CENTRO UNIVERSITÁRIO DE ANÁPOLIS – UniEVANGÉLICA
BACHARELADO EM ENGENHARIA DE COMPUTAÇÃO

FRAMEWORK PARA GESTÃO E DESENVOLVIMENTO DE *SOFTWARE* ÁGIL: UM ESTUDO DE CASO DESENVOLVIDO NAS DISCIPLINAS PRÁTICA EM FÁBRICA DE *SOFTWARE*

ISABELLA CAROLINA MORAIS DE BARROS
JÚLIO RODRIGUES LOBO

ANÁPOLIS - GO

2019

ISABELLA CAROLINA MORAIS DE BARROS
JÚLIO RODRIGUES LOBO

FRAMEWORK PARA GESTÃO E DESENVOLVIMENTO DE *SOFTWARE* ÁGIL: UM ESTUDO DE CASO DESENVOLVIDO NAS DISCIPLINAS PRÁTICA EM FÁBRICA DE *SOFTWARE*

Trabalho de Conclusão de Curso II apresentado como requisito parcial para a obtenção de grau do curso de Bacharelado em Engenharia de Computação do Centro Universitário de Anápolis – UniEVANGÉLICA.

Orientadora: Profa. Dra. Renata Dutra Braga

ANÁPOLIS - GO

2019

ISABELLA CAROLINA MORAIS DE BARROS
JÚLIO RODRIGUES LOBO

FRAMEWORK PARA GESTÃO E DESENVOLVIMENTO DE *SOFTWARE* ÁGIL: UM ESTUDO DE CASO DESENVOLVIDO NAS DISCIPLINAS PRÁTICA EM FÁBRICA DE *SOFTWARE*

Trabalho de Conclusão de Curso II apresentado como requisito parcial para a obtenção de grau do curso de Bacharelado em Engenharia de Computação do Centro Universitário de Anápolis – UniEVANGÉLICA.

Aprovado(a) pela banca examinadora em __ de _____ de 2019, composta por:

Renata Dutra Braga
Renata Dutra Braga
Presidente da Banca

Natasha Sophie
Natasha Sophie Pereira
Prof(a). Convidado(a)

Walquiria F. Marins
Walquiria Fernandes Marins
Prof(a). Convidado(a)

Aos nossos pais, familiares e companheiros que, com muito carinho e apoio, não mediram esforços para que chegássemos até esta etapa de nossas vidas.

E também a Professora Dra. Renata, pela paciência na orientação e incentivo que tornaram possível a conclusão deste trabalho.

“Ser pela liberdade não é apenas tirar as correntes de alguém, mas viver de forma que respeite e melhore a liberdade dos outros.”

(Nelson Mandela)

Resumo

Justificativa: A gestão da construção de um *software* muitas vezes é o ponto mais crítico das equipes, pois nem sempre as metodologias se encaixam na forma que a equipe trabalha o que dificulta alcançar um fluxo de desenvolvimento e uma agilidade adequada, que possa garantir uma boa qualidade do produto de *software* e uma maior integração da equipe. Por isso o uso de um framework e suas definições são importantes, pois com ele a equipe tem maior controle dos processos, dos problemas, das soluções e do tempo gasto para cada iteração, garantindo um processo ágil e interativo com toda a equipe de desenvolvimento e com o cliente melhorando a qualidade e o andamento do projeto. Objetivo: Avaliar um *framework* para gestão e desenvolvimento de *software* ágil, baseado nas especificidades de um projeto em andamento nas disciplinas de Prática em Fábrica de *Software*. Método: Trata-se de estudo exploratório, descritivo, com estudo de caso do projeto EduPlan e execução na Fábrica de Tecnologias Turing da instituição UniEvangelica nas disciplinas de PSF. Ele foi estruturado em quatro etapas: 1- identificação de boas práticas; 2- criação do *framework*; 3- aplicação; 4- registro dos resultados. Resultados: Onze metodologias foram analisadas durante a fase inicial desse estudo. Isso permitiu a elaboração de um *framework* com seis etapas, com as respectivas atividades, técnicas, boas práticas, artefatos e ferramentas. Conclusão: O desenvolvimento do presente estudo possibilitou a avaliação do framework que foi criado e validado em um estudo de caso nas disciplinas de PFS, onde utilizando métodos ágeis foi possível criar um processo ágil que se encaixasse especificamente na equipe e no projeto. E através dos estudos que foram feitos, foi possível extrair informações que foram cruciais ao elaborar o processo, pois fundamental para ter os resultados apresentados.

Palavras-chaves: metodologia ágil, *framework*, implementação, planejamento, resultados.

Abstract

Justification: The management of software construction is commonly the most critical point of the teams, because the methodologies will not always fit the way the team works, which makes it difficult to reach the perfect flow of development and agility that could guarantee a good software quality and a greater integration of the team. Therefore, the use of a framework and its definitions are important, because with it the team has greater control of the processes, problems, solutions and the time spent for each interaction, granting a fast and interactive process with all the development team and the customer, improving the quality and the progress of the project. Objective:Evaluate a framework to a fast management and development of the software, based on the specifications of a project in progress in the disciplines of practice in software factory. Method:It is about of a exploratory study, descriptive, with case study of the Project EduPlan and execution at Turing technology factory of UniEvangelica institution in PSF disciplines. It was structured in four stages: 1 - Identification of good practices; 2 - Framework creation; 3 - Aplication; 4 - Resgistry of the results; Results:Eleven methodologies were analized during the initial phase of this study. That allowed the elaboration of a 6 stages framework, with the respective activities, techniques, good practices, artefacts and tools. Conclusion: The development of the present study made it possible the evaluation of the framework created and validated in a case study in PFS disciplines, where using the fast methods, it was possible to create a fast process which would fit specifically on the team and in the project. And through studies that were made, it was possible to extract crucial informations when developing the process, because it was fundamental, to obtain the presented results.

Keywords: *agile methodology, framework, implementation, planning, results.*

Lista de figuras

Lista de Quadros

Sumário

1. INTRODUÇÃO

Uma das finalidades do uso de modelos de desenvolvimento de *software* é facilitar e organizar as etapas e tarefas que necessitam ser realizadas ao longo do projeto. Elas são estruturadas em etapas, em formato de linha de produção, ora iterativa ora sequencial. Quando se trata de métodos tradicionais, tais formatos podem impedir que os desenvolvedores atendam às expectativas e necessidades do cliente (PRESSMAN, 2016).

Ao considerar-se um cenário de constantes evoluções e mudanças, a necessidade de adaptabilidade torna-se primordial para o ingresso e permanência no mercado de produção de *software* (PRESSMAN, 2016). Um projeto de *software* ou qualquer tipo de projeto que venha a ser executado, deve contar com um planejamento detalhado de cada fase, e com um acompanhamento de cada processo. Caso contrário podem vir a ocorrer vários problemas como desorganização da equipe e a falta de acompanhamento do projeto. Esses problemas podem causar falhas futuras na implementação e nos testes e, consequentemente, alto custo no produto final. Por conseguinte, quando não há uma previsão pode haver demora na entrega dos produtos. Além disso, sem o devido ciclo, as entregas para os clientes sem data prevista, podem haver mudanças de requisitos ocasionando em aumento de prazo e aumento de custo. Ou seja, diversos inconvenientes que podem ser evitados com a utilização de uma metodologia.

Em contraposição a este cenário, tem-se as metodologias ágeis, capazes de ofertar total flexibilidade no desenvolvimento e maior interação entre os *stakeholders*. Com isso, o produto de *software*, é entregue em pequenos incrementos (partes), o que resulta em menor tempo, proporcionando a validação, inclusive o entendimento das necessidades, sendo possível contemplar as mudanças necessárias. A diferença entre métodos ágeis e Clássicos sequenciais está o enfoque maior nas pessoas e o seu conjunto de valores, princípios e práticas, além do trabalho em equipe e comunicação efetiva (PRIKLADNICKI, WILLI e MILANI, 2014). Como citado por Soares (2014) “Comparando métodos ágeis com as metodologias tradicionais pesadas mostrou que os projetos usando os métodos ágeis obtiveram melhores resultados em termos de cumprimento de prazos, de custos e padrões de qualidade”. Assim, pode-se destacar que o uso de métodos ágeis em relação a modelos tradicionais se encaixam adequadamente aos projetos que são executados nas disciplinas de Prática de Fábrica de *Software*, pois abrange as principais características que são ministradas nessa disciplina, que são o prazo, a interação com o cliente, a equipe, os custos e a qualidade.

Com a necessidade da construção de um produto de *software* nas disciplinas de PFS do Bacharelado em Engenharia de Computação identificou-se a necessidade da elaboração de

um *framework*. Por *framework* entenda a definição do passo a passo das etapas e atividades para desenvolvimento de um produto com funções pré-definidas para resolver um problema de um domínio específico. Ele foi elaborado por uma equipe, que contempla as suas particularidades referente à produção efetiva do produto de *software*. Para Prikladnicki, Willi e Milani (2014, p. 47) "um ambiente colaborativo tem foco nos resultados da equipe", então o importante e necessário é que todos os integrantes conseguiram entender cada parte do processo, e trabalharam de forma que cada um teve conhecimento de todas as áreas.

Considerando esse cenário, a pergunta que norteou a execução deste estudo foi: quais os elementos e boas práticas são necessárias para a definição de um *framework* que auxilie na gestão e desenvolvimento de um *software* ágil em andamento, nas disciplinas de Prática em Fábrica de *Software*?

A gestão da construção de um *software* muitas vezes é o ponto mais crítico das equipes, pois nem sempre as metodologias se encaixam na forma que a equipe trabalha e, além disso, muitas vezes não são adotados modelos de processos, dificultando alcançar um fluxo de desenvolvimento e com uma agilidade adequada, que possa garantir uma boa qualidade do produto de *software* e uma maior integração da equipe (KOSCIANSKI e SOARES, 2007). Por isso o uso de um *framework* e suas definições são importantes, pois com ele a equipe tem maior controle dos processos, dos problemas, das soluções e do tempo gasto para cada iteração, garantindo um processo ágil e interativo com toda a equipe de desenvolvimento e com o cliente melhorando a qualidade e o andamento do projeto.

Os fatores que dificultaram a adoção de um modelo de processo já existente foram a dificuldade de adaptação ao contexto do projeto e o andamento das disciplinas de Prática em Fábrica de *Software* onde são divididas em 3 e em cada uma delas se tem uma finalidade e um objetivo a se cumprir ao final das disciplinas. Em PFS I o objetivo são as métricas, definição de de riscos e parametrização do planejamento do *software* a ser produzido. Já em PFS II se tem o andamento do desenvolvimento e produção do *software*. Em PFS III é a etapa final dos testes e pretende-se que o *software* já esteja em faze de finalização e entrega.

Considerando esse cenário, a proposta do presente estudo foi definir e aplicar um *framework* que contemple toda a estrutura necessária, desde o conjunto de etapas, tarefas, recursos humanos, boas práticas e artefatos a serem gerados para que os participantes e responsáveis tenham uma boa organização na elaboração de um *software*. O objetivo foi utilizar como referência as principais características das metodologias ágeis já existentes as quais podem possibilitar a criação de um *framework* mais acessível e compatível com o projeto

realizado nas disciplinas de Prática em Fábrica de *Software*. Com isso foi aplicado um processo para gerir e desenvolver um produto ágil onde serão mostrados os resultados com as melhorias, utilizando o *framework* criado. (ISO, 2008)

Sendo assim, o tema proposto segue o princípio de que com a utilização de um *framework* adequado uma equipe está sujeita a maior facilidade na divisão de tarefas, mais integração entre os membros da equipe e uma visão mais ampla do andamento da produção do *software* (SOMMERVILLE, 2011). O trabalho foi importante também a partir do momento que é proposto planejar e organizar o desenvolvimento do *software*, mostrando que foi possível desenvolver de forma simples um projeto de criação de *software*, com entregas regulares e uma documentação adequada, mesmo por uma equipe pequena.

2. OBJETIVOS

2.1. Objetivo geral

- Avaliar um *framework* para gestão e desenvolvimento de *software* ágil, baseado nas especificidades de um projeto em andamento nas disciplinas de Prática em Fábrica de *Software*

2.2. Objetivos específicos

- Identificar na literatura metodologias para gestão e desenvolvimento de *software* baseado nas necessidades do projeto.
- Criar um *framework*, atendendo às necessidades do projeto.
- Validar o *framework aplicado* no projeto, descrevendo os resultados e propondo possíveis melhorias no processo (evolução contínua).

3. METODOLOGIA

O método de pesquisa realizada foi a pesquisa exploratória que estabelece critérios, métodos e técnicas para a elaboração de uma pesquisa e visa oferecer informações sobre o objeto desta e orientar a formulação de hipóteses (CERVO, SILVA e BERVIAN, 2006) na qual foi explorada no ambiente da fábrica e a pesquisa descritiva que realiza-se o estudo, a análise, o registro e a interpretação dos fatos do mundo físico sem a interferência do pesquisador. A finalidade da pesquisa descritiva foi observar, registrar e analisar os fenômenos ou sistemas técnicos, sem, contudo, entrar no mérito dos conteúdos. São exemplos de pesquisa descritiva as pesquisas mercadológicas e de opinião (BARROS e LEHFELD, 2007).

A elaboração do *framework* seguiu pelas seguintes etapas:

3.1. Identificação na literatura de metodologias para gestão e desenvolvimento de *software*.

A obtenção das literaturas foi através de pesquisa bibliográfica em bases de dados com Google Acadêmico, IEEE *Xplore*, Anais de eventos científicos, que dispõe de uma literatura com embasamento em metodologias ágeis entre outros artigos que citem metodologias de projeto de *software*.

Para a localização dessas fontes foram utilizadas palavras chaves como: metodologia ágil, *framework*, manifesto ágil, gestão ágil, processos de gestão de *software* e processos ágeis. Tiveram prioridade na escolha, fontes com datas superiores ao ano de 2011, mas não excluindo as com datas inferiores que tiveram relevância na execução da pesquisa.

3.2. Criação do *framework* para gestão e desenvolvimento de *software*.

Durante a disciplina Prática em Fábrica de *Software* I um grupo de acadêmicos foi formado para aplicação dos conceitos e desenvolvimento de um produto. Após leitura das fontes coletadas, foram selecionadas as atividades, técnicas, artefatos, boas práticas e ferramentas para construção do *framework*. As metodologias embasadas foram selecionadas através de critérios que atendessem uma equipe reduzida e que desse uma melhor oportunidade de aprendizado para todos os integrantes.

3.3. Aplicação do *framework*

A aplicação se deu nas disciplinas de Prática de Fábrica de *Software* na Fábrica de Tecnologia Turing, onde o *framework* foi aplicado na elaboração do projeto EduPlan. A equipe

implantou o *framework* criado em cada disciplina na elaboração do projeto tendo assim um resultado da execução do *framework* no ambiente acadêmico profissional.

3.4. Registro dos resultados, a partir da aplicação do *framework*

Os resultados obtidos ao final de cada *sprint* foram registrados nas ferramentas selecionadas e discutidos com a equipe responsável, como forma de melhorar a interação com a equipe e principalmente de realizar melhorias no *framework*.

Como forma de avaliar a eficiência do *framework* foi aplicado um questionário baseado na metodologia de Avaliação 360º que tem como objetivo avaliar, de forma imparcial para identificar e analisar qual a percepção que as pessoas ao redor de um profissional têm dele.

A metodologia foi adaptada para que a avaliação seja do *framework* ao invés de um funcionário/colaborador, dessa forma, sendo possível através do feedback da equipe de desenvolvimento, identificar e apontar possíveis melhorias no processo.

4. FUNDAMENTAÇÃO TEORICA

Um processo de gestão de projetos é um conjunto de fatores humanos e tecnológicos com o intuito de planejar todo o processo de forma que ocorra com qualidade e sem deixar nenhuma pendência. Pode ser divido em diversas áreas e subáreas como são descritas no SWEBOK (BURGESS, KELLY, et al., 2014).

SWEBOK (*Software Engineering Body of Knowledge* - Corpo de conhecimento de engenharia de *software*) é um documento desenvolvido juntamente com o IEEE (*Institute of Electrical and Electronic Engineers* - Instituto de Engenheiros Eletricistas e Eletrônicos) que serve de referência para as áreas de atuação de engenharia de *software*. Esse documento é classificado de forma hierárquica.

Outro guia importante é o PMBOK (*Project Management Body of Knowledge* - Corpo de Conhecimento em Gerenciamento de Projetos) que trata de diversas áreas que envolvem projeto. Tem o objetivo de melhorar a atuação do profissional e do projeto em questão.

Tendo em vista os problemas que ocorrem no planejamento de um *software*, estes dois guias auxiliam no modo de como cada área é tratada. Como por exemplo, na classificação de processos, que mostra as principais funções e o que é preciso ter para obter um bom planejamento.

4.1. Processo tradicional de desenvolvimento

Os processos tradicionais de desenvolvimento tem como principais modelos: Linear; Cascata, Prototipagem, Incremental, Espiral (PRESSMAN, 2016).

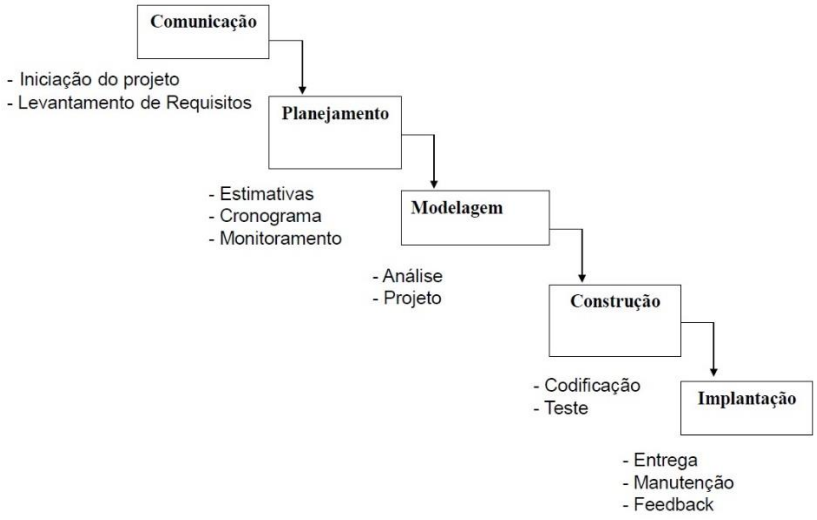
O modelo Linear possui abordagem sistemática sequencial para o desenvolvimento do *software*. Começa no nível de sistema e progride mediante análise de projeto, codificação, teste e manutenção (SBROCCO e MACEDO, 2012).

O modelo Cascata (Figura 1) foi o primeiro processo de desenvolvimento de *software* publicado. Composto de um modelo de fácil entendimento, baseado em uma sequência de etapas. Cada etapa tem associada ao seu término uma documentação padrão que deve ser aprovada para que se inicie a etapa imediatamente posterior (PRESSMAN, 2016).

Na fase de comunicação são estabelecidos os requisitos do produto, é feita a documentação e o estudo da viabilidade. O planejamento determina o cronograma e as estimativas. Depois é realizada a modelagem do sistema quanto à estrutura de dados, a arquitetura do *software*, interfaces e procedimentos. Após o cumprimento das etapas anteriores,

tem início a codificação do produto e os devidos testes. Ao final do ciclo de vida tem-se a entrega do *software*, as devidas manutenções e o feedback do cliente (PRESSMAN, 2011).

