UNIVERSIDADE FEDERAL DO PARÁ
CAMPUS DE CASTANHAL
FACULDADE DE COMPUTAÇÃO
CURSO DE BACHARELADO EM SISTEMAS DE INFORMAÇÃO

JUAN CARLOS DO NASCIMENTO BARATA

UMA PROPOSTA DE MELHORIA DO PROCESSO E QUALIDADE DO SOFTWARE USANDO BPMN: UM ESTUDO DE CASO DA COTIC PROEG

Castanhal – Pará
2018

JUAN CARLOS DO NASCIMENTO BARATA

UMA PROPOSTA DE MELHORIA DO PROCESSO E QUALIDADE DO SOFTWARE USANDO BPMN: UM ESTUDO DE CASO DA COTIC PROEG

Trabalho de Conclusão de Curso apresentado à Universidade Federal do Pará, como requisito para obtenção do título de Bacharel em Sistemas de Informação da Faculdade de Computação, Campus de Castanhal, sob a orientação da Prof.ª. Ms. Elziane Monteiro Soares.

Castanhal – Pará

2018

JUAN CARLOS DO NASCIMENTO BARATA

UMA PROPOSTA DE MELHORIA DO PROCESSO E QUALIDADE DO SOFTWARE USANDO BPMN: UM ESTUDO DE CASO DA COTIC PROEG

Trabalho de Conclusão de Curso apresentado à Universidade Federal do Pará, como requisito para obtenção do título de Bacharel em Sistemas de Informação da Faculdade de Computação, Campus de Castanhal, sob a orientação da Prof.ª. Ms. Elziane Monteiro Soares.

Orientadora: Prof.ª. Ms. Elziane Monteiro Soares

Data da aprovação: Castanhal-PA. _____/_____/_____

Banca Examinadora

_____

Prof.ª. Ms. Elziane Monteiro Soares

Orientadora – Universidade Federal do Pará – FACOMP/ Campus Castanhal

_____

Prof.º. Dr. Tássio Costa de Carvalho – Coorientador

Coorientador – Universidade Federal do Pará – FACOMP/ Campus Castanhal

_____

Prof.º. Dr. José Jailton Henrique Ferreira Junior

Membro Interno – Universidade Federal do Pará – FACOMP/ Campus Castanhal

_____

Especialista – Diego Assis da Silva Lisbôa

Membro Externo – Universidade Federal do Pará – Campus Belém

Castanhal – Pará

2018

AGRADECIMENTOS

Agradeço, primeiramente, a Deus por sua infinita misericórdia e pelo seu infinito amor por mim. Por ter me dado o privilégio de vivenciar esse grande momento em minha vida e cumprir tudo o que prometeu.

Aos meus pais Jean Carlos Corrêa Barata e Aldenora Oliveira do Nascimento pelo grande referencial de vida que são para mim, por terem me ensinado e me formado o homem de caráter que sou. Aos meus avós Amilton Garcia Barata, Zaid Corrêa Barata e Orlandina Oliveira do Nascimento por todos os conselhos dados, carinho ofertado e cuidado dedicado durante minha vida. A toda minha família pelo apoio oferecido em todos os momentos ao longo dessa graduação e por me motivar a sempre lutar pelas minhas metas de vida.

A minha querida noiva Layra Emilli Gonçalves Salomão e toda sua família pela paciência e compreensão quando precisei me ausentar para focar um pouco mais neste trabalho aqui apresentado. Por todo apoio e força concedido ao longo destes quase 2 anos que estamos juntos. Pelo amor e carinho de todos os dias, os quais estão sendo os combustíveis deste relacionamento.

A todos que fazem parte da COTIC, pessoas que ao longo do tempo passaram de colegas de trabalho para amigos, os quais serão lembrados eternamente por mim. Agradeço também por todo o conhecimento adquirido ao longo de minha permanência no estágio.

A Prof.ª. Ms. Elziane Monteiro Soares, ao Prof.º. Dr. Tássio Costa de Carvalho e ao meu Chefe de estágio Diego Assis da Silva Lisboa por toda a paciência dedicada, pelas críticas construtivas e por me orientarem na execução deste trabalho.

RESUMO

As técnicas para melhoria de processo se atualizam constantemente para manter as organizações competitivas e suprir as necessidades de seus clientes. Uma organização com a pretensão de alcançar maior produtividade e qualidade no desenvolvimento de software precisa ter seu processo muito bem definido e atualizado. Como forma de constante melhoria, a Notação de Modelagem de Processos de Negócio veio para alinhar a tecnologia da informação e processos para agregar valor ao negócio e, consequentemente, aos seus clientes. O presente trabalho propõe padronizar o processo de desenvolvimento de software da Coordenadoria de Tecnologia da Informação e Comunicação da PROEG UFPA e incluir uma nova metodologia ágil em uma das etapas do processo como forma de melhoria. Logo, os resultados obtidos através da inclusão do Test Driven Development no processo de desenvolvimento de software da referida empresa, melhoraram a qualidade dos produtos e agilizaram ainda mais o processo.

PALAVRAS-CHAVE: Desenvolvimento de Software, Processo de desenvolvimento de Software, Melhoria de Processo, Metodologia ágil, Notação de Modelagem de Processos de Negócio, Tecnologia da Informação e Comunicação, Test Driven Development.

ABSTRACT

As techniques to improve the upgrade process to keep organizations competitive and meet the needs of their customers. An organization with a claim to higher performance and quality in software development has its own process very well updated and updated. As a form of constant improvement, Business Process Modeling Notation for information technology and processes add value to the business and consequently to its customers. The present work aims to standardize the software development process of the Information and Communication Technology Coordination of PROEG UFPA and to include a new methodology in a stage of the improvement process. Therefore, the results obtained through the inclusion of TDD in the software development process of the company, improved the quality of the products and further expedited the process.

