintes atividades para a decomposição da funcionalidade:

- Escrever estórias
- Definir critérios de aceite
- Definir critério de "preparado" (*ready*)
- Definir critério de "pronto" (*done*)

O *blueprint* detalhado permite que a decomposição dos requisitos ocorra de modo que todas as informações obtidas junto aos clientes sejam traduzidas em estórias. O *Product Owner* deve buscar o alinhamento entre usuário, cliente e o time de desenvolvimento para os aspectos constantes no *blueprint* detalhado para a solução no formato de estórias do usuário. Ou seja, os critérios de aceitação, de preparado (*ready*) e de pronto (*done*) que compõem as estórias devem ser definidos, alinhados e formalizados entre todas as partes interessadas e envolvidas no processo de desenvolvimento do *software* ao longo deste processo.

3.1.3. Avaliação da implementação do produto

A classificação dos requisitos de um projeto de *software* pode ser determinada em (i) requisitos funcionais; (ii) requisitos não-funcionais, e; (iii) critérios de aceitação (Vidal, 2017). Abaixo estão as características e exemplos destes aspectos que devem ser observadas para que exista a possibilidade de avaliar e determinar se um produto, no caso, o *software*, está ou não em conformidade.

- Requisitos funcionais: descrevem as necessidades que os usuários e clientes do veem e o que deve existir no *software*, ou seja, determina o que o *software* necessita realizar. O exemplo de um requisito funcional pode ser definido como: “habilitar a escolha de uma nova modalidade de pagamento na tela em que o cliente escolhe a forma de pagamento no processo de compra de um produto”.

- Requisitos não-funcionais: definem as características de comportamento do software para que os requisitos funcionais sejam adotados. Como exemplo de um requisito não-funcional podemos citar: “o retorno de autorização da nova forma de pagamento não pode exceder os 10 segundos entre a solicitação e a aprovação de pagamento”.
- Critérios de aceitação: representa um ou mais objetivos que devem ser alcançados a partir do uso do *software* pelo usuário. Um exemplo de critério de aceitação é realizar a consistência de data de validade de um cartão de crédito utilizado como forma de pagamento, que não pode ser menor do que o mês e ano corrente.

3.1.4. Aprovação do protótipo pelo usuário para início do desenvolvimento

A técnica de prototipação é caracterizada por definir e detalhar as principais interfaces do produto, permitindo que seja demonstrado como será feita a interação do usuário com o *software* (Massari & Vidal, 2018). De acordo com Vidal (2017), o processo de construção de protótipos detalhados deve conter as seguintes características, nesta ordem:

- requisitos dos clientes
- gerar protótipo de baixa fidelidade
- aplicar testes de aceitação
- validar modelo, condicionado ao aceite do cliente
- oficializar protótipo, condicionado a aprovação formal do cliente

Etapa 4 - Iterações de desenvolvimento

A Etapa 4 - Iterações de desenvolvimento tem como objetivo realizar o desenvolvimento das hipóteses de solução definidas em conjunto com clientes e usuários distribuindo-as em iterações para que as novas funcionalidades sejam implantadas no *software*. Para que as entregas sejam mais frequentes são estabelecidos ciclos de menor duração denominados como iterações de desenvolvimento, além de fazer uso do conceito de entrega de um menor produto viável ou MVP, do inglês *Minimum Viable Product*. A Figura 23 traz o resumo do processo da Etapa 4, incluindo objetivos, facilitadores, participantes e ferramentas.