Figura 1 - Etapas do modelo cascata

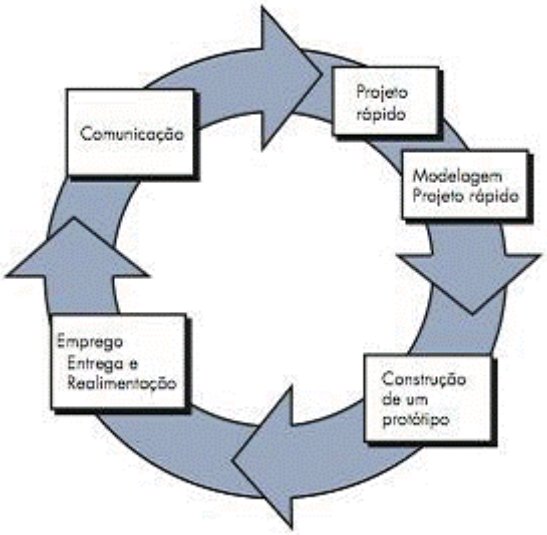


Fonte: PRESSMAN (2016)

No modelo Prototipagem (Figura 2) o desenvolvedor e o cliente encontram-se e definem os objetivos gerais do *software*. Identificam as necessidades conhecidas e delineiam áreas que necessitam de mais definições. Um "projeto rápido" é então realizado (SBROCCO e MACEDO, 2012).

O processo tem início com a comunicação. Em reunião dos integrantes com o cliente são definidos os objetivos, identificados os requisitos e esquematizado o projeto. O projeto rápido foca na representação dos principais aspectos do *software* para os usuários em questão, além disso ele é responsável pela construção do protótipo. Após a entrega do protótipo é solicitado um *feedback* do usuário para eventuais modificações no requisito (PRESSMAN, 2016).

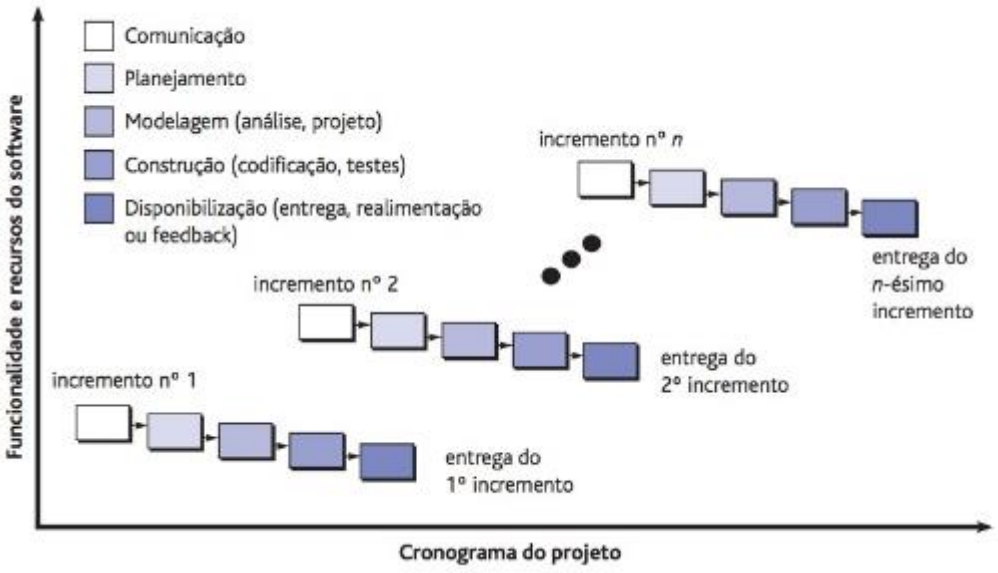
Figura 2 - Modelo prototipagem



Fonte: PRESSMAN (2016)

O modelo incremental (Figura 3) mescla elementos da modelo cascata com elementos da prototipação. O modelo é baseado na divisão de todas as funcionalidades que devem ser alcançadas e as sequências de atividades em incrementos. Ao final de cada iteração o incremento deve ser entregue seguido de um protótipo (SBROCCO e MACEDO, 2012).

Figura 3 - Modelo Incremental



Fonte: PRESSMAN (2016)

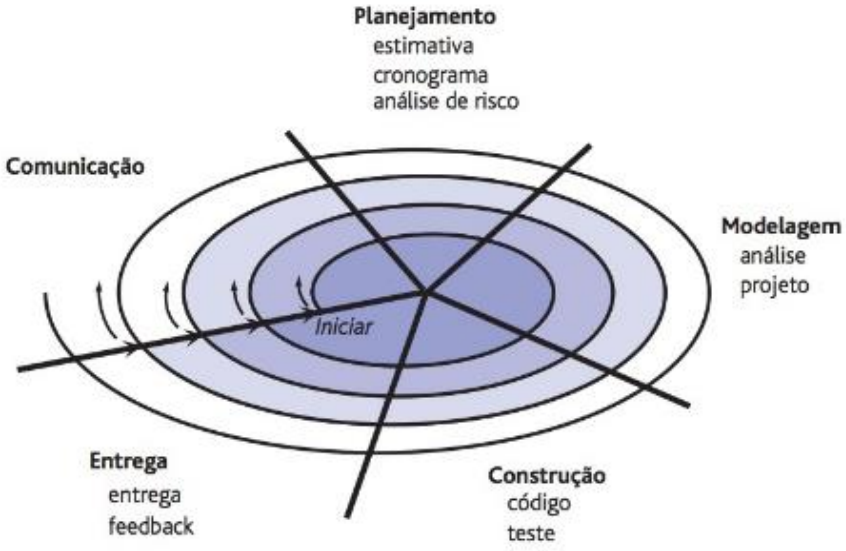
O modelo espiral (Figura 4) fornece a possibilidade para o desenvolvimento rápido de incrementos de *software*s que são divididos em versões.

> Durante as primeiras iterações, podemos produzir, por exemplo, protótipos, até que nas últimas iterações possamos produzir versões cada vez mais completas do sistema. O modelo espiral é desenvolvido em atividades que formam uma estrutura, também chamada de regiões de tarefa (SBROCCO e MACEDO, 2012, p. 65).

As regiões de tarefa fornecem um conjunto de tarefas, que podem ser ajustadas às características necessárias para o projeto de *software*, com a possibilidade de que se tenha mais ou menos tarefas dependendo do porte do produto a ser construído (SBROCCO e MACEDO, 2012).

Trata-se de um processo de Engenharia de *Software* que prega a prática de uma disciplina em que tarefas e responsabilidades são atribuídas a organizações que desenvolvem produtos de *software*. O seu objetivo é garantir que a produção de *software* tenha alta qualidade e esteja de acordo com as necessidades dos usuários finais (SOMMERVILLE, 2011).

Figura 4 - Modelo Espiral



Fonte: PRESSMAN (2016)

4.2. Processo ágil de desenvolvimento

O desenvolvimento ágil tem a finalidade de aperfeiçoar a produção de *software* que tem como base os seguintes princípios (SOMMERVILLE, 2011):

- Indivíduos e interação entre eles mais que processos e ferramentas;
- *Software* em funcionamento mais que documentação abrangente;
- Colaboração com o cliente mais que negociação de contratos;

- Responder a mudanças mais que seguir um plano.

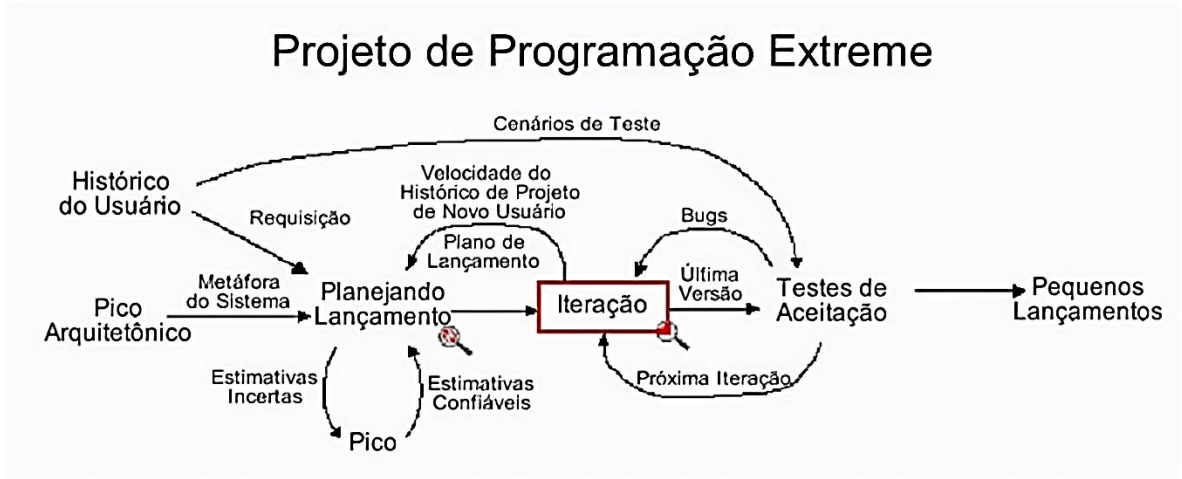
A partir desses princípios se tem condições de evoluir um *software* de forma continua, permitindo uma interação maior com o cliente, onde ele manuseia o *software* a medida em que os incrementos vão sendo disponibilizados (PRESSMAN, 2016).

A seguir estão dispostas as metodologias selecionadas para compor a criação do *framework* de forma a colaborar com as necessidades encontradas no decorrer do estudo:

4.2.1. XP

O XP (Figura 5), é voltado para o desenvolvimento do *software* no qual tudo é feito ao máximo, são empregadas as boas práticas de engenharia de *software*. Proporcionam, por conseguinte em um bom produto. O extremismo remete ao uso máximo dessas boas práticas que se apoiam e criam uma sinergia. (PRIKLADNICKI, WILLI e MILANI, 2014).

Figura 5 - Processo do Modelo XP



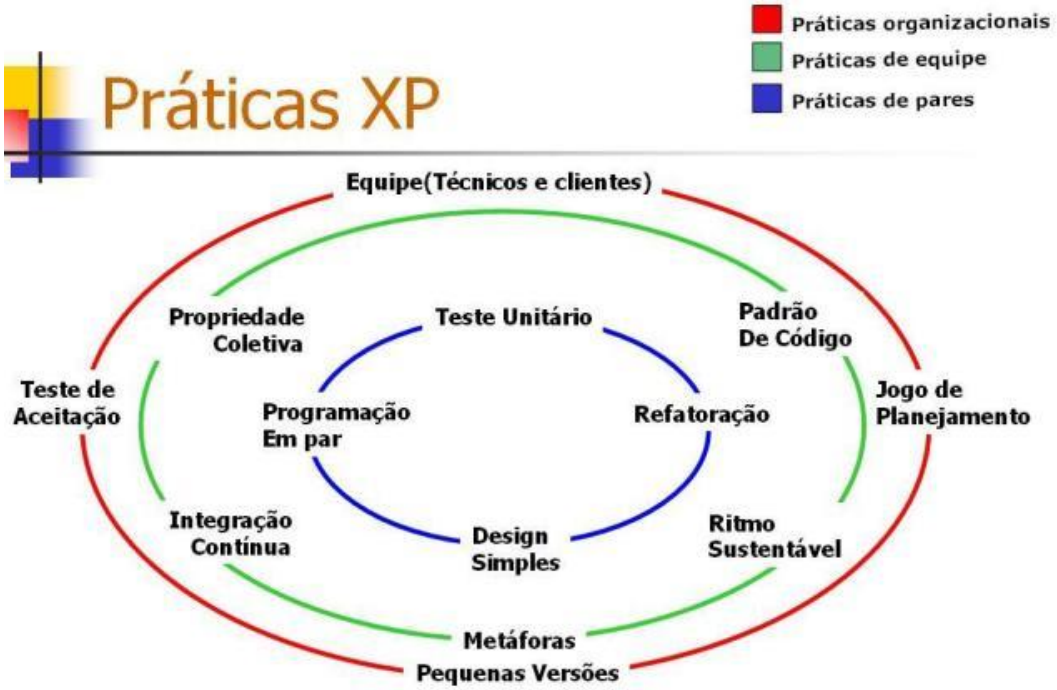
Fonte: ROCHA (2014)

Inicia-se por um plano geral que tem o objetivo de entender o problema como um todo e dividi-lo em partes para serem solucionados e a partir disso dar início ao desenvolvimento.

> O objetivo principal do XP é levar ao extremo esse conjunto de práticas que são ditas como boas na engenharia de *software*. Entre elas podemos citar o teste, visto que procurar defeitos é perda de tempo, nós temos que constantemente testar, se perder tempo com código sujo é ruim, melhoraremos o nosso código sempre que uma nova mudança for feita. Portanto, como podemos notar todas as coisas práticas do XP são levadas ao extremo (MEDEIROS, 2013, p. 1).

Considerado o objetivo citado e as práticas abordadas (Figura 6) com maior foco na Engenharia de *Software*, notou-se potencial para ser integrado à proposta desse trabalho, por ser uma metodologia que se adapta facilmente com o Scrum que será tratado no próximo tópico.

Figura 6 - Praticas da metodologia XP

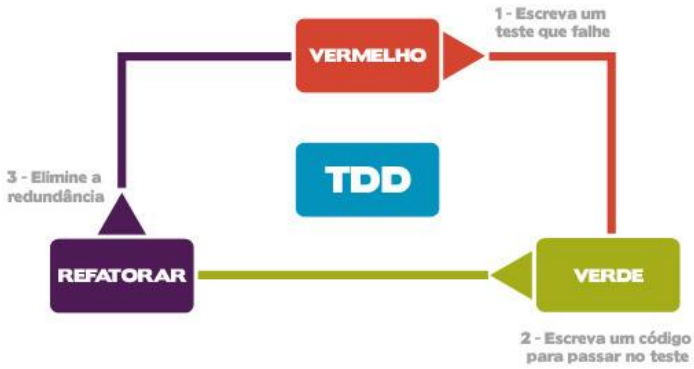


Fonte: ROCHA (2014)

4.2.2. TDD

O TDD (*Test-Driven Development* – Desenvolvimento orientado a testes) (Figura 7) é uma técnica de desenvolvimento orientada a testes que tem como objetivo identificar e corrigir falhas durante o projeto. A premissa é criar um caso de teste antes de começar a desenvolver o código. São baseadas em três etapas, na primeira, chamada de teste é criado um código que falhe, pois o teste ainda não foi implementado; no segundo, chamado de código, é implementado um código que seja validado no teste; no final o código é refatorado, porém agora atribui qualidade. Seguindo essas etapas de processo o desenvolvedor consegue melhorar o teste, as chances do código já estar correto são bem maiores. Além disso, o código criado é simples pois este método foge de complexidades e deixa o excesso de acoplamentos de fora da implementação (NAZÁRIO e RUFINO, 2015).

Figura 7 - Processo TDD



Fonte: Retirado do artigo TDD: fundamentos do desenvolvimento orientado a testes - DEVMEDIA

4.2.3. Scrum

O Scrum (Figura 8) tem como base um processo adaptável em que as funcionalidades mais importantes são priorizadas e ao longo do processo são entregues as partes já concluídas. Porém, em caso de necessidade de mudanças não há complicações (SBROCCO e MACEDO, 2012).

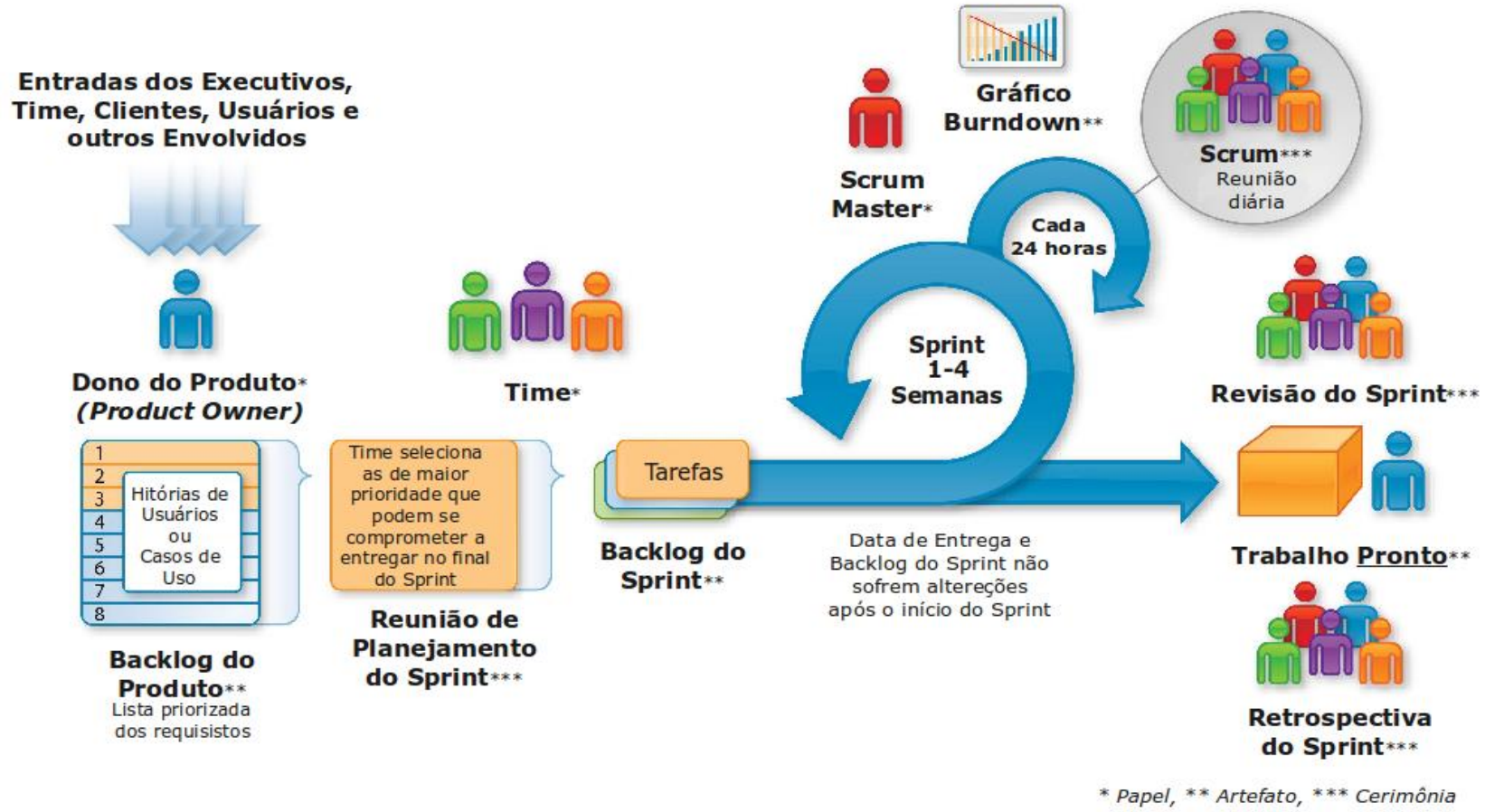
> Para Prikladnicki, Willi E Milani (2014, p. 113) "Em cada *sprint*, o trabalho é priorizado a partir de uma lista de requisitos, chamada de Backlog do Produto. O desenvolvimento a partir da priorização dos requisitos garante que se trabalhe no que tem mais prioridade e valor para o cliente. Ao final de cada *sprint*, um conjunto de funcionalidades prontas é entregue."

Fundamentando-se nessa premissa, é significativo que ao criar o *framework* essas *sprint*'s tenham o tempo correto para cada atividade e para isso será utilizado não um tempo determinado, mas estimado. Tempo que pode ser modificado a qualquer momento se eventualmente surgir impedimento, sem a necessidade de passar o problema para a próxima *Sprint*. O scrum também define algumas práticas como os papéis de cada indivíduo na equipe, a qual se divide em três papéis: O product owner, o scrum master e o TEAM ou equipe.

Os artefatos Product Backlog, *Sprint* Backlog e o Burndown Chart são gerados a partir de reuniões que acontecem durante uma *Sprint* o Daily Meeting, a *Sprint* Review, o *Sprint* Planning Meeting e a *Sprint* Retrospective tornando em um ciclo de etapas para cada *Sprint*.