KEYWORDS: Software Development, Software Development Process, Process Improvement, General Methodology, Business Process Modeling Notation, Information and Communication Technology, Test Driven Development.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIAÇÕES

COTIC - COORDENADORIA DE TECNOLOGIA DA INFORMAÇÃO E COMUNICAÇÃO
AIT - ASSESSORIA DE INFORMAÇÃO E TECNOLOGIA
PROEG - PRÓ-REITORIA DE ENSINO DE GRADUAÇÃO
UFPA - UNIVERSIDADE FEDERAL DO PARÁ
BPM - BUSINESS PROCESS MANAGEMENT
BPMN - BUSINESS PROCESS MODEL NOTATION
FDD - FUTURE DRIVEN DEVELOPMENT
TDD - TEST DRIVEN DEVELOPMENT
BPMI - BUSINESS PROCESS MODELING INITIATIVE
OMG - OBJECT MANAGEMENT GROUP
XP - EXTREME PROGRAMMING
CRC - CLASSES-RESPONSABILIDADES-COLABORADORES
SWOT – STRENGTHS-WEAKNESS-OPPORTUNITIES-THREATS
TI – TECNOLOGIA DA INFORMAÇÃO
UML – UNIFIED MODELING LANGUAGE

SUMÁRIO

1. INTRODUÇÃO

1.1 CONTEXTO

Por um longo período as empresas tinham o foco voltado para a linha de produção em massa, popularizada por Henry Ford no século XX. Porém, com a chegada da globalização e a evolução constante da tecnologia, as empresas viram a necessidade de conhecer melhor seus processos e aprimorá-los para agilizar suas entregas e melhorar o seu produto constantemente, a fim de manter-se competitiva no mercado (FREITAS, 2016). Não foi diferente com as empresas de TI (Tecnologia da Informação), as quais também passaram a aderir esses tipos de técnicas, procurando conhecer seus processos, melhorá-los, e gerenciá-los de forma cada vez mais eficiente.

As cooperativas que buscam melhorar seus processos utilizam técnicas e ferramentas de modelagem para visualizar e planejar seus processos. De acordo com SANT'ANNA e ABDALA (2004), "A modelagem de processos e o suporte automatizado de um ambiente integrado apresentam-se como facilitadores para definição e a efetiva implantação em uma organização desenvolvedora de software".

Uma abordagem utilizada para a modelagem de processo é o *Business Process Management* (BPM), o qual permite o mapeamento dos processos organizacionais da empresa, buscando a integralização funcional e agilizando as atividades desenvolvidas, auxiliando no caminho em que deve percorrer até o seu objetivo. Através das notações de modelagem de processos utilizando o *Business Process Model Notation* (BPMN), pode ser feita toda a documentação dos modelos criados, facilitando o entendimento dos processos aos participantes dos projetos.

Seguido dessas adequações ao mercado competitivo e para melhorar os processos de desenvolvimento de software, surgiram às metodologias ágeis, com o objetivo de valorizar a produtividade e melhorar os produtos. As metodologias mais conhecidas nesse ramo e que já são utilizadas pela COTIC são: *SCRUM, KANBAN, eXtreme Programming e Feature Driven Development* (FDD). Neste trabalho, será apresentada uma proposta de melhoria para o processo de desenvolvimento de software da empresa através da modelagem de processos com BPMN, e será inserido no processo os conceitos aplicados no *Test Driven Development* (TDD).

Segundo FREITAS (2016), "é de grande importância para as corporações conhecerem seus processos para analisar os seus pontos positivos que devem permanecer e os negativos a

corrigi-los ou eliminá-los. Além disso, é interessante que essas tenham conhecimento das características e técnicas usadas nos seus procedimentos".

1.2 MOTIVAÇÃO E JUSTIFICATIVA

Uma empresa, seja ela da esfera pública ou privada, possui diversos tipos de processos de negócio. De acordo com REBOUÇAS (2007), processo "é o conjunto estruturado e intuitivo das funções de planejamento, organização, direção e avaliação das atividades sequenciais, que apresentam relação lógica entre si". Com isso, a modelagem de processo ajuda a organizar, estruturar e definir as tarefas e seus procedimentos, trazendo mais clareza ao modelo utilizado.

Partindo do princípio de que "tudo que realizamos em uma organização está relacionado a um processo" (ARAUJO, GARCIA & MARTINES, 2011), a modelagem de processos também possibilita identificar as melhorias que podem fazer com que o processo em análise tenha maior eficácia e eficiência. Entretanto, muitas organizações têm dificuldades e/ou ainda não sabe como se beneficiar e maximizar seus ganhos através de processos e, por conseguinte, sua modelagem (CRUZ, 2009).

É por meio dos processos que os produtos ou serviços são entregues. Um processo de negócio é aquele que entrega valor ao cliente (GUERRINI, et al., 2014) enquanto que o processo crítico é aquele que atinge diretamente os objetivos organizacionais.

A modelagem é uma forma utilizada para a compreensão dos processos, independentemente de que tipo ele seja, contanto que seja entendida de forma a transparecer as atividades necessárias para redesenhar e documentá-los sempre que for necessário. A modelagem pode ser utilizada em qualquer tipo de seguimento, seja da esfera privada ou pública, a qual é o caso da COTIC, analisada nesse presente trabalho.

Portanto, o desenvolvimento da modelagem de processo de software da Coordenadoria de Tecnologia da Informação e Comunicação da Pró-reitoria de Ensino de Graduação (COTIC PROEG) é justificada pela possibilidade de compreender melhor os processos adotados, mapeá-los utilizando a modelagem, identificar e propor melhorias, automatizando-as a fim de agregar valor e eficácia ao serviço prestado pela mesma. Além disso, o estudo realizado com foco na COTIC, também pode servir de referência para ser aplicada em outros órgãos com as mesmas características.

1.3 OBJETIVOS

1.3.1 OBJETIVOS GERAIS

O objetivo deste trabalho é apresentar uma abordagem prescritiva para modelo de processo de desenvolvimento da COTIC, modelando todo o processo utilizado pela empresa através do BPMN, propondo melhorias com a utilização da metodologia do *Test DrivenDevelopment*, a fim de solucionar ou minimizar os problemas existentes no gerenciamento e desenvolvimento dos projetos de software da organização.

