UNIVERSIDADE FEDERAL DO CEARÁ

CAMPUS DE RUSSAS

CURSO DE GRADUAÇÃO EM ENGENHARIA DE SOFTWARE

MIRLANE BESERRA CAVALCANTE

PROPOSTA DE UM MODELO BASEADO NA COMBINAÇÃO DE METODOLOGIAS ÁGEIS DE DESENVOLVIMENTO DE SOFTWARE

RUSSAS

2018

MIRLANE BESERRA CAVALCANTE

PROPOSTA DE UM MODELO BASEADO NA COMBINAÇÃO DE METODOLOGIAS ÁGEIS DE DESENVOLVIMENTO DE SOFTWARE

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia de Software do Campus Russas da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Engenharia de Software.

Orientador: Prof. Ms. José Osvaldo Mesquita Chaves

RUSSAS

2018

MIRLANE BESERRA CAVALCANTE

PROPOSTA DE UM MODELO BASEADO NA COMBINAÇÃO DE METODOLOGIAS ÁGEIS DE DESENVOLVIMENTO DE SOFTWARE

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia de Software do Campus Russas da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Engenharia de Software.

Aprovada em:

BANCA EXAMINADORA

Prof. Ms. José Osvaldo Mesquita Chaves (Orientador)
Universidade Federal do Ceará (UFC)

Profa. Dra. Anna Beatriz dos Santos Marques
Universidade Federal do Ceará (UFC)

Profa. Dra. Marília Soares Mendes
Universidade Federal do Ceará (UFC)

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Biblioteca Universitária
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

C364p Cavalcante, Mirlane Beserra.
Proposta de um modelo baseado na combinação de metodologias ágeis de desenvolvimento de software / Mirlane Beserra Cavalcante. – 2018.
75 f. : il. color.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Campus de Russas, Curso de Engenharia de Software, Russas, 2018.
Orientação: Prof. Me. José Osvaldo Mesquita Chaves.

1. Scrum (Desenvolvimento de software). 2. eXtreme Programming. 3. Feature Driven Development. 4. Lean (Desenvolvimento de software). 5. OpenUP/Basic (Processo Unificado Aberto). I. Título.
CDD 005.1

AGRADECIMENTOS

A Deus por me conceder forças para superar as dificuldades, por iluminar os meus caminhos, por ser essencial em minha vida e ser autor do meu destino. Te agradeço, meu Deus, por todas as bênçãos que já recebi e por todos os dias cuidares de mim, por tudo que já superei e alcancei na minha vida. Obrigada por tudo, Senhor!

Agradeço ao Prof. Ms. Osvaldo Mesquita, meu orientador, pela disponibilidade, incentivo, atenção e apoio para realização deste trabalho. Por compartilhar informações e experiências profissionais, que muito contribuíram para o aprimoramento de conhecimentos. Grata sempre!

A minha mãe, Maria Fabíola, pela sua dedicação, atenção e apoio. Por todo amor concedido, carinho, conselhos, sou eternamente grata por tudo. Te amo, minha rainha.

Ao meu namorado, Adailson Raulino, pela paciência, e por entender a minha ausência em alguns momentos destinados a realização deste trabalho. Por todo amor e carinho concedidos, que tornam meus dias melhores.

Aos professores Ms. Marcos Vinícius de Andrade, Dra. Anna Beatriz dos Santos e Dra. Marília Mendes, integrantes da banca examinadora, pela disponibilidade e enriquecimento deste estudo, através de sugestões de melhorias para o projeto de pesquisa.

À todos os professores, que ao longo desses anos contribuíram, com seus ensinamentos, na composição de aprendizado.

A Universidade Federal do Ceará, em especial o campus de Russas, por proporcionar a oportunidade em cursar uma graduação. Ao seu corpo docente, direção, professores, discentes, servidores, todos que contribuem para manter a universidade como uma excelente referência de estudo.

Ao Núcleo de Soluções em Software por me proporcionar uma rica experiência em trabalhar com projetos de desenvolvimento de software. E por conceder a realização de um estudo de caso. Muito grata também aos envolvidos neste estudo, que muito contribuíram para a realização deste trabalho.

A todos da minha família que apoiam e ficam contentes pelas minhas conquistas, e pelos incentivos em realizar sonhos. Em especial aos meus irmãos, Francisco Alcilon, Alane Cavalcante e Jandecleia Cavalcante, minhas primas Letícia Bezerra e Girllen Wanne, minha cunhada Maria Ticiane e meu sobrinho Alan Robson.

Obrigada a todos que de alguma forma contribuíram para a realização deste trabalho e durante minha caminhada na graduação.

“O sucesso nasce do querer, da determinação e persistência em se chegar a um objetivo. Mesmo não atingindo o alvo, quem busca e vence obstáculos, no mínimo fará coisas admiráveis.”

(José de Alencar)

RESUMO

O aumento da procura por softwares gerou consequentemente uma elevação da disputa no mundo empresarial. Por esse motivo, é visto a grande importância em seguir boas práticas significativas no processo de desenvolvimento com finalidade de adquirir aumento na produtividade, qualidade do produto e dos projetos. Para isso é indispensável ter um processo bem definido que ajuda a evitar retrabalho, entender as expectativas do cliente, escolher as ferramentas corretas e adequadas de acordo com o orçamento estabelecido, e que auxilie no trabalho organizado e colaborativo em equipe. As metodologias ágeis são uma excelente proposta para se aderir à essas boas práticas de desenvolvimento e superar obstáculos no gerenciamento de projetos, respondendo com mais rapidez à mudanças, além de ter foco no cumprimento de orçamentos e prazos. No entanto, possivelmente um só método não será suficiente para atender todos esses fatores. Devido a essa questão e ao fato dos mesmos possuírem pontos positivos e negativos, esse trabalho sugere um modelo ágil, baseado na combinação de melhores práticas das metodologias Scrum, XP, FDD, Lean e OpenUp/Basic. A pesquisa tem modalidade quali-quantitativa, pois para realização deste trabalho foram utilizados os métodos de pesquisa bibliográfica e análise de documentos para coletar dados sobre metodologias ágeis e modelos já proposto por outros autores, e foi aplicado um questionário que obteve informações sobre dificuldades de equipes de desenvolvimento de software em relação ao processo de construção do produto. Além disso, foi realizado um estudo de caso para verificar benefícios do modelo, e posteriormente foi realizado entrevistas com os envolvidos no estudo de caso. Como resultados apresenta-se uma comparação das tarefas de antes e depois da implantação das práticas do modelo, opiniões da equipe de desenvolvimento de software que participou da pesquisa e sugestões de melhorias.

Palavras-chave: Scrum (Desenvolvimento de software). eXtreme Programming. Feature Driven Development. Lean (Desenvolvimento de software). OpenUP/Basic (Processo Unificado Aberto).

ABSTRACT

The increase in the demand for software has consequently generated a rise in the dispute in the business world. For this reason, it is important to follow significant good practices in the development process in order to increase productivity, product quality and projects. To do this, it is essential to have a well-defined process that helps to avoid rework, understand customer expectations, choose the right tools, according to the established budget, and assist in organized and collaborative teamwork. Agile methodologies are an excellent proposal to adhere to these good development practices and overcome obstacles in project management, responding faster to change, and focus on meeting budgets and deadlines. However, possibly a single method will not suffice to meet all these factors. Due to this issue and their positive and negative points, this work suggests an agile model based on the combination of best practices of Scrum, XP, FDD, Lean and OpenUp / Basic methodologies. The research has a qualitative-quantitative method, since for the accomplishment of this work the methods of bibliographic research and analysis of documents were used to collect data on agile methodologies and models already proposed by other authors, and a questionnaire was obtained with information about difficulties of teams of software development in relation to the process of product construction. In addition, a case study was carried out to verify benefits of the model, and interviews were subsequently conducted with those involved in the case study. As results we present a comparison of the tasks before and after the implantation of the practices of the model, opinions of the software development team that participated in the research and suggestions for improvements.

Keywords: Scrum (Software Development). eXtreme Programming. Feature Driven Development. Lean (Software Development). OpenUP/Basic (Unified Process Open).

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

CLF	Construir Lista de Funcionalidades
CPF	Construir por Funcionalidade
DMA	Desenvolver Modelo Abrangente
DPF	Detalhar por Funcionalidade
FDD	Feature Driven Development
GCM	Gerência de Configuração e Mudanças
LSD	Lean Software Development
MVP	Minimum Viable Product
N2S	Núcleo de Soluções em Software
PPF	Planejar por Funcionalidade
PO	Product Owner
OpenUP	Open Unified Process
TDD	Test Driven Development
TI	Tecnologia da Informação
UFC	Universidade Federal do Ceará
XP	Extreme Programming

SUMÁRIO

1 INTRODUÇÃO

O avanço tecnológico está cada dia mais presente e disponível à maioria da população. Devido a isso as empresas procuram desenvolver novos softwares como forma de aproveitar oportunidades, e manter seu negócio diante da grande competitividade no mercado (SOMMERVILLE, 2011). Apesar do software fazer parte do cotidiano das pessoas, sendo encontrado em vários locais pertencentes a diferentes negócios, tais como escolas, espaços de lazer e de segurança, ainda há no desenvolvimento de software um grande desafio para construir um produto com qualidade e menos erros, sem extrapolar o prazo e o orçamento previamente estipulados (SANTOS; CÓRDOVA, 2017).

Empresas que utilizam métodos ágeis percebem grandes vantagens como ganhos significativos em produtividade, foco no cumprimento de orçamentos e prazos, produção de software com mais rapidez e satisfação do cliente (COHN, 2011).

As metodologias ágeis se caracterizam por possuir um desenvolvimento iterativo e incremental, por dispor de equipes que trabalham de forma organizada e colaborativa, além de ter a presença do cliente durante as etapas do projeto para auxiliar no entendimento e nas possíveis mudanças de requisitos (PIMENTEL; PEGORARO, 2016).

Mesmo com todas as técnicas e práticas existentes as empresas de pequeno e médio porte encontram dificuldades na utilização dos modelos de desenvolvimento, pois muitas vezes a documentação abrangente é desnecessária e consome muito tempo, por outro lado a falta dela pode prejudicar por não deixar claro o detalhamento das atividades necessárias (COSTA, 2010).

Apesar dos métodos ágeis apoiarem bastante o desenvolvimento de sistemas, possivelmente a utilização de um único método não será suficiente para atender todos os fatores para adquirir um produto com uma excelente qualidade (FELLER, 2013). O sucesso das metodologias ágeis tem se confirmado em alguns casos. Por exemplo, com a adoção ao *Scrum* pode-se ter a obtenção de melhorias na gerência, satisfação do cliente e comunicação. Porém, outras como o XP não demonstram muita adesão a essas práticas de controle e gerenciamento de software (COELHO, 2011). Assim, é notado que cada metodologia pode considerar mais algumas práticas do que outras, ou seja, pode ter um foco mais específico. Outro problema percebido é o fato de que o desenvolvimento ágil acaba deixando a documentação do software de lado (JUNIOR; ROCHA; GOMES, 2016). Além disso, alguns métodos ágeis, como exemplo o *Scrum*, não especificam como a atividade de Gerência de Configuração e Mudança deve ser incorporada ao processo de desenvolvimento, onde fica a cargo da equipe decidir a forma de execução desse gerenciamento. Essa prática permite o acompanhamento e o controle de

artefatos, processos e ferramentas, com o propósito de evitar confusões dispendiosas, garantindo que durante o andamento do projeto os artefatos produzidos não entrem em conflito (MAGALHÃES; SANTOS; ALMEIDA, 2010). Devido à essas questões, neste trabalho é proposto um modelo que combina as melhores práticas de cada metodologia ágil abordada (*Scrum*, *Extreme Programming* (XP), *Feature Driven Development* (FDD), *Lean*, OpenUP/Basic).

As metodologias *Scrum*, XP, FDD, *Lean*, foram selecionadas com base nos trabalhos relacionados apresentados na Seção 2 deste trabalho. Pois, esses apresentam combinações de metodologias ágeis para desenvolvimento de *software*. Já o OpenUP/Basic foi escolhido pelo motivo de possuir as disciplinas Gerência de Configuração e Mudança, e Análise e Projeto de Sistemas que proporciona a documentação e modelagem do sistema. Essas são as práticas já pré-definidas para compor o modelo a ser elaborado neste presente trabalho. Justamente para minimizar os problemas citados anteriormente em relação à documentação e gerenciamento de artefatos.

Para realização deste trabalho foi utilizada pesquisa bibliográfica e análise de documentos (livros e artigos) para coletar dados sobre metodologias ágeis e modelos já propostos por outros autores. Essas informações foram obtidas em bibliotecas digitais, artigos encontrados em sites, como da empresa Softplan (SOFTPLAN, 2018), e periódicos. Além disso, foi feito uso de questionário para obter informações sobre possíveis dificuldades de membros participantes em projetos de *software*, em relação ao processo de desenvolvimento do produto. A modalidade da pesquisa é quali-quantitativa, pois além da qualitativa para análise de conceitos e princípios de metodologias ágeis, possui uma parte quantitativa para coleta de dados através de questionário, e em seguida uma análise sobre os resultados obtidos.

Alguns trabalhos sobre interação entre metodologias ágeis foram encontrados durante as pesquisas realizadas, sendo eles: Feller (2013) que apresenta uma proposta de modelo ágil por meio da junção de boas práticas das metodologias SCRUM, KANBAN, LEAN, FDD e XP; já em Santos e Córdova (2017) mostram um estudo de caso da aplicação de uma combinação dos métodos *Scrum*, Kanban e XP; e Jesus (2016) que propôs uma metodologia híbrida para *startups* com base na combinação de Scrum, XP e Lean *Startup* (uma variação do Lean). Porém, o trabalho de Feller (2013) se difere dos outros por propor um modelo com mais atividades relacionadas ao processo de desenvolvimento, combinando diferentes metodologias. Portanto, é o que mais se aproxima do contexto deste trabalho.

Feller (2013) formulou em seu trabalho um modelo para organizações que utilizam as metodologias *Scrum* e *Kanban* aplicando as boas práticas das metodologias *Scrum*, *Kanban*,

eXtreme Programming, *Feature Driven Development* e Lean com o intuito de potencializar os pontos positivos de cada uma. Porém, não considera as práticas de Gerência de Configuração e Mudanças, além de atividades de Análise e Projeto de Sistemas que podem ser obtidas através do método OpenUp/Basic (*Open Unified Process*).

O modelo proposto neste trabalho tem o intuito de diminuir problemas no desenvolvimento de *software*, tais como, requisitos mal entendidos, falta de documentação e artefatos sem registro de mudanças, visando beneficiar organizações de desenvolvimento de *software*, de pequeno e médio porte, que querem um processo que apoie a gestão de projetos, que tenha uma minimização da documentação e possua técnicas para obter as informações relevantes sobre cada mudança ocorrida durante o processo de desenvolvimento.

Este trabalho está organizado da seguinte forma: No capítulo 2 encontram-se os trabalhos relacionados a esta pesquisa, descrevendo as informações de cada um deles. No capítulo 3 estão presentes os objetivos, sendo divididos em objetivo geral (Subseção 3.1) e específicos (Subseção 3.2). No capítulo 4 apresenta-se a fundamentação teórica, discorrendo sobre metodologias ágeis (Subseção 4.1), SCRUM (Subseção 4.2), XP (Subseção 4.3), FDD (Subseção 4.4), Lean (Subseção 4.5) e OpenUP/Basic (Subseção 4.6). No capítulo 5 contém os procedimentos metodológicos do presente trabalho. No capítulo 6 os resultados desses procedimentos, separados em resultados da pesquisa de coleta de dados (Subseção 6.1), comparações entre as metodologias (6.2) e o modelo proposto (Subseção 6.3). No capítulo 7 é apresentado como foi realizado a etapa do estudo de caso, onde no capítulo 8 podem ser observados os resultados desse. E no capítulo 9, são apresentadas as conclusões da pesquisa e trabalhos futuros.

2 TRABALHOS RELACIONADOS

Os trabalhos relacionados que foram identificados dizem respeito a trabalhos envolvendo propostas de modelos ágeis que tenham como características as principais atividades e princípios do desenvolvimento de *software*. A seguir são apresentados os trabalhos com mais detalhes.

Santos e Córdova (2017) no seu trabalho, com título "Combinação de métodos ágeis no processo de desenvolvimento de *software*: Um estudo de caso", apresentam a experiência de combinar métodos ágeis em uma empresa de *software* situada no estado de Santa Catarina. Essa empresa não possuía framework ou metodologia definida para auxílio no desenvolvimento do produto. A metodologia de pesquisa se deu a partir de um estudo exploratório bibliográfico sobre as metodologias de desenvolvimento, e posteriormente um estudo de caso com a implantação de modelos ágeis foi realizado. Foram avaliados dados coletados por meio de uma ferramenta de gestão de incidente e serviços, e também por questionário aplicado aos membros da empresa.

A combinação foi feita através dos métodos Scrum, Kanban e XP. As boas práticas que foram mescladas são:

- Scrum: *Product Backlog* e *Sprint Backlog*, papéis *Scrum Master* e *Product Owner* (PO); Reuniões diárias;
- Prática do *Kanban*;
- XP: Etapa de planejamento, princípio equipe e cliente trabalhando juntos, princípio simplicidade;
- XP: Testes no final de cada Sprint.

A combinação proposta teve resultados satisfatórios nas análises realizadas, como o aumento significativo na eficiência da equipe, membros tornaram-se mais cooperativos entre si, e mais qualidade aos produtos de *software*.

O trabalho "Metodologias Ágeis de Gerenciamento de Projetos" (JESUS, 2016), apresenta uma proposta de uma metodologia híbrida para gerenciamento de projetos adaptado para *startups*. Realizado com a finalidade de agregar valor nos processos de desenvolvimento de *software*. O modelo é baseado na combinação das metodologias *Scrum*, XP e Lean *Startup*. A metodologia do referente trabalho desenvolveu-se por meio de estudos bibliográficos sobre metodologias ágeis e gerenciamento de projetos, elaboração do modelo híbrido, e após foi

realizado um estudo de caso com a implantação da metodologia em uma empresa, a qual não foi revelada pelo autor por questões de privacidade e foi chamada de Startup X.

A empresa Startup X conta com apenas quatro funcionários, e tem como foco o desenvolvimento de um *software* e um hardware de baixo custo para seus usuários finais. A qual já possui experiência no ramo de hardware, porém no que diz respeito a *software* ainda era algo novo. Por isso, a empresa ainda não continha processos maduros de desenvolvimento de *software*. Esse é um dos fatos que levou à elaboração da metodologia híbrida para gerenciamento dos projetos na Startup X, além disso foi considerado características da empresa como limitação de recursos, tamanho da equipe e necessidade de criação de solução disruptiva e inovadora para o mercado. A Tabela 1 mostra as caraterísticas do modelo proposto nesse trabalho.

Tabela 1 - Características da metodologia para Startup X.

Conceitos propostos pela metodologia	Origem	Adesão
Cliente presente	XP	Ausente
Código coletivo	XP	Presente
Padrão de código	XP	Presente
Projeto simples	XP	Presente
Releases curtos	XP	Ausente
Jogo de planejamento	XP	Ausente
Metáfora	XP	Ausente
Programação em pares	XP	Ausente
Desenvolvimento orientado a testes	XP	Opcional
Refatoração	XP	Presente
Integração contínua	XP	Presente
Semana de 40 horas	XP	Presente
Sprint	Scrum	Presente
Reunião de planejamento do *Sprint*	Scrum	Presente
Reunião diária	Scrum	Presente
Revisão do *Sprint*	Scrum	Opcional
Retrospectiva do *Sprint*	Scrum	Opcional
Backlog do produto	Scrum	Presente
Backlog do *Sprint*	Scrum	Presente
Minimum Viable Product (MVP)	*Lean Startup*	Presente

Fonte: JESUS (2016). Adaptada pela autora.