Segundo Schwaber, o SCRUM baseia-se em seis características: flexibilidade dos resultados, flexibilidade dos prazos, times pequenos, revisões frequentes, colaboração e orientação a objetos (SBROCCO E MACEDO,2012).

Figura 8 - Processo Scrum



Fonte: Desenvolvimentoagil.com (2014)

A escolha do Scrum se deu pelo seu foco na gestão e planejamento de projetos de *software*, que contribuirá para uma melhor organização no que se propõe o *framework* criado.

O Scrum possui 5 valores (Figura 9) que tem o objetivo de serem um meio de viabilizar entregas de produtos de *software* com uma qualidade satisfatória e ao mesmo tempo melhorar o convívio no ambiente de trabalho.

> Quando os valores de comprometimento, coragem, foco, abertura e respeito são assumidos e vividos pelo Time Scrum, os pilares do Scrum de transparência, inspeção e adaptação tornam-se vivos e constroem a confiança para todos. Os membros do Time Scrum aprendem e exploram estes valores à medida que trabalham com os eventos, papéis e artefatos do Scrum (SCHWABER e SUTHERLAND, 2016).

Figura 9 - Valores Scrum

Fonte: CRUZ (2016)

4.2.4. OpenUP

O Processo Unificado Aberto (Figura 10), do inglês *Open Unified Process* - OpenUP foi selecionado por ser uma metodologia que fornece características como abordagens iterativa e incremental em um ciclo de vida estruturado, com um conceito ágil e focado na colaboração do desenvolvimento de *software*. O OpenUP é uma metodologia livre de ferramentas e de baixo formalismo que pode ser estendido a uma variada gama de tipos de projetos e não apenas ao desenvolvimento de *software* (MEIRA, 2010).

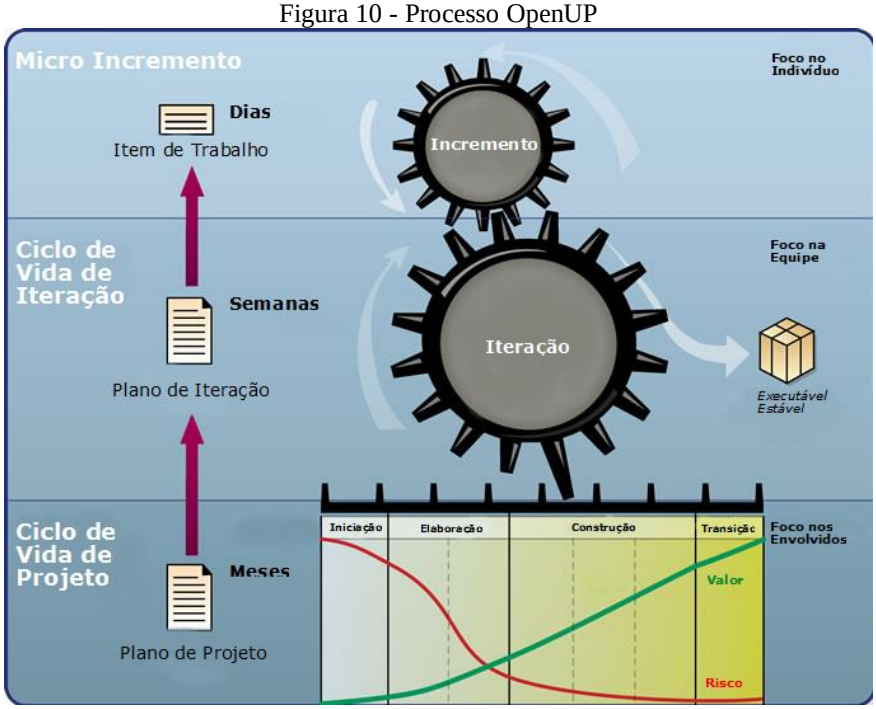
O OpenUP é dividido em quatro etapas (PRIKLADNICKI, WILLI e MILANI, 2014):

- Iniciação: eliminar riscos de negócio.
- Elaboração: eliminar riscos arquiteturais e técnicos.
- Construção: eliminar riscos de não construção das funcionalidades principais.
- Transição: eliminar riscos de homologação, testes beta, implantação do sistema em produção e migrações de sistemas legado.

Os papéis são distribuídos em seis funções. Cada um dispõe de uma atividade fundamental para o processo. São eles: os *stakeholders*; gerente de projeto; arquiteto; analista; desenvolvedor e tester. (PRIKLADNICKI, WILLI e MILANI, 2014).

As práticas que o método exerce dispõem de um projeto evolutivo, que constantemente melhora seu processo. Contam com desenvolvimento dirigido por testes e casos de uso para manter sempre de acordo com os requisitos, integração contínua e gestão de mudanças.

Figura 10 - Processo OpenUP



Fonte: MEIRA (2010)

4.2.5. Lean

O Lean (Figura 11) apesar de ser considerado um modelo de processo ágil de *software* no ramo da Engenharia de *Software*, foi criado no século XIX na produção de tecidos no Japão, o objetivo do processo era manter pessoas observando as máquinas e parando a produção ao encontrar defeitos.

Com isso as boas práticas que surgiram a partir disso, foram as organizações das equipes quanto ao projeto por ser baseada em equipes multidisciplinares, nas quais todos os desenvolvedores tem conhecimento necessário para construir o produto. Forma-se assim um ambiente de melhoria pois a cada defeito encontrado, mudanças ocorrem para que seja melhorado. O fluxo de entrega é constante para manter a qualidade do produto e ter sempre no processo a participação do cliente. Além disso, um fator importante desta prática é eliminar o

desperdício, categorizar de forma a saber o que deve ser eliminado para evitar o desperdício em um projeto de *software* (PRIKLADNICKI, WILLI e MILANI, 2014).

Figura 11 - Modelo Lean

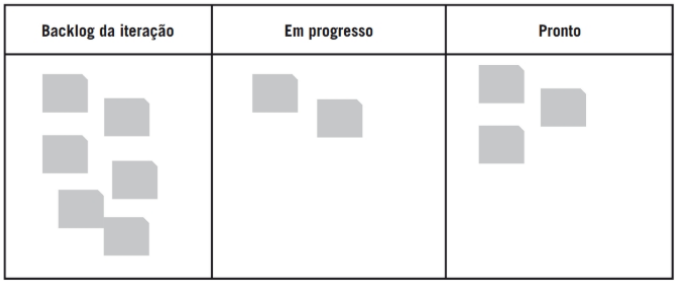
Fonte: OLIVEIRA (2017)

4.2.6. Kanban

O Kanban (Figura 12) é um método que utiliza como princípio os artefatos de maior "valor" para o projeto. Segue o fluxo para que esse artefato seja produzido com qualidade. Seu objetivo é evidenciar em um mapa visual os problemas que surgem e dar uma visão melhor de como lidar com a situação para que o fluxo continue. Ao surgir um problema, esse mapa tende a se modificar para favorecer o aperfeiçoamento, e seguir assim, um fluxo evolucionário.

Seguem três etapas: a primeira é a discussão entre a equipe sobre as características, riscos, priorização e outros critérios relevantes; em seguida, são feitas perguntas sobre o fluxo do processo; por último, são projetadas no mapa mental as informações decorrentes das perguntas feitas na primeira e segunda etapa, a fim de detalhar todo o processo. (PRIKLADNICKI, WILLI e MILANI, 2014).

Figura 12 - Modelo Kanban



Fonte: PRIKLADNICKI, WILLI e MILANI (2014)

4.3. Processo híbrido

Os tópicos que foram discorridos anteriormente trataram de alguns processos já criados e estudados na gestão e desenvolvimento de *software*. Porém, a utilização de processo híbrido adquiriu um papel importante na área de gestão, pelo fato de que ao utilizar um método padrão, a criação de um método encaixe no projeto de uma forma mais viável, pois esse tipo de processo seleciona os métodos ágeis que sejam necessários para o projeto, e é moldado de acordo com as necessidades do time de desenvolvimento e gestão. Dando um melhor resultado para os quesitos necessários para se obter um bom produto e com uma boa qualidade.

Como exemplo, têm-se o nosso próprio projeto, que é baseado neste processo onde estudamos os melhores e mais utilizados métodos e selecionamos os que mais se encaixavam com o perfil do projeto conseguindo um melhor resultado, pois foi criado especialmente para um tipo de projeto e equipe.

Figura 13 - Metodologia de Desenvolvimento de *Software*

Fonte: DataSUS (2018)

4.4. Técnicas e artefatos

4.4.1. *Elevator statement*

Elevator Pitch (Figura 14) é uma técnica muito utilizada por vendedores para efetivação das vendas, e consiste em uma pequena conversa com duração de apenas um minuto (MARQUES, 2017).

Figura 14 - Elevator Pitch

Fonte: MARQUES (2017)

4.4.2. Entrevista

Segundo Egger (2017), antes de se fazer uma entrevista devemos entender o público do produto, criar um roteiro, definir um momento para falar com o cliente e definir uma amostragem para validar suas hipóteses. Todas as etapas sendo importantes para que a entrevista seja concluída com sucesso, e o cliente saia satisfeito com o que foi proposto.

Além disso na hora da entrevista é importante seguir o roteiro, mas sem interromper o cliente respeitar o tempo do cliente.

Roteiro de entrevista com o cliente.

- Apresentação da equipe para o cliente
- Apresentação do produto para o cliente. O que ele faz? Suas funcionalidades? Sua importância? Por que seria melhor com ele?
- O nosso produto adequa as necessidades do cliente, no quesito funcionalidade, o produto seria de ajuda para o cliente.
- O cliente acrescentaria mais funcionalidades? Ou é suficiente.
- Qual a opinião do cliente sobre tudo que foi discutido.
- O cliente conseguiu esclarecer todas as dúvidas.

4.4.3. Cenários (divisão dos fluxos)

“Um caso de uso captura um contrato... [que] descreve o comportamento do sistema sob várias condições, à medida que o sistema responde a uma solicitação de um de seus envolvidos...”. Alistair Co- ckburn, ou seja um caso de uso conta como e quando o usuário interage no sistema.

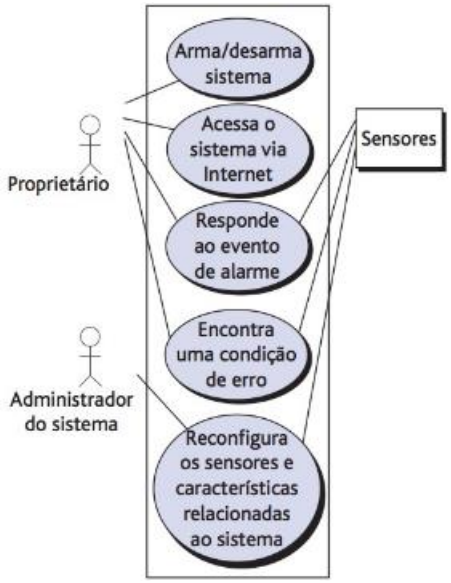
Na elaboração dos casos de uso a primeira etapa é definir os atores que vão compor o sistema, cada uma desempenhando seu papel, e após identificado os atores, os casos de uso serão elaborados (PRESSMAN, 2016).

Jacobson sugere algumas perguntas que devem ser respondidas por um caso de uso:

- Quem é o ator primário e quem é (são) o(s) ator(es) secundário(s)?
- Quais são as metas do ator?
- Que precondições devem existir antes de uma jornada começar?
- Que tarefas ou funções principais são realizadas pelo ator?
- Que exceções poderiam ser consideradas à medida que uma jornada é descrita?
- Quais são as variações possíveis na interação do ator?
- Que informações de sistema o ator adquire, produz ou modifica?
- O ator terá de informar o sistema sobre mudanças no ambiente externo?
- Que informações o ator deseja do sistema?
- O ator gostaria de ser informado sobre mudanças inesperadas?

Entretanto o caso de uso (Figura 15) tem como objetivo contar uma história detalhada sobre o sistema, então deve conter todas as informações que envolvem os atores tanto primário quanto secundário no sistema.

Figura 15 - Caso de uso



Fonte: PRESSMAN (2011)

4.4.4. História de usuário

Em metodologia ágil é muito comum utilizar as histórias de usuário que capturam o que um usuário faz ou necessita fazer como parte de sua função de trabalho, É uma técnica que captura o "quem", "o quê" e "por quê" de um requisito (ALFF, 2019).

“Uma história de usuário só é considerada válida se seguir todas as características do modelo INVEST” (BRASILEIRO, 2017).

O INVEST é a junção das inicias das principais características que uma história de usuário deve ter. Independente, Negociável, Valiosa, Estimável, Pequena (*Small*) e Testável (BRASILEIRO, 2017).

Figura 16 - INVEST

Fonte: BRASILEIRO *(*2017*).*

5. RESULTADOS

O estudo de modelos de processos de desenvolvimento de *software* permitiu a identificação de elementos necessários para a elaboração de um *framework* híbrido que atendesse às necessidades de um projeto, executado durante as disciplinas de Prática em Fábrica de *Software*. Na oportunidade, o grupo colaborou com as informações e conhecimentos, evoluindo o produto gradativamente ao decorrer do projeto.

Por isso a importância de utilizar não somente um *framework,* mas utilizar a base de cada um como forma de priorizar as problemáticas e dificuldades encontradas pelo grupo, com o intuito de melhorar a qualidade e interação do projeto de acordo com sua necessidade. Dessa forma não só na fase de documentação mas também nas fases seguintes toda a equipe tenha as informações que precisam quando chegar a hora de programar e testar o produto.

5.1. *Framework* elaborado

O *framework* foi elaborado com base em diferentes modelos. A sua essência foi a adoção de metodologias ágeis. Ele foi estruturado em seis etapas:

- Requisitos: É onde a equipe entende o real problema dos *stakeholders*, e conseguem buscar soluções para este problema, levantando os requisitos necessários para desenvolver o projeto.
- Planejamento da *Sprint*: É quando acontece as reuniões da equipe para apurar o desenvolvimento do projeto.
- Codificação: Os programadores, encontram as ferramentas que vão ser utilizadas, e a partir dos processos anteriores acontece o desenvolvimento do projeto.
- Testes: Após a codificação são implementados os testes para ter certeza que está tudo correto para o cliente.
- Transição: Realizar os ajustes na funcionalidade e qualidade geral da nova versão do *software*, além de realizar o treinamento e capacitação do usuário, tendo como objetivo a melhoria da operação desse *software*.
- Retrospectiva: A equipe verifica a necessita de mudanças no processo para a próxima *sprint*.

Figura 17 - Processo FGD – Framework de Gestão e Desenvolvimento

Fonte: Elaborado pelos autores

Para cada etapa do processo, um conjunto de atividades foi identificado com as respectivas técnicas, boas práticas, artefatos e ferramentas (Quadro 1, 2, 3, 4, 5 e 6).

Ressalta-se que nas Quadros as linhas tachadas foram as que houveram mudanças e foram melhoradas ao decorrer do processo dando lugar as que estão em escrita verde.

Quadro 1 - Etapa de Requisitos

Etapa: Requisitos				
Atividade	**Técnica**	**Boas práticas**	**Artefatos**	**Ferramentas[1]**
Definição da visão	Elevator statement (BENASSI, FERREIRA JUNIOR e AMARAL, 2012)	Pitch/Comunicação	*Vision Box*	Word versão 2016
	Entrevista (PRESSMAN, 2011)	Comunicação efetiva	Documento de visão	Word versão 2016
Identificação dos Casos de uso	UML (PRESSMAN, 2011)	Divisão dos fluxos	Diagrama de casos de uso	Astah versão 8.0
~~Histórias de usuário~~	~~Jogo de planejamento~~	~~INVEST~~	~~Cartões de histórias de usuário~~	~~Trello~~
~~Priorização de requisitos~~	~~Jogo de planejamento~~	~~Definição de status de prioridade~~	~~Cartões de histórias de usuário~~	~~Trello~~
Elaboração das Histórias de usuário	Histórias de usuário (PRIKLADNICKI, WILLI e MILANI, 2014)	INVEST (BRASILEIRO, 2017)	Cartões de histórias de usuário incluídas no product backlog	Kanban do Gitlab
Priorização de requisitos	Numeração de 1 a 5, onde 5 indica a maior prioridade	Priorização baseada em valor (SBOK, 2016)	Cartões de histórias de usuário	Kanban do Gitlab

Fonte: Elaborado pelos autores

[1] As ferramentas propostas foram aquelas utilizadas durante o Projeto EduPlan. Ressalta-se que estas podem ser substituídas.

Quadro 2 - Etapa de planejamento da *sprint*

Etapa: Planejamento da *sprint*				
Atividade	**Técnica**	**Boas práticas**	**Artefatos**	**Ferramentas**
~~Planejamento da *sprint*~~	~~Cerimônia (SBROCCO e MACEDO, 2012)~~	-	~~Baclog do produto~~	~~Trello~~
~~Meta da *sprint*~~	~~Cerimônia~~	~~Respeitar a capacidade da equipe~~	~~Backlog da *sprint*~~	~~Trello~~
Planejamento da *sprint*	Cerimônia (SBROCCO e MACEDO, 2012)	Sessões de planejamento com grupos de usuário	Backlog da *sprint*	Kanban do Gitlab
Definição da Meta da *sprint*	Cerimônia	Respeitar a capacidade da equipe	Backlog da *sprint*	Kanban do Gitlab
Definição da estimativa	Planning poker	Jogo de planejamento	Cartões das histórias de usuário	Kanban do gitlab

Fonte: Elaborado pelos autores

Quadro 3 - Etapa de codificação

Etapa: Codificação				
Atividade	**Técnica**	**Boas práticas**	**Artefatos**	**Ferramentas**
Elaboração do Design de interface	Simplicidade, de forma intuitiva	Comunicação	Protótipos de tela	*Framework* de interface
Codificação do incremento	Programação em par (SBROCCO e MACEDO, 2012)	Padronização do código	Código-fonte	Angular JS

Fonte: Elaborado pelos autores

Quadro 4 - Etapa de testes

Etapa: Testes				
Atividade	**Técnica**	**Boas práticas**	**Artefatos**	**Ferramentas**
Definição dos Teste de unidade	Pseudocontrolador (PRESSMAN, 2011)	Foco no componente alvo	Plano de teste	JUnit
Definição dos Teste de aceitação	Cenários	Reunião de revisão do produto	Critérios de aceitação	Aplicação (Dado, Quando, Então)
Definição do Teste de performance	Sobrecarga	Conduzido pelos desenvolvedores	Relatório de tempo de resposta	JMeter
Revisão do produto	Cerimônia	Inspeção, transparência e adaptação (SBOK, 2016)	Incremento aprovado pelo P.O	Reunião

Fonte: Elaborado pelos autores

Quadro 5 - Etapa de transição

Etapa: Transição				
Atividade	**Técnica**	**Boas práticas**	**Artefatos**	**Ferramentas**
Análise de mudanças	Processo de aprovação de mudanças	Flexibilidade e estabilidade	Mudança aprovada ou reprovada – Backlog atualizado	Reuniões
~~Teste beta~~	~~Feito sem roteiro definido~~	~~Conduzido pelos usuários~~	~~Casos de teste~~	~~Aplicação~~
Treinamento dos usuários	Apoio prático in loco	Sessões presenciais de treinamento	Material de suporte	Produto funcional
Implantação	Implantação contínua	Estratégias de implantação	Entregáveis do produto	Produto funcional

Fonte: Elaborado pelos autores

Quadro 6 - Etapa de retrospectiva

Etapa: Retrospectiva				
Atividade	**Técnica**	**Boas práticas**	**Artefatos**	**Ferramentas**
Retrospectiva	Cerimônia	Comunicação, adaptabilidade	Lições aprendidas	Reunião

Fonte: Elaborado pelos autores

5.2. Aplicação do *framework* no projeto Eduplan

O *framework* foi aplicado em 26 *sprint*s do projeto Eduplan. A equipe foi composta por 8 membros, que assumiram os papeis de scrum master, product owner e equipe (Quadro 7).