1.3.2 OBJETIVOS ESPECÍFICOS

- Estudar as metodologias usadas pela instituição;
- Identificar os processos que são adotados pela COTIC;
- Realizar a modelagem de processo utilizado atualmente;
- Sugerir melhorias para o processo com base na modelagem realizada;
- Apresentar a modelagem do processo melhorado;

1.4 METODOLOGIA

Com a finalidade de criar sistemas para a Pró-reitoria de Ensino de Graduação, agilizando os processos deste setor da Universidade Federal do Pará, a COTIC, anteriormente chamada de AIT (Assessoria de Informação e Tecnologia), foi fundada em meados de agosto do ano de 2009, e somente em outubro de 2017 passou a se chamar por este nome. Para alcançar seus objetivos com mais rapidez e organização, a coordenadoria passou a utilizar as metodologias ágeis em seus processos.

De acordo com CASTRO, CARDOSO, LISBOA (2011), inicialmente foi utilizado o *Scrum* como principal metodologia e algumas práticas do *eXtreme Programming* nos primeiros processos, porém, atualmente, utilizam o KANBAN. Estas metodologias são usadas de forma adaptável com a finalidade de otimizar o processo.

Dessa forma, a COTIC faz utilização de diferentes técnicas de diversas metodologias ágeis, sempre buscando os melhores métodos a serem utilizados em sua forma de trabalho para ter um processo que satisfaça os interesses da organização e, principalmente, do seu cliente.

As características das metodologias utilizadas na COTIC serão descritas e detalhadas na Tabela 1 a seguir:

Tabela 1: Metodologias e características adotadas pela COTIC.

Metodologia	Características usadas
Scrum	● Ciclo do Scrum usado diariamente ● Artefatos: ○ Product Backlog ○ Selected Product Backlog ○ Sprint Planning ○ Burndown Chart ● Papéis: rotatividade do Scrum Master pelo time ao iniciar uma nova iteração (sprint) ● Eventos (sprint e reuniões) ○ Sprint Planning ○ Daily Scrum ○ Sprint Review ○ Sprint Retrospective
XP	● Codificação e Testes ● Práticas: ○ Cliente presente ○ Programação em par ○ Refactoring ○ Código padronizado
KANBAN	● Limitar a quantidade de serviço a ser executado (WIP) ● Métricas e otimização do fluxo de serviço ● Explicitar os processos ● Quadro Kanban - facilitar a visualização do fluxo de serviço

Fonte: Autor (2018).

Além das características acima citadas, a COTIC faz uso de outras ferramentas, as quais são a *Gamificação*, que segundo FREITAS (2016) é o uso de jogos para engajar os membros do time, os motivando a resolverem problemas, incrementar a participação e comprometimento

por parte dos membros,e o *Planning poker*, que segundo SABBAGH (2013), "é a técnica mais conhecida e utilizada por times ágeis para realizar as estimativas de itens do Product Backlog".

Diante disso, nota-se que a COTIC se preocupa com a melhoria constante de seu processo e adota diversas técnicas sem fazer utilização de apenas uma específica. Dessa forma acaba criando o seu próprio processo de desenvolvimento de software, de acordo com suas próprias necessidades e, caso necessite realizar mudanças ou novas adaptações ao seu processo visando melhorá-lo, esta coordenadoria não exita em assim o fazer.

Dessa forma, a metodologia de pesquisa e implantação utilizada neste trabalho foi dividida em quatro etapas (pesquisa bibliográfica, levantamento de dados e procedimentos, escolha das ferramentas para o desenvolvimento do trabalho e, por fim, uma nova modelagem para o processo da COTIC), as quais serão descritas a seguir:

Primeiramente, foi feita uma pesquisa bibliográfica sobre as práticas de desenvolvimento com a utilização da engenharia de software, das principais metodologias ágeis e práticas de modelagem de processos a fim de proporcionar um maior entendimento dos assuntos abordados para desenvolver este trabalho.

Em seguida, foi feito um levantamento de dados e procedimentos que são utilizados na COTIC, através de entrevistas presenciais com os integrantes da organização para entender o processo adotado pela instituição.

Na terceira etapa, foi feito a escolha da ferramenta e a notação que utilizada para modelar os processos da instituição. Com isso, desenvolveu-se as modelagens dos processos de acordo com a forma que a organização trabalha atualmente As Is, possibilitando estudos para identificar possíveis melhorias no processo atual.

Como etapa final do trabalho, foi realizada uma nova modelagem To Be para o processo da COTIC, mostrando a introdução de uma nova técnica de desenvolvimento de software, como o TDD, a fim de agregar mais qualidade ao processo de desenvolvimento da empresa.

1.5 ORGANIZAÇÃO DO TRABALHO

Este trabalho foi divido em 5 capítulos, sendo eles:

- Introdução: contextualiza o tema de gerenciamento de processos de negócio, apresenta a questão de pesquisa, além dos objetivos e dos métodos de pesquisa utilizados.
- Trabalhos correlatos: mostra a utilização de 4 (quatro) trabalhos de conclusão de curso e 1 (um) artigo que possuem conteúdos importantes para a elaboração e fundamentação desta monografia. São apresentados de forma resumida, porém, enfatizando a sua

importância e colaboração para fundamentar os temas e pontos cruciais deste trabalho, tratando do gerenciamento de processos, linguagens de modelagem de software, modelagem e automação de processos.

- Fundamentação teórica: contextualiza a adoção e evolução dos métodos ágeis de desenvolvimento de software e gerenciamento de processos de negócio, desde seus surgimentos até os dias mais atuais. É apresentado em forma de subseções, temas como: gestão de processo ágeis, modelo de maturidade de processos de negócio, ciclo de vida dos principais métodos ágeis, modelagem, análise e técnicas de processo de negócio, notação de modelagem de processos de negócio, sistemas de gestão de processos de negócio e ferramentas, métodos e técnicas de automação de processos.
- Estudo de caso com a modelagem do processo de software da COTIC: mostra os 4 passos utilizados para desenvolver a aplicação prática das modelagens do processo utilizado na coordenadoria, passando pela identificação e mapeamento dos processos, levantamento dos dados para a criação dos modelos, desenvolvimento do modelo atual e a proposta de melhorias através da remodulação para o processo atual utilizado.
- Considerações finais: ponto de conclusão do trabalho, discussão sobre o desenvolvimento da modelagem e automação dos processos, além de apontar os pontos de melhoria através da remodelagem. Então, são apresentadas as dificuldades encontradas durante a realização desta monografia e, em seguida, é feito um agradecimento à COTIC-PROEG por aceitar o desenvolvimento deste trabalho. Por fim, é apresentado os trabalhos futuros.