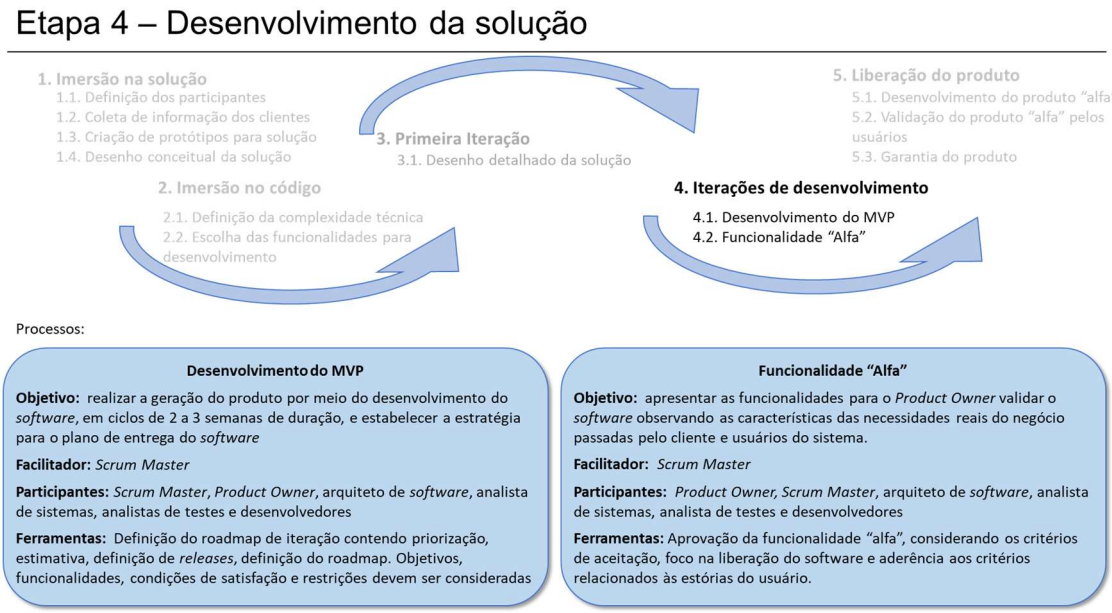

Figura 23. Processos e ferramentas da Etapa 4: Iterações de desenvolvimento

Fonte: Autor

Ao final de cada iteração prevista na Etapa 4, os resultados são apresentados para a gestão de produtos. Um dos aspectos a serem observados nesta etapa está relacionado ao modelo de contratação do time que ficará responsável pelo desenvolvimento do *software*. Uma das premissas é que haja um time dedicado, que poderá ser designado por um time de desenvolvimento próprio da empresa ou pela contratação de uma empresa ou fábrica de *software* que ficará responsável pela execução das atividades para gerar a funcionalidade "alfa".

Para projetos maiores e mais complexos, o mecanismo chamado de funcionalidade "alfa" é gerado para capturar o *feedback* dos usuários. A funcionalidade alfa consiste na implementação da solução à versão corrente do *software* disponível aos clientes, quando houver um sistema já em uso. O usuário pode inclusive fazer uso da funcionalidade em ambiente controlado, que em certas ocasiões pode apresentar qualidade inferior se comparado ao ambiente oficial do sistema. Velocidade de processamento reduzida e eventuais instabilidades podem ser características inferiores experimentadas pelo cliente ao utilizar o *software* implantado em um ambiente controlado (Bosch & Bosch-Sijtsema, 2011). A Etapa 4 – Iterações de desenvolvimento é subdividida em dois processos, como pode ser visto na Tabela 42.

Tabela 42: Processos da etapa 4. Iterações de desenvolvimento subsequentes

Processo	***Descrição***
4.1	Desenvolvimento do MVP
4.2	Funcionalidade “alfa”

Fonte: Elaborado pelo autor

Cada um dos processos acima será detalhado nas sessões a seguir, incluindo objetivos, responsável (facilitador), participantes, ferramentas e técnicas.

4.1. Desenvolvimento do MVP

O processo 4.1. Desenvolvimento do MVP tem como objetivo realizar a geração do produto por meio do desenvolvimento do *software*. A Tabela 43 apresenta o facilitador, responsável pela execução do processo, e os atores participantes.

Tabela 43: Facilitador e participantes do processo 4.1. Desenvolvimento do MVP

Papel	Atores
Facilitador	*Scrum Master*
Participantes	*Scrum Master, Product Owner*, arquiteto de *software*, analista de sistemas, analistas de testes e desenvolvedores

Fonte: Elaborado pelo autor

O desenvolvimento do software deverá ocorrer por meio de iterações com uma duração previamente estabelecida entre o time de desenvolvimento e o cliente. Embora a literatura exponha que cada iteração deve ter no máximo 30 dias de duração (Vidal, 2017), a execução desta etapa deve ser mais curta, entre duas a três semanas. A limitação das iterações em um período menor de tempo está fundamentada no fato deste método promover entregas com menor duração e com maior frequência. Deste modo, este processo possui como ferramenta a definição do *roadmap* de iteração, que permite estabelecer a estratégia para o plano de entrega do *software* e alinhado com a expectativa dos clientes e usuários.