Quadro 7 - Mapa das *sprints*

SPRINT	**INÍCIO DA SPRINT**	**FIM DA SPRINT**	**OBJETIVOS DA SPRINT**
1	FEV/18 DIA 05	FEV/18 DIA 19	Levantar de requisitos do produto
2	FEV/18 DIA 26	MAR/18 DIA 05	Analisar requisitos e estimativa (*planning poker*)
3	MAR/18 DIA 19	ABR/18 DIA 01	Elaborar documento de visao
4	ABR/18 DIA 02	ABR/18 DIA 15	Elaborar casos de uso e histórias de usuário
5	ABR/18 DIA 16	ABR/18 DIA 29	*Elaborar vision box* e *elevator estatement*
6	ABR/18 DIA 30	MAI/18 DIA 13	Elaborar Protótipo das telas
7	MAI/18 DIA 14	MAI/18 DIA 27	Analisar de riscos
8	MAI/18 DIA 28	JUN/18 DIA 10	Analisar de riscos
9	AGO/18 DIA 12	AGO/18 DIA 26	Elaborar documento de arquitetura
10	AGO/18 DIA 27	SET/18 DIA 09	Elaborar diagrama de implantação
11	SET/18 DIA 10	SET/18 DIA 23	Revisar a documentação

12	SET/18 DIA 24	OUT/18 DIA 07	Modelar o banco de dados
13	OUT/18 DIA 08	OUT/18 DIA 21	Criar tela de login e criar docente
14	OUT/18 DIA 22	NOV/18 DIA 04	Criar novo plano e editar plano
15	NOV/18 DIA 05	NOV/18 DIA 18	Listar docente, deletar docente
16	NOV/18 DIA 19	DEZ/18 DIA 02	Deletar plano e deletar docente
17	DEZ/18 DIA 03	DEZ/18 DIA 16	Visualizar plano e editar plano
18	FEV/19 DIA 17	MAR/19 DIA 02	Imprimir plano
19	MAR/19 DIA 03	MAR/19 DIA 16	Versionamento com gitlab e documentação da política de versionamento
20	MAR/19 DIA 17	MAR/19 DIA 30	Elaborar plano de teste e caso de teste
21	ABR/19 DIA 01	ABR/19 DIA 13	Executar teste unitário
22	ABR/19 DIA 14	ABR/19 DIA 27	Teste de performance
23	ABR/19 DIA 28	MAI/19 DIA 11	Teste de interface
24	MAI/19 DIA 12	MAI/19 DIA 25	Implantar o devops
25	MAI/19 DIA 26	JUN/19 DIA 08	Executar teste de aceitação
26	JUN/19 DIA 09	JUN/19 DIA 22	Encerramento

O objetivo do Eduplan foi desenvolver um *software* onde o docente ao fazer o plano de ensino, só será necessário preencher campos específicos, diminuindo o tempo gasto do docente, e facilitando o gestor a verificar os planos de ensino.

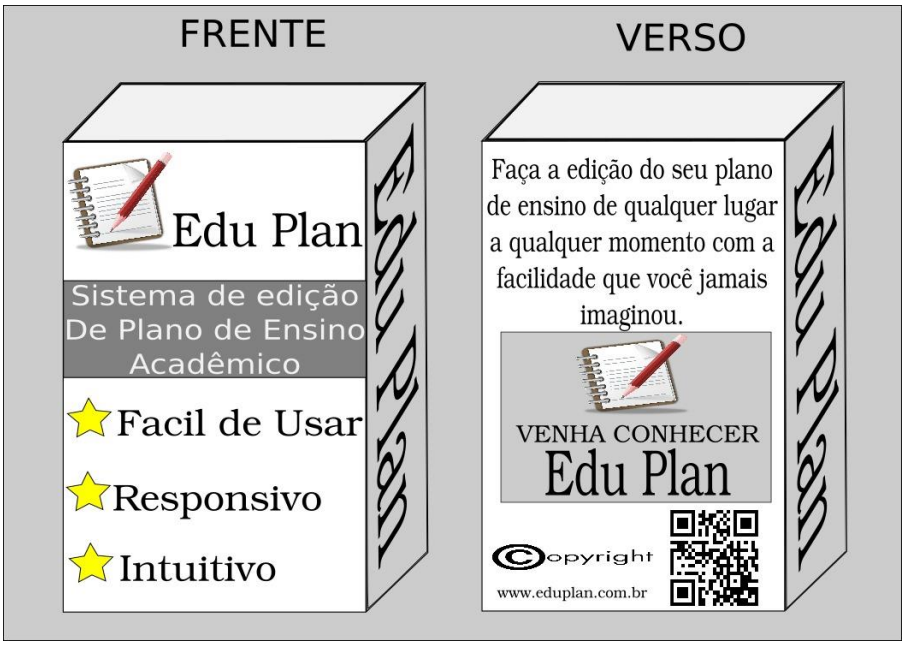
A aplicação das atividades, técnicas, boas práticas e ferramentas do *framework* gerou os seguintes artefatos, organizados de acordo com as etapas do processo.

1) Vison Box (definição da visão)

O Vision Box é uma dinâmica que deve ser feita (se possível) com os *stakeholders*, time, usuários e clientes. É uma prática lúdica que os participantes montam uma caixa de produto apresentando:

- Nome do produto
- Três ou quatro pontos chaves para vender o produto
- Principais funcionalidades
- Principais requisitos operacionais

Figura 18 - Vision box



Fonte: Elaborado pelos autores

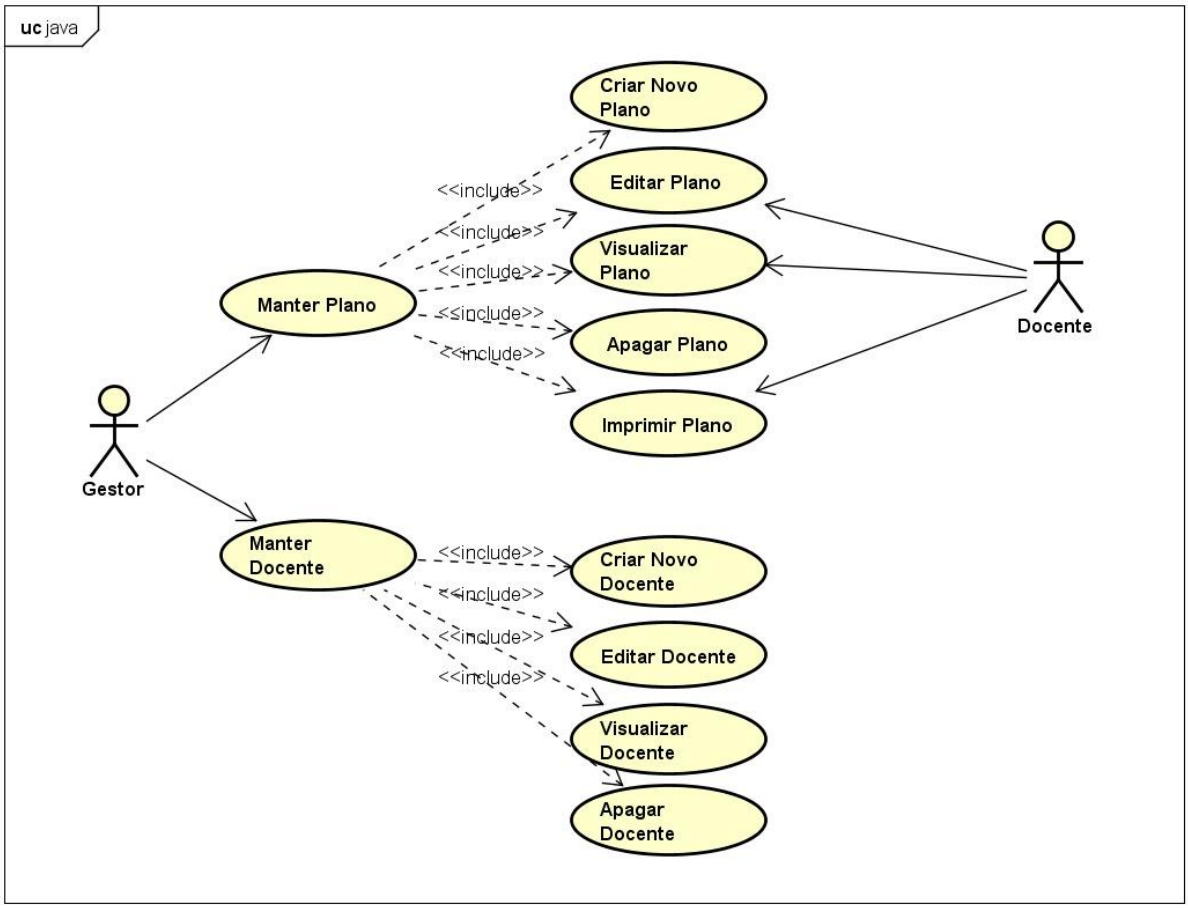
2) Documento de visão (definição da visão)

O documento de visão está no apêndice A.

3) Diagramas UML (identificação dos casos de uso)

Do conjunto de diagramas identificados, somente parte foi selecionada para apresentar aqui (Figura 19).

Figura 19 - Caso de uso



Fonte: Elaborado pelos autores

4) Histórias de usuário (elaboração das histórias de usuário)

Somente parte das histórias de usuário foi selecionada para incluir aqui (Quadro 8).

Quadro 8 - Histórias de usuário

2. CRITÉRIOS DE ACEITE

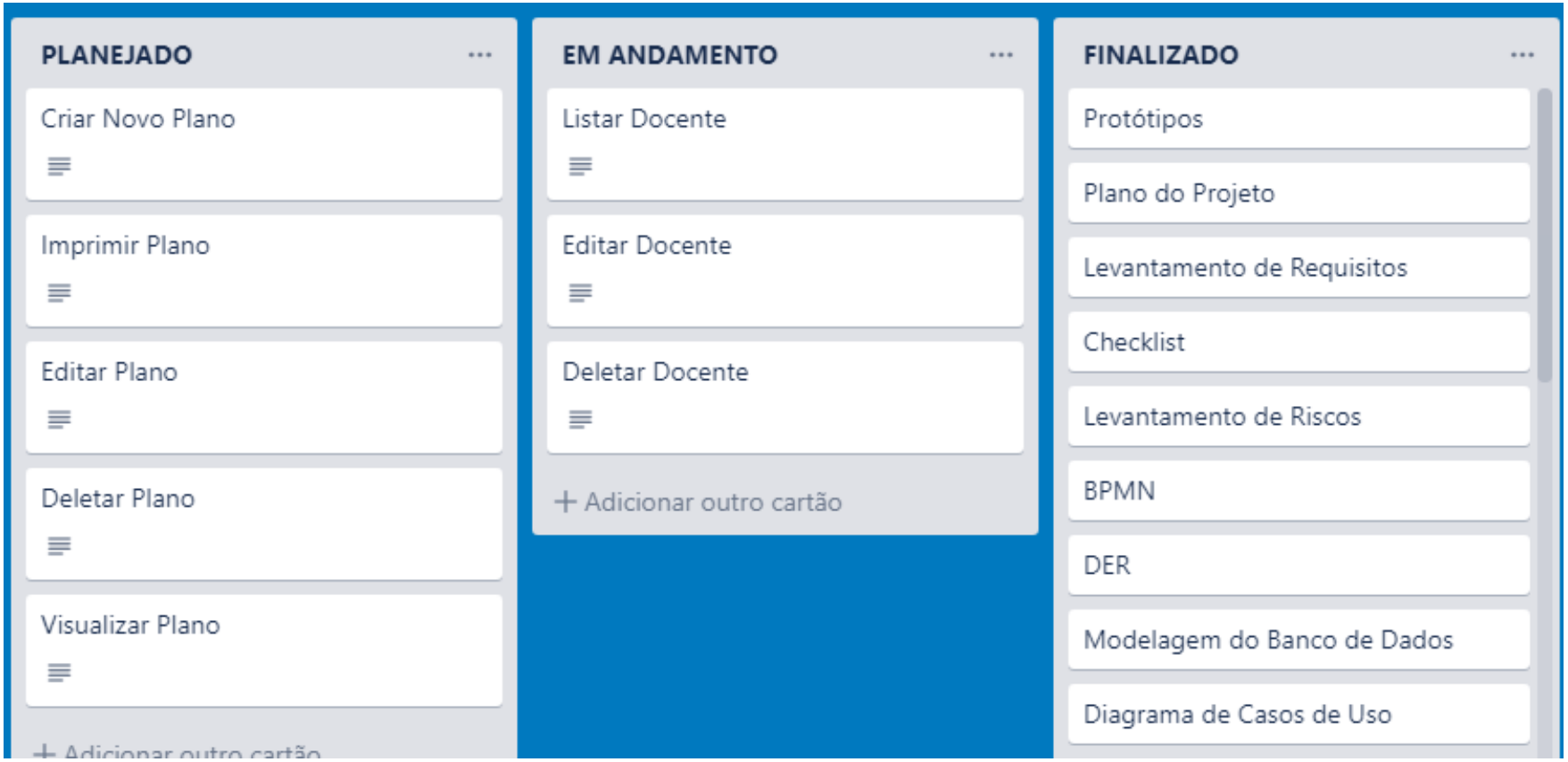
Cenário 01	**Login**
Descrição	COMO Diretor/Docente DEVO fazer login PARA acessar as funcionalidades do sistema. Fluxo Principal: COMO Diretor /Docente DEVO inserir meus dados de acesso E clicar em 'login' ENTÃO o sistema confirmará a solicitação E redirecionará para a página de início. Fluxo de Exceção: COMO Diretor /Docente DEVO inserir meus dados de acesso E clicar em 'login' E o sistema ao verificar a informação não encontrar as mesmas no BD ENTÃO o modal "Login/Senha incorreto".
Cenário 02	**Cadastrar Usuário**
Descrição	COMO Diretor DEVO cadastrar docentes PARA adicionar novos docentes ao sistema. Fluxo Principal: COMO Diretor DEVO logar no sistema E clicar no menu ‘Usuários’ E clicar em ‘Novo’ E ser encaminhado para a página de cadastro E preencher as informações solicitadas E selecionar 'Salvar' ENTÃO o sistema enviará a informação para o Banco de Dados E retornará o modal "Salvo com Sucesso" Fluxo de Exceção: COMO Diretor DEVO logar no sistema E clicar no menu ‘Usuários’ E clicar em ‘Novo’ E ser encaminhado para a página de cadastro E preencher as informações solicitadas E selecionar 'Salvar' E o sistema enviará a informação para o Banco de Dados E não foi possível salvar a informação ENTÃO será retornado o modal 'Falha ao salvar, tente novamente'

Fonte: Elaborado pelos autores

***5) Backlog* do produto** (elaboração das histórias de usuário)

Na imagem (Figura 20) estão descritas as funcionalidades e artefatos em seus respectivos status de execução.

Figura 20- Backlog do produto

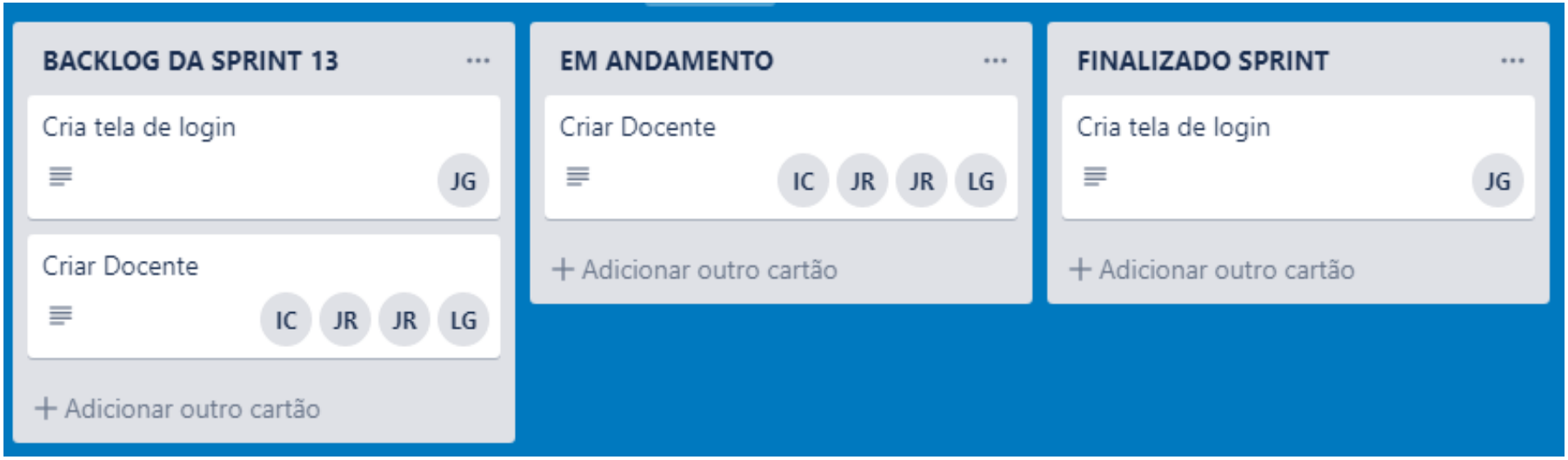


Fonte: Elaborado pelos autores

6) Backlog da *Sprint* (planejamento da *sprint*)

A imagem (Figura 21) mostra o planejamento da *Sprint* 13.