A implantação completa da metodologia híbrida não ocorreu devido a limitações de tempo e por necessidade em concluir acordos comerciais e projetos por parte da Startup X. Assim apenas algumas práticas foram implantadas tais como Sprint, as reuniões diárias, integração contínua, MVP, *Backlog* do *Sprint* e testes.

Pelo fato que a empresa não possuía métricas estabelecidas nem processos definidos antes do modelo proposto, não foi possível apresentar uma comparação entre os resultados de processos anteriores e depois da implantação na empresa. Porém, mesmo com a implantação imparcial foi perceptível algumas melhoras relatadas pela equipe e pelos responsáveis pela startup. A equipe utilizou a ferramenta *Trello* para guardar os itens do *Backlog*, ou melhor, para simular o quadro *Kanban*. Servindo como uma espécie de documentação e descrição de requisitos, pois as tarefas listadas estão relacionadas aos requisitos do *software*.

Mesmo com a implantação imparcial foi comprovado a melhora nos resultados depois que foi seguido os processos e práticas determinadas, e que isso pode contribuir muito para o crescimento da empresa. Porém há melhorias que podem ser feitas nas práticas propostas por esta metodologia, para que contribua ainda mais para o crescimento das empresas. Como por exemplo o acréscimo de mais atividades relacionadas ao processo de desenvolvimento de *software*.

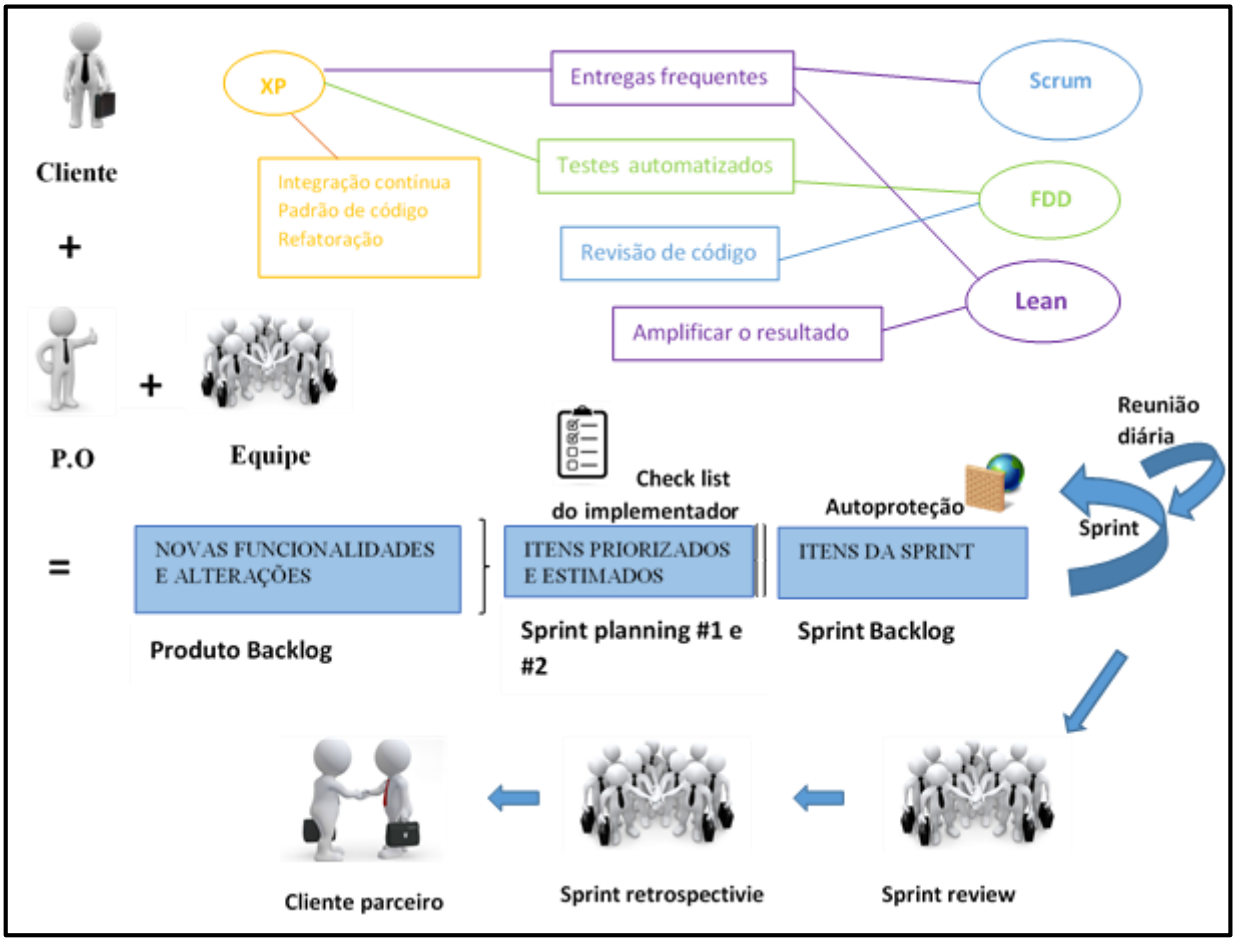
Já o trabalho “Proposta de modelo ágil: União de boas práticas das metodologias SCRUM, KANBAN, LEAN, FDD e XP” (FELLER, 2013), apresenta uma combinação mais robusta de métodos ágeis. A pesquisa teve como objetivo formular um modelo para empresas que utilizam *Scrum* e *Kanban* baseado na união das metodologias citadas anteriormente. Para isso foi utilizado a metodologia de pesquisa bibliográfica e estudo de caso.

Além de considerar as melhores práticas de cada metodologia analisada para formar o modelo ágil, foi levado em consideração as necessidades de uma empresa, citada como XPTO por questões de privacidade. Esta tem experiência no ramo de desenvolvimento de *software*, pois está atuando no mercado há mais de vinte anos. Sem o uso de metodologias de desenvolvimento foi relatado por membros de uma equipe da empresa dificuldades tais como especificação ineficiente, prioridades mal estabelecidas e refeitas a todo momento. Esses acabam gerando insatisfação do cliente, funcionalidades em desacordos com as solicitações, atraso em entregas de versão e baixa qualidade do produto final.

Inicialmente foi feita a implantação dos métodos ágeis *Scrum* e *Kanban*, onde o primeiro era utilizado para gerenciar o desenvolvimento de alterações e novas implementações, enquanto o *Kanban* era aplicado no desenvolvimento de correções. Mesmo depois dessa implantação foram relatados problemas que ocorreram durante as *Sprints*, sendo eles: Especificação imprecisa, alteração de prioridades, estimativas ineficientes, desmotivação para manter o *burndown*, problemas com código fonte, alto índice de reprovação e falta de confiança do cliente.

Com a finalidade de resolver esses problemas foi proposto um modelo ágil levando também em consideração as metodologias inicialmente implantadas na empresa XPTO. As práticas consideradas para formular esse modelo são mostradas na Figura 1:

Figura 1 - Práticas do modelo de Feller



Fonte: Feller (2013). Adaptado pela autora.

Apesar desses modelos, apresentados anteriormente, possuírem atividades essenciais para um processo de desenvolvimento de *software*, neles não foram consideradas outras tarefas que são importantes para aumentar a taxa de sucesso dos projetos como, por exemplo, atividades referentes à documentação, ao registro e gestão de mudanças, e entre outras. Tarefas estas que farão parte do conjunto de práticas do modelo proposto neste trabalho.

3 OBJETIVOS

3.1 Objetivo geral

O presente trabalho teve como objetivo principal sugerir um modelo ágil, baseado na combinação de metodologias ágeis já existentes, para dar suporte ao desenvolvimento de *software* em empresas de pequeno e médio porte, visando obter a redução de problemas nos projetos realizados, tais como produto final com erros e requisitos mal entendidos.

3.2 Objetivos específicos

Para alcançar o objetivo geral foram estabelecidas as seguintes metas:

1. Levantar dados sobre as dificuldades encontradas por equipes de desenvolvimento durante o processo de construção do *software*.
2. Identificar quais aspectos positivos das metodologias ágeis podem ser utilizados em conjunto para auxiliar o processo de desenvolvimento de *software*.
3. Elaborar um modelo baseado nas metodologias SCRUM, XP, FDD, LEAN e OPENUP/BASIC.
4. Verificar benefícios e melhorias para o modelo proposto.

4 FUNDAMENTAÇÃO TEÓRICA

Nas subseções a seguir são apresentados os conceitos necessários para um bom entendimento no decorrer deste trabalho. Onde são descritas sobre metodologias ágeis (Subseção 4.1), SCRUM (Subseção 4.2), XP (Subseção 4.3), FDD (Subseção 4.4), *Lean* (Subseção 4.5) e OpenUP/Basic (Subseção 4.6).

4.1 Metodologias Ágeis

As metodologias ágeis apresentam como uma das características o fato de serem adaptativas ao invés de serem prescritivas. Ou seja, elas se adaptam melhor às mudanças durante o desenvolvimento do projeto, ao invés de tentar analisar previamente tudo o que pode ou não acontecer no decorrer do projeto. Pois as metodologias ágeis trabalham com constante feedback, e isso auxilia bastante nas possíveis mudanças dos requisitos. Essas alterações podem ser críticas nas metodologias tradicionais, pois não apresentam meios de se adaptar rapidamente às mudanças. Outro fator importante das metodologias ágeis são as entregas constantes de partes do *software* funcionando. Assim, com essa entrega frequente o cliente não precisa esperar muito tempo para ver o *software* funcionando e perceber que não era bem isso que ele esperava (LIBARDI; BARBOSA, 2010).

No ano de 2001, uma reunião ocorrida nas montanhas nevadas do estado norte-americano de Utah, marcou o surgimento e propagação de paradigmas dos métodos ágeis**.** Onde desencadeou o conhecido manifesto ágil, o qual se tornou um incentivo para a indústria de *software* e para os dezessete profissionais de *software* que estavam presente na reunião. O manifesto ágil possui quatro valores que são: Indivíduos e interação entre eles mais que processos e ferramentas; *software* funcionando mais que documentação abrangente; colaboração do cliente mais que negociação de contratos; responder à mudanças mais que seguir um plano (BERNARDO, 2018).

Além desses quatro valores mencionados acima, foi estabelecido um conjunto de princípios que norteiam todas as metodologias descritas nas subseções posteriores. Beck *et. al* (2001) apresentam esses conforme listados a seguir:

1. Nossa maior prioridade está em satisfazer o cliente por meio da entrega adiantada e contínua de *software* de valor;

2. Aceitar mudanças de requisitos, mesmo no fim do desenvolvimento. Processos ágeis se adequam a mudanças, para que o cliente possa tirar vantagens competitivas;
3. Entregar *software* funcionando com frequência, na escala de semanas até meses, com preferência aos períodos mais curtos;
4. Pessoas relacionadas à negócios e desenvolvedores devem trabalhar em conjunto e diariamente, durante todo o curso do projeto;
5. Construir projetos ao redor de indivíduos motivados. Dando a eles o ambiente e suporte necessário, e confiar que farão seu trabalho;
6. O Método mais eficiente e eficaz de transmitir informações para, e por dentro de um time de desenvolvimento, é através de uma conversa cara a cara;
7. *Software* funcional é a medida primária de progresso;
8. Processos ágeis promovem um ambiente sustentável. Os patrocinadores, desenvolvedores e usuários, devem ser capazes de manter indefinidamente, passos constantes;
9. Contínua atenção à excelência técnica e bom design, aumenta a agilidade;
10. Simplicidade: A arte de maximizar a quantidade de trabalho que não precisou ser feito;
11. As melhores arquiteturas, requisitos e designs emergem de times auto organizáveis;
12. Em intervalos regulares, o time reflete em como ficar mais efetivo, então, se ajustam e otimizam seu comportamento de acordo.

4.2 SCRUM

Scrum é um modelo ágil voltado para a gestão de projeto que surgiu em uma indústria automobilística, e que pode ser adaptado a várias áreas diferentes da produção de *software* (WAZLAWICK, 2013).

Scrum é definido como um *framework* estrutural para tratar e resolver problemas complexos e adaptativos, que vem sendo usado desde o início de 1990. Este é caracterizado em leve, simples de entender e extremamente difícil de dominar. E consiste nos times do *Scrum* associados a papéis, eventos, artefatos e regras (SCHWABER; SUTHERLAND, 2013).

O responsável pelo projeto é o PO que tem atividades como indicar quais os requisitos mais importantes a serem tratadas em cada Sprint. Ele deve conhecer as necessidades do cliente. Já *Scrum Master* é o membro que deve conhecer bem o modelo *Scrum*, sendo assim, um facilitador e um solucionador de conflitos. E o papel *Scrum Team* é referente a equipe de desenvolvimento onde todos interagem para desenvolver o produto (WAZLAWICK, 2013).

Os requisitos a serem implementados em cada projeto são mantidos em uma lista denominada *Product Backlog*, porém não é obrigatório ser uma lista completa já no início do projeto. A partir dele é determinado cada funcionalidade a ser feita em cada *Sprint*, que é o ciclo de desenvolvimento de poucas semanas (geralmente de 2 a 4 semanas). No início de cada Sprint é realizada um *Sprint Planning Meeting* para a equipe priorizar os elementos do *Product Backlog* que devem entrar na *Sprint Backlog*, ou melhor lista de funcionalidades que vão ser implementadas no ciclo de desenvolvimento. No cotidiano de cada Sprint acontece a *Scrum Dailly Meeting* com presença dos membros da equipe. Já ao final da Sprint é realizado a *Scrum Review Meeting*, onde é avaliado a funcionalidade desenvolvida, que se for aprovada será entregue ao cliente. Eventualmente acontece a *Sprint Retrospective*, onde é avaliado os processos de trabalho (WAZLAWICK, 2013).

A técnica de *Planning Poker* pode ser utilizada para estimar o esforço necessário para o desenvolvimento de cada funcionalidade do *Backlog*, onde cada membro da equipe tem cartas correspondentes aos valores da sequência de *Fibonacci*. Após explicação do item cada integrante da equipe apresenta a carta que acredita ser equivalente ao esforço necessário para a codificação (FELLER, 2013). Cada um dos participantes dá sua nota de complexidade para cada item e caso haja consenso entre eles a complexidade é validada e atribuída a funcionalidade. Em caso de divergência deve acontecer um debate entre os membros que tiveram maior diferença, onde cada um pode explicar o motivo que levou a determinada atribuição de nota. Em seguida é realizada uma nova rodada de estimativa para o mesmo item, e assim segue até o consenso (VARASCHIM, 2008).

4.3 *Extreme Programming* (XP)

XP foi desenvolvida com o objetivo de impulsionar práticas reconhecidamente boas, como por exemplo desenvolvimento iterativo. Os requisitos são expressos como estórias do usuário, que são implementados como uma série de tarefas (SOMMERVILLE, 2001). Estórias de usuário são caraterizadas como uma curta e simples descrição da necessidade do cliente. Que deve explicar bem para quem, o que e por que está sendo criada (BERNARDO, 2018).

A metodologia XP é baseada em conjunto de valores, alguns princípios e várias práticas**.** Destinada para times de até dez programadores, projetos de curto e médio prazo (LOCAWEB, 2018).

Teles (2018) mostra os cinco valores de XP que são descritos a seguir:

1. **Comunicação** – mesmo existindo vários meios de comunicação, alguns são mais eficazes que outros. Como exemplo tem-se o diálogo presencial, ou seja conversa cara-a-cara, que é mais eficaz que videoconferência. Já o último mencionado é mais eficaz que telefonema, que são melhores que e-mail e entre outros exemplos. Por isso, quem trabalha com XP prefere o uso do diálogo presencial, com o objetivo de garantir que os *stakeholders* do projeto tenham a chance de se compreender da melhor maneira possível.
2. ***Feedback*** – em projetos XP os desenvolvedores procuram entregar novas funcionalidades no menor prazo possível, possibilitando que o cliente compreenda rapidamente o que está acontecendo. E os clientes, por sua vez, procuram se manter em constante comunicação com os desenvolvedores para prover informações precisas sobre qualquer dúvida que eles venham a ter ao longo do processo de desenvolvimento.
3. **Coragem** – o XP possui mecanismos de proteção e práticas que são voltadas para proteger o *software* de inúmeras formas. Assim passa mais confiança, para os membros da equipe, em aderir as mudanças. As equipes XP consideram as mudanças inevitáveis e procuram se adaptar a elas com segurança e coragem, ou seja, com confiança em seus mecanismos de proteção e práticas, tais como desenvolvimento orientado a testes, programação em par e integração contínua.
4. **Simplicidade** – o XP utiliza este conceito para assegurar que a equipe se concentre em fazer, primeiro, apenas aquilo que é claramente necessário e evite fazer o que poderia vir a ser necessário, mas ainda não se provou essencial.
5. **Respeito** – esse valor é considerado o mais básico e essencial em um projeto. Saber escutar, compreender e respeitar o ponto de vista do outro é essencial para o sucesso de um projeto de *software*.

XP define papéis como: Programador que implementa o código e realiza testes; técnico (*coach*) responsabilidade por guiar os outros membros da equipe; *tracker* que é o responsável por verificar o progresso de cada iteração e avalia se os objetivos podem ser alcançados com os

recursos providos; consultor que é membro que auxilia o time com assuntos (problemas) técnicos específicos e o cliente responsável pelos estórias de usuário (VASCO; VITHOFT; ESTANTE, 2006).

XP possui princípios básicos como *feedback* rápido, presumir simplicidade, mudanças incrementais, abraçar mudanças e trabalho de alta qualidade (WAZLAWICK, 2013). Além desses princípios XP também possui várias práticas, essas são descritas em Campelo (2003) como:

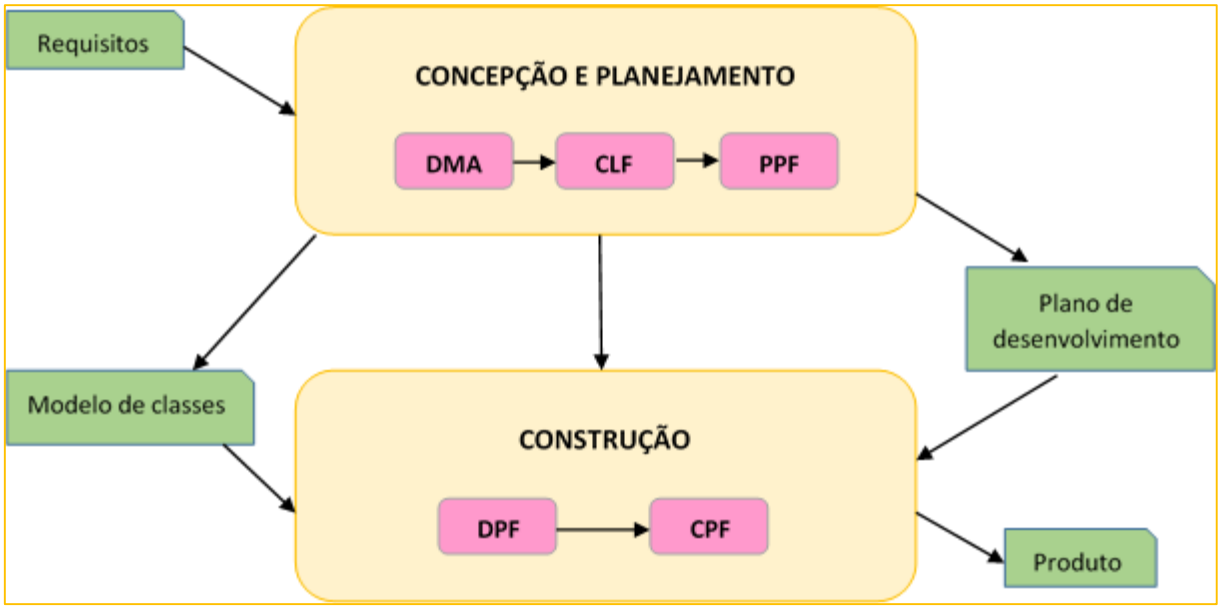
1. ***Releases* pequenos** – O *release* ocorrem em um curto espaço de tempo e possui *software* de valor.
2. **Jogo do planejamento** – É elaborado um plano para cada *release*, que consiste em determinar o escopo da iteração.
3. ***Design* simples** – O sistema deve ser simples de modo que possa solucionar o problema do cliente, fazendo somente o que foi solicitado.
4. **Testes** – São realizados automatizados, sendo testes unitários e testes de aceitação.
5. **Refatoração** – é um conjunto de técnicas para modificar o código do sistema sem alterar nenhuma funcionalidade, visando simplificá-lo, deixando o código mais fácil de entender.
6. **Programação em pares** – Dois programadores trabalham juntos na mesma máquina, onde enquanto um programador digita o outro observa.
7. **Propriedade coletiva** – O código fonte não pertence a um único programador. Todos da equipe são responsáveis. Todos podem alterar qualquer parte do código, a qualquer momento.
8. **Integração contínua** – Sempre que uma tarefa é completada o código deve ser integrado.
9. **Semana de 40 horas** – O trabalho dos membros da equipe não deve exceder 40 horas semanais, pois pode gerar cansaço e indisposição.
10. **Cliente presente** – O cliente é um membro da equipe, e deve estar sempre disponível para atender às dúvidas dos outros integrantes.
11. **Código padrão** – Para fins de bom entendimento e facilitação do processo de manutenção do produto a ser desenvolvido, o time deve seguir padrões pré-acordados de codificação deixando ele bem escrito e estruturado.