2. TRABALHOS CORRELATOS

Este capítulo tem por objetivo apresentar o processo de revisão e análise de alguns trabalhos relacionados sobre automação de processos de negócio que serviram de base para o desenvolvimento desta monografia. Boa parte da pesquisa ocorreu por meio de busca em site de referência em artigos acadêmicos: Google Scholar[1].

No total, foram encontrados cerca de 12 artigos na plataforma. Em seguida, foi realizada a leitura integral dos trabalhos para verificar a relevância do conteúdo contido nos trabalhos. Ao todo, foram selecionados 5 artigos, os quais serão apresentados resumidamente a seguir.

O autor PIZZA desenvolveu o trabalho de conclusão de curso com o tema *A metodologia Business Process Management (BPM) e sua importância para as organizações,* onde foi abordado com mais detalhes os pontos positivos da utilização do BPM a nível de gerenciamento de empresas e seus projetos.

O foco maior deste trabalho foi em explicar e exemplificar a utilização das notações BPM e BPMN 2.0, a ponto de mostrar detalhadamente como funciona e a maneira que deve ser aplicada cada artefato das notações. A monografia foi escrita no ano de 2012 e defendida na Faculdade de Tecnologia de São Paulo, e se tornou uma das referências nacionais mais utilizadas em trabalhos que envolvem este tema.

Um artigo a ser enfatizado é o dos autores SANT'ANNA e ABDALA, o qual trouxe o tema *Modelagem do Processo de Gerenciamento da Configuração de Software para um Ambiente Integrado,* onde foi feito um estudo para mostrar a importância de ter um processo de desenvolvimento e o quanto é essencial para a engenharia de software uma empresa possuir um processo bem definido. O desafio do artigo é apresentar dois modelos ISO/IEC (12207 e 15504) para definir o processo da *Gerência da Configuração de Software.*

Para o desenvolvimento deste trabalho, foi utilizada a notação padrão UML da linguagem de processos SPEM para facilitar a atividade de modelagem. Ao longo do artigo, ele exemplifica a utilização da modelagem de processo através do diagrama de atividades e propõe a ampliação da lista através do desenvolvimento do diagrama de casos de uso para mostrar a relação entre os papéis e as principais definições de trabalho, e o desenvolvimento do diagrama de estado para ilustrar o comportamento do modelo.

O autor GONÇALVES trouxe no ano de 2016 o tema *Guia para Modelagem e Automação de Processos de Negócios Acadêmicos: estudos de caso com processos da UFSC,*

[1]https://scholar.google.com.br/

onde foi feito um estudo de caso para modelar e automatizar os processos de negócios acadêmicos da instituição. O desafio era focar na integração do controle das atividades e subprocessos executados por sistemas manuais e automatizados.

Neste trabalho, foi utilizado o padrão BPMN 2.0 para a modelagem prática e automação dos processos. Como resultado, o guia de modelagem e automação desenvolvido serviu como forma de redução para o tempo de execução das tarefas e da burocracia de alguns processos, uma vez que a tramitação é feita eletronicamente, sem a necessidade de assinaturas e o envio de documentos pessoalmente. Além de permitir que os envios nos processos possuam uma atualização em tempo real.

Com o trabalho de conclusão de curso no ano de 2013 e tematizado de *Modelagem de Software: um estudo de caso em uma empresa de distribuição de frutas,* o autor RODRIGUES enfatiza a utilização das modelagens de software durante o desenvolvimento de um sistema para uma microempresa. O objetivo do trabalho é criar os diagramas de todo o sistema e modelá-los através da notação padrão UML da linguagem de processos SPEM e descrever todos os passos do desenvolvimento do produto.

Para o desenvolvimento da monografia, foi utilizada a ferramenta StarUML, por ser um programa gratuito e obter uma vasta biblioteca de modelagem de diagramas diferentes, o que possibilitou a realização de quase todos os diagramas necessários para o estudo do autor. Outra ferramenta utilizada, foi o Microsoft Office Access 2007, escolhida com a finalidade de modelar o diagrama de implantação, já que proporciona uma visualização privilegiada dos layouts do sistema a ser oferecido.

O trabalho que de fato teve maior atenção foi o da autora FREITAS, o qual destacou o tema *Descrição de um processo de software usando SPIDER_ML: um estudo de caso da AIT-PROEG,* onde foi realizado um estudo no ano de 2016 sobre as principais metodologias ágeis de desenvolvimento de software, afim de entender como elas são utilizadas e mescladas no decorrer de todo o processo da Assessoria de Informação e Tecnologia da PROEG UFPA.

Para realizar o desenvolvimento da monografia, foi utilizada a linguagem de modelagem de processo SPIDER_ML, onde foi desenvolvida a modelagem de 3 processos (chamadas de “disciplinas” durante o trabalho), os quais são: Disciplinas Solicitações, Disciplinas Serviços e Disciplinas User Stories. Estes modelos serviram para encontrar pontos de melhorias no processo do setor contemplado.

A partir da análise dos trabalhos citados acima, podemos perceber que a utilização da modelagem de processos ágeis de desenvolvimento de software foi considerado no escopo de apenas uma dessas pesquisas. Como diferencial, o presente trabalho, em comparação aos

trabalhos relacionados, fornecerá um mapeamento das principais abordagens ágeis (Scrum, Kanban, XP e TDD), e fornecerá o embasamento necessário para a elaboração da abordagem de boas práticas utilizando as metodologias ágeis, dando apoio ao processo de aquisição pela COTIC PROEG e de empresas de desenvolvimento de software semelhantes.