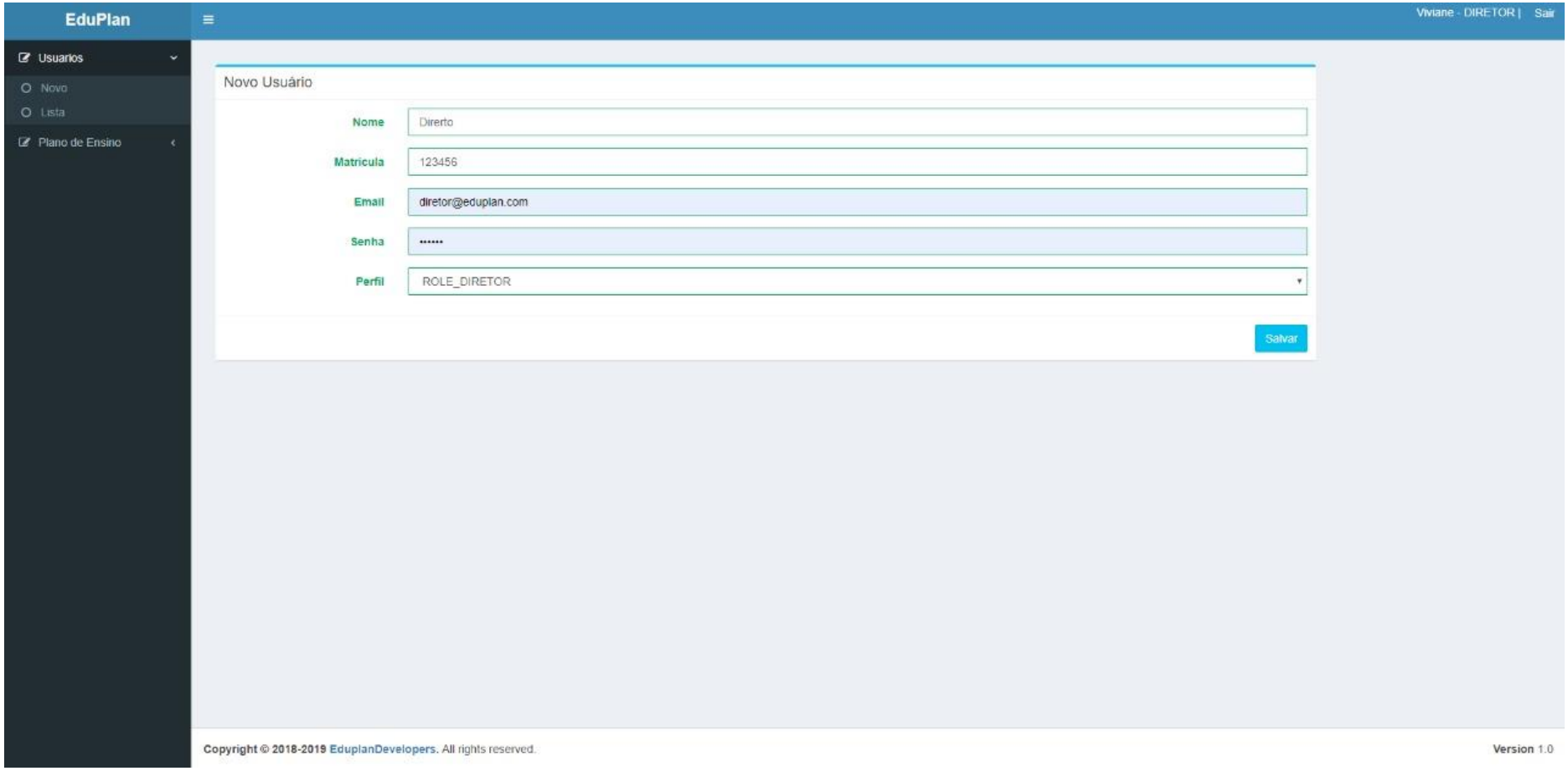
Figura 21 - Backlog da sprint *13*



Fonte: Elaborado pelos autores

7) Protótipos de tela (elaboração do design de interface)

Figura 22 - Tela de cadastro de usuário

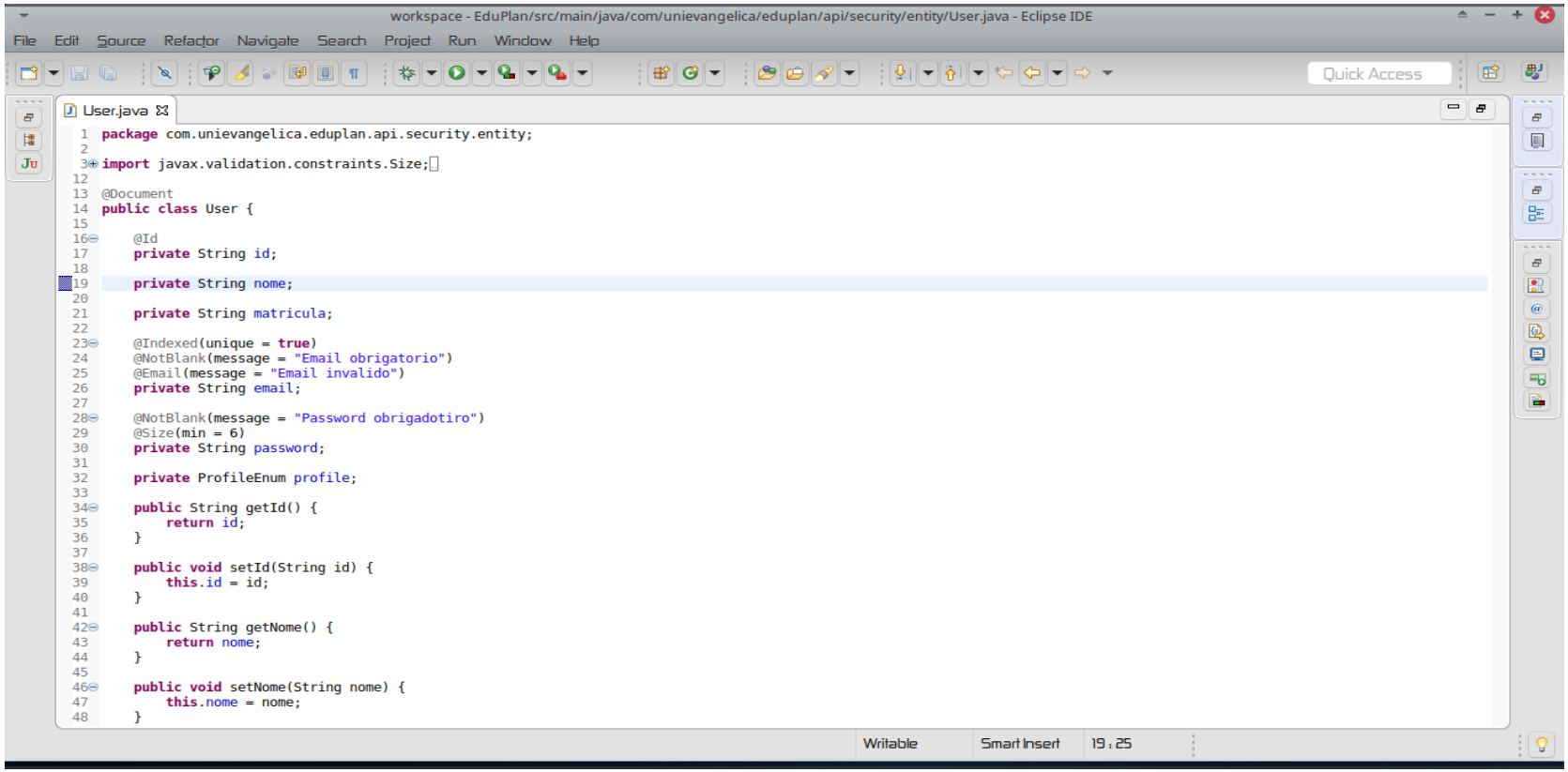


Fonte: Elaborado pelos autores

8) Código fonte (codificação do incremento)

Parte do código foi selecionado para ser apresentada aqui (Figura 23).

Figura 23 - Trecho da implementação do cadastro de usuários



Fonte: Elaborado pelos autores

9) Plano de teste (elaboração dos testes)

O planejamento dos testes foi feito na *Sprint* 20 onde foram selecionados os testes de unidade, performance, interface e aceitação. O plano de testes está no apêndice B.

5.3. Avaliação do *framework*

A avaliação do *framework* foi feita através de um questionário, disponibilizado na plataforma Google *Forms* (disponível no link: < https://forms.gle/5NvSa2aqGwPTELdn6>). Os integrantes da equipe avaliaram o processo como um todo, desde, o modelo de comunicação, as atividades e etapas, completude das informações e sua consistência, satisfação com a maneira que foram executadas as *sprints* e também foi dada uma a abertura para sugestões de melhorias e mudanças.

O questionário foi composto por onze questões, sendo elas respondidas por quatro membros do projeto.

1. De acordo com as etapas do processo implementado, na sua opinião precisa de uma adequação quanto a etapas, atividades, papéis/responsabilidade, ferramentas, boas práticas e artefatos? se sim qual?
2. As etapas propostas no processo, foram suficientes para um bom planejamento?
3. O processo contribuiu para que todas as tarefas fossem cumpridas nas *sprint*s?
4. Em relação ao grupo teve alguma dificuldade de comunicação? Tem alguma boa prática que você sugere? (Comunicação)
5. O processo ajudou na elaboração da documentação? Tem algum artefato faltando ou excedendo?
6. O processo ajudou na elaboração do desenvolvimento do *software*? Quais ferramentas você destacaria?
7. O processo ajudou na elaboração dos testes, quanto aos artefatos, boas práticas, ferramentas, atividades e técnicas?
8. Na sua opinião, observando todas as etapas do processo e de acordo com o que foi feito, a equipe conseguiu atingir os objetivos esperados?
9. Dê uma nota de 0 a 10 do processo FGD (*framework* de gestão e desenvolvimento)

10. Os métodos ágeis que foram implementados na elaboração do Eduplan contribuíram para que concluísse os objetivos de maneira qualitativa e quantitativa?
11. Na sua opinião se não utilizasse os métodos ágeis que foram implementados seria possível ter o mesmo rendimento e produtividade?

A análise das respostas permitiram identificar que o time do projeto quanto à adequação das etapas do processo acrescentaram algumas observações durante a prática do processo, algumas foram modificadas, outras não se encaixam no que foi proposto para o *framework*.

“Acredito que todas as etapas estão bem coesas e têm colaborado para o bom andamento do projeto.’’
“No planejamento da *sprint* falta uma discussão da *sprint* após concluir a meta, para haver uma melhor comunicação da equipe’’
“Faltou testes de integração’’
“Na transição poderia ter uma etapa de versionamento E no planejamento uma etapa de riscos.’’

A maioria dos respondentes acreditam que as etapas do processo foram suficientes para um bom planejamento (Figura 24).

Figura 24 – Gráfico referente as respostas da questão 2

Fonte: Elaborado pelos autores

Com relação ao cumprimento das tarefas durante as sprins os membros do projeto afirmaram que o processo contribui e que:

“Os poucos impedimentos que tivemos foi devido a falhas técnicas ou impedimentos terceiros.”
“Houve uma boa organização.”
“Ajudou a manter uma organização das *sprint*s”

Em relação ás dificuldades de comunicação os respondentes sugeriram uma conclusão de *sprint e* a maioria relatou uma comunicação efetiva.

“A comunicação sempre foi efetiva.”
“Uma conclusão de *sprint*.’’
“Todos tiveram uma boa interação.”
“A comunicação é efetiva.”

De acordo os respondentes o processo foi o suficiente para conseguir elaborar a documentação do projeto e não excedeu , nem faltou nenhum documento.

“Temos todos os artefatos.”
“Foi o suficiente.”
“Foi o suficiente.”
“Os artefatos foram o suficiente.”

Quanto a elaboração do desenvolvimento de *software* alguns membros do projeto afirmaram que foi importante o processo principalmente para tomada de decisões e outros destacaram o angular JS como uma importante ferramenta nessa etapa.

“Clarificou as etapas e ajudou a dividir melhor as responsabilidades. Ao priorizar técnicas para cada artefato, também tornou-se mais fácil o desenvolvimento dos mesmos.”
“Angular JS.”
“Angular js.”
“Ajudou, pois quando havia dificuldade com as ferramentas a equipe era bem participativa, resolvendo as questões pendentes e colocando nas *sprint*s as dificuldades, fazendo com que fosse concluída rapidamente.”

Quanto a etapa dos testes os membros do projeto concordaram com a contribuição do processo na fase dos testes e sugeriram a inclusão do teste de integração, porém não foi necessario até o devido momento do projeto.

“Contribuiu para melhor organização da equipe e dos artefatos a serem elaborados e desenvolvidos.”

“Faltando só a inclusão do teste de integração.”

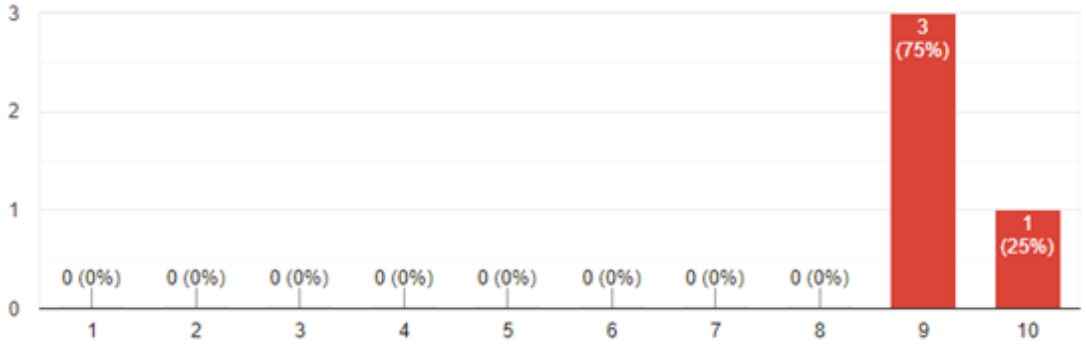
Os respondentes ao observar as etapas do processo concluiram que foi possivel atingir os objetivos esperados 50% respondeu que sim e os outros 50% também responderam que sim, mas deram uma breve explicação da sua afirmação (Figura 25).

Figura 25 - Gráfico referente as respostas da questão 8

Fonte: Elaborado pelos autores

Da nota de 0 a 10 sobre o processo FGD 75% dos membros deram nota 9 e 25% deu nota 10 (Figura 26).

Figura 26 - Gráfico referente as respostas da questão 9



Fonte: Elaborado pelos autores

Os participantes afirmaram que com os métodos ágeis conseguiram alcançar de maneira qualitativa e quantitativa de tudo que foi proposto quanto a elaboração do Eduplan.

"Tanto qualitativa, pois foram alcançados vários artefatos por *sprint* possibilitando uma entrega mais ágil. Assim como melhorou a qualidade dos artefatos."
"Foi concluído tudo que foi proposto."

De acordo os membros da equipe não seria possível ter o mesmo rendimento e produtividade sem a utilização dos métodos ágeis, pois contribuiu para um bom planejamento e não teria os mesmos resultados gerados sem a utilização do processo.

"Levaria mais tempo para engajar a equipe e produzir os resultados gerados com o processo."
"Com a metodologia o planejamento melhorou o rendimento."
"O processo contribuiu para um bom planejamento."

6. CONSIDERAÇÕES FINAIS

O desenvolvimento do presente estudo possibilitou a avaliação do *framework* que foi criado e validado em um estudo de caso nas disciplinas de Pratica em Fábrica de *Software*. Na oportunidade, foram utilizados métodos ágeis que tornou possível criar um processo ágil que se adequasse as expectativas de equipe e do projeto.

A utilização de um processo para gestão e desenvolvimento foi fundamental para desenvolver um *software* que atendesse as especificidades do projeto, e que esclarecesse o que precisava ser feito e como deveria ser feito. Através do passo a passo do processo todos os integrantes do time tiveram uma visão melhor do projeto, evitando problemas e falta de comunicação que foi fundamental e efetiva em todo o processo.

Outro momento importante foi a validação do processo que deu a equipe uma oportunidade de identificar as melhorias que ocorreram desde o início, e uma visão de todo o processo com foco no time, identificando melhorias através do questionário.

Contudo, o processo que foi criado garantiu a qualidade e produtividade do *software* que foi desenvolvido nas disciplinas de PFS, tendo como foco principal a equipe e o desenvolvimento ágil, possibilitando uma boa adaptação do que precisava ser feito e o que conseguiria ser feito em cada *Sprint*.

REFERÊNCIAS BIBLIOGRAFICAS

BARROS, A. J. D. S.; LEHFELD, N. A. D. S. **Fundamentos de metodologia científica**. 3ª. ed. São Paulo: Pearson Prentice Hall, 2007.

BECK, K. et al. Agile Manifesto. **Manifesto for Agile *Software* Development**, 2015. Disponivel em: <http://agilemanifesto.org/>. Acesso em: 21 Maio 2018.

BENASSI, J. L. G.; FERREIRA JUNIOR, L. D.; AMARAL, D. C. **Identificação das propriedades da visão do produto no gerenciamento ágil de projetos de produtos manufaturados**. Universidade de São Paulo. Porto Alegre, p. 17. 2012.

BURGESS, A. et al. **Guide to the *Software* Engineering Body of Knowledge - SWEBOK**. IEEE Computer Society. [S.l.]. 2014.

BRASILEIRO, Roberto. **Veja agora 08 dicas para criar excelentes histórias de usuário.** Disponível em: <http://www.metodoagil.com/historias-de-usuario/>. Acesso em: 28 de mar. 2019

CERVO, A. L.; SILVA, R. D.; BERVIAN, P. A. **Metodologia Científica**. 6ª. ed. São Paulo: Pearson, 2006.

CUNHA, F. *Framework*. **Dicionario Informal**, 2010. Disponivel em: <https://www.dicionarioinformal.com.br/*framework*/>. Acesso em: 01 out. 2018.

EGGER, Carolina. **Entrevistas com clientes – Como extrair informações valiosas?** Disponível em: < https://blog.contentools.com.br/marketing-de-conteudo/entrevistas-com-clientes/>. Acesso em: 18 fev. 2019

ISO. **ISO/IEC 12207:2008 - Systems and *software* engineering — *Software* life cycle processes**. International Organization for Standardization. Piscataway. 2008.

KOSCIANSKI, A.; SOARES, M. D. S. **Qualidade de *Software*:** aprenda as metodologias e tecnicas mais modernas para o desenvolvimento de *software*. 2. ed. São Paulo: Novatec Editora, 2007.

MARQUES, J.R. **Como funciona a Avaliação 360 graus?** Disponível em: <https://www.ibccoaching.com.br/portal/coaching/como-funciona-avaliacao-360-graus/>. Acesso em: 02 abr. 2019

MDS - Metodologia de Desenvolvimento de *Software*. **DATASUS**. Disponivel em: <http://datasus.saude.gov.br/metodologias/mds-*software*>. Acesso em: 18 out. 2018.

MEDEIROS, H. DevMedia. **Práticas em XP:** Extreme Programming, 2013. Disponivel em: <https://www.devmedia.com.br/praticas-em-xp-extreme-programming/29330>. Acesso em: 08 nov. 2018.

MEIRA, F. L. Introdução ao Processo Unificado Aberto. **OPEN UP - PROCESSO UNIFICADO ABERTO**, 2010. Disponivel em: <http://open2up.blogspot.com>. Acesso em: 07 nov. 2018.

NAZÁRIO, R. L.; RUFINO, R. **O Conceito de TDD no Desenvolvimento de *Software***. Universidade Paranaense. Paranavaí, p. 5. 2015.

OLIVEIRA, G. O que você precisa saber sobre o método Lean de melhoria de processos. **Excellence blog**, 2017. Disponivel em: <https://blog.softexpert.com/melhoria-de-processo-lean/>. Acesso em: 2018 nov. 2018.

PRESSMAN, R. S. **Engenharia de *Software*:** uma abordagem profissional. 9ª. ed. São Paulo: Pearson Makron Books, 2011.

PRIKLADNICKI, R.; WILLI, R.; MILANI, F. **Métodos ágeis para desenvolvimento de *software***. Porto Alegre: Bookman, 2014.

ROCHA, F. G. **TDD: fundamentos do desenvolvimento orientado a testes**. DEVMEDIA. [S.l.]. 2013.

ROCHA, F. G. **Integrando XP as principais metodologias agéis**. DEVMEDIA. [S.l.]. 2014.

ROCHA, F. G. Integrando XP as principais metodologias agéis. **DEVMEDIA**. Disponivel em: <https://www.devmedia.com.br/integrando-xp-as-principais-metodologias-ageis/30989>. Acesso em: 18 out. 2018.

SBROCCO, J. H. T. D. C.; MACEDO, P. C. **Metodologias ágeis:** engenharia de *software* sob medida. São Paulo: Érica, 2012.

SCHWABER, K.; SUTHERLAND, J. **Scrum Guide**. Tradução de Fábio Cruz. [S.l.]: [s.n.], 2016. Disponivel em: <https://www.scrumguides.org>. Acesso em: 08 nov. 2018.

SCRUM. **DesenvolvimentoAgil.com.br**. Disponivel em: <https://www.desenvolvimentoagil.com.br/scrum/>. Acesso em: 18 out. 2018.

SOARES, M. D. S. **Comparação entre Metodologias Ágeis e Tradicionais para o Desenvolvimento de *Software***. Universidade Presidente Antônio Carlos. Conselheiro Lafaiete, p. 6. 2014.

SOMMERVILLE, I. **Engenharia de *Software*.** Tradução de Ivan Bosnic e Kalinka G. de O. Gonçalves. 9ª. ed. São Paulo: Pearson Prentice Hall, 2011.