Diariamente é realizada a *Stand-up meeting* que é uma breve reunião com o objetivo de compartilhar informações sobre o projeto e suas atividades. Permitindo que os membros relatem o que aconteceu no dia anterior de trabalho. Além disso, o ambiente de trabalho da equipe deve ser um reflexo informativo do projeto, de modo que permita em instantes ter uma ideia do andamento do projeto, como exemplos gráficos de progresso e cartões com estórias em paredes (TELES, 2018).

4.4 *Feature-Driven Development* (FDD)

FDD foi apresentado em 1997 por Peter Coad e Jeff de Luca como a evolução de um processo mais antigo (WAZLAWICK, 2013). Este método enfatiza o uso da programação orientada a objetos, e possui apenas duas fases que são “Concepção e planejamento” e “Construção” como demonstrado na Figura 2.

Figura 2 - Estrutura do FDD



Fonte: Horácio (2012). Adaptada pela autora.

Os papéis principais são: gerente de projeto, arquiteto chefe, gerente de desenvolvimento, programador chefe, proprietário de classe e especialista de domínio (ALMEIDA, 2018). O FDD é composto por práticas divididas em cinco disciplinas, estas são apresentadas em Figueiredo (2007) como mostra a seguir:

1. **Desenvolver Modelo Abrangente (DMA):** Formar o time de modelagem; conduzir o Domain Walkthrough (estudo dirigido sobre o domínio do negócio), estudar documentação, desenvolver modelos de pequenos grupos, desenvolver o modelo da equipe, refinar o modelo abrangente, e escrever notas;
2. **Construir Lista de Funcionalidades (CLF):** Formar time da lista de funcionalidades e construir a lista de funcionalidades;
3. **Planejar Por Funcionalidade (PPF):** Formar o time de planejamento, determinar a sequência do desenvolvimento, atribuir atividades de negócio aos programadores, atribuir classes aos desenvolvedores;
4. **Detalhar Por Funcionalidade (DPF):** Formar as equipes de funcionalidade, estudar documentação e anotações de negócio, desenvolver diagrama de sequência, refinar o modelo abrangente, escrever prólogo de métodos e classes, e inspeção de design;
5. **Construir Por Funcionalidade (CPF):** Implementar classes e métodos, inspecionar código, teste unitário e promover a build.

4.5 Lean

A metodologia Lean existe desde que seus princípios foram definidos, no início do século 20, a indústria automotiva de veículos Toyota fez alguns ajustes de acordo com as suas necessidades e para assim criar um sistema mais adequado à sua linha de produção, no final dos anos 80 (GAEA, 2018).

SCHEREDER (2013) mostra que Lean contém princípios básicos e os apresentam como a seguir:

1. **Valor:** Isso significar especificar todas as características do produto desejadas pelo usuário, a fim de identificar as atividades que contribuem para que o produto atenda aos requisitos do cliente.
2. **Mapear o fluxo de valor:** Levar em consideração todas as ações necessárias que levam à produção de um bem ou serviço;
3. **Fluxo:** Após o valor ser definido é feito o fluxo de valor para mapear as etapas com suas atividades.

Lean também possui diversas ferramentas de apoio como setup rápido, automação, tecnologia da informação, sistema *Kanban*, prevenção de falhas, procedimento de trabalho padrão e melhorias de atividades, que são vistas como as principais (SCHEREDER, 2013). Além dessas, Lean também possui a ferramenta 5S (organização do local de trabalho) que contribui para melhorar a produtividade, aumentar a visibilidade das tarefas e garantir um fluxo de trabalho mais padronizado (GAEA, 2018).

Os princípios do *Toyota Way* – demanda puxada (que tem no mecanismo sinalização local com gráficos, reuniões diárias e testes), qualidade total, teoria das restrições, melhoria contínua e flexibilidade – antes aplicados na montadora japonesa (e posteriormente em vários setores), inspiraram também a indústria de *software*, provocando o surgimento do *Lean* para Desenvolvimento de *Software* (LSD) (BARANDAS, 2017).

Ao aplicar *Lean* em desenvolvimento de *software* teremos produção de alta qualidade realizada com mais rapidez e menor custo, pois essa metodologia busca atender as necessidades do cliente da maneira mais simples e com menor custo possível (FELLER, 2013).

Barandas (2017) informa que o LSD tem sete princípios, sendo estes apresentados a seguir:

1. **Eliminar desperdícios:** evitar tudo aquilo que não agrega valor para o cliente, como funcionalidades a mais.
2. **Embutir qualidade:** integração contínua, testes automatizados, inspeção de código e análise constante de *feedbacks* são meios capazes de incorporar qualidade nos processos.
3. **Criar conhecimento: pode ser feito através de treinamentos ou mentorias, ciclos de** *feedback*, desenvolvimento iterativo e revisões do produto. Que ajuda a enriquecer a fonte de aprendizado para amadurecer a equipe envolvida.
4. **Adiar decisões e comprometimentos:** significa diminuir incertezas e se basear em acontecimentos mais concretos. Uma estratégia é ter capacidades e práticas que permitam abraçar as mudanças mais tardiamente possível.
5. **Entregar rápido:** garante que o cliente receberá o que ele precisa no momento certo. *Kanban*, iterações e simplicidade são práticas que contribuem para uma maior velocidade no desenvolvimento.
6. **Respeitas as pessoas:** está relacionado a oferecer as pessoas um espaço para colaborarem e se motivarem a trabalhar como equipe e engajarem no trabalho. Auto-gestão, trabalho em equipe e *feedbacks* são exemplos de práticas que auxiliam.

7. **Ver o todo:** Otimizar tudo começa com visualizar o todo, do início ao fim do desenvolvimento do produto, e o que pode ser medido pode ser melhorado. Para isso é essencial mapear fluxos de valor e implementar métricas de tarefas, processos e interações.

4.6 Processo Unificado Aberto (OpenUP/Basic)

O OpenUP/Basic aplica uma abordagem iterativa e incremental dentro de um ciclo de vida estruturado e tem uma filosofia pragmática e ágil que foca na natureza colaborativa do desenvolvimento de software. É baseado no processo unificado, considerado mínimo, completo e extensível, além de ser caracterizado por valorizar a colaboração entre a equipe e os benefícios aos interessados (SENE, 2018).

Como consta em Sene (2018), o OpenUp/Basic conta com três camadas que interagem entre si, sendo elas:

1. **Micro-incremento:** representam pequenas unidades de trabalho que produzem um passo do progresso do projeto, constante e mensurável;
2. **Ciclo de vida de iteração:** que estrutura como os micro-incrementos são aplicados para entregar partes (resultados) estáveis e coesos do sistema;
3. **Ciclo de vida do projeto:** que é dividido em quatro fases sendo a concepção, elaboração, construção e transição.

Cunha (2007) mostra que o OpenUp/Basic possui quatro princípios que são citados e descritos a seguir:

1. **Colaboração:** Pessoas com diferentes habilidades trabalham em conjunto para construir o produto final de acordo com o especificado.
2. **Equilíbrio:** Membros de projetos devem trabalhar para que o produto desenvolvido traga benefícios aos *stakeholders* da melhor maneira diante das adversidades previstas no projeto.
3. **Foco:** O desenvolvimento focado na arquitetura permitirá que desenvolvedores realizem melhores decisões técnicas garantindo uma evolução eficiente do sistema.
4. **Evolução:** É necessário dividir o projeto em pequenas iterações para demonstrar agregação de valor incremental entre os ciclos e obter constantes *feedbacks* por parte dos *stackholders.*

A equipe de desenvolvimento é composta por papéis como o gerente de projeto, arquiteto, analista, desenvolvedor, testador e *any role* (qualquer papel) (DOURADO, 2018).

Como é mostrado em Cunha (2007) as disciplinas contidas OpenUp/Basic são estas descritas a seguir:

1. **Requisitos:** pode ser definida como a responsável por elicitar, analisar, especificar, validar e gerenciar os requisitos referentes ao sistema.
2. **Análise e Projeto:** auxilia na criação do *design*, futuramente implementado por desenvolvedores a partir dos requisitos do sistema. Tem como principais objetivos transformar requisitos em *design* do futuro sistema, desenvolver uma arquitetura robusta e adaptar ao *design* as características do ambiente de desenvolvimento.
3. **Gerenciamento de Projeto:** tem como objetivos principais manter o foco da equipe no desenvolvimento e entrega para avaliação dos *stackholders*, criar um ambiente de alta produtividade, manter *stackholders* e a equipe informados sobre o andamento do projeto, prover *framework* para acompanhamento dos riscos e adaptação do projeto aos mesmos e priorizar a sequência de trabalhos.
4. **Gerenciamento de Configuração e Mudanças:** Tem como objetivos manter um conjunto consistente dos produtos dos trabalhos realizados através de sua evolução, manter *builds* do software, prover maneiras eficientes para adaptação de mudanças, e fornecer dados para avaliação de progresso.
5. **Desenvolvimento:** relacionada ao desenvolvimento das iterações, onde são feitos e executados os testes de desenvolvedor e a implementação do produto final, sendo gerados *builds* estáveis em cada ciclo.
6. **Testes:** define conjunto de atividades necessárias para planejamento, implementação, execução e avaliação de testes do sistema. Essa disciplina desafia as proposições, riscos e incertezas provenientes do trabalho das outras disciplinas utilizando demonstrações concretas e imparciais.

5 PROCEDIMENTOS METODOLÓGICOS

Os procedimentos metodológicos adotados para o desenvolvimento dessa pesquisa se constituem nas seguintes etapas: Pesquisa de coleta de dados com membros de equipes de desenvolvimento de *software*; Pesquisa documental centrada na busca por aspectos das metodologias ágeis (SCRUM, XP, FDD, LEAN E OPENUP/BASIC); Elaboração da proposta do modelo ágil; Aplicação de método para validação. Nas subseções deste capítulo estas etapas citadas são explanadas.

5.1 Pesquisa de Coleta de Dados

Esta atividade teve por objetivo obter informações relevantes sobre possíveis dificuldades em processos de desenvolvimento de *software*. Para essa pesquisa foi elaborado e aplicado um questionário (Apêndice A) para membros (desenvolvedores, analistas, testadores, arquitetos, etc) de equipes de projetos de desenvolvimento de *software*. Onde foi coletado relatos de dificuldades, como falha no entendimento de requisitos; falta de planejamento; falta de documentação; e atrasos em entregas de funcionalidades. Os resultados dessa etapa podem ser vistos na Seção 6.1.

5.2 Pesquisa documental

Essa etapa foi feita através de pesquisa documental de livros, sites e artigos que retratem as características e comparações de metodologias ágeis de desenvolvimento de *software*, neste caso mais especificamente estas: SCRUM, XP, FDD, LEAN E OPENUP/BASIC. Com o objetivo de identificar as práticas dessas metodologias. A fundamentação teórica (Seção 4) foi a base para essa etapa, pois foi feito o uso da mesma fonte (livros, sites artigo, etc) para essa pesquisa documental, que tem a finalidade de fazer uma comparação (Seção 6.2) das práticas pertencentes à essas metodologias.

5.3 Elaboração do Modelo Ágil

Após a análise dos dados obtidos através do questionário e dos documentos utilizados para realizar a pesquisa documental, foi elaborado o modelo ágil baseado nas atividades, princípios, ferramentas e entre outras características presentes nas metodologias ágeis que servirão como base para esta proposta. Além das características já definidas (Práticas de

documentação, modelagem e Gerência de Configuração e Mudanças), as demais foram escolhidas com base nos relatos coletados através do questionário e nas semelhanças de boas práticas (Seção 6.2). O resultado dessa etapa, ou melhor, o modelo proposto pode ser observado na Subseção 6.3.

5.4 Aplicação de método para validação

Logo em seguida foi feito a implantação das práticas em um projeto de desenvolvimento de *software*, com o intuito de fazer uma verificação da satisfação do modelo. Sendo assim, foi realizado um estudo de caso e entrevistas com os participantes. Após, os dados obtidos foram analisados com o intuito de verificar a aderência do modelo.

5.5 Análise dos resultados

Foi realizada uma análise de artefatos, neste caso, número de tarefas finalizadas ou pendentes, problemas das *Sprints* anteriores à implantação do modelo e também das posteriores. Além disso, os relatos dos participantes do estudo de caso durante uma entrevista foram também analisados para obter as conclusões deste trabalho.

6 PROPOSTA DE MODELO PARA DESENVOLVIMENTO DE SOFTWARE

Este capítulo descreve como foi a execução das etapas dos procedimentos metodológicos para propor o modelo de desenvolvimento de *software*. São apresentados os resultados da pesquisa de coleta de dados, comparações entre as metodologias e o modelo elaborado.

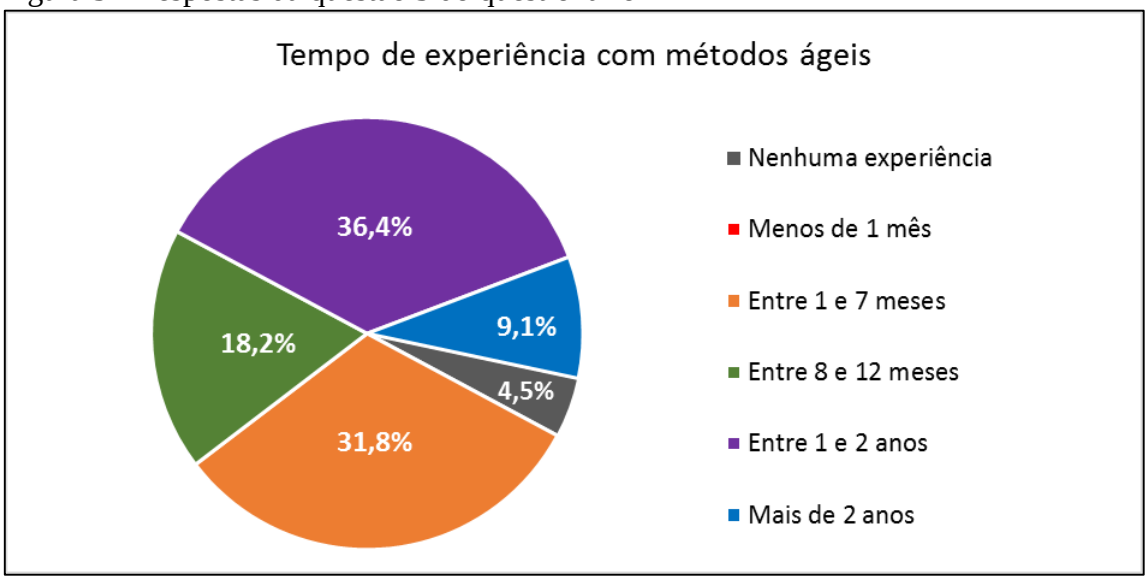
6.1 Resultados da Pesquisa de Coleta de Dados

Foi aplicado um questionário online (Apêndice A) utilizando a ferramenta Formulário do Google, com início no dia 28 de agosto de 2018 e término em 23 de setembro de 2018, para quem participa ou já participou de projetos de desenvolvimento de *software*. Os participantes foram membros de empresas (não identificadas por questão de privacidade) de desenvolvimento de *software* e estagiários do Núcleo de Soluções em Software (Seção 7.1) dos semestres antecedentes ao segundo semestre de 2018.

O número de respostas obtidas foram 22, respondidas por perfis de analistas de sistemas, arquitetos de softwares, desenvolvedores e PO. Treze desses respondentes assinalaram que a equipe que participam possui no máximo quatro membros, outros quatro dos respondentes em equipes com cinco ou até oito membros. Dois dos participantes da pesquisa por meio do questionário responderam que participam de equipes com nove ou mais membros. E três dos respondentes informaram que não estão em projetos de desenvolvimento de software. Estas informações foram obtidas através das respostas da primeira e segunda questão do questionário. Todas as respostas obtidas são mostradas no Apêndice B.

A terceira questão solicitava que o respondente informasse qual o tempo estimado de experiência em projetos com métodos ágeis. Como mostra a Figura 3, é possível perceber que a maioria tem alguma experiência, e que somente um assinalou não ter nenhuma.

Figura 3 - Respostas da questão 3 do questionário

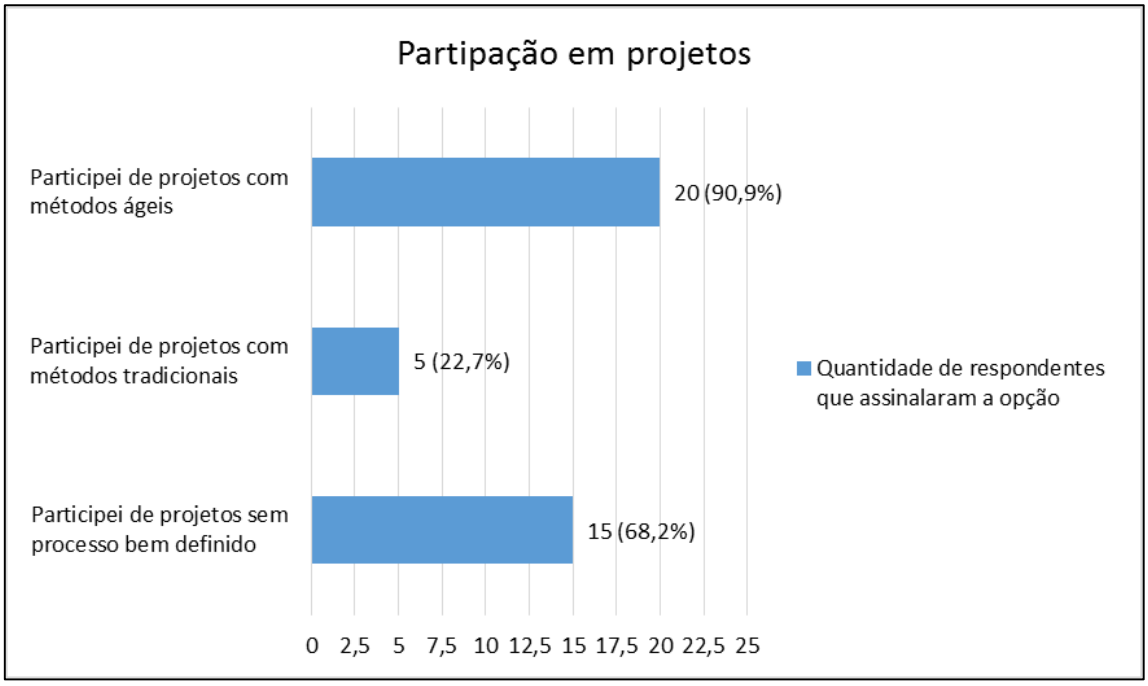


Fonte: Elaborada pela autora.