3. REFERENCIAL TEÓRICO

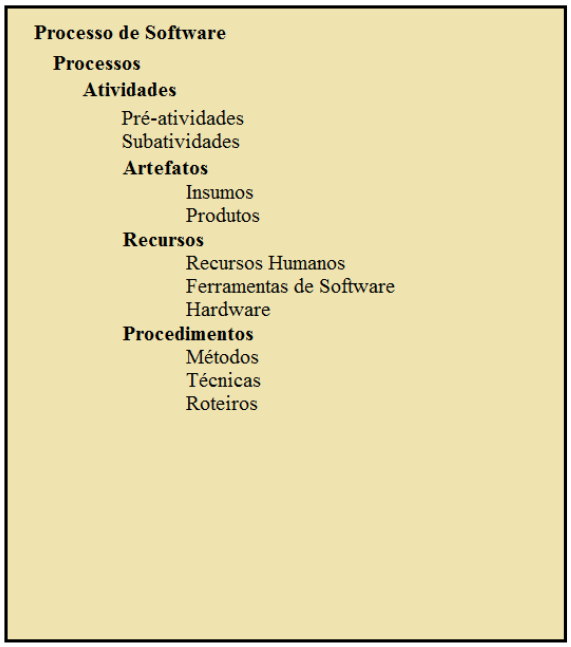
3.1 PROCESSO DE SOFTWARE

De acordo com FALBO (2005), o processo de software é um conjunto de atividades, métodos, práticas e transformações norteadoras no desenvolvimento de um software e sua documentação. A eficácia de um processo deve conter relações claras entre as atividades, seus artefatos produzidos, ferramentas e procedimentos necessários, seguido de habilidade, treinamento e motivação dos envolvidos no processo, proporcionando estabilidade, controle e organização para o desenvolvimento do projeto.

Segundo PRESSMAN (2016), a organização e o gerenciamento vieram ganhando força dos últimos anos, pois garantem um planejamento do desenvolvimento e que a entrega no prazo seja realizada, questão muito debatida nos processos de softwares que pode ser resolvida ou amenizada.

Os processos de software são decompostos em diversos processos, os quais são de desenvolvimento, garantia de qualidade e gerência de projetos. Por sua vez, os processos são compostos por atividades, que também são compostos por subatividades como descrito na Figura 1.

Figura 1: Elementos que compõem um processo de software.



Fonte: ADAPTADO DE FALBO (2005)

Segundo EBERT (2009), considera-se no processo de software os seguintes pontos:

- Recursos: são necessários para a execução das atividades;
- Atividades: são executadas pelos recursos e geram algum artefato;
- Artefatos: são produtos gerados por atividades, e que podem ser matéria-prima para outras atividades do processo;

3.2 MODELAGEM DE PROCESSO

No dicionário, a palavra modelo significa uma "representação, em escala reduzida, de objeto, um conceito, uma função ou atividade a ser reproduzida em dimensões normais". Segundo MOORE (2013), a modelagem de processos é um agregado de atividades que tem como objetivo representar um negócio existente, e requer um conjunto de habilidades e técnicas para compreender, comunicar e gerenciar componentes de processos de negócio.

Segundo DERNIAME, KABA e WASTELL (1999), o modelo de processo deve possuir quatro elementos básicos, os quais são:

- Atividade: são os passos do processo, os quais detalham o modo de execução do processo.
- Produtos: são artefatos gerados durante ou no fim de um processo.
- Papéis: são as responsabilidades de quem executa a atividade.
- Ferramentas: são usadas na produção do software.

> A modelagem de processos ajuda empresas a conseguirem um melhor controle e gerenciamento dos processos. Para empresas de software, a modelagem de processos de software é de extrema importância para a qualidade do produto final. (EBERT, 2009).

Os principais objetivos e benefícios de se modelar o processo são:

- Possibilitar a comunicação e o entendimento do processo;
- Padronizar o processo;
- Apoiar a evolução do processo;
- Facilitar o gerenciamento do processo;

3.3 LINGUAGEM DE MODELAGEM DE PROCESSO

Segundo STOROLLI, ZANOLLA e BORSOI (2009), para representar o modelo de processo são usadas linguagens de modelagem de processos. Os modelos obtidos com essas

linguagens obedecem a critérios sintáticos e semânticos dos conceitos notacionais. Esses conceitos compõem o vocabulário básico para representar a arquitetura de processo.

Neste trabalho, iremos utilizar a linguagem *BPM* para modelar os processos da COTIC. A seguir, iremos falar mais detalhadamente dessa linguagem de modelagem.

3.3.1 Business Process Management (BPM)

Segundo CRUZ (2008), BPM é um conjunto de múltiplos elementos, conceitos e metodologias que juntos tem a finalidade de tratar de forma holística processos de negócio.

Com a junção desses elementos, o BPM tem como objetivo, possibilitar a organização, maior visibilidade e a integração de seus ambientes e das atividades de cada colaborador em seu processo de negócio (PIZZA, 2012).

Quando a empresa decide utilizar-se da modelagem BPM, primeiramente precisa mapear seus processos, para que seja possível definir qual será o escopo do projeto, não importa o método utilizado para obtenção desse mapeamento, o importante é que tanto escopo quanto método estejam claramente definidos. (CRUZ, 2008)