APÊNDICE A

EduPlan
Documento visão

Versão <5.0>

Documento visão

1. Introdução

A finalidade deste documento é coletar, analisar e definir necessidades e recursos de nível superior do EduPlan. Ele se concentra nos recursos necessários aos envolvidos e aos usuários-alvo e nas razões que levam a essas necessidades. Os detalhes de como o EduPlan satisfaz essas necessidades são descritos no caso de uso e nas especificações suplementares.

- Descrição do Problema

O problema	*Falta de agilidade no preenchimento do plano de ensino.*
afeta	*Docentes*
cujo impacto é	*Excesso de retrabalho*
uma boa solução seria	*A criação de uma plataforma onde os dados fossem pré-armazenados facilitando e agilizando o processo de produção dos planos de ensino.*

- Sentença de Posição do Produto

Para	*Instituições Universitárias*
Que	*Necessitem agilizar o processo de elaboração do plano de ensino das disciplinas*
O EduPlan	*É um sistema web*
Que	*Facilita trabalho do docente ao elaborar o plano de ensino de forma padronizada*

2. Descrições dos Envolvidos e Usuários

- Resumo dos Envolvidos

Nome	Descrição	Responsabilidades
Analistas de requisito	*Ira analisar os requisitos e fazer o levantamento dos mesmos, ficando responsável pela documentação do projeto*	- *Garante a agilidade do processo* - Garante uma adequação ao que o cliente busca no *software*
Desenvolvedores	*Ira desenvolver e dar suporte técnico ao sistema*	- *garante que o sistema terá manutenção* - *monitora o andamento do projeto* - *dar o suporte necessário para o bom funcionamento do sistema*

- Resumo dos Usuários

Nome	Descrição	Responsabilidades	Envolvidos
Gestor	*Os Gestores validarão*	- *Coordena o trabalho* - *Produz relatório* - *Analisa os planos de ensino*	-
Docentes	*Os docentes irão preencher os campos abertos do plano de ensino*	- *Preenchem de forma correta e com as normas da instituição o plano de ensino*	-

- Ambiente do Usuário

Supõe-se que os usuários tenham um computador com acesso a internet para poder acessar.

- Alternativas e Concorrência

Até o momento não há uma concorrência.

3. Visão Geral do Produto

- Perspectiva do Produto

O produto tem como meta ajudar os docentes a produzirem os planos de ensino da instituição, de modo que diminua o retrabalho de refazer toda a documentação a cada semestre.

O sistema visa deixar certos campos já estabelecidos e colocando para o docente preencher somente o necessário. Após isso será encaminhado o plano de ensino para os gestores responsáveis, onde será analisado e logo aceito ou rejeitado, caso seja rejeitado o plano de ensino volta para o docente fazer as correções necessárias

4. Recursos do Produto

O *software* basicamente no início faz o login do docente e do gestor. Quando o docente loga é criado um novo plano com opção de editar e adicionar. Quando o gestor loga ele analisa e aceita ou rejeita o novo plano. Se aceito o docente tem a opção de imprimir o plano e se rejeita é encaminhado para correção.

APÊNDICE B

PLANO DE TESTE

VERSÃO 2.0

18/03/2019

1. Objetivos

Este documento documenta e expõe o plano de teste a ser aplicado no *software* Eduplan. Os objetivos a serem alcançados com os testes visam:

1. Identificar informações existentes do projeto e os componentes de *software* que devem ser testados.
2. Listar os requisitos de teste recomendados.
3. Recomendar e descrever as estratégias de teste a serem empregadas.
4. Identificar os recursos requeridos e fornecer uma estimativa dos esforços de teste.
5. Listar os artefatos produzidos com as tarefas de teste.

2. Escopo

O presente plano de teste foi desenvolvido visando a aplicação, e organização dos testes a serem executados nos requisitos do *software* EduPlan. Além dos testes nos releases dos requisitos apontados como [pronto], também planeja-se os teste de caixa preta para teste dos códigos fonte e interface do sistema, de forma a serem incrementados conforme desenvolvimento dos mesmos.

Este Plano de Teste é aplicado ao teste de todos os requisitos do EduPlan, conforme definido no Documento de Visão, e Especificações de Caso de Uso.

3. Referências

[1] Documento de História de Usuário

[2] Documento de Arquitetura

[3] Plano de Projeto

[4] Caso de uso UML

4. Requisitos de Teste

Os requisitos (casos de uso, requisitos funcionais, requisitos não funcionais) tidos como alvos dos testes são organizados conforme lista a seguir. Essa lista representa poderá ser testado. Detalhes sobre cada teste serão determinados posteriormente à medida que os Casos de Teste forem identificados e os Scripts de Teste forem desenvolvidos.

Teste de Integridade dos Dados e do Banco de Dados

- Verificar acesso ao Banco de Dados, visando login de usuários e cadastro de docentes.
- Verificar acessos de leitura simultâneos ao registro.
- Verificar interrupção durante atualizações do Cadastro de Docentes.
- Verificar a recuperação correta de atualizações dos dados do banco de dados.

Teste Funcional

- Verificar Caso de Uso Login [6]
- Verificar Caso de Uso Manter Docente [5]
- Verificar Caso de Uso Manter Plano[10]

❖ Especificação Complementar, seção 4.1: "Todos os erros do sistema devem ser registrados. Os erros fatais do sistema devem resultar em um encerramento ordenado do sistema."

❖ Especificação Complementar, seção 4.1: " As mensagens de erro do sistema devem incluir uma descrição em texto do erro, o código de erro do sistema operacional (se aplicável), o módulo que detectou a condição de erro, um stamp de dados e um time stamp."

Teste de Ciclo de Negócio

- Verificar a integridade dos planos de ensino após inserção de novos planos e alteração de planos existentes.
- Verificar os planos por vários semestres e vários anos.
- Verificar se após alterado um plano de ensino o mesmo é alterado em semestres anteriores.

Teste da Interface com o Usuário

- Verificar a facilidade de navegação utilizando um conjunto de amostras de telas.
- Verificar se as telas de amostra estão em conformidade com as histórias de usuário.
- Segundo documento de visão: O Sistema deve ter fácil utilização e deve ser apropriado para o mercado de destino, para docentes com experiência em computadores.
- Segundo Plano de Projeto: A interface do Eduplan deverá ser projetada para facilidade de utilização e deverá ser apropriada para uma comunidade de usuários experiente com computadores.

Teste de Desempenho

- Verificar o tempo de resposta para acesso ao cadastro de docente.
- Verificar o tempo de resposta para acesso ao cadastro de planos.
- Verificar o tempo de resposta para listagem de informações.
- Verificar o tempo de resposta para login.
- Verificar o tempo de resposta para a submissão remota de cadastros.

Teste de Carga

- Verificar a resposta do sistema quando estiver carregado com 250 usuários com logon efetuado.
- Verificar a resposta do sistema quando existir 50 acessos simultâneos de docentes aos planos de ensino.

Teste de Estresse

- Verificar a resposta do sistema durante o máximo de logins de usuários.

Teste de Volume

- Definir tamanho do Banco de dados, e verificar a resposta do sistema quando o Banco de Dados estiverem em 90% da capacidade.

Teste de Segurança e Controle de Acesso

- Verificar o Logon a partir de um PC local.
- Verificar o Logon a partir de um PC remoto.
- Verificar a segurança de Logon por meio de mecanismos de nome de usuário e senha. Especificação Complementar: Toda a funcionalidade deve estar disponível remotamente por uma conexão à Internet.

5. Estratégia de Teste

A Estratégia de Teste apresenta a abordagem recomendada para o teste dos aplicativos de *software*. A seção anterior dos Requisitos de Teste descrevia *o que* será testado; esta descreve *como* será testado.

- As principais considerações para a estratégia de teste são as técnicas a serem utilizadas e o critério para saber quando o teste está concluído.
- Além das considerações fornecidas para cada teste a seguir, o teste deve ser executado apenas utilizando bancos de dados conhecidos e controlados, em ambientes protegidos.
- A estratégia de teste a seguir é genérica por natureza e foi desenvolvida para ser aplicada aos requisitos listados neste documento.

Tipos de Teste

1. *Teste de Integridade dos Dados e do Banco de Dados*

Os bancos de dados e os processos de banco de dados devem ser testados como sistemas separados. Esses sistemas devem ser testados sem os aplicativos (como a interface para os dados). É necessário executar pesquisas adicionais referentes ao Banco de Dados escolhido a fim de identificar as ferramentas / técnicas que poderão existir para suportar os testes identificados a seguir.

Objetivo do Teste:	Assegurar que os processos e métodos de acesso ao Banco de Dados funcionem corretamente e sem corrupção de dados.

Técnica:	• Chamar cada processo e método de acesso a banco de dados, propagando cada um com dados válidos e inválidos (ou pedidos de dados). • Inspecionar o banco de dados para assegurar que os dados foram preenchidos conforme planejado e que todos os eventos do banco de dados ocorreram adequadamente ou revisar os dados retornados para assegurar que os dados corretos foram recuperados (pelas razões corretas)
Critérios de Conclusão:	Todos os processos e métodos de acesso ao banco de dados funcionam conforme projetado e sem nenhuma corrupção de dados.
Considerações Especiais:	• Os testes podem exigir drivers e/ou SGBD para digitar ou modificar dados diretamente nos bancos de dados. • Os processos devem ser chamados manualmente. • Bancos de dados pequenos ou de tamanho mínimo (número limitado de registros) devem ser utilizados para aumentar a visibilidade de quaisquer eventos não aceitáveis.

2. Teste de Funcionalidade

Os testes do aplicativo devem ter foco em quaisquer requisitos de destino que possam ser rastreados diretamente para casos de uso (ou funções de negócios) e regras de negócios. A meta desse teste é verificar a adequada aceitação, o processamento e a recuperação dos dados, e a implementação apropriada das regras de negócios. Esse tipo de teste baseia-se em técnicas de caixa preta, ou seja, verificar o aplicativo (e seus processos internos) interagindo com o aplicativo por meio da GUI e analisar a saída (resultados). A seguir é identificado um esboço do teste recomendado para cada aplicativo:

Objetivo do Teste:	Assegurar a navegação correta do aplicativo, além da entrada, processamento e recuperação de dados.

Técnica:	• Executar cada caso de uso, fluxo de caso de uso ou função, utilizando dados válidos e inválidos, para verificar o seguinte: • Os resultados esperados ocorrerão quando forem usados dados válidos. • As mensagens de erro / aviso apropriadas sejam exibidas quando dados inválidos forem utilizados. • Cada regra de negócio será adequadamente aplicada.
Critérios de Conclusão:	• Todos os testes planejados foram executados. • Todos os defeitos identificados foram tratados e documentados.

3. ***Teste de Ciclo de Negócio***

O Teste de Ciclo de Negócios deve emular as atividades executadas no sistema ao longo do tempo. Deverá ser identificado um período como, por exemplo, um ano, e deverão ser executadas as transações e atividades que ocorreriam durante esse período de um ano. Isso inclui todos os ciclos diários, semanais e mensais, assim como os eventos sensíveis a datas como, por exemplo, lembretes.

Objetivo do Teste:	Assegurar que os processos de segundo plano e do aplicativo corretos funcionem de acordo com os planejamentos e os modelos de negócios requeridos.

Técnica:	• O teste simulará vários ciclos de negócios, executando o seguinte: • Os testes utilizados para o teste de funções do aplicativo serão modificados / melhorados para aumentar o número de vezes que cada função é executada, a fim de simular vários usuários diferentes ao longo de um período de tempo especificado. • Todas as funções sensíveis a datas ou tempo serão executadas usando datas ou períodos de tempo válidos e inválidos. • Todas as funções que ocorrerem segundo um planejamento periódico serão executadas / iniciadas no momento adequado. • O teste incluirá o uso de dados válidos e inválidos para verificar se: • Os resultados esperados ocorrerão quando forem usados dados válidos. • As mensagens de erro / aviso apropriadas sejam exibidas quando dados inválidos forem utilizados. • Cada regra de negócio será adequadamente aplicada.
Critérios de Conclusão:	• Todos os testes planejados foram executados. • Todos os defeitos identificados foram tratados.

4. *Teste da Interface com o Usuário*

O teste da Interface com o Usuário verifica a interação de um usuário com o *software*. A meta do Teste de UI é assegurar que a Interface com o Usuário forneça ao usuário o acesso e a navegação adequados por meio das funções dos aplicativos. Além disso, o Teste de UI assegura que os objetos contidos na UI funcionem conforme esperado e estejam em conformidade com padrões corporativos ou do segmento de mercado.

Objetivo do Teste:	Verifique o seguinte: • A navegação pelo aplicativo reflete os requisitos e funções de negócios, incluindo a navegação janela a janela, campo a campo e o uso de métodos de acesso (teclas de tabulação, movimentos do mouse e teclas aceleradoras) • Objetos e características da janela, tais como menus, tamanho, posição, estado e foco estão em conformidade com os padrões.
Técnica:	• Criar / modificar testes para cada janela a fim de verificar a navegação adequada e os estados de objeto para cada janela e objeto do aplicativo.
Critérios de Conclusão:	Verificação com êxito de cada janela permanecer consistente com a versão de benchmark ou dentro do padrão aceitável
Considerações Especiais	• Nem todas as propriedades de objetos personalizados e de terceiros podem ser acessadas.

5. ***Teste de Desempenho***

O teste de desempenho mede tempos de resposta, taxas de transação e outros requisitos sensíveis ao tempo. A meta do teste de Desempenho é verificar e validar se os requisitos de desempenho foram alcançados. O teste de desempenho normalmente é executado várias vezes, cada uma utilizando uma "carga de segundo plano" diferente no sistema. O teste inicial deve ser executado com uma carga "nominal", semelhante à carga normal observada (ou prevista) no sistema de destino. Um segundo teste de desempenho é executado utilizando uma carga de pico.

Além disso, os testes de desempenho podem ser utilizados para traçar o perfil e ajustar o desempenho de um sistema como uma função de condições, como a carga de trabalho ou configurações de hardware.

NOTA: As transações a seguir se referem a "transações comerciais lógicas." Essas são transações são definidas como funções específicas que se espera que um usuário do sistema execute utilizando o aplicativo, como incluir ou modificar um determinado registro.

Objetivo do Teste:	Validar o Tempo de Resposta do Sistema para funções de negócios ou transações designadas sob as duas condições a seguir: - volume normal previsto - volume de pior caso previsto
Técnica:	• Utilizar Scripts de Teste desenvolvidos para Teste de Modelo de Negócio (Teste de Funcionalidade). • Modificar arquivos de dados (a fim de aumentar o número de transações) ou modificar scripts a fim de aumentar o número de iterações ocorrido em cada transação. • Os scripts devem ser executados em uma máquina (o melhor é avaliar o desempenho de um único usuário, uma única transação) e repetidos com vários clientes (virtuais ou reais, *consulte as considerações especiais a seguir).*
Critérios de Conclusão:	• Transação Única / usuário único: Conclusão com êxito dos scripts de teste sem nenhum defeito e na alocação de tempo esperada / requerida (por transação) • Várias Transações / vários usuários: Conclusão com êxito dos scripts de teste sem nenhum defeito e dentro de alocação de tempo aceitável.

Considerações Especiais:	O teste abrangente do desempenho inclui ter uma carga "em segundo plano" no servidor. Há vários métodos que podem ser usados para executar esse teste, incluindo: "Encaminhar as transações" diretamente para o servidor, geralmente na forma de chamadas SQL. Criar carga "virtual" de usuários para simular muitos (geralmente várias centenas) de clientes. Para se obter essa carga, geralmente são usadas ferramentas de Emulação de Terminal Remoto. Essa técnica também pode ser utilizada para carregar a rede com "tráfego". Utilizar vários clientes físicos, cada qual executando scripts de teste para inserir carga no sistema. O teste de desempenho deverá ser executado em uma máquina dedicada ou em um período de tempo dedicado. Isso permitirá o controle total e a medição exata. Os bancos de dados utilizados para teste de Desempenho deverão ter tamanhos reais ou ser igualmente escalados.

6. ***Teste de Carga***

As medidas do teste de carga sujeitam o sistema em teste a cargas de trabalho variáveis para avaliar a capacidade do sistema em continuar a funcionar corretamente sob essas diferentes cargas de trabalho. A meta desse teste de carga é determinar e assegurar que o sistema funcione adequadamente com uma carga de trabalho superior à carga máxima esperada. Além disso, o teste de carga avalia as características de desempenho (tempos de resposta, taxas de transação e outros aspectos sensíveis ao tempo).

NOTA: As transações a seguir se referem a "transações comerciais lógicas." Essas são transações são definidas como funções específicas que se espera que um usuário do sistema execute utilizando o aplicativo, como incluir ou modificar um determinado contrato.

Objetivo do Teste:	Verificar o Tempo de Resposta do Sistema para casos de negócios ou transações designadas sob condições de carga de trabalho variáveis.

Técnica:	• Utilizar os testes desenvolvidos para o Teste do Ciclo de Negócio. • Modificar os arquivos de dados (a fim de aumentar o número de transações) ou os testes a fim de aumentar o número de vezes que cada transação ocorre.
Critérios de Conclusão:	• Várias Transações / vários usuários: Conclusão com êxito dos testes sem nenhum defeito e dentro de alocação de tempo aceitável.
Considerações Especiais:	• Os testes de carga devem ser executados em uma máquina dedicada e em um período de tempo dedicado. Isso permitirá o controle total e a medição exata. • Os bancos de dados utilizados para teste de carga deverão ter tamanhos reais ou ser igualmente escalados.

7. *Teste de Estresse*

O teste de estresse foi projetado para localizar erros devidos a falta de recursos ou competição por recursos. Pouca memória ou espaço em disco podem revelar defeitos no *software* que não são aparentes sob condições normais. Outros defeitos podem resultar da competição por recurso compartilhado, como bloqueios de banco de dados ou largura da banda de rede. O teste de estresse identifica a carga de pico que o sistema pode manipular.

NOTA: As referências às transações a seguir referem-se a transações comerciais lógicas.

Objetivo do Teste:	Verificar se o sistema e o *software* funcionam corretamente e sem erros sob as seguintes condições de estresse: • pouca ou nenhuma memória disponível no servidor (RAM e DASD) • número máximo (real ou fisicamente capaz) de usuários conectados (ou simulados) • vários usuários executando as mesmas transações nos mesmos dados / contas • conjunto / volume de transações no pior caso (consulte o teste de desempenho acima). NOTAS: A meta do teste de estresse também pode ser definida como identificar e documentar as condições sob as quais o sistema FALHA em continuar funcionando corretamente.

Técnica:	• Utilizar os testes desenvolvidos para o Teste de Desempenho. • Para testar recursos limitados, os testes devem ser executados em uma única máquina e a RAM e DASD no servidor devem ser reduzidos (ou limitados). • Para os testes de estresse restantes, deverão ser utilizados vários clientes, executando-se os mesmos testes ou testes complementares a fim de produzir o conjunto / volume de transações no pior caso.
Critérios de Conclusão:	Todos os testes planejados são executados e os limites do sistema especificados são alcançados / excedidos sem o *software* ou falha do *software* (ou as condições sob as quais a falha do sistema ocorre estão fora das condições especificadas).

8. ***Teste de Volume***

O Teste de Volume sujeita o *software* a grandes quantidades de dados para determinar se serão atingidos limites que farão com que o *software* falhe. O teste de volume também identifica o volume ou a carga máxima contínua que o sistema pode manipular durante um determinado período de tempo. Por exemplo, se o *software* estiver processando um conjunto de registros de banco de dados para gerar um relatório, um Teste de Volume utilizará um grande banco de dados de testes e verificará se o *software* se comportou normalmente e gerou o relatório correto.

Objetivo do Teste:	Verifica se o aplicativo / sistema funciona com êxito sob os seguintes cenários de alto volume: • número máximo (real ou fisicamente capaz) de clientes conectados (ou simulados), todos executando a mesma função de negócio em pior caso (desempenho) por um período extenso. • o tamanho máximo do banco de dados foi alcançado (real ou escalado) e várias consultas / transações de relatório são executadas simultaneamente.