4.1.1. Definição do *roadmap* de iteração

O *roadmap* de iteração é composto pelas informações que caracterizam a entrega ao final de cada iteração. O *roadmap* de iteração deve conter os itens do *backlog* do MVP que serão trabalhados, metas, critérios de aceite e de “pronto” (Vidal, 2017).

Nesta fase do projeto, deve-se buscar a estruturação das etapas de cada uma das *releases* e iterações buscando a definição dos seguintes aspectos: priorização, estimativa, definição de *releases* e definição do *roadmap* (Massari & Vidal, 2018):

- Priorização: com base no *backlog* do produto, o *Product Owner* deve priorizar as entregas de acordo com o valor de negócio, esforço, complexidade e risco.

- Estimativa: o time de desenvolvimento deve estimar cada item do *backlog* do produto com a unidade de esforço reconhecida e acordada com todos (ex. *story points*, horas, etc.).
- Definição de *releases*: o *Product Owner* deve determinar quantos releases do produto serão realizados ao longo de um período (linha de tempo) e quais os objetivos de negócio que serão atendidos a cada lançamento (*release*).
- Definição do *roadmap*: listar as iterações necessárias para que cada release seja concluído, determinando a meta de cada iteração e a velocidade exercida pelo time de desenvolvimento.

Cabe reforçar que o *Product Owner* deve considerar, a cada *release*, as seguintes informações (Massari & Vidal, 2018):

- Qual o objetivo do negócio que será atendido
- Quais funcionalidades, processos e estórias que a compõe
- Quais as condições de satisfação ou critérios de aceite
- Quais as restrições existentes (prazo, custo, pessoas, recursos, processo e disponibilidade).

4.2. Funcionalidade "alfa"

O processo 4.2. Funcionalidade "alfa" tem como objetivo apresentar as funcionalidades para o *Product Owner* validar o *software* observando as características das necessidades reais do negócio passadas pelo cliente e usuários do sistema. A Tabela 44 apresenta o facilitador, responsável pela execução do processo, e os atores participantes.

Tabela 44: Facilitador e participantes do processo 4.1. Desenvolvimento do MVP

Papel	Atores
Facilitador	*Scrum Master*
Participantes	*Product Owner*, *Scrum Master,* arquiteto de *software*, analista de sistemas, analista de testes e desenvolvedores

Fonte: Elaborado pelo autor

O *Scrum Master* é quem desempenha o papel de gestor do projeto de desenvolvimento do *software* e entrega as funcionalidades ao *Product Owner*. Para que a funcionalidade do *software* seja dada como concluída, os critérios de aceite definidos no processo 3.1. Desenho detalhado da solução junto ao cliente deverão nortear a validação do que foi desenvolvido no processo 4.1 Desenvolvimento do MVP. Sendo assim, o *Product Owner* é capaz de avaliar se

existe aderência entre o produto gerado e as expectativas dos clientes e usuários. Porém, nos casos onde o Product Owner entender que existe a necessidade de envolvimento de clientes e usuários neste processo, devem ser convocadas as mesmas pessoas definidas no processo 1.1. Definição dos participantes.

A partir do retorno sobre o aceite da funcionalidade por parte do *Product Owner*, o time do projeto já poderá avaliar novas necessidades dos clientes para o desenvolvimento de novas funcionalidades no *software*. As necessidades que podem ser requisitos ou "dores" dos clientes devem ser organizadas em iterações que respeitem o limite de duração estipulado entre o time do projeto e cliente em cada ciclo de desenvolvimento.

4.2.1. Aprovação da funcionalidade "alfa"

O *Product Owner* realiza a avaliação do *software* de modo a identificar se a funcionalidade desenvolvida está adequada ao uso e se está em conformidade para a resolução dos problemas sinalizados pelos clientes e usuários. Os critérios de "pronto" e aceitação devem ser confirmados, a partir dos exemplos estabelecidos por Vidal (2017):

- A definição do que é ou não pronto ajuda no entendimento dos critérios de aceitação;
- A liberação do produto, no caso o *software* deve ser preponderante, ou seja, sobrepõe a ausência de documentação;
- É necessário que os critérios relacionados à estória do usuário sejam revalidados.

Sendo assim, o *Product Owner*, em função do conhecimento que possui em relação ao *software*, poderá realizar a aprovação da funcionalidade em nome dos usuários observando se os critérios de aceite anteriormente definidos estão sendo atendidos, mesmo que nesta etapa não haja o envolvimento direto dos usuários e clientes do *software*. Caso o *Product Owner* não detenha um nível de conhecimento adequado do produto ou esteja em dúvida em relação ao atendimento dos critérios de aceite, os usuários e clientes do *software* devem ser envolvidos nesta etapa.