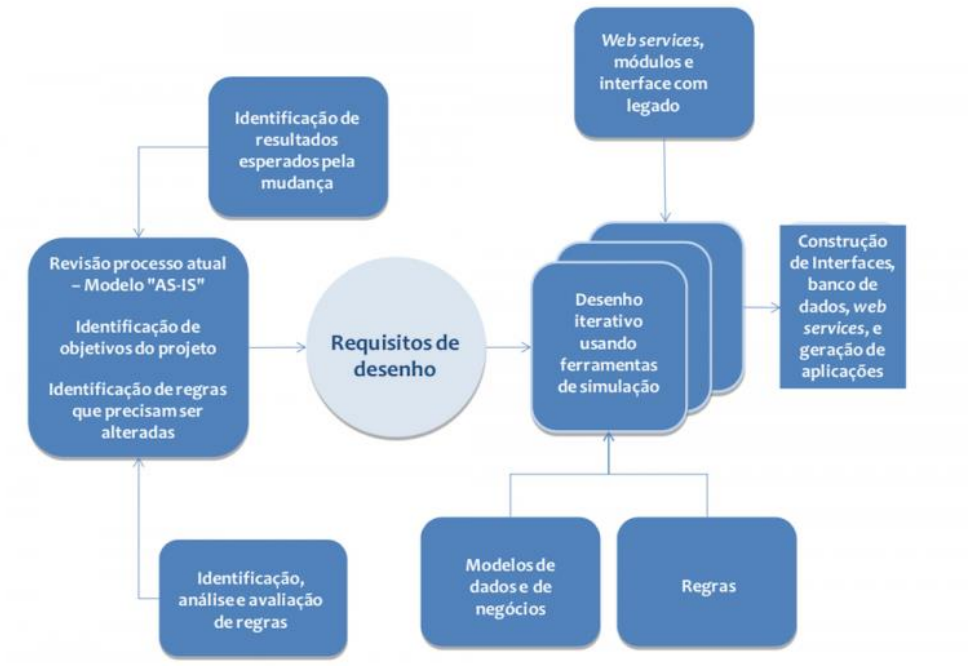
3.3.1.1 O ciclo do BPM

Segundo o *Guide to the Business Process Management Body of Knowledge* (BPM CBOK), o ciclo do BPM é composto por seis fases, os quais serão descritos e definidos a seguir:

- Planejamento: é o início de tudo. Nessa fase é necessário analisar e revisar toda a documentação existente, com a finalidade de identificar e entender de que maneira os processos estão alinhados, para se obter uma visão ampla dos processos existentes na empresa. Para isso, existem quatro etapas a serem executadas. As etapas são:
  - Emoldurar a organização.
  - Identificar os processos primários, de gestão e de apoio.
  - Identificar indicadores de desempenho.
  - Preparar para análise de processos.
- Análise: N essa fase é necessário observar como os processos ocorrem em sua exatidão dentro da empresa, para que possamos avaliar os processos da empresa e assim começar a pensar no desenho da modelagem desses processos.
- Desenho: Após a identificação dos gargalos, falhas e deficiências dos processos analisados, é necessário alinhar os objetivos da empresa e desenhar um novo processo. Para isso, é necessário fazer simulações e prototipagem com base em

cenários e incluir as melhorias necessárias. A seguir, uma ilustração dos passos para a criação de um novo processo, Figura 2:

Figura 2: Desenho de um novo processo.



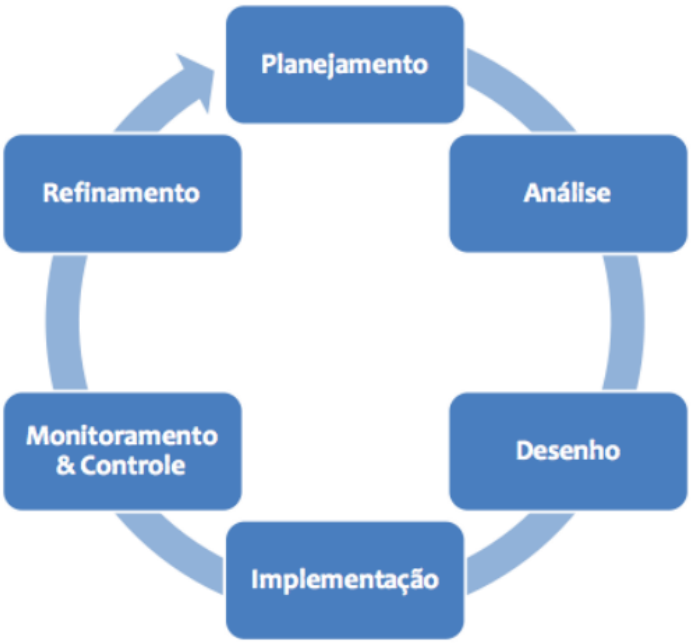
Fonte: CBOK ABPMP

- Implementação: Pode ser executada de duas formas. Através de uma implementação sistêmica com auxílio de ferramentas específicos, ou através de uma implementação não sistêmica sem a utilização de ferramentas. Independente da forma que for escolhida, o objetivo será pôr em execução os processos como foram definidos e documentados, em seguida, criar o fluxo de trabalho.
- Monitoramento e Controle: Fase onde é possível descobrir se os processos implementados estão de acordo com os objetivos da empresa através de quatro indicadores de avaliação. Os indicadores de desempenho mais utilizados são:
  - Tempo de execução do processo.
  - Custo com o processo.
  - Capacidade efetivo do processo.
  - Qualidade do produto produzido ao cliente a partir do processo.
- Refinamento: Também chamado de transformação dos processos através de uma evolução planejada e monitorada por meio dos resultados obtidos. Fase em que

a proposta de melhoria contínua dos processos é empregada a partir do monitoramento e controle utilizado no passo anterior do ciclo.

A Figura 3 ilustra o funcionamento do ciclo do BPM, em seis passos:

Figura 3: Ciclo do BPM



Fonte: IGEPP (PORTO, Gilberto)