Com as respostas da quarta questão foi possível obter o nível de facilidade/dificuldade no aprendizado das práticas dos métodos ágeis utilizados pelos participantes dessa pesquisa, onde tiveram que selecionar um número da escala de 1 (muita dificuldade) a 5 (facilidade em aprender). Foi perceptível que o nível 2 foi assinalado somente uma vez, ou seja, somente um respondente atribuiu 2 ao seu nível de aprendizado. E que nove respondentes escolheram 5 como nível de aprendizado. As demais atribuições podem ser vistas no Apêndice B. E na quinta questão, foi solicitado expor quais foram as dificuldades em relação ao aprendizado em métodos ágeis, seja teórico e/ou prático, essas foram: Decidir quais requisitos implementar na *Sprint*, implantar algumas práticas no dia a dia e conseguir utilizar os artefatos do *Scrum* em conjunto.

Na sexta questão foi solicitado que assinalasse se já participou em projetos sem processo definido, com métodos tradicionais e/ou ágeis. Através da Figura 4, é perceptível que a maioria já participou de projetos com o uso de métodos ágeis, e que boa parte também já teve participação em projetos com processos não definidos.

Figura 4 - Respostas da questão 6 do questionário

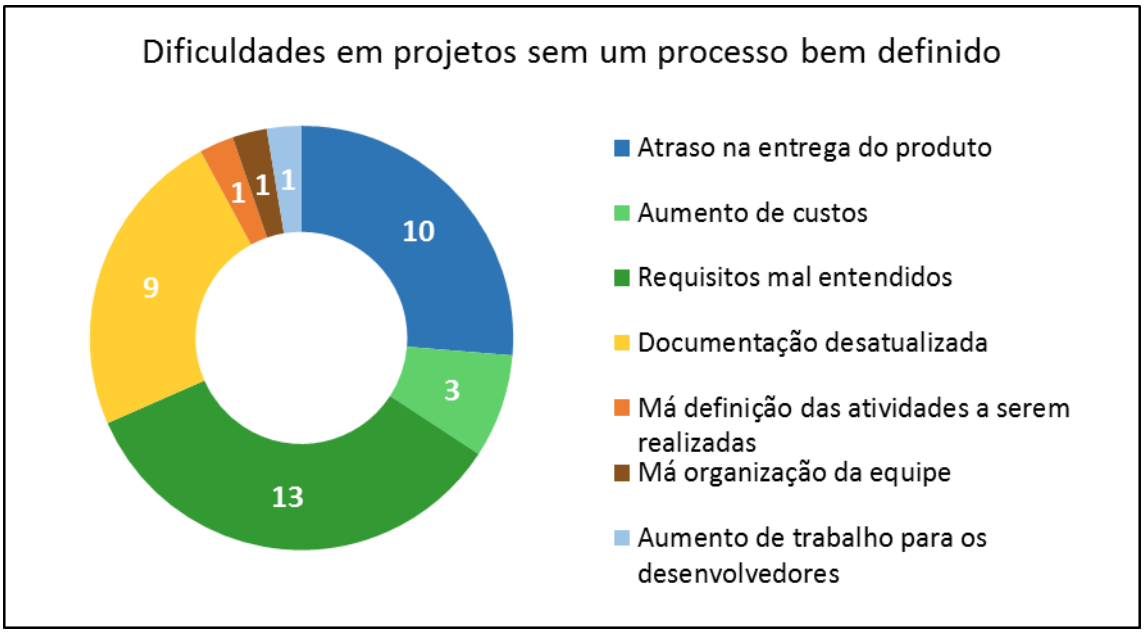


Fonte: Elaborada pela autora.

A sétima, oitava e nona questões foram elaboradas com o intuito de identificar quais foram as dificuldades enfrentadas em cada projeto que o respondente já participou. Mais precisamente a sétima é em relação as dificuldades em projetos sem processos definidos, a oitava questão sobre métodos tradicionais e a nona sobre metodologias ágeis.

A Figura 5 mostra as respostas obtidas na sétima questão, onde percebe-se que as opções mais assinaladas foram “Requisitos mal entendidos”, “Atraso na entrega do produto” e “Documentação desatualizada”.

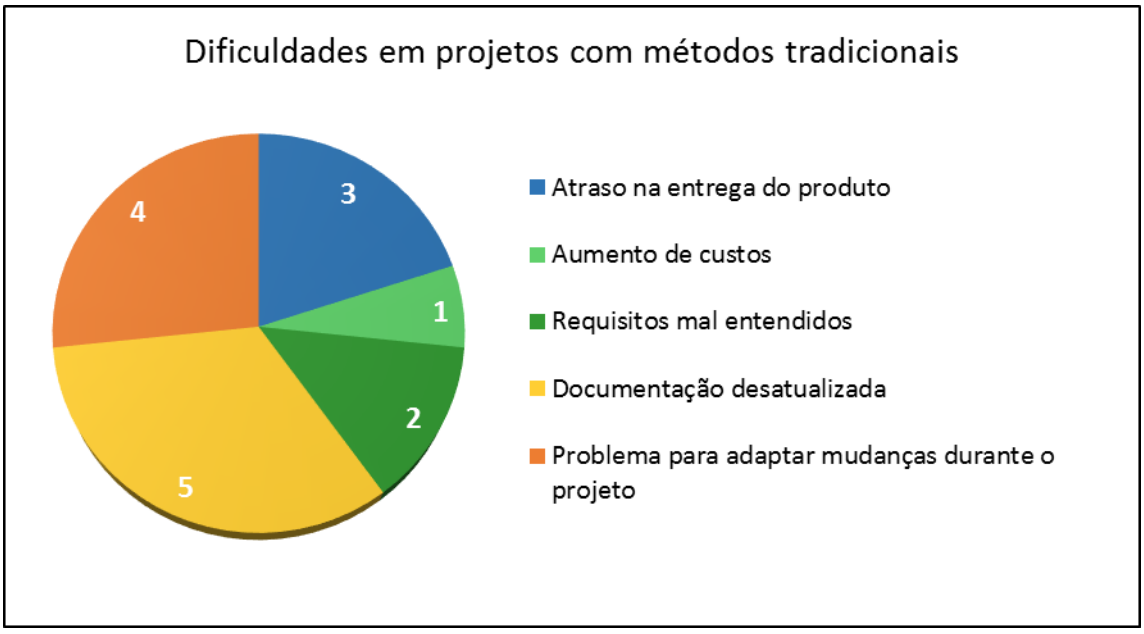
Figura 5 - Respostas da questão 7 do questionário



Fonte: Elaborada pela autora.

A Figura 6 mostra as respostas obtidas na oitava questão, onde é possível perceber que as opções mais assinaladas foram “Problemas para adaptar mudanças durante o projeto” e “Documentação desatualizada”.

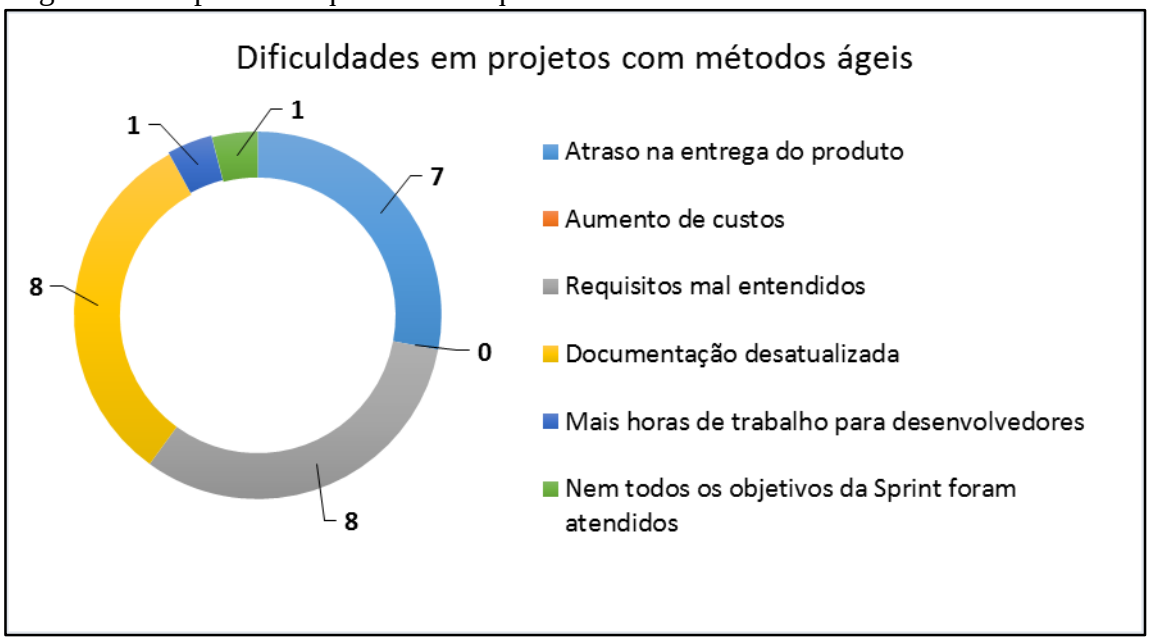
Figura 6 - Respostas da questão 8 do questionário



Fonte: Elaborada pela autora.

A Figura 7 mostra as respostas obtidas na nona questão, onde é possível perceber que as opções mais assinaladas foram “Requisitos mal entendidos”, “Documentação desatualizada” e “Atraso na entrega do produto”.

Figura 7 - Respostas da questão 9 do questionário



Fonte: Elaborada pela autora.

O quadro 1 mostra as maiores dificuldades referentes ao processo de aprendizagem de metodologias ágeis, relatadas pelos respondentes na quinta questão. Pode-se perceber, que as dificuldades estão mais relacionadas à pouca experiência ou conhecimento das práticas utilizadas no modelo de desenvolvimento.

Quadro 1 - Respostas da questão 5 do questionário

Falando especificamente do SCRUM, as dificuldades foram em relação as escolhas de quais estórias ou requisitos implementar durante uma Sprint. Pode parecer estranho, mas é um pouco difícil deixar nas mãos da equipe de desenvolvimento quais atividades cada programador vai realizar. Na minha opinião, isso pode tornar o processo de desenvolvimento um pouco sem rumo quando os membros da equipe escolhem as atividades que serão realizadas sem seguir uma forma lógica, em que uma atividade precisa ser feita antes da outra, ou coisas assim.
As poucas dificuldades, foram implantar algumas poucas práticas no dia a dia, porém foi superado com pouco tempo de prática.
Encaixar todos os artefatos do SCRUM.

Fonte: Elaborada pela autora.

O Quadro 2 mostra opiniões sobre possíveis melhorias em métodos ágeis, apontadas pelos respondentes na décima questão. É perceptível que os pontos citados estão relacionados à documentação, gerenciamento do entendimento dos requisitos do sistema e planejamento das atividades a serem feitas.

Quadro 2 - Respostas da questão 10 do questionário

A pouca documentação pode dificultar em manutenções. Ágil + Prescrição uma boa saída.
Se posso apontar algo a melhorar nas metodologias seria em relação a documentação. Acredito que, pelo fato das metodologias valorizarem mais o *software* funcionando do que documentação, as pessoas as adotam pensando que não vão precisar documentar nada sobre o produto desenvolvido. Acho que deveria haver algo que reforçasse o uso e a necessidade da documentação ou até a descrição de alguns tipos de documentos que podem ser utilizados em processos ágeis.
Um bom gerenciamento e controle sobre a definição do sistema e seus requisitos.
Dar uma dica sobre como iniciar o projeto.
Modelagem com diagramas para facilitar o entendimento.
Um documento de requisitos mais simples, para facilitar o entendimento, e um mapeamento do mesmo para as funcionalidades.
Focar um pouco mais na documentação.
Questão de prazo como trabalhamos com pequenas entregas temos a impressão que nunca paramos de trabalhar, e esse círculo sempre tá retornando e nunca vamos ficar ociosos.
Planejar bem o que deve ser entregue em cada ciclo.
Reuso deveria ser uma base para a utilização de métodos ágeis.
Vejo que um pouco mais de tempo no entendimento do produtos seus requisitos e do domínio teria ajudado bastante algumas etapas seguintes do desenvolvimento.
Uma melhor gestão da documentação do projeto em questão e também a melhor gerência das atividades a serem feitas e por quem essas atividades são feitas.

Fonte: Elaborada pela autora.

6.2 Comparativo de Metodologias

Com a finalidade de identificar semelhanças entre as metodologias utilizadas para formular o modelo proposto neste trabalho, foi realizado um estudo comparativo entre elas. As fontes (livros, sites e outros trabalhos) que serviram como base para a realização desse estudo, são as mesmas presentes na Fundamentação Teórica (Seção 4), onde estão contidas informações sobre cada metodologia. Para facilitar o entendimento da análise comparativa, atividades do desenvolvimento de *software* e práticas correspondentes às metodologias (SCRUM, XP, FDD, LSD e OpenUP/BASIC) estão elencados na Tabela 2. Onde os campos da tabela estão preenchidos com o nome especifico da prática ou com apenas um X quando a metodologia

possui a determinada prática, e os demais estão vazios por não ter sido identificado durante a análise de documentos a prática na referente metodologia.

Tabela 2 - Comparativo de metodologias

Metodologia/Prática	**Scrum**	**XP**	**FDD**	**LSD**	**OpenUP**
Desenvolvimento iterativo	*Sprint*	*Releases* pequenos	Iteração	Entregar rápido	Ciclo de vida da iteração
Reunião diária	*Scrum Daily Meeting*	*Daily Stand-up Meeting*		Reuniões diárias (sinalização local)	
Lista de funcionalidades	*Product Backlog*	Estórias de usuário	Lista de funcionalidades		Requisitos do sistema
Papéis	*Scrum Master, Product Owner, Scrum Team*	*Coach*, programador, *tracker*, testador, cliente e consultor	Gerente de projeto, especialista do negócio, arquiteto chefe, gerente de desenvolvimento, programadores chefes, programadores (*class owner*), testadores		Stakeholders, Gerente de Projeto, Arquiteto, Analista, Desenvolvedor, Testador e Any Role (Qualquer Papel)
Lista de tarefas da iteração	*Sprint Backlog*		Pacote de funcionalidades		Micro-incrementos da iteração
Acompanhamento do progresso das iterações	*Scrum Board*	Ambiente informativo	Relatórios de progresso	*Kanban*	X
Reunião para definir tarefas da iteração	*Sprint Planning Meeting*	Jogo do planejamento			
Reunião de revisão da iteração	Sprint Retrospective				
Testes		X	X	Testes automatizados	X
Integrar os trabalhos feitos		Integração contínua	Promover a *build*	Integração contínua	X
Propriedade do código		Propriedade coletiva	Classes proprietárias		

Refatoração		X		X	
Entregas frequentes	X	X	X	X	X
Implementação da funcionalidade	X	X	X	X	X
Código padrão		X		X	
Cliente presente	X	X	X	X	X

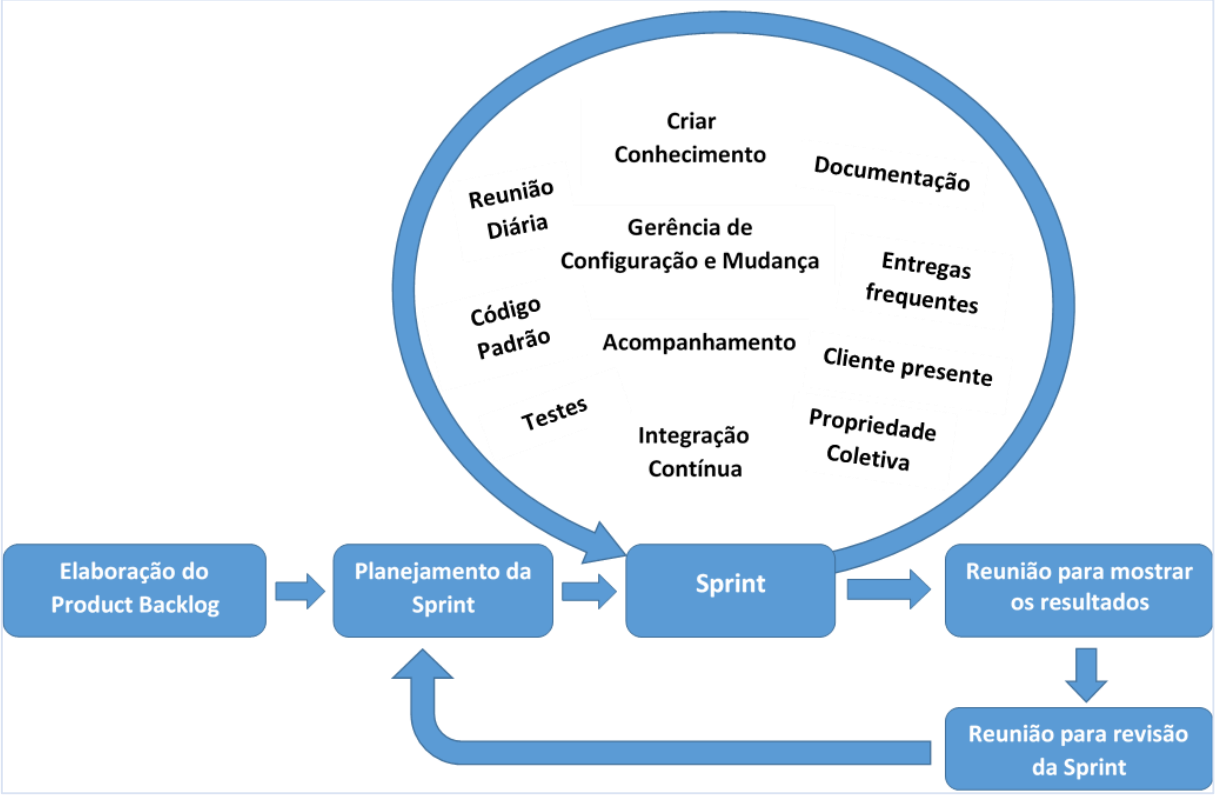
Fonte: Elaborada pela autora

6.3 O Modelo Proposto

A partir dos dados coletados nas etapas anteriores, respostas do questionário e a comparação das práticas das metodologias, foi realizada a combinação de boas práticas para formar um modelo de processo de desenvolvimento. As práticas Gerência de Configuração e Mudanças (GCM) e a relacionada a documentação (Análise e Projeto de Sistemas) são as pré-definidas para constituir o modelo, conforme especificado na introdução deste documento (Seção 1).

As práticas do *Scrum* serão utilizadas por atender as dificuldades relacionadas a planejamento citadas anteriormente na Seção 6.1, pois é focada no gerenciamento de projetos. Essas práticas e as demais selecionadas para formular o modelo são apresentadas na Figura 8 e descritas a seguir.

Figura 8 - Visão Geral do Modelo



Fonte: Elaborada pela autora.

Papéis: *Scrum Master*; *Product Owner* (PO); Consultor; *Scrum Team*, formada por: Desenvolvedor, Monitor (*tracker*), Testador e Analista.