Técnica:	• Utilizar os testes desenvolvidos para o Teste de Desempenho. • Deverão ser usados vários clientes, executando-se os mesmos testes ou testes complementares a fim de produzir o conjunto / volume de transações no pior caso (consulte teste de estresse acima) durante um longo período de tempo. • O tamanho máximo do banco de dados é criado (real, escalado ou preenchido com dados representativos) e vários clientes são utilizados para executar consultas / transações de relatório simultaneamente por longos períodos de tempo.
Critérios de Conclusão:	Todos os testes planejados foram executados e os limites do sistema especificados são alcançados / excedidos sem o *software* ou falha do *software*.
Considerações Especiais:	• Qual período de tempo seria considerado aceitável em condições de alto volume?

9. ***Teste de Segurança e Controle de Acesso***

O Teste de Segurança e de Controle de Acesso tem como foco duas áreas principais de segurança:

1. Segurança do aplicativo, incluindo o acesso aos Dados ou às Funções de Negócios.
2. Segurança do sistema, incluindo login e acesso remoto ao sistema.

A segurança do aplicativo assegura que, com base na segurança desejada, os usuários têm restrição a funções específicas ou estão limitados aos dados que estão disponíveis a eles. Por exemplo, todos têm permissão para inserir dados e criar novas contas, mas apenas os administradores poderão excluí-los. Se houver segurança no nível dos dados, o teste assegura que o usuário em questão acesse somente as informações ao qual o mesmo possua permissão.

A segurança do sistema assegura que apenas os usuários, para os quais o acesso ao sistema foi concedido, sejam capazes de acessar os aplicativos e apenas por meio dos gateways apropriados.

Objetivo do Teste:	Segurança de Função / Dados: Verificar se o usuário pode acessar apenas as funções / dados para os quais seu tipo de usuário tenha recebido permissão. Segurança do Sistema: Verificar se apenas os usuários com acesso ao sistema e aplicativo(s) têm permissão para acessá-los.

Técnica:	• Segurança de Função / Dados: Identificar e listar cada tipo de usuário e as funções / dados para os quais cada tipo tem permissão. • Criar testes para cada tipo de usuário e verificar a permissão criando transações específicas para cada tipo de usuário. • Modificar o tipo de usuário e executar novamente os testes para os mesmos usuários. Em cada caso, verificar se as funções / dados adicionais estão corretamente disponíveis ou se têm seu acesso negado. • Acesso ao Sistema (consulte considerações especiais a seguir)
Critérios de Conclusão:	Para cada tipo de usuário conhecido, a função / dados apropriados estão disponíveis e todas as transações funcionem como esperado e sejam executadas nos testes de Função de Aplicativo anteriores
Considerações Especiais:	• O acesso ao sistema deve ser revisado / discutido com o administrador da rede ou do sistema apropriado. Talvez esse teste não seja necessário, pois pode ser uma função de administração da rede ou do sistema.

2. Ferramentas

As seguintes ferramentas serão pesquisadas e empregadas para o teste do sistema:

	Ferramenta	**Versão**
Gerenciamento de Teste	A Definir	A Definir
Design de Teste	A Definir	A Definir
Controle de Defeitos	A Definir	A Definir
Teste Funcional	A Definir	A Definir

Teste de Desempenho	A Definir	A Definir
Gerador de Perfil ou Monitor de Cobertura de Teste	A Definir	A Definir
Outras Ferramentas de Teste	A Definir	A Definir
Gerenciamento de Projeto	A Definir	A Definir
Ferramentas de SGBD	A Definir	A Definir

6. Recursos

Esta seção apresenta os recursos recomendados para o teste do EduPlan, suas principais responsabilidades e seu conhecimento ou configuração de habilidades.

- **Trabalhadores**

Esta Quadro mostra as premissas de equipe para as tarefas de teste.

Trabalhador	**Recursos Mínimos Recomendados** **(número de trabalhadores alocados em período integral)**	**Responsabilidades Específicas/Comentários**
Gerente de Testes	Jônatas Gabriel	Fornece supervisão de gerenciamento Responsabilidades: • Fornecer direção técnica • Adquirir recursos apropriados • Relatório de gerenciamento

Designer de Teste	Melquisedeque Morais	Identifica, prioriza e implementa casos de teste Responsabilidades: • Gerar plano de teste • Gerar Conjunto de Teste • Avaliar eficácia do esforço de teste
Testador do Sistema	Lucas Silva Melquisedeque Morais	Executa os testes Responsabilidades: • Executar testes • Registrar resultados • Recuperar-se de erros • Documentar defeitos

8. **Produtos de Trabalho**

Os produtos de trabalho das tarefas de teste definidas neste Plano de Teste estão descritos na Quadro a seguir.

Note que alguns desses produtos de trabalho são produzidos várias vezes; uma para cada iteração ou ciclo de teste. Outros produtos de trabalho, tal como o Plano de Teste, são atualizados a cada iteração de desenvolvimento.

Produtos de Trabalho	**Proprietário**	**Revisão / Distribuição**	**Data Comprometida**
Plano de Teste	Adriana L.G. Rodrigues	Time e Scrum Master	25 de março
Ambiente de Teste	Melquisedeque Morais e Lucas Silva	-	A Definir
Conjunto de Testes	Melquisedeque Morais e Lucas Silva	Revisão por time	A Definir
Conjuntos de Dados de Teste	Melquisedeque Morais e Lucas Silva	Revisão por time	A Definir

Scripts de Teste	Melquisedeque Morais e Lucas Silva	-	A Definir
Stubs de Teste, Drivers	Melquisedeque Morais e Lucas Silva	-	A Definir
Relatórios de Defeitos do Teste	Melquisedeque Morais e Lucas Silva	Time e Scrum Master	A Definir
Resultados de Testes	Melquisedeque Morais e Lucas Silva	Gerente de Testes	A Definir
Relatório de Avaliação de Teste	Melquisedeque Morais e Lucas Silva	Time e Scrum Master	A Definir

Aprovações

Título	Nome e Assinatura	Data
Product Owner	Adriana L.G. Rodrigues	18/03/2019
Scrum Master	Júlio Rodrigues	18/03/2019
Time	Isabella Carolina	18/03/2019
Time	Jônatas Gabriel	18/03/2019
Time	Lucas Silva	18/03/2019
Time	Jonathas Ramos	18/03/2019

Nota: Quaisquer alterações neste documento deverão ser submetidas ao processo de controle de projeto para aprovações antes de serem incorporadas a este documento

APÊNDICE C

Framework para Gestão e Desenvolvimento de *Software* Ágil: Um Estudo de Caso Desenvolvido nas Disciplinas de Prática em Fábrica de *Software*

Isabella Carolina M. Barros, Júlio R. Lobo, Renata D. Braga

Bacharelado em Engenharia de Computação – Centro Universitário de Anápolis (UniEvangélica)
75083-515 – Anápolis – GO - Brasil

isabellacarolina80@gmail.com,Juliorlobo@hotmail.com,
professorarenatabraga@gmail.com

Resumo. *Com a necessidade de uma metodologia adaptada às necessidades de uma equipe de desenvolvimento, este trabalho teve como objetivo geral avaliar um framework para gestão e desenvolvimento de software ágil, baseado nas especificidades de um projeto em andamento nas disciplinas de Prática em Fábrica de Software. Os resultados permitiram definir um processo ágil com as seguintes etapas: requisitos, planejamento de Sprint, codificação, testes, transição e retrospectiva.*

Abstract. *With the need of a methodology adapted to the needs of a development team, this work had as general objective to evaluate a framework for management and development of agile software, based on the specificities of an ongoing project in the disciplines of Software Factory Practice. The results allowed to define an agile process with the following steps: requirements, Sprint planning, coding, testing, transition and retrospective*

1. Introdução

Ao considerar-se um cenário de constantes evoluções e mudanças, a necessidade de adaptabilidade torna-se primordial para o ingresso e permanência no mercado de produção de *software* [PRESSMAN 2016]. Um projeto de *software* ou qualquer tipo de projeto que venha a ser executado, deve contar com um planejamento detalhado de cada fase, e com um acompanhamento de cada processo. Caso contrário, vários problemas surgem, como a desorganização da equipe e a falta de acompanhamento do projeto.

Esses problemas podem causar falhas futuras na implementação e nos testes e, consequentemente, alto custo no produto final. Por conseguinte, quando não há uma previsão, pode haver demora na entrega dos produtos. Além disso, sem o devido ciclo, as entregas para os clientes sem data prevista, podem haver mudanças de requisitos ocasionando em aumento de prazo e aumento de custo. Ou seja, diversos inconvenientes que podem ser evitados com a utilização de uma metodologia.

Em contraposição a este cenário, tem-se as metodologias ágeis, capazes de ofertar total flexibilidade no desenvolvimento e maior interação entre os *stakeholders*. Com isso, o produto de *software*, é entregue em pequenos incrementos (partes), o que resulta em menor tempo, proporcionando a validação, inclusive o entendimento das necessidades,

sendo possível contemplar as mudanças necessárias. A diferença entre métodos ágeis e Clássicos sequenciais está o enfoque maior nas pessoas e o seu conjunto de valores, princípios e práticas, além do trabalho em equipe e comunicação efetiva [PRIKLADNICKI et al. 2014].

Nesse contexto, a pergunta que está norteando a execução deste estudo é: quais os elementos e boas práticas são necessárias para a definição de um framework que auxilie na gestão e desenvolvimento de um *software* ágil em andamento, nas disciplinas de Prática em Fábrica de *Software*?

Esse estudo tem como objetivo geral avaliar um *framework* para gestão e desenvolvimento de *software* ágil, baseado nas especificidades de um projeto em andamento nas disciplinas de Prática em Fábrica de *Software*.

2. Metodologia

Trata-se de uma pesquisa exploratória, que estabelece critérios, métodos e técnicas para a elaboração de uma pesquisa e visa oferecer informações sobre o objeto desta e orientar a formulação de hipóteses [CERVO et al 2006].

A elaboração do framework foi organizado nas seguintes etapas:

2.1.Identificação na literatura de metodologias para gestão e desenvolvimento de *software*

A obtenção das literaturas foi através de pesquisa bibliográfica em bases de dados com Google Acadêmico, IEEE Xplore, Anais de eventos científicos, que dispõe de uma literatura com embasamento em metodologias ágeis entre outros artigos que citem metodologias de projeto de *software*.

Para a localização dessas fontes foram utilizadas palavras chaves como: metodologia ágil, *framework*, manifesto ágil, gestão ágil, processos de gestão de *software* e processos ágeis. Tiveram prioridade na escolha, fontes com datas superiores ao ano de 2011, mas não excluindo as com datas inferiores que tiveram relevância na execução da pesquisa.

2.2.Criação do framework para gestão e desenvolvimento de *software*.

Durante a disciplina Pratica em Fábrica de *Software* I um grupo de acadêmicos foi formado para aplicação dos conceitos e desenvolvimento de um produto. Após leitura das fontes coletadas, foram selecionadas as atividades, técnicas, artefatos, boas práticas e ferramentas. As metodologias embasadas foram selecionadas através de critérios que atendessem uma equipe reduzida e que desse uma melhor oportunidade de aprendizado para todos os integrantes.

2.3. Aplicação do framework

A aplicação se dará nas disciplinas de Prática de Fábrica de *Software* na Fábrica de Tecnologia Turing, onde o framework será aplicado na elaboração do projeto EduPlan. A

equipe implantará o *framework* criado em cada disciplina na elaboração do projeto tendo assim um resultado da execução do *framework* no ambiente acadêmico profissional.

2.4.Registro dos resultados, a partir da aplicação do framework

Os resultados obtidos ao final de cada sprint serão registrados nas ferramentas selecionadas e discutidos com a equipe responsável, como forma de melhorar a interação com a equipe e principalmente de realizar melhorias no *framework*.
Como forma de avaliar a eficiência do framework foi aplicado um questionário baseado na metodologia de Avaliação 360º que tem como objetivo avaliar, de forma imparcial para identificar e analisar qual a percepção que as pessoas ao redor de um profissional têm dele.
A metodologia foi adaptada para que a avaliação seja do framework ao invés de um funcionário/colaborador, dessa forma, sendo possível através do feedback da equipe de desenvolvimento, identificar e apontar possíveis melhorias no processo.

3. Referencial Teórico

Um processo de gestão de projetos é um conjunto de fatores humanos e tecnológicos com o intuito de planejar todo o processo de forma que ocorra com qualidade e sem deixar nenhuma pendência. Pode ser divido em diversas áreas e subáreas como são descritas no *Guide to the Software Engineering Body of Knowledge*, conhecido pela sigla SWEBOK [BURGESS, KELLY, et al., 2014].

Os processos tradicionais de desenvolvimento têm como principais modelos: Linear; Cascata, Prototipagem, Incremental e Espiral. O modelo Linear possui uma abordagem sistemática sequencial para o desenvolvimento do *software*. Começa no nível de sistema e progride mediante análise de projeto, codificação, teste e manutenção [SBROCCO e MACEDO, 2012].

O modelo Cascata (Figura 1) foi o primeiro processo de desenvolvimento de *software* publicado, é de fácil entendimento e é baseado em uma sequência de etapas. Cada etapa tem associada ao seu término uma documenta padrão que deve ser aprovada para que se inicie a etapa imediatamente posterior [PRESSMAN 2011].

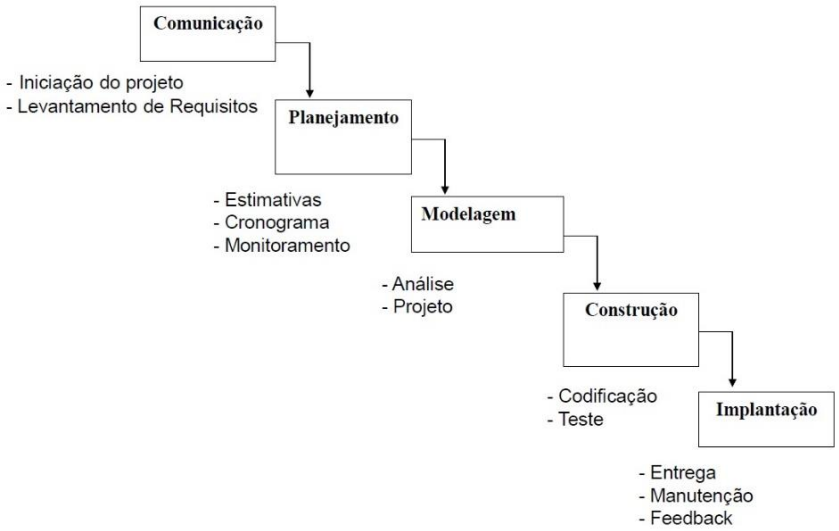


Figura 1 - Processo de desenvolvimento de *software* Cascata. Fonte: [Pressman, 2016]

Depois dos processos tradicionais foram surgindo os modelos ágeis como o Extreming Programing (Figura 2) que é voltado para o desenvolvimento de *software* ágil, no qual tudo é feito ao máximo. Nele, são empregadas as boas práticas de Engenharia de *Software* visando a produção de um produto de qualidade. O extremismo remete ao uso máximo dessas boas práticas que se apoiam e criam uma sinergia [PRIKLADNICKI, WILLI e MILANI, 2014].

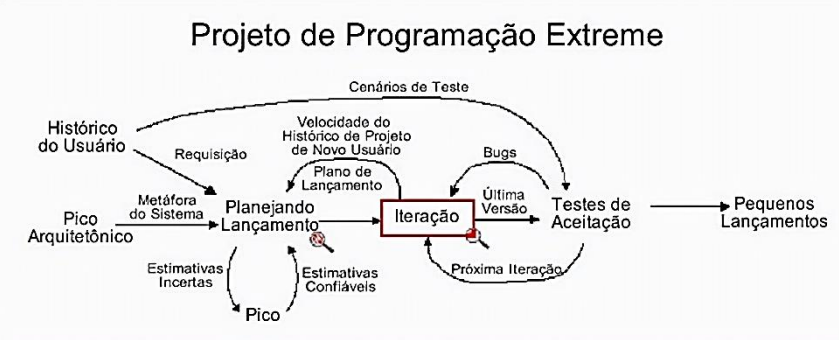


Figura 2 - Método XP. Fonte: Retirado do artigo Integrando XP as principais metodologias ágeis - DEVMEDIA

Também temos o TDD (*Test-Driven Development* – Desenvolvimento orientado a testes) (Figura 3) é uma técnica de desenvolvimento orientada a testes que tem como objetivo identificar e corrigir falhas durante o projeto [NAZÁRIO e RUFINO, 2015].

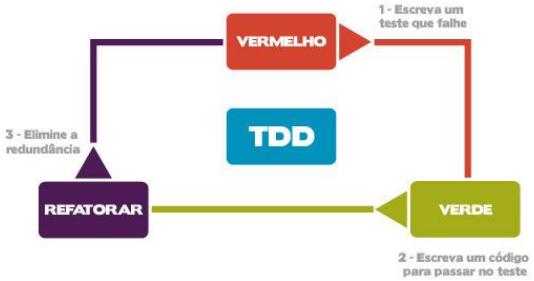


Figura 3 - Método TDD. Fonte: Retirado do artigo TDD: fundamentos do desenvolvimento orientado a testes - DEVMEDIA

Para a documentação temos como estrutura o *Scrum* (Figura 4) que tem como base um processo adaptável em que as funcionalidades mais importantes são priorizadas e ao longo do processo são entregues as partes já concluídas. Porém, em caso de necessidade de mudanças não há complicações [SBROCCO et al 2012].

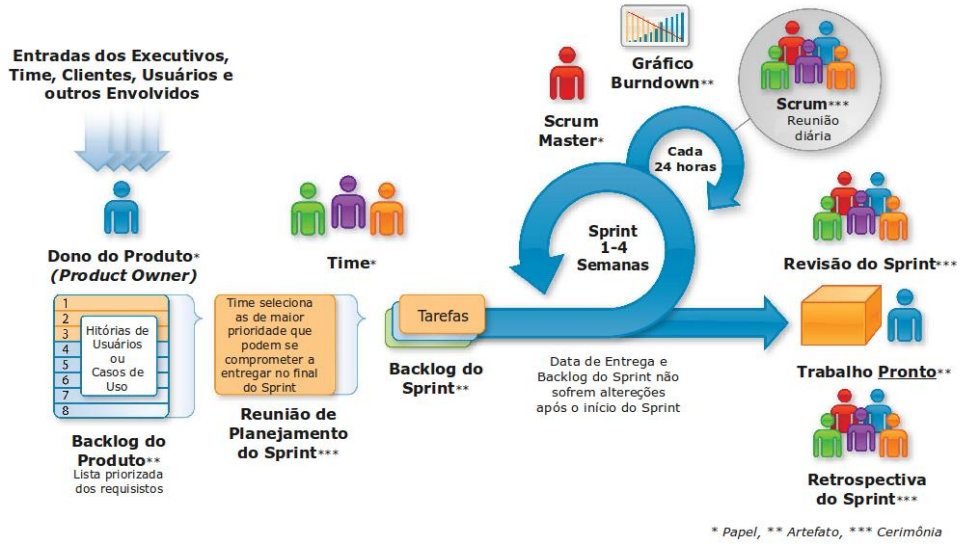


Figura 4 - Método Scrum. Fonte: Site desenvolvimentoagil.com

Também muito utilizado para a visualização do projeto o *Kanban* (Figura 5) é um método que utiliza como princípio os artefatos de maior "valor" para o projeto. Segue o fluxo para que esse artefato seja produzido com qualidade. Seu objetivo é evidenciar em um mapa visual os problemas que surgem e dar uma visão melhor de como lidar com a situação para que o fluxo continue. Ao surgir um problema, esse mapa tende a se modificar para favorecer o aperfeiçoamento, e seguir assim, um fluxo evolucionário.