Etapa 5 – Liberação do produto

O objetivo da Etapa 5 - Liberação do produto é realizar o desenvolvimento, validação, autorização para implantação e garantia dos produtos "alfas". Cada produto "alfa" é composto por várias funcionalidades que farão parte do produto a partir de sua validação e aceitação por

parte dos usuários. A Figura 24 traz o resumo dos processos da Etapa 5, incluindo objetivos, facilitadores, participantes e ferramentas.

Figura 24. Processos e ferramentas da Etapa 5: Liberação do produto

Fonte: Autor

O produto "alfa" é composto por todas as funcionalidades desenvolvidas nas iterações de desenvolvimento previstas na Etapa 4. Iterações de desenvolvimento. Após o agrupamento e empacotamento destas funcionalidades, o produto deve ser validado pelos clientes e usuários do *software*. Em alguns casos a validação do produto "alfa" pode ocorrer em um ambiente controlado, facilitando que os clientes avaliem as novas funcionalidades de forma operacional. Após a validação, o produto "alfa" é liberado para o ambiente de produção, sendo que o time de desenvolvimento ainda deverá participar da etapa de garantia do produto, até que as funcionalidades estejam implantadas e funcionais em definitivo. Caso alguma inconformidade seja identificada ao longo do período de garantia, o time do projeto deverá atuar para realizar o reparo e devolver a funcionalidade em pleno funcionamento aos usuários e clientes do *software*. A etapa 5 é subdividida em três processos, como pode ser visto na Tabela 45.

Tabela 45: Processos da etapa 5. Liberação do produto

Processo	***Descrição***
5.1	Desenvolvimento do produto "alfa"
5.2	Validação do produto "alfa" pelos usuários
5.3	Garantia do produto

Fonte: Elaborado pelo autor

Cada um dos processos acima será detalhado nas sessões a seguir, incluindo objetivos, responsável (facilitador), participantes, ferramentas e técnicas.

5.1. Desenvolvimento do produto “alfa”

O processo 5.1. Desenvolvimento do produto “alfa” tem como objetivo realizar a unificação de várias funcionalidades para a incorporação ao produto “alfa” que será liberado para um grupo de usuários finais. A Tabela 46 apresenta o facilitador, responsável pela execução do processo, e os atores participantes.

Tabela 46: Facilitador e participantes do processo 5.1. Desenvolvimento do produto “alfa”

Papel	Atores
Facilitador	*Product Owner*
Participantes	*Scrum Master, Product Owner*, arquiteto de *software*, analista de sistemas, analistas de testes e desenvolvedores

Fonte: Elaborado pelo autor

A implantação do produto “alfa” em ambiente controlado possibilita que os usuários tenham contato com o *software* e realizem um teste do sistema por meio de situações reais, possibilitando que eventuais erros ou inconformidades sejam identificados antes da implantação em definitivo. Sendo assim, os critérios de aceitação identificados anteriormente poderão ser validados pelos usuários e clientes do *software* em ambiente controlado.

5.1.1. Consolidação das funcionalidades “alfa” aprovadas

Esta técnica permite realizar o processo de unificação das funcionalidades por meio de atividades de engenharia de *software*, permitindo que a solução desenvolvida seja incorporada ao produto concebido para o atendimento das necessidades dos usuários e a resolução das “dores” dos clientes. Todas as funcionalidades desenvolvidas, após este agrupamento farão parte do produto “alfa” e serão testadas e validadas pelos usuários-chave ao longo do processo 5.2. Validação do produto “alfa” pelos usuários.

5.2. Validação do produto “alfa” pelos usuários

O processo 5.2. Validação do produto “alfa” pelos usuários tem como objetivo obter a formalização dos clientes do *software* para que o produto seja implantado. A Tabela 47 apresenta o facilitador, responsável pela execução do processo, e os atores participantes.

Tabela 47: Facilitador e participantes do processo 5.2. Validação do produto “alfa” pelos usuários

Papel	Atores
Facilitador	Gerente de produto
Participantes	*Scrum Master, Product Owner*, arquiteto de *software*, analista de sistemas, desenvolvedores, usuários-chave

Fonte: Elaborado pelo autor

O processo em questão prevê a existência da formalização para que o produto seja implantado em ambiente de produção pelos clientes e usuários. Como nesta etapa devem ser realizadas as validações das funcionalidades do *software*, uma das ferramentas para que essa validação ocorra é a cerimônia de revisão da iteração, que avalia e mede o resultado da entrega (Vidal, 2017).