Elaboração do *Product Backlog* (*Scrum*): Uma reunião inicial com o cliente, de responsabilidade do PO ou analista, deve acontecer, com o objetivo de esclarecer as funcionalidades do sistema para formular o *Product Backlog*. Nessa também é estabelecido pelo cliente a prioridade de cada uma. Após esse encontro, o PO se reuni com o time de desenvolvimento (*Scrum Team*) para apresentar e discutir os requisitos extraídos da reunião inicial. Como resultado é elaborada uma lista contendo as funcionalidades do sistema e as suas prioridades. É interessante adicionar uma prática advinda do XP:

- **Estórias de Usuário (XP):** O analista, com auxílio do PO, implementa a lista de funcionalidades adicionando as descrições de cada uma, em forma de estórias de usuário.

Sprint (*Scrum*): É o período de tempo definido para executar um conjunto de atividades. Deve possuir uma duração máxima de quatro semanas. Ao final desse período, uma parte do produto deve ser entregue como resultado.

Planejamento da *Sprint* (*Scrum*): Com o *Product Backlog* já contendo as funcionalidades (Estórias de Usuário) a serem implementadas, o *Scrum Team* realiza uma reunião para definir as tarefas a serem executadas durante uma Sprint. A prioridade das Estórias de Usuário é o critério principal para essa escolha. O resultado desse planejamento, é uma lista com as atividades (*Sprint Backlog*) a serem feitas durante uma determinada *Sprint*. Foi utilizado em conjunto com a seguinte técnica:

- ***Planning Poker* (*Scrum*):** Utilizada para estimar a duração (em horas) de cada tarefa do *Sprint Backlog*. Os membros da *Scrum Team* que executam as tarefas, fazem a estimativa de cada tarefa individualmente.

Reunião Diária (*Scrum*): Ao longo da *Sprint* é realizada uma reunião a cada dia, antes de iniciar o desenvolvimento das atividades, entre os integrantes da equipe. Cujo o objetivo é a sincronização do conhecimento, através de três perguntas básicas que deve ser respondida pelos participantes: O que fiz desde a última reunião? O que farei até a próxima reunião? Existe algum impedimento?

Reunião para mostrar os resultados (*Scrum*): Ao final da *Sprint*, os resultados (funcionalidades construídas e/ou seus artefatos) são apresentados e discutidos. Onde o cliente pode demonstrar sua satisfação e o que precisa ser modificado nas funcionalidades entregue.

Reunião para revisão da *Sprint* (*Scrum*): Após o final de uma *Sprint*, e antes de iniciar uma nova, a *Scrum Team* deve realizar uma reunião com o objetivo de analisar e refletir os resultados. Ou seja, o que deu certo e errado durante o desenvolvimento das tarefas do *Sprint Backlog*, o que precisa ser melhorado.

Além dessas práticas descritas anteriormente, fez-se necessário selecionar outras com características mais voltadas a parte técnica e que estão presentes durante o desenvolvimento da Sprint. Essas são detalhadas a seguir:

Gerenciar documentação do projeto (OpenUP/Basic): Elaborar/Atualizar documentações como documento de requisitos, manuais do usuário e do desenvolvedor. E também construir artefatos como modelagem de domínio (diagrama de classes) e casos de uso.

Gerência de configuração e mudança (OpenUP/Basic): Conter um sistema para gerenciar as alterações nos artefatos é preciso, uma vez que estas são inevitáveis. Cuja a finalidade é de assegurar uma evolução sincronizada do conjunto de itens produzidos pela equipe.

Testes (XP, FDD, *Lean* e OpenUP/Basic): A equipe (desenvolvedores e testadores) projeta e executa uma série de testes a fim de descobrir possíveis erros e para assegurar que o código-fonte se ajuste aos requisitos.

Acompanhamento do progresso das iterações (*Scrum* e LSD): Através de um quadro de tarefas é feito o monitoramento do progresso da *Sprint*. Onde se tem a divisão das tarefas a serem feitas, o que está sendo feito e o que já está feito.

Criar Conhecimento (LSD): Quando o ambiente de desenvolvimento dispõe de um banco de lições aprendidas é possível reduzir o tempo em que uma tarefa é executada. Pois, caso alguém se depare com um problema já adicionado em um banco de soluções, provavelmente o tempo de correção diminuirá. Ou então, até mesmo *templates* para construir determinados artefatos (documento de requisitos, manuais, telas, entre outros) auxiliam na agilidade para executar atividades.

Propriedade coletiva (XP): O que é produzido por um implementador não é sua exclusividade, ou seja, qualquer outro membro pode ter acesso e modificá-lo.

Integração contínua (XP e LSD): A ideia é que os desenvolvedores realizem a sincronização do código fonte produzido, para garantir que tudo esteja funcionando corretamente em conjunto.

Cliente presente (*Scrum*, XP, FDD, *Lean* e OpenUP/Basic): Quando o cliente está próximo do desenvolvimento do *software* gera uma certeza de que a execução do projeto está no caminho certo, pois é possível ter uma maior colaboração do cliente.

Entregas frequentes (*Scrum*, XP, FDD, *Lean* e OpenUP/Basic): Com a finalidade de obter um *feedback* do cliente, através da entrega de pequenas versões do produto.

Código padrão (XP): Através da padronização da arquitetura do código, é mais fácil o compartilhamento e entendimento do código fonte entre todos os membros presentes no desenvolvimento do sistema.

7 ESTUDO DE CASO PARA AVALIAR O MODELO PROPOSTO

Neste capítulo, é apresentado o estudo de caso deste trabalho, no qual foi feita uma avaliação do modelo proposto, ou seja, para verificar se o mesmo traria benefícios para o processo de desenvolvimento de *software*. A implantação foi realizada no Núcleo de Soluções em Software (N2S), mediante autorização do responsável pelo Núcleo (Apêndices C e D). Feita através da aplicação das práticas em uma equipe composta por um time de desenvolvimento e outro de Análise e Projeto de Sistemas. Os membros da equipe trabalham em um projeto que é continuidade da construção de um *software*, que teve início no primeiro semestre de 2018. A implantação do modelo foi iniciada a partir da terceira *Sprint* do segundo semestre de 2018. A seguir, são apresentadas as características do N2S (Seção 7.1), como foi a apresentação do modelo aos participantes (7.2) e também como foi feita a aplicação das práticas (Seção 7.3).

7.1 Núcleo de Soluções em Software

O estudo de caso foi realizado no Núcleo de Soluções em Software (N2S), este foi criado com o objetivo de atender a comunidade acadêmica da UFC em Russas e a comunidade russana com soluções de TI. O Núcleo fica localizado na Universidade Federal do Ceará/Campus Russas. E atualmente possui uma coordenação que é formada por dois professores, sendo um coordenador e o outro vice coordenador, e por um servidor que é supervisor de projetos. As atividades do N2S iniciaram em 2017 possuindo, além da coordenação, apenas quatro alunos de graduação da UFC – Campus Russas. Esses eram participantes de projetos de extensão e BIA (Bolsa de Iniciação Acadêmica), que realizaram projetos juntamente com os docentes e servidores envolvidos. O núcleo fornece a oportunidade de estágio para discentes dos cursos de Ciência da Computação e Engenharia de Software da UFC - Campus Russas. Tendo como missão garantir a excelência no desenvolvimento de *software*, contribuindo para a formação profissional e para o crescimento tecnológico do Vale do Jaguaribe. E a sua visão é ser uma referência em excelência no desenvolvimento de produtos de *software* por meio da integração do ensino e prática profissional (N2S, 2018).

Atualmente, no segundo semestre de 2018, as equipes de estagiários são formadas por treze discentes, onde dez estão divididos em quatro times de desenvolvimento, e os outros três estão em dois times de Análise e Projeto de Sistema. Essas equipes estão alocadas em quatro projetos de desenvolvimento de software.

7.2 Apresentação do Modelo aos Participantes

Como um dos primeiros passos para o estudo de caso, foi realizada uma reunião com os participantes (*Scrum Master*, desenvolvedores e analistas) da pesquisa. Onde a autora deste trabalho, em uma apresentação mais informal, mostrou as características do modelo e explicou como seria a aplicação das suas práticas durante o desenvolver do projeto do *software*. Após isso, eles afirmaram sua aceitação em participar da pesquisa, por meio de assinatura de um termo de consentimento (Apêndice E). Outra reunião de apresentação do modelo, no mesmo molde da primeira, foi realizada para o PO do projeto. Vale relatar que o total de participantes são sete, onde são três desenvolvedores, dois analistas, um *Scrum Master* e um PO.

7.3 Aplicação das práticas no N2S

Antes da realização desta pesquisa, o processo de desenvolvimento de *software* no N2S já contava com a utilização de práticas em comum as que são pertencentes ao modelo proposto neste trabalho, sendo a maioria delas advindas do *Scrum*, como reuniões diárias, *Sprint* (duração de duas semanas), planejamento da *Sprint*, onde eram apenas selecionadas as atividades a fazer e atribuição dos responsáveis por executá-las. Além das práticas de integração contínua, utilização de ferramenta para gerenciar mudanças, código com propriedade coletiva, padronização de código e testes. Outro ponto importante em mencionar é que o *Product Backlog* já estava feito, pois como citado anteriormente o projeto é uma continuidade do desenvolvimento de um *software* iniciado anteriormente.

E como relatado na seção anterior, os papéis foram *Scrum Master*, PO e *Scrum Team* (desenvolvedores e Analistas). Por esse motivo a divisão especificamente dos papéis *Scrum Master*, PO, desenvolvedor e analista se mantiveram, e todos os membros participantes da pesquisa exerciam o papel de consultor, com a percepção que os analistas e o PO podem sanar dúvidas principalmente em relação aos requisitos do sistema, o *Scrum Master* auxiliar no processo como um todo. Já os papéis referentes ao *Scrum Team*, foram distribuídos entre os membros. Logo adiante pode ser visto os papéis de cada um dos sete participantes deste estudo de caso:

1. **Participante 1** – *Scrum Master* e consultor;
2. **Participante 2** – PO e consultor;
3. **Participante 3** – desenvolvedor, testador e consultor;
4. **Participante 4** – desenvolvedor, testador e consultor;

5. **Participante 5** – analista, testador e monitor;
6. **Participante 6** – analista, testador e monitor;
7. **Participante 7** – desenvolvedor, testador e consultor.

Logo após a apresentação do modelo aos envolvidos, realizada em conjunto com a reunião diária, foi feita a retrospectiva da *Sprint* anterior e, em seguida, o planejamento de uma nova *Sprint*. Nessa última prática citada, *Scrum Team* selecionou as tarefas para compor o *Sprint Backlog*, levando em consideração as tarefas pendentes da *Sprint* antecedente. Em seguida, foi feito o uso da técnica *Planning Poker* para estimar a duração de cada tarefa realizada por desenvolvedores do *Sprint Backlog*, antes a autora deste trabalho explicou para todos como funciona essa técnica. E após isso, os analistas foram os facilitadores com a responsabilidade de informar qual a tarefa que estava sendo estimada no dado momento e anotar a estimativa dada por cada participante. Inicialmente um breve resumo da tarefa era dado por algum dos desenvolvedores, depois cada um anotava em um papel a estimativa (notas) em horas, mínimo meia-hora e máximo vinte (máximo de horas de trabalho de cada membro da equipe em uma *Sprint*). A pontuação foi formada pela média das notas dadas por cada um, porém, em alguns casos que o intervalo entre as três foram muito distantes, cada membro teve que argumentar o motivo de ter dado a devida nota e uma nova rodada era feita. Esses passos eram executados individualmente para cada tarefa planejada para a *Sprint*.

No horário inicial do trabalho da equipe foi realizado as reuniões diárias, onde membros da *Scrum Team* juntamente com o *Scrum Master* se fazem presentes. Por questão de indisponibilidade de horário, para o PO, nos dias determinados para acontecer a reunião pra mostrar os resultados da *Sprint*, esta prática e a de entrega frequente não foram devidamente aplicadas, pois o PO não pôde estar presente, porém os resultados foram mostrados através do quadro de tarefas e da ferramenta de Gerência de Configuração e Mudanças (GCM), os quais o PO tem acesso. Além disso, os desenvolvedores, analistas e *Scrum Time* podem notar e ver resultados durante as reuniões e interações entre si. Levando em consideração que PO trabalha na mesma localização do N2S, ou melhor, na própria UFC – Campus Russas, assim o cliente estava sempre presente representado pelo PO.

A equipe realizava de forma contínua a integração dos itens de trabalhos, utilizando uma ferramenta para gerenciar as mudanças no código, todos os desenvolvedores no final do dia de trabalho adicionavam a uma plataforma online as mudanças que executaram no código fonte do projeto. A partir disso, qualquer membro podia implementar alterações no código de forma organizada sem afetar o trabalho dos demais.

A utilização de uma ferramenta online para *kanban* permitiu o acompanhamento do progresso das atividades planejadas para a *Sprint* e a visualização do fluxo de trabalho, onde era responsabilidades dos membros com papel de monitor (*tracker*) manter os quadros atualizados refletindo o que foi feito na iteração.

Os documentos gerados foram: documento com uma listagem dos requisitos, estórias de usuário, uma breve descrição do sistema, casos de uso, diagrama de casos de uso, diagrama de classes e diagramas de atividades, além do manual de usuário.

Os testes realizados até a quinta *Sprint* foram de natureza exploratória, e executados por desenvolvedores e consequentemente testadores, a medida que iam implementando as funcionalidades. Testes manuais baseado nos critérios de aceitação das estórias de usuário foram planejados para serem realizados por todos os testadores quando o *Product Backlog* estiver todo implementado.

O N2S já continha uma base de conhecimento online para registro de lições aprendidas, porém não era dado uma grande ênfase a essa forma de armazenar aprendizados, ou seja, os membros não tinham muita noção da importância em manter a base com soluções de problemas recorrentes, *templates*, manuais, entre outros. Após a implantação do modelo, principalmente os analistas tiveram a responsabilidade de acrescentar mais lições e conhecimentos necessários para auxiliar as próximas equipes que vierem fazer parte do N2S.

8 RESULTADOS

Neste capítulo, são apresentados os resultados obtidos com o estudo de caso, que foi a implantação das práticas do modelo proposto. Estes, foram obtidos por meio de análise de artefatos (quantidade de tarefas concluídas e pendentes) e por *feedback* de participantes através de entrevista.

8.1 Análise de Artefatos

Foi elaborada uma comparação de tarefas de *Sprints* do projeto de desenvolvimento de sistema. As *Sprints* selecionadas foram a partir da terceira até a sexta do primeiro semestre de 2018 (será referenciado como 2018.1), e a primeira até a quinta *Sprint* do segundo semestre (2018.2). Como no semestre de 2018.1 não tinha o papel de analista separado do papel de desenvolvedor, a própria equipe de desenvolvimento era a responsável por fazer a análise de requisitos e a documentação do sistema, e nas duas primeiras *Sprints* foram realizadas somente atividades desse tipo, ou seja, não sendo feita implementação de funcionalidades, pois inicialmente foi feita toda a etapa de levantamento de requisitos e definição do *Product Backlog*. Sendo assim, as tarefas de análise não serão constadas, pra ser uma comparação justa, onde somente as tarefas de desenvolvedores serão levadas em consideração na análise de artefatos.

Para realizar essa comparação foi verificada a quantidade de tarefas nos quadros de cada *Sprint*, e foram elencados na Tabela 3, seguindo esta divisão: número de tarefas que no final da *Sprint* não foram iniciadas (por fazer), de tarefas que ainda estavam sendo feitas (fazendo), de tarefas que estavam em teste (testando) e as concluídas (feito).

Tabela 3 - Comparação de tarefas das Sprints

Sprint	Por fazer	Fazendo	Testando	Feito
3_2018.1	0	1	3	2
4_2018.1	0	0	0	6
5_2018.1	3	1	0	2
6_2018.1	0	2	0	0
1_2018.2	0	1	0	3
2_2018.2	0	2	1	2
3_2018.2	0	3	1	1
4_2018.2	0	2	1	2
5_2018.2	0	0	0	5

Fonte: Elaborada pela autora.

Os resultados obtidos após as *Sprints* de 2018.1 foram: *Product Backlog* definido com estórias de usuário, documento com descrição do sistema, listagem de boa parte dos requisitos funcionais, regras de negócio e diagramas (de casos de uso, de classes, de banco de dados e de estado), detalhamento de casos de uso, criação das tabelas no banco de dados, além da implementação para atender quatro casos de uso do sistema, sendo duas funcionalidades consideradas de complexidade simples e duas com complexidade média.

Já os resultados das *Sprints* de 2018.2 foram: revisão e atualização do documento de requisitos, diagrama de casos de uso e do diagrama de classes atualizados, diagramas de atividades (de casos de uso, de classes, de banco de dados e de estado), manual de usuário do sistema, detalhamento de quatro casos de uso, modificação em tabelas do banco de dados, além da implementação para atender cinco casos de uso do sistema, onde dois com complexidade simples, outros dois com complexidade média e um com complexidade alta.

Por meio de observação a autora deste trabalho identificou algumas dificuldades principais durante o decorrer da realização das atividades das *Sprints*. Essa observação foi possível devido a autora ser integrante da equipe do projeto nos dois semestres, ou melhor, participante desde que iniciou as atividades para desenvolver o sistema referente ao projeto em que foi realizado o estudo de caso.

As dificuldades e problemas observados durante as *Sprints* de 2018.1, foram relacionadas principalmente à gerência de mudanças, pois não foi utilizado pela equipe o uso rígido da plataforma de GCM. Geralmente cada membro da equipe implementava funcionalidades diferentes, e em certos momentos foi feito a integração dos artefatos gerados por cada um. Porém, isso demandou mais tempo do que o necessário, porque tinha que analisar o que cada um alterou e depois sincronizar tudo. Os problemas foram com tecnologias, mais especificamente com o *TomCat* (servidor que permite a execução de aplicações para web), que muitas vezes não executava o código e isso impossibilitava os testes dos desenvolvedores. Outro problema foi que devido a um erro no momento de integrar as alterações na ferramenta de GCM, o sistema parou de executar. E foi preciso a equipe gastar um bom tempo de trabalho para identificar o erro e consertá-lo.

Foi observado durante as *Sprints* de 2018.2, que grande parte dos problemas foram em relação a estrutura de arquivos do projeto (código fonte), pois devido anteriormente não ter sido feito corretamente o gerenciamento dos artefatos e suas alterações, consequentemente a integração destes foi afetada. Pois, no momento de integrar (em 2018.1) os trabalhos de cada um, não foi feito uma verificação para saber se todos os arquivos necessários para executar corretamente o sistema foram adicionados. Em 2018.2 aconteceram alguns erros na tecnologia

utilizada para executar o sistema (*TomCat*), onde esta ferramenta parou de funcionar porque não estava reconhecendo um dos arquivos do projeto. Pois no momento de adicionar na plataforma de GCM, esse arquivo não foi junto aos outros e automaticamente foi criado um novo arquivo com a mesma extensão, porém vazio, ou seja, impossibilitando a sua execução. Isso ocorreu durante uma das *Sprints* e como é algo que não estava planejado, ocasionou atraso porque antes dos desenvolvedores continuarem as alterações no código, foi visto a necessidade da equipe verificar o motivo do problema e solucioná-lo. Outra dificuldade, observada durante o *Planning Poker*, foi que os membros não tinham experiência com estimativas e isso gerou uma certa dificuldade em atribuir estimativas mais próximas do tempo necessário para execução das tarefas. Porém, foi percebido que aos poucos a equipe foi superando essa dificuldade. E também foi visto que isso ajudou a selecionar uma quantidade de tarefas necessárias que davam para ser feitas na *Sprint*, sem precisar ficar modificando o quadro de tarefas após o planejamento. Apesar de algumas não terem sido realizadas devido aos problemas inesperados que aconteceram durante a *Sprint*.