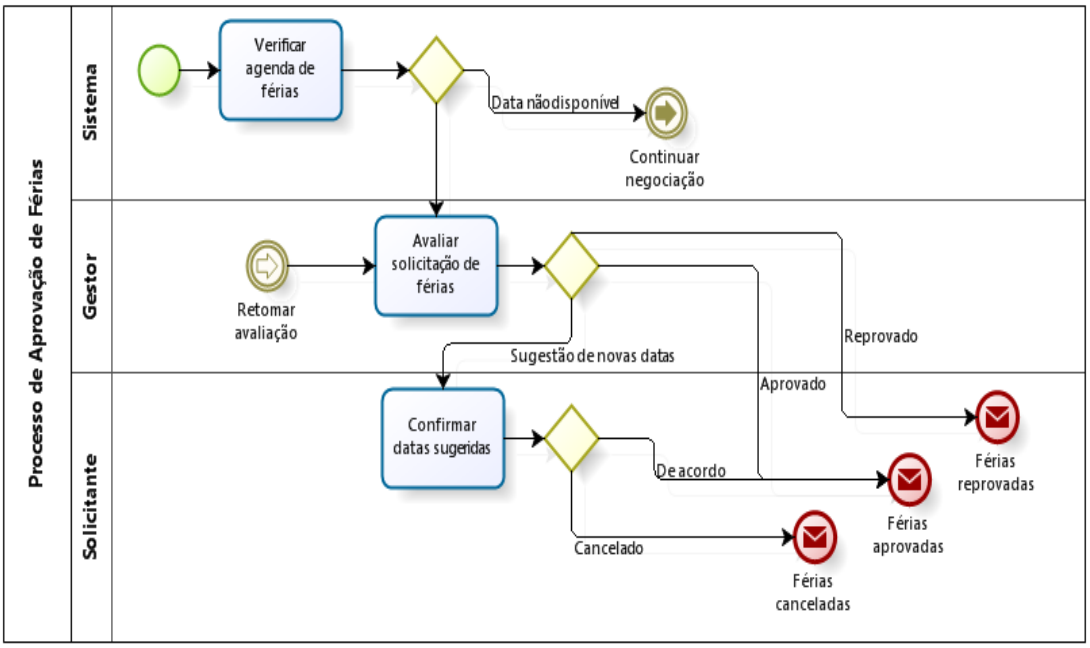
3.3.2 Business Process Modeling Notation (BPMN)

Foi desenvolvido pelo *Business Process Modeling Initiative* (BPMI) e, atualmente, mantida pelo *Object Management Group* (OMG) após uma fusão das duas organizações em 2005. Em março de 2011, foi lançada a versão 2.0 do BPMN (PIZZA, 2012).

Segundo PIZZA (2012), *Business Process Modeling Notation* (BPMN) é uma notação da metodologia de gerenciamento de processos de negócio e trata-se de uma série de ícones padrões para o desenho de processos, o que facilita o entendimento do usuário. A modelagem é uma etapa importante da automação, pois é nela que os processos são descobertos e desenhados. É nela também que pode ser feita alguma alteração no percurso do processo visando a sua otimização.

O objetivo da notação BPMN é representar os processos BPM através de representações gráficas, possibilitando a visualização do processo de negócio em seu estado atual, e após analisado o processo, representar como ficará com a alteração do processo. A Figura 4 a seguir demonstra um exemplo de como funciona o BPMN.

Figura 4: Exemplo de notação BPMN



Fonte: IPROCESS

Na Figura 4, mostra-se um exemplo de um processo de solicitação de férias de um funcionário desde a verificação da agenda de férias, passando pela a avaliação da solicitação, até sua validação ou reprovação. Percebe-se que o processo é demonstrado por inteiro de forma simples e direta, facilitando a identificação e o entendimento do processo de forma holística por todos.

Na modelagem atual do processo, também chamada de As Is (traduzido inglês para: como é), é realizada uma avaliação através de uma reunião com os integrantes do processo para descobrir os principais pontos de melhorias que podem ser implementadas no processo. Para a coleta dessa avaliação são utilizadas técnicas de análise, troca de opiniões com a *brainstorm* (traduzido do inglês para: tempestade de ideias) e sugestões de melhorias.

Na fase To be, pretende-se criar um ambiente de discussão entre as partes envolvidas de forma a melhorar o processo em questão, inová-lo ou mesmo questionar se ele se faz necessário e se agrega valor à organização BALDAM ET AL (2008). Segundo PIZZA (2012), as modelagens do estado futuro mais comuns são:

- Melhoria contínua
- FAST
- Benchmarking
- Adoção de melhores práticas
- Redesenho do processo

- Inovação de processo

Essa fase tem como objetivo definir o que deve ser feito com os processos identificados durante a modelagem do estado atual (As Is) e alinhá-los com os objetivos da empresa. Se a decisão for redesenhar os processos, é necessário que seja desenvolvido um novo modelo de processos com as melhorias e soluções previstas para a situação atual.

3.3.2.1 Notação BPMN

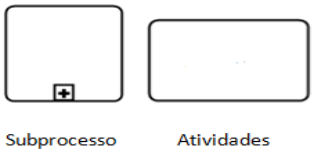
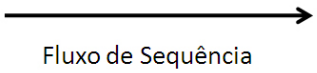
Segundo a OMG (2011), a notação oficial BPMN é constituída de três elementos principais: *Eventos, Atividades e Gateway*. Um evento representado por um círculo que podem ser divididos em três categorias: *Start, Intermediate e End*, e cada um possui uma notação gráfica diferente. Esses eventos afetam diretamente o fluxo dos processos proporcionando um impacto no processo como um todo.

- Início do evento: É onde o fluxo do processo se inicia, sendo obrigatoriamente o primeiro elemento de qualquer processo.
- Evento intermediário**:** Trata-se de um evento que acontece no meio do fluxo dos processos. É um dos eventos que mais possui variações e mais complexo, ao todo são nove tipos diferentes.
- Final do evento: É o último evento do fluxo de qualquer processo. Mais precisamente, o *End event* é onde encerra-se um fluxo em particular ou todo o processo.

> Os eventos são ligados através de objetos de conexão de fluxo e são três: Sequência, Mensagem e Associação. Há também artefatos que são usados para complementar os diagramas, como Dados, Envolvidos e Anotações. Todos esses elementos são agrupados em *Pools e Lanes.* (GONÇALVES, 2016).

Durante o desenvolvimento deste trabalho, os principais elementos da presente notação (BPMN 2.0) serão utilizados para modelar os processos utilizados pela COTIC-PROEG. Esses elementos serão descritos na Tabela 2 abaixo.