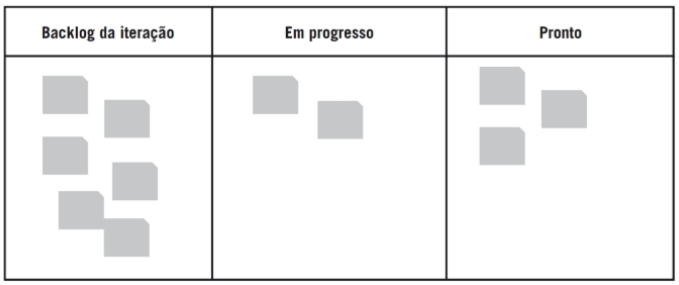


Figura 5 - Exemplo de um Kaban. Fonte: [Prikladnicki et al. 2014]

Esse conjunto de metodologias pode ser utilizado para a construção de um processo híbrido, selecionando os métodos ágeis que sejam necessários para o projeto e que atenda às especificidades do time de desenvolvimento e gestão (Figura do MDS - Metodologia de Desenvolvimento de *Software*).

Figura 6 – Metodologia de Desenvolvimento de *Software*. Fonte: DATASUS

4. Resultados

O estudo de modelos de processos de desenvolvimento de *software* permitiu a identificação de elementos necessários para a elaboração de um framework híbrido que atendesse às necessidades de um projeto, executado durante as disciplinas de Prática em Fábrica de *Software*. Ao longo desse projeto, o grupo colabora com as informações e conhecimentos, evoluindo gradativamente ao decorrer do projeto (Figura 7).

Figura 7 - Processo híbrido elaborado para gestão e desenvolvimento de *software* ágil. Fonte: Os autores.

Por isso a importância de utilizar não somente uma metodologia, mas utilizar a base de cada um como forma de priorizar as problemáticas e dificuldades encontradas pelo grupo, com o intuito de melhorar a qualidade e interação do projeto de acordo com sua necessidade. Dessa forma não só na fase de documentação, mas também nas fases

seguintes toda a equipe tenha as informações que precisam quando chegar a hora de programar e testar o produto.

A Quadro a seguir apresenta as atividades, boas práticas e ferramentas propostas pelo *Framework.*

Quadro 1 - Etapa de Requisitos

Etapa: Requisitos				
Atividade	**Técnica**	**Boas práticas**	**Artefatos**	**Ferramentas[2]**
Definição da visão	Elevator statement (BENASSI, FERREIRA JUNIOR e AMARAL, 2012)	Pitch/Comunicação	*Vision Box*	Word versão 2016
Levantamento de requisitos	Entrevista (PRESSMAN, 2011)	Comunicação efetiva	Documento de visão	Word versão 2016
Identificação dos Casos de uso	UML (PRESSMAN, 2011)	Divisão dos fluxos	Diagrama de casos de uso	Astah versão 8.0
Elaboração das Histórias de usuário	Histórias de usuário (PRIKLADNICKI, WILLI e MILANI, 2014)	INVEST (BRASILEIRO, 2017)	Cartões de histórias de usuário incluídas no product backlog	Kanban do Gitlab
Priorização de requisitos	Numeração de 1 a 5, onde 5 indica a maior prioridade	Priorização baseada em valor (SBOK, 2016)	Cartões de histórias de usuário	Kanban do Gitlab

Fonte: Elaborado pelos autores

[2] As ferramentas propostas foram aquelas utilizadas durante o Projeto EduPlan. Ressalta que estas podem ser substituídas.

Quadro 2 - Etapa de planejamento da sprint

Etapa: Planejamento da *sprint*				
Atividade	**Técnica**	**Boas práticas**	**Artefatos**	**Ferramentas**
Planejamento da *sprint*	Cerimônia (SBROCCO e MACEDO, 2012)	Sessões de planejamento com grupos de usuário	Backlog da *sprint*	Kanban do Gitlab
Definição da Meta da *sprint*	Cerimônia	Respeitar a capacidade da equipe	Backlog da *sprint*	Kanban do Gitlab
Definição da estimativa	Planning poker	Jogo de planejamento	Cartões das historias de usuário	Kanban do gitlab

Fonte: Elaborado pelos autores

Quadro 3 - Etapa de codificação

Etapa: Codificação				
Atividade	**Técnica**	**Boas práticas**	**Artefatos**	**Ferramentas**
Elaboração do Design de interface	Simplicidade, de forma intuitiva	Comunicação	Protótipos de tela	*Framework* de interface
Codificação do incremento	Programação em par (SBROCCO e MACEDO, 2012)	Padronização do código	Código-fonte	Angular JS

Fonte: Elaborado pelos autores

Quadro 4 - Etapa de testes

Etapa: Testes				
Atividade	**Técnica**	**Boas práticas**	**Artefatos**	**Ferramentas**
Definição dos Teste de unidade	Pseudocontrolador (PRESSMAN, 2011)	Foco no componente alvo	Plano de teste	JUnit
Definição dos Teste de aceitação	Cenários	Reunião de revisão do produto	Critérios de aceitação	Aplicação (Dado, Quando, Então)
Definição do Teste de performance	Sobrecarga	Conduzido pelos desenvolvedores	Relatório de tempo de resposta	JMeter
Revisão do produto	Cerimônia	Inspeção, transparência e adaptação (SBOK, 2016)	Incremento aprovado pelo P.O	Reunião

Fonte: Elaborado pelos autores

Quadro 5 - Etapa de transição

Etapa: Transição				
Atividade	**Técnica**	**Boas práticas**	**Artefatos**	**Ferramentas**
Análise de mudanças	Processo de aprovação de mudanças	Flexibilidade e estabilidade	Mudança aprovada ou reprovada – Backlog atualizado	Reuniões

Fonte: Elaborado pelos autores

Treinamento dos usuários	Apoio prático in loco	Sessões presenciais de treinamento	Material de suporte	Produto funcional
Implantação	Implantação contínua	Estratégias de implantação	Entregáveis do produto	Produto funcional

Quadro 6 - Etapa de retrospectiva
Fonte: Elaborado pelos autores

Etapa: Retrospectiva				
Atividade	**Técnica**	**Boas práticas**	**Artefatos**	**Ferramentas**
Retrospectiva	Cerimônia	Comunicação, adaptabilidade	Lições aprendidas	Reunião

5. Aplicação do framework no projeto Eduplan

O *framework* foi aplicado em 26 sprints do projeto Eduplan (Quadro 7). A equipe foi composta por 8 membros, que assumiram os papeis de scrum master, product owner e equipe.

O objetivo do Eduplan foi desenvolver um *software* onde o docente ao fazer o plano de ensino, só será necessário preencher campos específicos, diminuindo o tempo gasto do docente, e facilitando o gestor a verificar os planos de ensino.

A aplicação das atividades, técnicas, boas práticas e ferramentas do *framework* gerou os seguintes artefatos, organizados de acordo com as etapas do processo.

Quadro 7 – Quadro das Sprints do Eduplan

SPRINT	INÍCIO DA SPRINT	FIM DA SPRINT	OBJETIVOS DA SPRINT
1	FEV/18 DIA 05	FEV/18 DIA 19	LEVANTAMENTO DE REQUISITO
2	FEV/18 DIA 26	MAR/18 DIA 05	ANALISE DE REQUISITO E ESTIMATIVA (PLANNING POKER
3	MAR/18 DIA 19	ABR/18 DIA 01	DOCUMENTO DE VISAO
4	ABR/18 DIA 02	ABR/18 DIA 15	CASOS DE USO E HISTORIAS DE USUÁRIO
5	ABR/18 DIA 16	ABR/18 DIA 29	VISION BOX E ELEVATOR ESTATEMENT
6	ABR/18 DIA 30	MAI/18 DIA 13	PROTOTIPO DAS TELAS
7	MAI/18 DIA 14	MAI/18 DIA 27	ANALISE DE RISCOS
8	MAI/18 DIA 28	JUN/18 DIA 10	ANALISE DE RISCOS
9	AGO/18 DIA 12	AGO/18 DIA 26	DOCUMENTO DE ARQUITETURA
10	AGO/18 DIA 27	SET/18 DIA 09	DIAGRAMA DE IMPLANTAÇÃO
11	SET/18 DIA 10	SET/18 DIA 23	REVISÃO DA DOCUMENTAÇÃO
12	SET/18 DIA 24	OUT/18 DIA 07	MODELAGEM DO BANCO DE DADOS
13	OUT/18 DIA 08	OUT/18 DIA 21	CRIAR TELA DE LOGIN E CRIAR DOCENTE

14	OUT/18 DIA 22	NOV/18 DIA 04	CRIAR NOVO PLANO E EDITAR PLANO
15	NOV/18 DIA 05	NOV/18 DIA 18	LISTAR DOCENTE, DELETAR DOCENTE
16	NOV/18 DIA 19	DEZ/18 DIA 02	DELETAR PLANO E DELETAR DOCENTE
17	DEZ/18 DIA 03	DEZ/18 DIA 16	VISUALIZAR PLANO E EDITAR PLANO
18	FEV/19 DIA 17	MAR/19 DIA 02	IMPRIMIR PLANO
19	MAR/19 DIA 03	MAR/19 DIA 16	VERSIONAMENTO COM GITLAB E DOCUMENTAÇÃO DA POLITICA DE VERSIONAENTO
20	MAR/19 DIA 17	MAR/19 DIA 30	PLANO DE TESTE E CASO DE TESTE
21	ABR/19 DIA 01	ABR/19 DIA 13	TESTE UNITARIO
22	ABR/19 DIA 14	ABR/19 DIA 27	TESTE DE PERFORMANCE
23	ABR/19 DIA 28	MAI/19 DIA 11	TESTE DE INTERFACE
24	MAI/19 DIA 12	MAI/19 DIA 25	IMPLANTAÇÃO DO DEVOPS
25	MAI/19 DIA 26	JUN/19 DIA 08	TESTE DE ACEITAÇÃO
26	JUN/19 DIA 09	JUN/19 DIA 22	ENCERRAMENTO

Fonte: Elaborado pelos autores

6. Avaliação do framework

A avaliação do *framework* foi feita através de um questionário, disponibilizado na plataforma Google *Forms* (disponível no link: < https://forms.gle/5NvSa2aqGwPTELdn6>). Os integrantes da equipe avaliaram o processo como um todo, desde, o modelo de comunicação, as atividades e etapas, completude das informações e sua consistência, satisfação com a maneira que foram executadas as *sprints* e também foi dada uma a abertura para sugestões de melhorias e mudanças.

O questionário foi composto por onze questões, sendo elas respondidas por quatro membros do projeto.

12. De acordo com as etapas do processo implementado, na sua opinião precisa de uma adequação quanto a etapas, atividades, papéis/responsabilidade, ferramentas, boas práticas e artefatos? se sim qual?
13. As etapas propostas no processo, foram suficientes para um bom planejamento?
14. O processo contribuiu para que todas as tarefas fossem cumpridas nas *sprint*s?
15. Em relação ao grupo teve alguma dificuldade de comunicação? Tem alguma boa prática que você sugere? (Comunicação)
16. O processo ajudou na elaboração da documentação? Tem algum artefato faltando ou excedendo?
17. O processo ajudou na elaboração do desenvolvimento do *software*? Quais ferramentas você destacaria?
18. O processo ajudou na elaboração dos testes, quanto aos artefatos, boas práticas, ferramentas, atividades e técnicas?
19. Na sua opinião, observando todas as etapas do processo e de acordo com o que foi feito, a equipe conseguiu atingir os objetivos esperados?
20. Dê uma nota de 0 a 10 do processo FGD (*framework* de gestão e desenvolvimento)
21. Os métodos ágeis que foram implementados na elaboração do Eduplan contribuíram para que concluísse os objetivos de maneira qualitativa e quantitativa?

22. Na sua opinião se não utilizasse os métodos ágeis que foram implementados seria possível ter o mesmo rendimento e produtividade?

A análise das respostas permitiram identificar que o time do projeto quanto à adequação das etapas do processo acrescentaram algumas observações durante a prática do processo, algumas foram modificadas, outras não se encaixam no que foi proposto para o *framework*.

“Acredito que todas as etapas estão bem coesas e têm colaborado para o bom andamento do projeto.’’

“No planejamento da *sprint* falta uma discussão da *sprint* após concluir a meta, para haver uma melhor comunicação da equipe’’

“Faltou testes de integração’’

“Na transição poderia ter uma etapa de versionamento E no planejamento uma etapa de riscos.’’

A maioria dos respondentes acreditam que as etapas do processo foram suficientes para um bom planejamento (Figura 24).

Figura 24 – Gráfico referente as respostas da questão 2

Fonte: Elaborado pelos autores

Com relação ao cumprimento das tarefas durante as sprins os membros do projeto afirmaram que o processo contribui e que:

“Os poucos impedimentos que tivemos foi devido a falhas técnicas ou impedimentos terceiros.”

“Houve uma boa organização.”

“Ajudou a manter uma organização das *sprint*s”

Em relação ás dificuldades de comunicação os respondentes sugeriram uma conclusão de *sprint e* a maioria relatou uma comunicação efetiva.

“A comunicação sempre foi efetiva.”

“Uma conclusão de *sprint*.’’

“Todos tiveram uma boa interação.”

“A comunicação é efetiva.”

De acordo os respondentes o processo foi o suficiente para conseguir elaborar a documentação do projeto e não excedeu , nem faltou nenhum documento.

“Temos todos os artefatos.”

“Foi o suficiente.”

“Foi o suficiente.”

“Os artefatos foram o suficiente.”

Quanto a elaboração do desenvolvimento de *software* alguns membros do projeto afirmaram que foi importante o processo principalmente para tomada de decisões e outros destacaram o angular JS como uma importante ferramenta nessa etapa.

“Clarificou as etapas e ajudou a dividir melhor as responsabilidades. Ao priorizar técnicas para cada artefato, também tornou-se mais fácil o desenvolvimento dos mesmos.”

“Angular JS.”

“Angular js.”

“Ajudou, pois quando havia dificuldade com as ferramentas a equipe era bem participativa, resolvendo as questões pendentes e colocando nas *sprint*s as dificuldades, fazendo com que fosse concluída rapidamente.”

Quanto a etapa dos testes os membros do projeto concordaram com a contribuição do processo na fase dos testes e sugeriram a inclusão do teste de integração, porém não foi necessario até o devido momento do projeto.

“Contribuiu para melhor organização da equipe e dos artefatos a serem elaborados e desenvolvidos.”

“Faltando só a inclusão do teste de integração.”

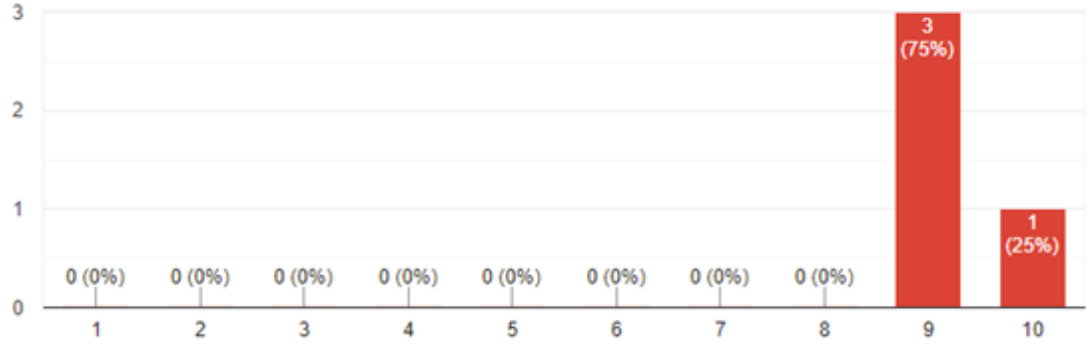
Os respondentes ao observar as etapas do processo concluiram que foi possivel atingir os objetivos esperados 50% respondeu que sim e os outros 50% também responderam que sim, mas deram uma breve explicação da sua afirmação (Figura 25).

Figura 25 - Gráfico referente as respostas da questão 8

Fonte: Elaborado pelos autores

Da nota de 0 a 10 sobre o processo FGD 75% dos membros deram nota 9 e 25% deu nota 10 (Figura 26).

Figura 26 - Gráfico referente as respostas da questão 9



Fonte: Elaborado pelos autores

Os participantes afirmaram que com os métodos ágeis conseguiram alcançar de maneira qualitativa e quantitativa de tudo que foi proposto quanto a elaboração do Eduplan.

"Tanto qualitativa, pois foram alcançados vários artefatos por *sprint* possibilitando uma entrega mais ágil. Assim como melhorou a qualidade dos artefatos."

"Foi concluído tudo que foi proposto."

De acordo os membros da equipe não seria possível ter o mesmo rendimento e produtividade sem a utilização dos métodos ágeis, pois contribuiu para um bom planejamento e não teria os mesmos resultados gerados sem a utilização do processo.

"Levaria mais tempo para engajar a equipe e produzir os resultados gerados com o processo."

"Com a metodologia o planejamento melhorou o rendimento."

"O processo contribuiu para um bom planejamento."

5. Considerações Finais

A definição de um processo híbrido, que atenda às especificidades do projeto e da equipe, é importante, pois as atividades que têm que ser exercidas são feitas de forma iterativa dentro dos prazos pré estabelecidos e que por consequências da organização deste ciclo resulte em um projeto de qualidade, como pôde ser visto na experiência da fábrica.

O principal impacto desejado é o de viabilizar para que outros alunos possam utilizar do framework no gerenciamento de seus projetos, tanto no âmbito acadêmico, como profissional ou, ainda, como base para criação de metodologias próprias, enriquecendo o aprendizado tanto nas disciplinas de Prática em Fábrica de *Software*, como no mercado de trabalho, sendo uma experiência de grande valia tanto no currículo acadêmico como profissional.

O desenvolvimento do presente estudo possibilitou a avaliação do framework que foi criado e validado em um estudo de caso nas disciplinas de PFS, onde utilizando métodos ágeis foi possível criar um processo ágil que se encaixasse especificamente na equipe e no projeto. A utilização de um processo para gestão e desenvolvimento foi fundamental para desenvolver um *software* que atendesse as especificidades do projeto, e que esclarecesse o que precisava ser feito e como deveria ser feito.

Referências

MDS - Metodologia De Desenvolvimento De *Software*. Datasus. http://Datasus.Saude.Gov.Br/Metodologias/Mds-*Software*.

Medeiros, H. (2013) "Práticas Em Xp: Extreme Programming". Https://Www.Devmedia.Com.Br/Praticas-Em-Xp-Extreme-Programming/29330.

Nazário, R. L.; Rufino, R. O Conceito De Tdd No Desenvolvimento De *Software*. Universidade Paranaense. Paranavaí, P. 5. 2015.

Pressman, R. S. (2011) Engenharia De *Software*: Uma Abordagem Profissional. 9ª. Ed. Pearson Makron Books.

Prikladnicki, R.; Willi, R.; Milani, F. (2014) Métodos Ágeis Para Desenvolvimento De *Software*. Bookman.

Rocha, F. G. (2013) Tdd: Fundamentos Do Desenvolvimento Orientado A Testes.

Rocha, F. G. (2014) Integrando Xp As Principais Metodologias Agéis. Devmedia. [S.L.].

Rocha, F. G. (2018) "Integrando Xp As Principais Metodologias Agéis". https://Www.Devmedia.Com.Br/Integrando-Xp-As-Principais-Metodologias-Ageis/30989.

Sbrocco, J. H. T. D. C.; Macedo, P. C. (2012) Metodologias Ágeis: Engenharia De *Software* Sob Medida. Érica.

Schwaber, K.; Sutherland, J. Scrum Guide. (2016) Tradução De Fábio Cruz. [S.L.]: [S.N.], https://Www.Scrumguides.Org.