8.2 Entrevista com os participantes

Através de entrevista estruturada, realizada pela autora deste trabalho, com quatros membros da *Scrum Team*, sendo três desenvolvedores e um analista, foi possível obter um *feedback* individual desses participantes. O roteiro dessa entrevista pode ser visto no Apêndice F. Em contrapartida à entrevista, foi solicitado que os participantes respondessem um pequeno questionário, que pode ser visto no Apêndice G. Com os relatos dos entrevistados, foi obtido as informações demonstradas a seguir nas Tabelas 4, 5, 6 e 7. Por questão de privacidade os participantes foram nomeados como entrevistado A, B, C ou D.

Tabela 4 - Relatos do entrevistado A

Questões	Respostas
1	Melhorou o planejamento das *Sprints*, e as estimativas das tarefas foram mais concisa ao passar de cada Sprint.
2	Na organização, estimativas ajudaram a saber o número de tarefas a se colocar na *Sprint*. Antes da implantação, eu ficava ocioso porque termina a tarefa e não tinha mais outras no *Sprint Backlog*, e era adicionado mais tarefas na *Sprint*. Ficava mudando as tarefas no *Trello*, pois o que era planejado no início eram menos tarefas do que realmente foi feito.
3	Não tenho sugestões de melhorias para as práticas

4	Muito útil e divertido. Onde os membros podem dar dica de como fazer uma determinada tarefa. Pois antes de cada um estimar, um dos membros fala o que tem que ser feito e como pretende fazer.
5	Não tive dificuldades devido a implantação.
6	Banco de lições é bom, pois a equipe tem onde procurar problemas/soluções semelhantes que outras pessoas registraram, pois isso diminui o tempo para solucionar.
7	Foi devido a problemas com ferramentas (*TomCat* que ficou umas duas semanas sem funcionar, por causa da estrutura do projeto que estava errada). E também, porque estava com uma tarefa que precisou fazer algo que eu não tinha muita experiência.

Fonte: Elaborada pela autora.

Tabela 5 - Relatos do entrevistado B

Questões	Respostas
1	A mudança visível foi a prática de estimar as tarefas, apesar de ainda não ter tido o resultado satisfatório (acertar o tempo para execução de cada tarefa).
2	Já trabalhava com algo semelhante, mas melhorou o planejamento, ajudou a entender a complexidade das tarefas (funcionalidades) em vez de só quantidades de tarefas nas *Sprints*.
3	Não tenho sugestão.
4	Com a prática é possível ir aprendendo criar a *Sprint Backlog*, em pensar na complexidade em vez de só na quantidade de tarefas. Seria produtivo se fosse usado uma ferramenta (se tivesse uma prototipação de baixa fidelidade na fase de análise) diminuiria a probabilidade de errar a estimativa.
5	Não tive dificuldades devido a implantação.
6	Continuar implementando o banco de lições, pois essa prática impede que uma empesa faça algo que já aprendeu antes, ou seja, que reinvente a roda. É importante documentar aprendizados e conhecimentos. Se não documentar é como se perdesse o que foi feito antes, fica sem um histórico.
7	Problemas externos ao modelo. Relacionados a conexão, extras ao que foi planejado, problemas inesperados.

Fonte: Elaborada pela autora.

Tabela 6 - Relatos do entrevistado C

Questões	Respostas
1	Mudança no entendimento da duração de cada atividade, depois de implantar a prática de estimar a duração, que foi dada uma percepção, durante a Sprint, qual o tempo necessário para realizar determinadas tarefas.
2	Ajudou principalmente na comunicação entre os membros, principalmente por causa das reuniões onde pode-se tirar dúvidas, e ajudar ou ser ajudado por outros membros.
3	Para o *Planning Poker*, seria melhor se tivesse um histórico de equipes passadas informando o tempo que realmente foi gasto em tarefas das *Sprints* passadas, pois poderia ter tarefas semelhantes a que está sendo estimada e isso ajudaria no momento da esquipe estimar. Pois as estimativas dadas a maioria foram erradas, devido à pouca experiência da equipe.
4	A necessidade das cartas é desnecessária, poderia ser apenas uma conversa entre os membros participantes do *Planning Poker*, onde cada um diz o que acha e no final entrariam em consenso em qual seria a estimativa de cada tarefa da *Sprint*.
5	Não tive dificuldades.
6	Precisava ter mais soluções no banco de lições e deve ser continuada a implementação do banco.
7	Os atrasos não foram devido a implantação do modelo e sim à problemas com tecnologias, como o *TomCat*.

Fonte: Elaborada pela autora.

Tabela 7 - Relatos do entrevistado D

Questões	Respostas
1	Noção do tempo para cada tarefa. Maturidade de planejar as atividades.
2	Organização e colaboração na resolução de problemas, onde o membro pode expor os problemas, principalmente nas reuniões, e os outros membros discutiam possíveis soluções.
3	Planejar melhor a forma de testar, quando for testar deve verificar se está com a versão atual do projeto, pois muitas vezes quando ia testar na máquina de outro membro não funcionava por esse motivo.
4	Gostei da ideia de cada um dar um tempo (estimativa), porque fez eu ver estimativas diferentes da minha opinião. A opinião de cada membro ajuda a planejar o tempo das atividades. Demorava um pouco o planejamento porque era meio disperso. Deveria fazer junto a reunião diária, pois todos já estariam discutindo sobre o que irá ser feito.
5	Não tive dificuldades devido a implantação.
6	A ideia é boa, mas muitas vezes os problemas foram individual do projeto. Ajudaria se já tivesse um passo a passo (no banco de lições) que serve de

	modelo padrão de como iniciar o projeto (estrutura do projeto, arquitetura).
7	Atrasos foram devido aos problemas vindos do semestre passado, porque precisou corrigir muitas coisas na estrutura do projeto, que ocasionou os principais problemas nas ferramentas (*TomCat*, conexão com o banco).

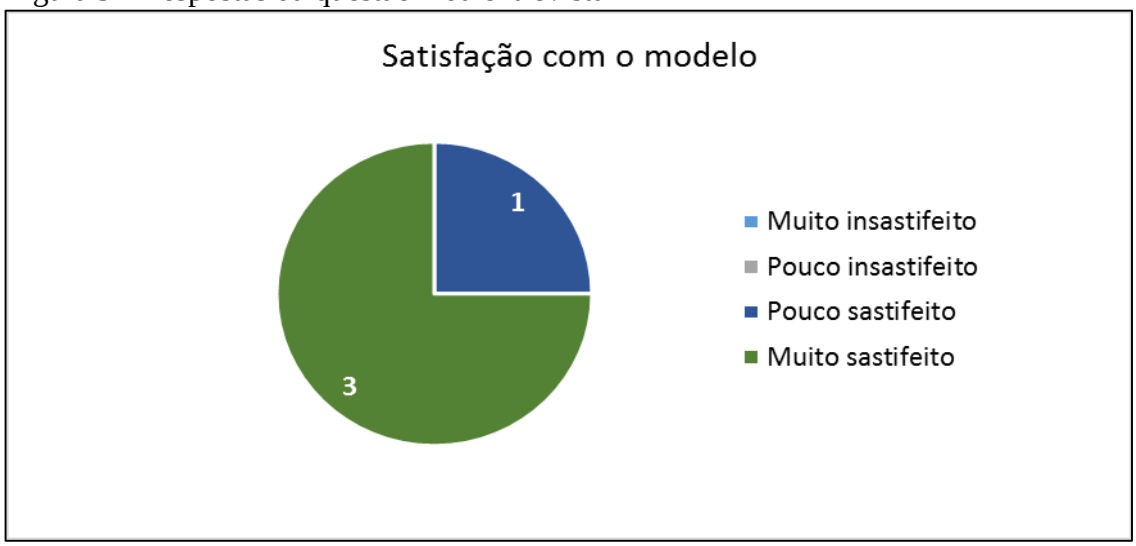
Fonte: Elaborada pela autora.

Após a análise das informações expostas nas Tabelas 4, 5, 6 e 7 que todos apontaram mudanças referentes ao planejamento das *Sprints*. Foi percebido que de um planejamento para outro, teve uma melhoria gradativa em relação a equipe conseguir selecionar uma quantidade de tarefas que realmente foram concluídas até o final da *Sprint*, sem precisar ficar modificando o *Sprint Backlog* (adicionando mais tarefas após o planejamento) e também não colocaram um número excessivo de tarefas. Pois, foi dada uma maior importância em se preocupar com a complexidade das atividades a serem realizadas. Outro resultado relatado foi que melhorou a comunicação entre os membros, onde nas interações entre eles foi possível expor os problemas ocorridos e diante disso soluções eram discutidas. A questão sobre o banco de lições teve respostas semelhantes, mostrando que essa prática foi bem aceita pela equipe, e ainda foi relatado que devem ser adicionadas mais soluções e conhecimentos, pois ainda possui poucas. Já com relação aos atrasos de tarefas foi informado que aconteceram devido a problemas com a estrutura do projeto (código fonte) e funcionamento de tecnologias. E todos afirmaram que os atrasos não foram devido a implantação do modelo.

Para o *Planning Poker*, uma sugestão dada para ajudar no momento de estimar as tarefas é que poderia ter um histórico (de equipes passadas) informando o tempo gasto em cada tarefa das *Sprints* passadas, para servir de modelo. Pois, caso as tarefas sejam semelhantes a que está sendo estimada é possível ter mais noção de quanto tempo é preciso para terminar a tarefa. A segunda sugestão é que deveria ser apenas uma conversa onde cada um diz o que acha, e no final entrariam em consenso para definir qual seria a estimativa de cada tarefa da *Sprint*. Outra sugestão, é fazer as estimativas juntamente com a reunião diária, porque todos já estariam falando sobre as tarefas. Uma sugestão para possivelmente diminuir a probabilidade de errar a estimativa é que poderia ter uma prototipação de baixa fidelidade feita na fase de análise.

Como dito anteriormente no início desta seção, foi aplicado a cada entrevistado um pequeno questionário, logo após responderem a última pergunta da entrevista. As informações obtidas podem ser observadas a seguir.

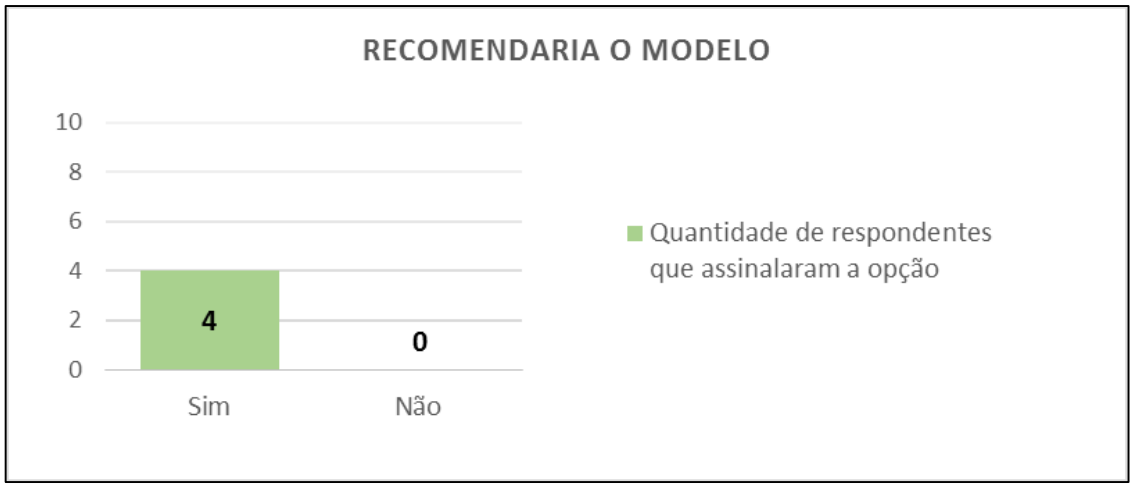
Figura 9 - Respostas da questão 1 da entrevista



Fonte: Elaborada pela autora

Na primeira questão foi solicitado que o entrevistado assinalasse a opção referente a seu nível de satisfação com o modelo. Através da Figura 9, é perceptível que todos estão satisfeitos, sendo que 3 assinalaram muito satisfeito e somente um assinalou pouco satisfeito.

Figura 10 - Respostas da questão 2 da entrevista



Fonte: Elaborada pela autora

A segunda pergunta solicitou que o respondente informasse se faria uma recomendação de uso do modelo proposto neste trabalho e, como mostra a Figura 10, todos assinalaram a opção ‘Sim’.

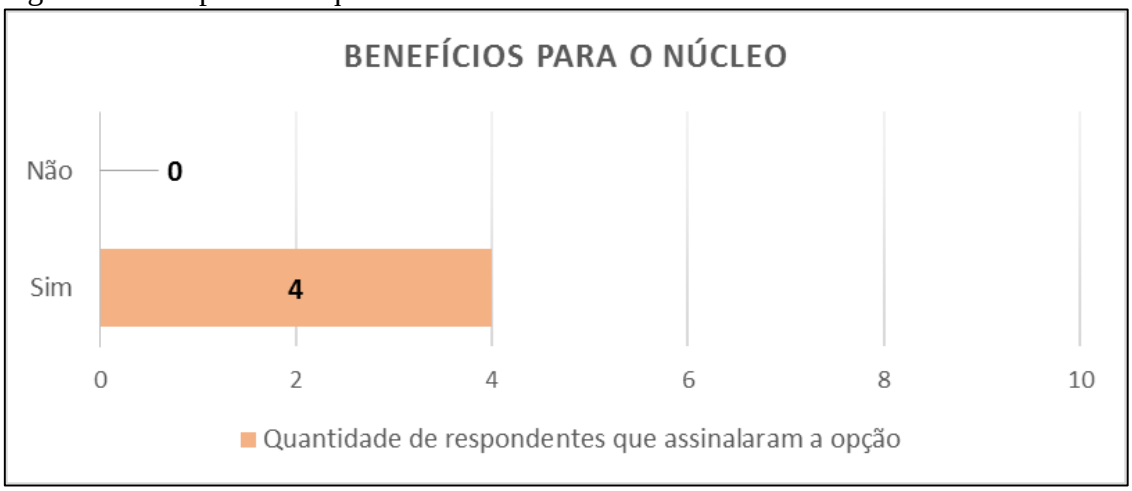
Quadro 3 - Resposta da questão 3 da entrevista

Manter um histórico com estimativas passadas para auxiliar as estimativas futuras.
Creio que não precisa, todas tarefas que foram aplicadas na implementação tiveram sempre bom apoio nas práticas do modelo.
Uma etapa de análise de riscos.
Sim. Tirar um tempo da Sprint para validar o que já foi feito em relação ao que foi planejado, como testes de segurança e de comportamento.

Fonte: Elaborada pela autora

Na terceira questão foi requisitado que fossem informadas práticas que não estão presentes no modelo e que o entrevistado considera que deveria ser adicionadas, como forma de complementar e auxiliar mais o desenvolvimento de software. O quadro 3, mostra que a maioria assinalou sugestão de prática para acrescentar ao modelo, sendo estas: Histórico com estimativas passadas, análise de riscos e validação das tarefas ainda durante a *Sprint*.

Figura 11 - Respostas da questão 4 da entrevista



Fonte: Elaborada pela autora

A quarta questão solicitou que assinalasse se concordava ou não com o seguinte fato: "*A adoção das práticas do modelo no desenvolvimento de software trouxe benefícios (melhor planejamento, mais foco na resolução de problemas futuros com o banco de lições, entre outros) para o N2S*". A partir da Figura 11 é possível observar que todos demonstraram que concordam que trouxe benefícios, os quais foram relatados durante a entrevista.

A última questão do questionário foi o espaço para que os respondentes colocassem, caso tivessem, comentários, sugestões e críticas sobre a realização do estudo de caso e também sobre a aplicação de metodologias ágeis no desenvolvimento de software. A única resposta obtida foi: *"Houve uma certa dispersão por parte da equipe durante a estimativa das atividades*

e acabava demorando um pouco”. Sendo que está informação também foi dita durante a entrevista.

Os problemas/dificuldades apontados, que podem ser extraídos após a análise dos dados da entrevista e também do questionário, foram que a maioria das estimativas feitas pelos desenvolvedores foi diferente do real tempo gasto, devido à pouca experiência da equipe. E o outro relatado foi que demorava um pouco a realização da prática do *Planning Poker*.

9 CONCLUSÕES E TRABALHOS FUTUROS

Este trabalho apresentou um modelo com práticas baseadas nas metodologias *Scrum*, XP, FDD, Lean e OpenUP/Basic. Elaborado com o objetivo de auxiliar nas atividades relacionadas ao desenvolvimento de *software*, apresentando também como foi a elaboração do modelo, a utilização das suas práticas em uma equipe pertencente a um projeto de desenvolvimento de software, e uma avaliação obtida através de estudo de caso.

Apesar de apontadas algumas melhorias ao modelo em relação ao planejamento de atividades de um determinado período, as práticas foram bem aceitas pelos envolvidos na implantação. Mesmo com o fato da equipe já trabalhar com um conjunto de práticas pertencentes ao modelo, foram relatadas mudanças e resultados positivos pelos participantes e pela autora deste trabalho. Entretanto, devido a problemas externos, as *Sprints* tiveram atrasos e o planejado não foi entregue, porém, observado e informado pela equipe, não foi devido à implantação do modelo, e sim à impedimentos referentes a problemas inesperados com funcionamento de tecnologias.

Após o estudo de caso de avaliação do modelo proposto, foi possível perceber que a união de boas práticas de metodologias apresentou bons resultados, partindo também do pressuposto que o modelo abrange tanto a questão gerencial, com as práticas do *Scrum*, quanto o lado técnico, se valendo das práticas do XP, FDD, Lean e OpenUP/Basic. Foram observados como benefícios das práticas um bom planejamento, melhor e comunicação e interação entre os envolvidos principalmente no momento das reuniões, e gradativamente foi percebido um melhor planejamento da *Sprint*, devido ao uso do *Planning Poker* para estimar a duração de tarefas, que ajuda a definir a quantidade de atividades para adicionar na *Sprint Backlog*. E foi relatado também, a importância em registrar mais lições aprendidas com soluções de problemas, *templates* de documentos de requisitos, manuais de usuário, manuais de desenvolvedor com primeiros passos para iniciar o projeto, entre outros. Foi observado, que após a implantação a ideia de colocar um papel de consultor no projeto foi de grande valia, pois os membros sempre que necessário procuravam ajuda de outros integrantes da equipe, seja procurando alternativas de como implementar uma certa funcionalidade ou até mesmo solicitando explicação de requisitos do sistema. Isso colabora para uma melhor interação entre os envolvidos no desenvolvimento do *software*.

Por meio dos resultados obtidos, uma sugestão da autora desse trabalho, e também participante da *Scrum Team*, é adicionar uma lista dos problemas ocorridos durante a Sprint e como foi resolvido no quadro de tarefas, pois deixa mais acessível o momento em que ocorreu

e quem participou da resolução, e facilita para debater os imprevistos durante a revisão da *Sprint*.

Como trabalhos futuros, pretende-se formular um processo que estabeleça o passo a passo a ser seguido para implementar as práticas do modelo proposto neste trabalho, com as melhorias sugeridas incorporadas, e posteriormente implantar em mais equipes de desenvolvimento de *software*.

REFERÊNCIAS

ALMEIDA, Vitor. **FDD:** desenvolvimento guiado por funcionalidades. Disponível em: http://www.itnerante.com.br/profiles/blogs/fdd-desenvolvimento-guiado-por-funcionalidades. Acesso em: 16 nov. 2018.

BARANDAS, Guilherme Mayrink. **Concepção de aplicações de software de Business Analytics com metodologias ágeis** . 2017. 34 p. Trabalho de Conclusão de Curso (Graduação em Administração de Empresas) - Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, 2017. Disponível em: https://www.maxwell.vrac.puc-rio.br/33559/33559.PDF. Acesso em: 15 nov. 2018.