5.2.1. Decisão de implantação da solução em ambiente de produção

Este processo visa concluir o ciclo de desenvolvimento do produto, estabelecendo a entrega formal das funcionalidades do *software* de modo a solucionar os problemas dos usuários e clientes. A base para a decisão de implantação da nova funcionalidade pode ser desdobrada nos seguintes aspectos, como sugerido por Massari e Vidal (2018):

- Atendimento às necessidades dos clientes e usuários
- Atendimento às necessidades do negócio
- Atender ao *time-to-market*, ou seja, ao tempo necessário para disponibilidade da funcionalidade ao mercado
- E, principalmente, atender à visão/objetivo do produto definida anteriormente.

Caso algum dos critérios não tenha sido satisfatoriamente atendido, cabe ao *Product Owner* identificar o motivo e retornar à etapa do processo correspondente ao problema identificado. Para que esta situação seja melhor ilustrada, quando há uma não conformidade em relação à um requisito legal, caberá ao time do projeto retornar ao processo de 1.1 Definir participantes para que um representante da área jurídica seja inserido no contexto do projeto para sinalizar os requisitos legais que deverão ser incorporados ao *software*. Desde modo, os requisitos serão novamente obtidos e mapeados e deverão fazer parte do processo de desenvolvimento. Este processo deverá percorrer todas as etapas e processos que sucedem a identificação de requisitos, tais como detalhamento técnico, iterações de desenvolvimento e liberação do produto.

O processo de desenvolvimento de *software* se encerra com a decisão de implantação da nova funcionalidade, a partir da formalização do aceite por parte dos usuários-chave que possuam autonomia para decisão. Embora as atividades relacionadas à estratégia de

implantação e engenharia de *software* sejam mandatórias, em função de haver variações entre as diferentes organizações, elas não serão abordadas neste método.

5.3. Garantia do produto

O processo 5.3. Garantia do produto tem como objetivo coletar informações relacionadas ao produto após a implantação em tempo de projeto e realizar o suporte ao *software*. A Tabela 48 apresenta o facilitador, responsável pela execução do processo, e os atores participantes.

Tabela 48: Facilitador e participantes do processo 5.3. Garantia do produto

Papel	Atores
Facilitador	*Product Owner*
Participantes	*Scrum Master, Product Owner*, arquiteto de *software*, analista de sistemas, desenvolvedores e usuários-chave

Fonte: Elaborado pelo autor

O time do projeto necessita dar o suporte necessário caso haja a identificação de não conformidades após a implantação do *software*. Deste modo, este processo visa assegurar que as necessidades identificadas junto aos clientes e usuários ao longo de todo o projeto estejam sendo atendidas de forma adequada pelo comportamento do *software*. As ferramentas deste processo são: coleta de informações sobre o funcionamento do produto e suporte aos usuários.

5.3.1. Coleta de informações sobre o funcionamento do produto

Obter as informações sobre o funcionamento do *software,* incluindo se os requisitos funcionais e não-funcionais estão sendo atendidos em sua totalidade. A identificação do comportamento do sistema deve ser feita primeiramente pelo *Product Owner* junto aos usuários do sistema de modo a entender e assegurar que as funcionalidades originalmente solicitadas estejam de fato trazendo o retorno esperado. Caso seja identificada alguma não conformidade, o time do projeto deverá atuar no suporte aos usuários. A partir do momento em que o *software* esteja formalmente implantado e em conformidade com os requisitos, a equipe do projeto poderá ser liberada para atuar em outras iniciativas.

5.3.2. Suporte aos usuários

A partir da identificação de uma não conformidade, atuar na correção dos problemas identificados e realizar o alinhamento com os usuários do *software*. A atuação do *Product*

Owner, suportado pelo time de desenvolvimento, deve ser desdobrada nas seguintes atividades sempre que um problema no funcionamento do *software* for identificado:

- Detalhar o problema relatado
- Realizar o reparo no sistema
- Analisar se existe problemas operacionais que estejam resultando no comportamento inadequado do sistema
- Realizar os testes da correção
- Avaliar se a não conformidade foi solucionada na causa-raiz, a partir de uma nova coleta de informações sobre o funcionamento do produto.