BECK, KENT; CUNNINGHAM, W.; HUNT, A.; MARTIN, ROBERT C.; THOMAS, D. BEEDLE, M.; FOWLER, M.; JEFFRIES, R.; MELOR, S.; BENNEKUM, A.; GRENNING, J; KERN, J.; SCHWABER, K.; COCKBURN, A.; HIGHSMITH, J.; MARICK, B.; SUTHERLAND, J. **Princípios por trás do Manifesto Ágil.** Disponível em: http://www.manifestoagil.com.br/principios.html. Acesso em: 16 nov. 2018.

BERNARDO, Kleber. **O que são métodos ágeis?**. Disponível em: http://www.culturaagil.com.br/o-que-sao-metodos-ageis/. Acesso em: 16 nov. 2018.

BERNARDO, Kleber. **Estória de usuário. Você saberia contar?**. Disponível em: https://www.culturaagil.com.br/estoria-de-usuario-voce-saberia-contar/. Acesso em: 16 nov. 2018.

CAMPELO, Renata Endriss Carneiro. **XP-CMM2:** um guia para utilização de Extreme Programming em um ambiente nível 2 do CMM. 2003. 207 p. Dissertação (Mestrado em Ciência da Computação) - Universidade Federal de Pernambuco, Recife, 2005. Disponível em: http://www.cin.ufpe.br/~hermano/download/dissertacoes/2003-XP-CMM2--

Renata%20Campelo.pdf. Acesso em: 15 nov. 2018.

COELHO, Cristiane dos Santos. **RELATO DE EXPERIÊNCIA NA IMPLANTAÇÃO DE UM MÉTODO ÁGIL EM UMA EQUIPE DE DESENVOLVIMENTO DE SOFTWARE**. 2011. 57 p. Monografia (Monografia de graduação) - Universidade Federal de Lavras, LAVRAS - Minas Gerais, 2015. Disponível em: http://repositorio.ufla.br/bitstream/1/30660/1/MONOGRAFIA_Relatorio_de_experiencia_na_implantacao_de_um_metodo_agil_em_uma_equipe_de_desenvolvimento_de_software.pdf. Acesso em: 16 nov. 2018.

COHN, Mike. **Desenvolvimento de Software com Scrum:** Aplicando métodos ágeis com sucesso. Porto Alegre: Bookman, 2011.

COSTA, Wesley Novaes Fioreze. **Modelagem de um novo processo de desenvolvimento de software com base em metodologias ágeis.** IFSP – Instituto Federal de Educação, Ciência e Tecnologia de São Paulo, São Paulo, p. 01-05, 2010. Disponível em: https://docplayer.com.br/39369884-Modelagem-de-um-novo-processo-de-desenvolvimento-de-software-com-base-em-metodologias-ageis.html. Acesso em: 16 nov. 2018.

CUNHA, Carlos Eduardo Albuquerque da. **Utilizando OpenUP/Basic para desenvolvimento de aplicações WEB.** 2007. 73 p. Trabalho de Conclusão de Curso (Graduação em Ciência da Computação) - Universidade Federal de Pernambuco, Recife, 2007. Disponível em: http://www.cin.ufpe.br/~tg/2006-2/ceac.pdf. Acesso em: 15 nov. 2018.

DOURADO, Antônio Miguel Batista. **OpenUP:** Um processo prático e otimizado para desenvolvimento de pequenos projetos de software. Disponível em: http://aberto.univem.edu.br/bitstream/handle/11077/714/OpenUP.pdf?sequence=1&isAllowed=y. Acesso em: 16 nov. 2018.

FELLER, Leandro. **Proposta de Modelo Ágil:** União de boas práticas das metodologias Scrum, Kanban, Lean, FDD e XP. 2013. 40 f. Trabalho de Conclusão de Curso (Graduação Tecnológica) - Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas. Faculdade de Tecnologia do SENAI, Florianópolis, 2013. Disponível em: http://www.softplan.com.br/tratadoaprendizagem/wp-content/files_mf/1415619953TCC_LeandroFeller.pdf . Acesso em: 15 nov. 2018.

FIGUEIREDO, Alexandre Magno. **Um guia de rápido aprendizado para a Feature-Driven Development.** p. 01-23, 2007 Disponível em: http://homes.dcc.ufba.br/~mauricio052/Engenharia%20de%20Software%20I/FDD/FDD%20Em%20Uma%20Casca%20De%20Banana.pdf. Acesso em: 16 nov. 2018.

GAEA. **Guia completo para entender a metodologia lean de uma vez por todas.** Disponível em: https://gaea.com.br/guia-completo-para-entender-a-metodologia-lean-de-uma-vez-por-todas/. Acesso em: 17 nov. 2018.

HORÁCIO, Jorge. **FDD** – Feature-Driven Development. 2012. Disponível em: https://jorgeaudy.com/2012/07/24/fdd-feature-driven-development/. Acesso em: 17 nov. 2018.

JESUS, Italo Paolo Saturnino de. **Metodologias ágeis de gerenciamento de projetos para startups.** 2016. 56 f. Monografia (Especialização em Informática: ênfase em Engenharia de Software) - Universidade Federal de Minas Gerais – Departamento de Ciência da Computação, Minas Gerais, 2016. Disponível em: http://www.bibliotecadigital.ufmg.br/dspace/bitstream/handle/1843/ESBF-A9EH4J/monografia_italo_paolo_final.pdf?sequence=1. Acesso em: 17 nov. 2018.

JUNIOR, Nilton Freitas; ROCHA, Davi; GOMES, Ariadne. Proposta de um novo Modelo de aplicação da metodologia ágil SCRUM para documentação. In: CONGRESSO NACIONAL DE EXCELÊNCIA EM GESTÃO, 12; INORVASE, 3, 2016, Rio de Janeiro. **Anais** [...]. Rio de janeiro: Inorvase, 2016. Disponível em: http://www.inovarse.org/sites/default/files/T16_221.pdf. Acesso em: 17 nov. 2018.

LIBARDI, Paula L.O; BARBOSA, Vladimir. **Métodos Ágeis**. 2010. 35 f. Monografia (Conclusão da disciplina tópicos em computação no curso de pós graduação)- Faculdade de Tecnologia - FT, Universidade Estadual de Campinas - UNICAMP, LIMEIRA - SP, 2010. Disponível em: http://www.ft.unicamp.br/liag/Gerenciamento/monografias/monogafia_metodos_ageis.pdf. Acesso em: 17 nov. 2018.

LOCAWEB. **Programação eXtrema** – eXtreme Programming ou simplesmente XP. Disponível em: http://blog.locaweb.com.br/artigos/metodologias-ageis/programacao-extrema-extreme-programming-ou-simplesmente-xp/. Acesso em: 17 nov. 2018.

MAGALHÃES, Cleyton Vanut Cordeiro de; SANTOS, Ronnie Edson de Souza; ALMEIDA, Isledna Rodrigues de. **Aplicação da Gerência de Configuração em Métodos Ágeis de Desenvolvimento de Software**. 2010. 7 p. Disponível em: https://pt.scribd.com/doc/38125200/Gerencia-de-Configuracao-e-Mudanca-nos-Metodos-Ageis-de-Desenvolvimento-de-Software. Acesso em: 17 nov. 2018.

N2S, Núcleo de Soluções em Software. **Maximizando potenciais, desenvolvendo soluções**. 2018. Disponível em: http://n2s.russas.ufc.br. Acesso em: 16 nov. 2018.

PIMENTEL, Luan Félix; PEGORARO, Raquel Aparecida. Análise de fatores de sucesso da adoção de métodos ágeis em micro e pequenas empresas de software e identificação de métricas que possibilitem auxiliar no seu monitoramento. In: JORNADA DE INICIAÇÃO CIENTÍFICA E TECNOLÓGICA, 6., 2016, Chapecó-SC. **Anais** [...]. Chapecó-SC: UFFS, v.1, n.6, 2016, p. 01-04. Disponível em: https://periodicos.uffs.edu.br/index.php/JORNADA/article/view/4744/2610. Acesso em: 17 nov. 2018.

SANTOS, Deisy Braz; CÓRDOVA, Paulo Roberto. Combinação de Métodos Ágeis no Processo de Desenvolvimento de Software: um estudo de caso. **Ignis**, Santa Catarina, v.6, n.1, p.06-23, jan./abr. 2017. Disponível em: https://periodicos.uniarp.edu.br/ignis/article/view/1133/570. Acesso em: 17 nov. 2018.

SCHEREDER, Joceli da Guia Chandelier. **Implantação da Metodologia Lean:** Um estudo de caso dos comportamentos observados em empregados de empresa privada da área de controladoria. 2013. 40 f. Monografia (Especialização em Controladoria,) - Universidade Federal do Paraná, Curitiba, 2013. Disponível em:

https://acervodigital.ufpr.br/bitstream/handle/1884/40216/R%20-%20E%20-%20JOCELI%20DA%20GUIA%20CHANDELIER%20SCHEREDER.pdf?sequence=1&isAllowed=y. Acesso em: 17 nov. 2018.

SCHWABER, Ken; SUTHERLAND, Jeff. **Guia Scrum.** p. 01-19, 2013. Disponível em: https://www.scrumguides.org/docs/scrumguide/v1/Scrum-Guide-Portuguese-BR.pdf. Acesso em: 17 nov. 2018.

SENE, Rafael Perie. **OpenUP:** Uma visão geral. Disponível em: https://www.tiespecialistas.com.br/openup-uma-visao-geral/. Acesso em: 17 nov. 2013.

SOFTPLAN: **Uma história de sucesso e relevância para o mercado nacional**. Disponível em: http://www.softplan.com.br/. Acesso em: 16 nov. 2018.

SOMMERVILLE, Ian. **Engenharia de Software.** 9ª Edição. Editora Pearson, 2011.

TELES, Vinícius Manhães. **EXTREME PROGRAMMING**. Disponível em: http://www.desenvolvimentoagil.com.br/xp/. Acesso em: 17 nov. 2018.

VARASCHIM, Jacques Douglas. **Implantando o SCRUM em um Ambiente de Desenvolvimento de Produtos para Internet**. 2008. 21 p. Monografia (Monografia em Ciência da Computação) - Pontifíca Universidade do Rio de Janeiro, Rio de Janeiro, 2009. Disponível em: ftp://ftp.inf.puc-rio.br/pub/docs/techreports/09_07_varaschim.pdf. Acesso em: 17 nov. 2018.

VASCO, Carlos G.; VITHOFT, Marcelo Henrique; ESTANTE, Paulo Roberto C. **Comparação entre Metodologias RUP e XP**. 2006. Disponível em: http://www.ppgia.pucpr.br/~alcides/Teaching/mestrado/FundamentosEngenhariaSoftware/artigos/ResumosApresentacoes/RUPvsXP_draft.pdf. Acesso em: 15 nov. 2018.

WAZLAWICK, Raul Sidnei. **Engenharia de Software**: Conceitos e práticas. [S.l.]: Elsevier Editora Ltda, 2013. 360 p.

APÊNDICE A - Questionário sobre Metodologias de Desenvolvimento de Software

1. Atualmente, a(s) equipe(s) de desenvolvimento que participa é considerada(s)?*

Escolha uma ou mais opções.

- [ ] Pequena (Máximo 4 membros)
- [ ] Média (Entre 5 e 8 membros)
- [ ] Grande (9 ou mais membros)
- [ ] Não estou em nenhum projeto no momento

2. Qual papel exerce, ou já exerceu, em projetos de software? *

Escolha uma ou mais opções.

- [ ] Analista de sistemas
- [ ] Arquiteto de software
- [ ] Desenvolvedor
- [ ] Outros _____

3. Qual o tempo estimado da sua experiência em projetos com métodos ágeis? *

- [ ] Nenhum experiência
- [ ] Menos de 1 mês
- [ ] Entre 1 e 7 meses
- [ ] Entre 8 e 12 meses
- [ ] Entre 1 e 2 anos
- [ ] Mais de 2 anos

4. Se você já tem experiência com métodos ágeis, responda: Qual foi o nível de facilidade/dificuldade no aprendizado das práticas dos métodos ágeis utilizados?

	1	2	3	4	5	
Tive muita dificuldade	☐	☐	☐	☐	☐	Tive grande facilidade em aprender

5. Se teve dificuldades, responda: Quais foram as maiores dificuldades enfrentadas no aprendizado dos métodos ágeis?

_____

_____

_____

_____

_____

6. Com relação a sua participação em projetos, marque as opções referentes aos métodos que já utilizou. *
Escolha uma ou mais opções.

☐ 1. Participei de projetos sem processo bem definido

☐ 2. Participei de projetos com métodos tradicionais (Ex.: RUP)

☐ 3. Participei de projetos com métodos ágeis (Ex.: Scrum, XP, FDD)

7. Se marcou a opção 1 na pergunta anterior, responda: Quais as dificuldades/problemas enfrentadas em projetos sem um processo bem definido?
Escolha uma ou mais opções.

☐ Atraso na entrega do produto

☐ Aumento de custos

☐ Requisitos mal entendidos

☐ Documentação desatualizada

☐ Outros _____

8. Se marcou a opção 2 na pergunta 6, responda: Quais as dificuldades/problemas enfrentadas em projetos que utilizam métodos tradicionais?
Escolha uma ou mais opções.

- [ ] Atraso na entrega do produto
- [ ] Aumento de custos
- [ ] Requisitos mal entendidos
- [ ] Problemas para adaptar mudanças durante o projeto
- [ ] Documentação desatualizada
- [ ] Outros _____

9. Se marcou a opção 3 na questão 6, responda: Quais as dificuldades/problemas enfrentadas em projetos que utilizam métodos ágeis?
Escolha uma ou mais opções.

- [ ] Atraso na entrega do produto
- [ ] Aumento de custos
- [ ] Requisitos mal entendidos
- [ ] Documentação desatualizada
- [ ] Outros _____

10. Se tem conhecimento (na teoria e/ou prática) em métodos ágeis, responda: Na sua percepção, que pontos precisam ser melhorados e/ou abrangidos por métodos ágeis?
Ex.: Modelagem com diagrama mais utilizados, documento básico de requisitos, gerenciamento das alterações nas versões do software, entre outros.

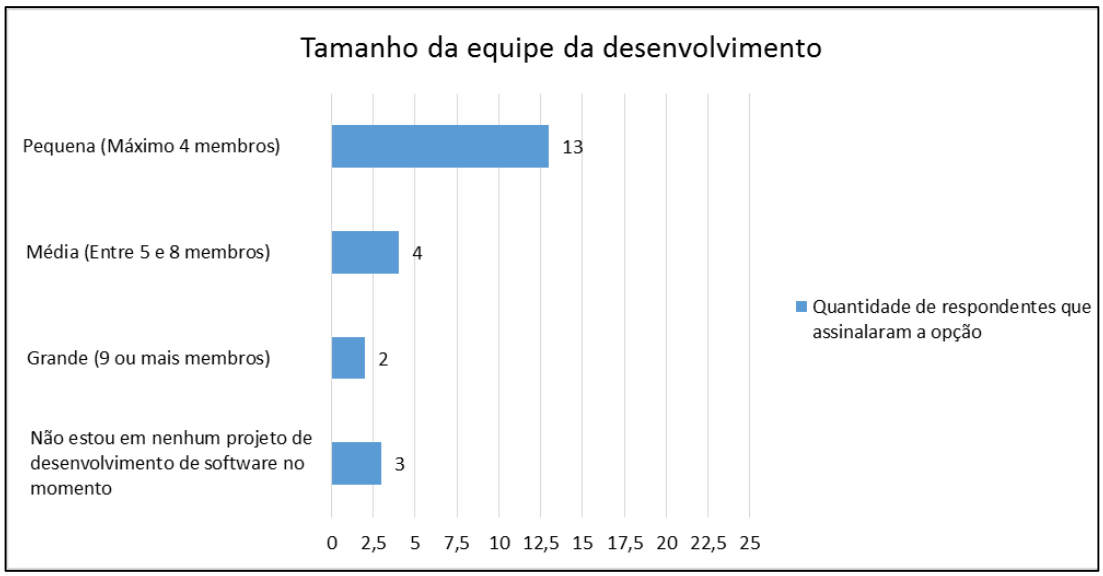
_____

_____

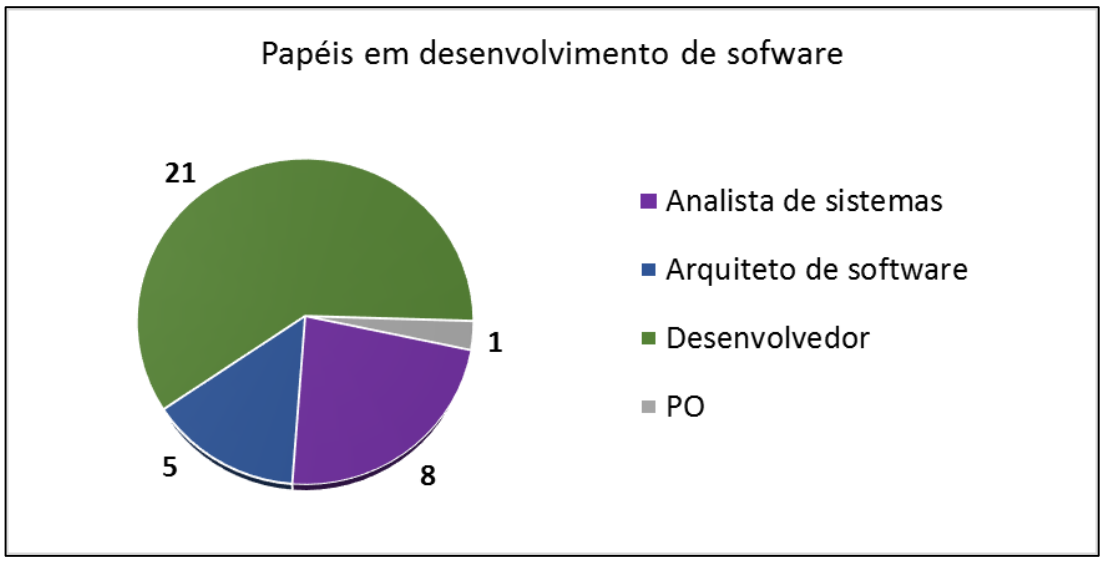
11. Espaço livre para observações sobre metodologias de desenvolvimento de software.
Sugestões para melhorias de práticas de desenvolvimento de software, críticas ou elogios.

APÊNDICE B – Respostas do Questionário

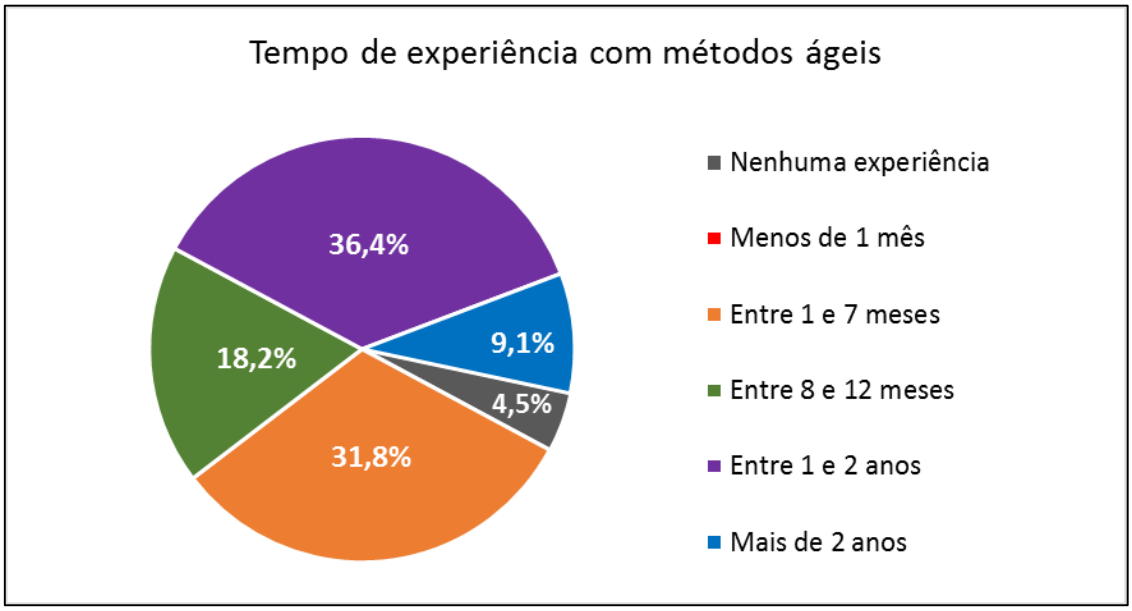
1. Atualmente, a(s) equipe(s) de desenvolvimento que participa é considerada(s)?
22 respostas



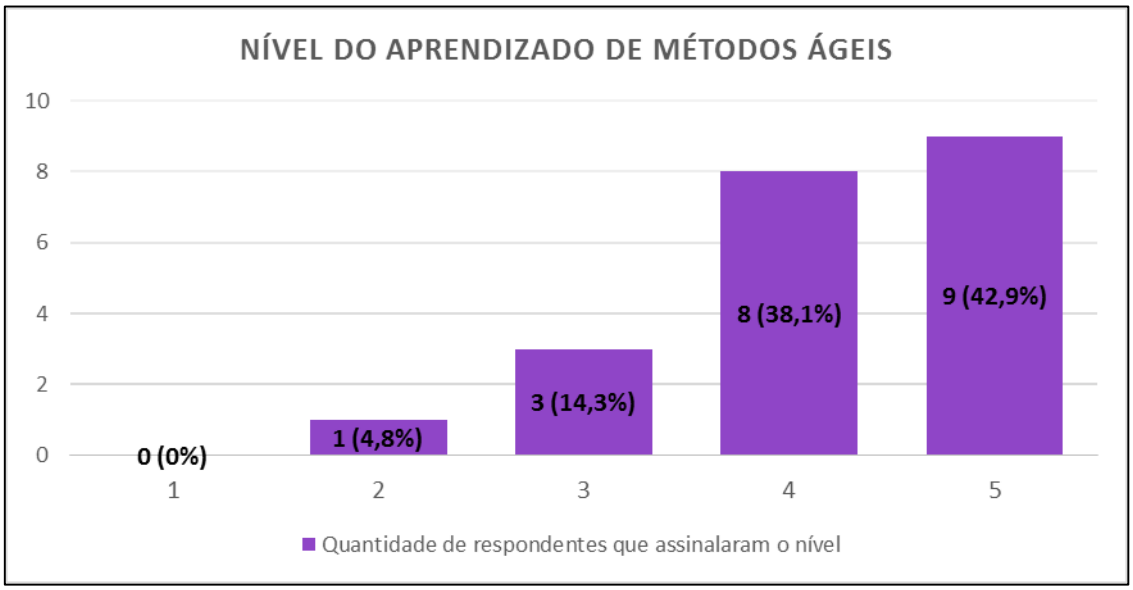
2. Qual papel exerce, ou já exerceu, em projetos de software?
22 respostas



3. Qual o tempo estimado da sua experiência em projetos com métodos ágeis?
22 respostas



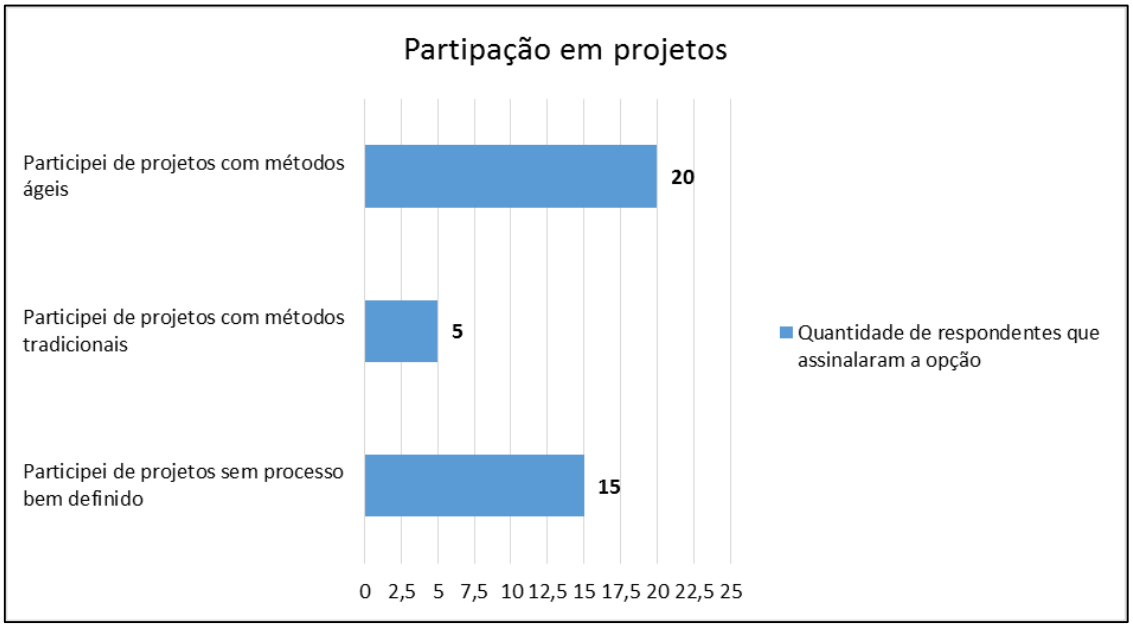
4. Se você já tem experiência com métodos ágeis, responda: Qual foi o seu nível de facilidade/dificuldade no aprendizado das práticas dos métodos ágeis utilizados?
21 respostas



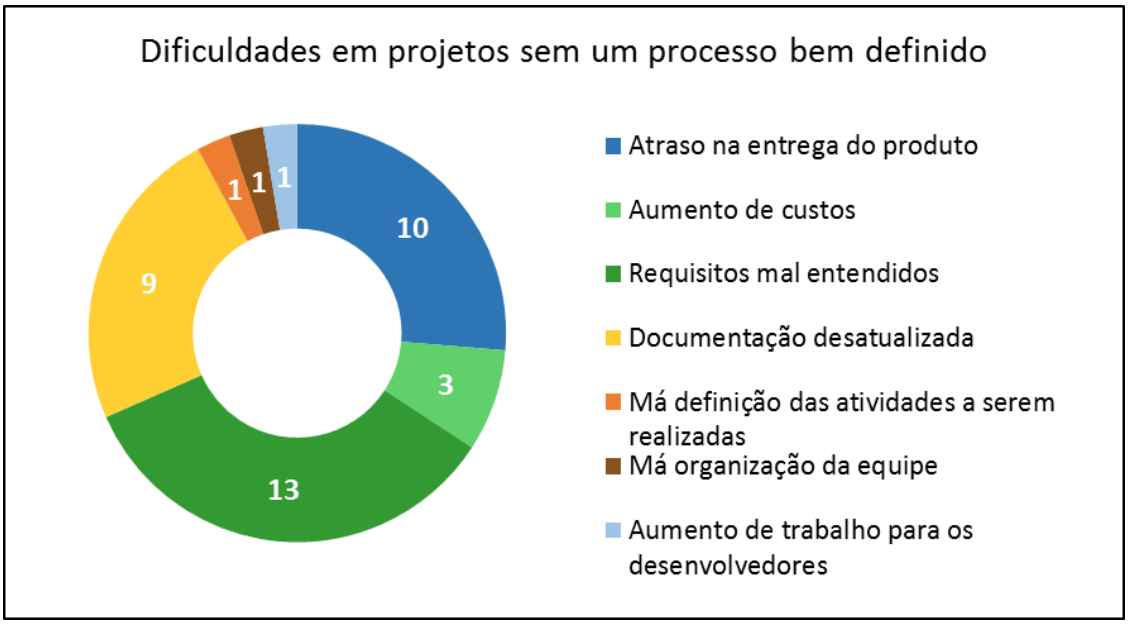
5. Se teve dificuldades, responda: Quais foram as maiores dificuldades enfrentadas no aprendizado dos métodos ágeis?
6 respostas

Falando especificamente do SCRUM, as dificuldades foram em relação as escolhas de quais estórias ou requisitos implementar durante uma Sprint. Pode parecer estranho, mas é um pouco difícil deixar nas mãos da equipe de desenvolvimento quais atividades cada programador vai realizar. Na minha opinião, isso pode tornar o processo de desenvolvimento um pouco sem rumo quando os membros da equipe escolhem as atividades que serão realizadas sem seguir uma forma lógica, em que uma atividade precisa ser feita antes da outra, ou coisas assim.
O processo não estava definido.
As poucas dificuldades foram implantar algumas práticas no dia a dia, porém foi superado com pouco tempo de prática.
Encaixar todos os artefatos do Scrum.
Atender a prazos.

6. Com relação a sua participação em projetos, marque as opções referentes aos métodos que já utilizou.
22 respostas

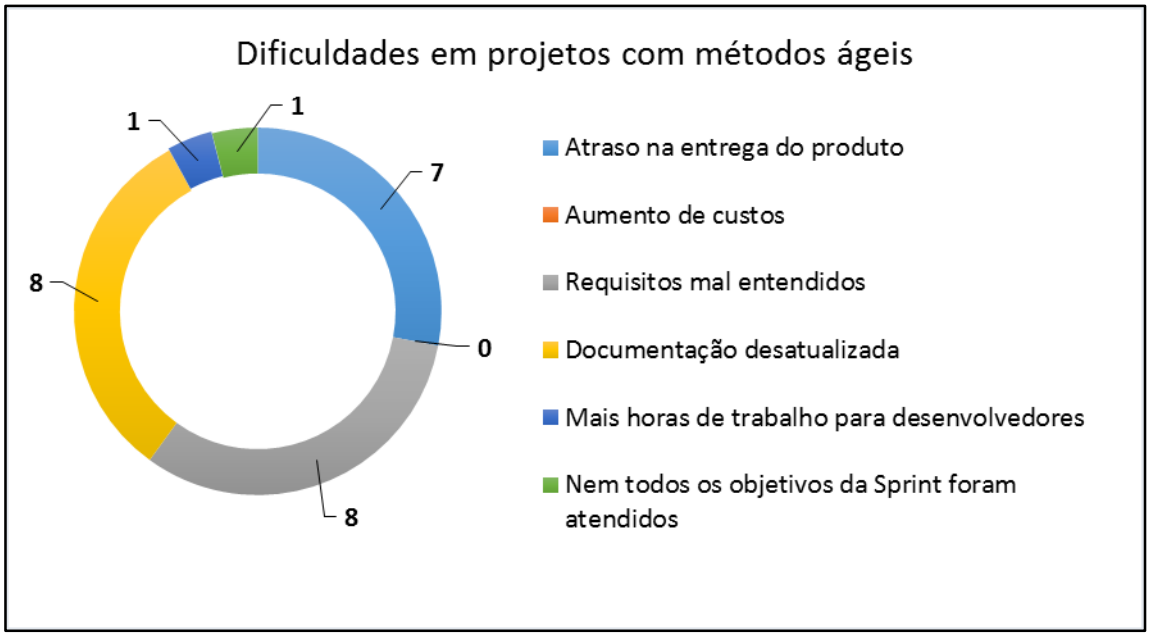


7. Se marcou a opção 1 na pergunta anterior, responda: Quais foram as possíveis dificuldades enfrentadas em projetos sem um processo bem definido?
15 respostas



8. Se marcou a opção 2 na pergunta 6, responda: Quais foram as dificuldades enfrentadas em projetos com métodos tradicionais?
5 respostas

9. Se marcou a opção 3 na questão 6, responda: Quais foram as dificuldades enfrentadas em projetos com métodos ágeis?
18 respostas



10. Se tem conhecimento (na teoria e/ou prática) em métodos ágeis, responda: Na sua percepção, que pontos precisam ser melhorados e/ou abrangidos por métodos ágeis?
15 respostas

A documentação, pouca documentação pode dificultar em manutenções. Ágil + Prescrição uma boa saída.
Apesar das dificuldades apresentas anteriormente, o problema por mim citado é mais em relação aos membros participantes do processo do que em relação ao processo em si. Se posso apontar algo a melhorar nas metodologias seria em relação a documentação. Acredito que, pelo fato das metodologias valorizarem mais o software funcionando do que documentação, as pessoas as adotam pensando que não vão precisar documentar nada sobre o produto desenvolvido. Acho que deveria haver algo que reforçasse o uso e a necessidade da documentação ou até a descrição de alguns tipos de documentos que podem ser utilizados em processos ágeis.
Um bom gerenciamento e controle sobre a definição do sistema e seus requisitos.
Dar uma dica sobre como iniciar o projeto.
Modelagem com diagramas para facilitar o entendimento
Um documento de requisitos mais simples, para facilitar o entendimento, e um mapeamento do mesmo para as funcionalidades.
Focar um pouco mais na documentação.
Questão de prazo como trabalhando com pequenas entregas temos a impressão que nunca paramos de trabalhar é esse círculo sempre tá retornando e nunca vamos ficar ociosos.
Na minha opinião, eu como desenvolvedor não gosto de estar criando atividades no trello (product backlog ou sprint backlog), acho que perdemos tempo com isso.
Planejar bem o que deve ser entregue em cada ciclo.
Reuso deveria ser uma base para a utilização de métodos ágeis.

Não vejo necessidade de melhorar os métodos, mas sim na consciência do gerente em definir prazos tão curtos.
Vejo que um pouco mais de tempo no entendimento do produtos seus requisitos e do domínio teria ajudado bastante algumas etapas seguintes do desenvolvimento.
Uma melhor gestão da documentação do projeto em questão e também a melhor gerência das atividades a serem feitas e por quem essas atividades são feitas.
Adaptação a governança de TI onde os processos não são ágeis.

Espaço livre para observações sobre metodologias de desenvolvimento de software.
4 respostas

As metodologias ágeis são por natureza dinâmicas, porém este dinamismo vêm por vezes ao custo do entendimento de alguns requisitos, que necessitam serem melhores detalhados, ou diagramados.
Qualquer metodologia funciona melhor quando as pessoas envolvidas no processo estão motivadas e felizes.
Vejo que é uma tendência dar cada vez mais atenção aos quesitos humanos atrelados ao desenvolvimento de software, vendo isso acho que metodologias que dão um maior enfoque a essas questões terão melhores resultados e espaço no futuro.
O ágil é muito aplicado e está sendo crescentemente incorporado em grandes corporações. Porém há muita resistência por parte da diretoria. Os cases de sucesso são os grande motivadores para essa mudança.

APÊNDICE C – Termo de autorização do N2S

TERMO DE AUTORIZAÇÃO

O Núcleo de Soluções em Software (N2S) autoriza a discente Mirlane Beserra Cavalcante a desenvolver o seu Trabalho de Conclusão de Curso em Engenharia de Software, intitulado "Proposta de um Modelo Baseado na Combinação de Metodologias Ágeis de Desenvolvimento de Software", aplicando as práticas do modelo proposto nesse trabalho.

Russas - CE, _____ de Outubro de 2018.

_____

Responsável pelo Núcleo de Soluções em Software (N2S)

APÊNDICE D – Modelo do Termo de autorização para divulgação de informações

UNIVERSIDADE FEDERAL DO CEARÁ

CAMPUS DE RUSSAS

TERMO DE AUTORIZAÇÃO PARA DIVULGAÇÃO DE INFORMAÇÕES

Nome: Núcleo de Soluções em Software (N2S)

Endereço completo: Rua Felipe Santiago, 411, Cidade Universitária, Russas – CE

Representante: José Osvaldo Mesquita Chaves

Telefone: (88) 3411-2143 Email: n2s@ufc.br

Tipo de produção intelectual: ( ) Monografia; ( ) Relatório técnico; ( ) Relatório de estágio; ( ) Trabalho de Conclusão de Curso; ( ) Outro: _____

Título/subtítulo: _____

_____

Autoria: _____ Matrícula: _____

Orientador: _____

Nome do curso: _____

Campus: _____

Como representante do Núcleo de Soluções em Software, declaro que as informações e/ou documentos disponibilizados para o trabalho citado:

( ) Podem ser publicados sem restrição.

( ) Possuem restrição parcial por um período de _____ anos, não podendo ser publicadas as seguintes informações e/ou documentos: _____

_____

( ) Possuem restrição total para publicação por um período de_____anos, pelos seguintes motivos: _____

_____

_____ _____

Representante do N2S Local e data

APÊNDICE E – Modelo do Termo de Consentimento dos Participantes

TERMO DE CONSENTIMENTO

Este é um convite para participação no projeto de pesquisa "Proposta de um Modelo Baseado na Combinação de Metodologias Ágeis de Desenvolvimento de Software", sob a responsabilidade da discente Mirlane Beserra Cavalcante, graduanda do curso de Engenharia de Software da Universidade Federal do Ceará (UFC) – Campus Russas, sob a orientação do Professor Ms. José Osvaldo Mesquita Chaves.

Sua participação será voluntária e se dará por meio da utilização das práticas do modelo, realizada durante a implantação no Núcleo de Soluções em Software (N2S). Estando ciente que esse estudo contém a realização de uma entrevista individual com cada participante, aplicada pela discente. Se aceitar participar, estará contribuindo para o referido Trabalho de Conclusão de Curso (TCC).

Os resultados da pesquisa serão analisados e publicados. Mas sua identidade não será divulgada, sendo guardada em sigilo. Portanto, estará ciente que este documento será emitido em duas vias, que serão ambas assinadas pelo participante, pelo orientador e pela pesquisadora. Ficando uma via deste termo para o participante.

Eu _____, concordo em participar da pesquisa, sabendo que não terei nenhuma despesa e nenhuma remuneração.

_____

Assinatura do participante

_____ _____

Assinatura da pesquisadora Assinatura do Orientador do projeto

Russas - CE, _____ de Outubro de 2018.

APÊNDICE F – Roteiro da Entrevista

1. Percebeu alguma mudança após a implantação da metodologia ágil?

2. Aspectos que a metodologia ajudou? (Organização, integração da equipe, motivação)

3. Alguma prática que poderia ser melhorada?

4. Qual sua opinião sobre o Planning Poker? Trouxe benefícios? Tem melhorias a sugerir? Dificuldades enfrentadas durante a execução?

5. Sua opinião sobre o banco de lições, se é uma boa prática e útil para resolução de problemas?

6. Teve problemas/dificuldades durante o processo de desenvolvimento devido a implantação do modelo? Se sim, quais foram?

7. Houve atrasos de algumas tarefas, qual sua opinião sobre isso, ou melhor, quais motivos você acha que levou a isso? Foi devido a utilização das práticas do modelo? Ou está ligado à outros fatores, como desmotivação, problemas com ferramentas?

APÊNDICE G – Questionário pós-entrevista

1. Seu nível de satisfação com o modelo?

2. Recomendaria o uso desse modelo?

☐ Sim

☐ Não, pelo motivo:

3. Considera que deveria ser acrescentadas mais práticas ao modelo? Se sim, quais?

4. Você considera que a adoção das práticas no desenvolvimento de software trouxe benefícios (melhor planejamento, mais foco na resolução de problemas futuros com o banco de lições, entre outros) para o núcleo?

☐ Sim

☐ Não, pelo motivo:

5. Espaço para comentários, sugestões ou/e críticas sobre a realização do estudo de caso e/ou sobre a aplicação de metodologias ágeis no desenvolvimento de software