Tabela 2: Lista dos elementos que compõem o BPMN 2.0

Elemento	Descrição	Notação gráfica
Evento	Ocorrido durante o processo que causam algum efeito no fluxo e, geralmente, possuem uma causa específica ou impacto no processo. São representados visualmente por um círculo, podendo ser de três tipos: início, intermediário e fim.	
Atividade	São representadas visualmente por retângulos com as bordas arredondadas. É utilizado para representar um serviço que a empresa realiza. As atividades podem ser divididas em dois tipos: subprocessos e tarefas.	
Gateway	São utilizados para controlar as divergências e convergências dos fluxos do processo. O *gateway* é determinante para saber se o fluxo irá se dividir, unir ou ramificar. São representados visualmente por losangos e os marcadores internos que irão determinar qual o tipo daquele *gateway*.	
Fluxo de Sequência	É utilizado para identificar a ordem que o fluxo segue dentro do processo. Visualmente, são representados por uma seta com linha contínua.	
Associação	É utilizado para associar informações a artefatos. A associação é representada visualmente por uma linha pontilhada ligando os elementos gráficos.	
Pool	É uma representação gráfica de um participante na colaboração de um processo, podendo ou não ser detalhada internamente na forma que o processo será executado.	
Objeto de Dados	São representadas visualmente por uma folha de papel com a ponta superior direita dobrada. Os objetos de dados são portadores de informações sobre o que as atividades precisam para serem colocadas em prática e/ou o que essas atividades produzem.	

Fonte: OMG (2011).

Existem outros elementos, uns bem mais complexos e outros nem tanto que não foram mostrados e nem serão utilizados nesta monografia, mas podem ser acessados no site oficial da OMG[2].

[2]https://www.omg.org/

3.4 METODOLOGIA DE DESENVOLVIMENTO DE SOFTWARE

Com a necessidade de evolução contínua e adequação ao mercado competitivo, as empresas mais antigas precisaram adequar sua forma de trabalhar a esse mercado, pois as execuções de seus projetos estavam ficando ultrapassadas e as entregas de seus produtos ficando lentas, comparadas as entregas realizadas pelos seus competidores.

Acompanhada dessa evolução, surgiram as metodologias ágeis de desenvolvimento de software, trazendo técnicas para reduzir os prazos de execução e agilizar as entregas dos projetos sendo flexível a mudanças (SILVA, 2013), com isso, o aumento da demanda e busca por profissionais capacitados para gerenciar projetos com o auxílio de boas práticas e ferramentas para alcançar os objetivos de cada projeto é inevitável.

3.4.1 Metodologia ágil

As metodologias ágeis surgiram com o objetivo de fornecer melhorias contínuas aos softwares, agilizar seu desenvolvimento e agregar mais valor a esses projetos, porém estão sendo utilizados não somente para este segmento, mas em vários tipos de projetos por conta de seus conceitos e princípios, os quais ajudam a esclarecer requisitos mal interpretados, criação de novos produtos, planejamento de orçamento e muitas outras áreas. (RIBEIRO, 2015)

3.4.2 Manifesto ágil

Os métodos ágeis tiveram início com o manifesto ágil, o qual surgiu em fevereiro de 2001, durante uma reunião no estado norte-americano de Utah no resort de inverno e verão Snowbird. Ali marcava-se o surgimento e propagação do paradigma de desenvolvimento de softwares ágeis com a presença de dezessete pessoas, elas dariam início naquele momento mesmo sem saber ao manifesto ágil. (BERNARDO, 2014). Segundo Gomes (2013), estamos descobrindo maneiras melhores de desenvolver softwares, fazendo-o nós mesmos e ajudando outros a fazerem o mesmo. Através deste trabalho, passamos a valorizar:

- **Indivíduos e interações** 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;

Ou seja, mesmo havendo valor nos itens à direita, valorizamos mais os itens à esquerda.

O manifesto ágil também é composto por doze princípios:

1. Nossa maior prioridade é satisfazer o cliente, através da entrega adiantada e contínua de software de valor.

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

O Manifesto Ágil não é contra os modelos adotados pela abordagem tradicional. Ele inclui técnicas utilizadas na Engenharia de Software, porém não segue o padrão proposto pelas metodologias tradicionais (HIGHSMITH, 2002). A seguir serão apresentados os tipos de metodologias ágeis que iremos utilizar neste trabalho.

3.5 SCRUM

O termo teve sua primeira referência em 1986 no artigo *The New New Product Development Game* escrito por Hirotaka Takeuchi e Ikujiro Nonaka ao apresentar conceitos e fundamentos de times ágeis com foco em entregas de valor e melhorias contínuas.

“O Scrum é uma estrutura processual (framework) de suporte ao desenvolvimento e manutenção de produtos complexos. O Scrum é composto por equipes scrum e suas funções, por eventos, artefatos e regras. Cada componente dentro desta estrutura serve um propósito

específico e é essencial ao uso e sucesso do Scrum." (SUTHERLAND e SCHWABER, 2011). A equipe scrum define um período, denominado iteração ou sprint, para que cada projeto possa ser realizado de forma limitada de acordo com o planejamento do time para a iteração que está em andamento.

Atualmente, o Scrum é uma das metodologias ágeis mais utilizadas pelas organizações por sua capacidade de agilizar o processo, "de forma que o cliente (interno ou externo) receba uma versão que agregue valor ao seu negócio" (DANTAS, 2003). Minimizar os riscos de erros e insucessos, e maior produtividade das equipes que fazem parte do projeto também fazem parte das características dessa metodologia.

3.5.1 Os três pilares do Scrum

Segundo SUTHERLAND e SCHWABER (2011), três pilares apoiam a implementação de controle de processo empírico. Esses pilares são:

- Transparência: os processos e suas tarefas devem ser muito bem claras e visíveis aos seus responsáveis para que todos compartilhem do mesmo entendimento.
- Inspeção: tem como objetivo inspecionar os artefatos Scrum, a fim de detectar possíveis variações no desenvolvimento do projeto. É importante que a inspeção seja feita por pessoas qualificadas na área a ser vistoriada para que este pilar seja benéfico.
- Adaptação: caso o time scrum perceba que um aspecto ou mais de um processo divergiu do limite aceitável do planejamento, a ponto de comprometer o projeto ou produto, deve haver um ajuste o mais rápido possível.

3.5.2 O ciclo de vida do Scrum

De acordo com VARASCHIM (2009), o scrum é composto por três fases, as quais são desenvolvidas em iterações (sprints) com duração entre duas e seis semanas.

A primeira fase é a reunião de planejamento (*Sprint Planning*) onde o Time Scrum e o Product Owner definem o que será desenvolvido durante a iteração, sendo responsabilidade do cliente (*Product Owner*) realizar a priorização do trabalho a ser feito.

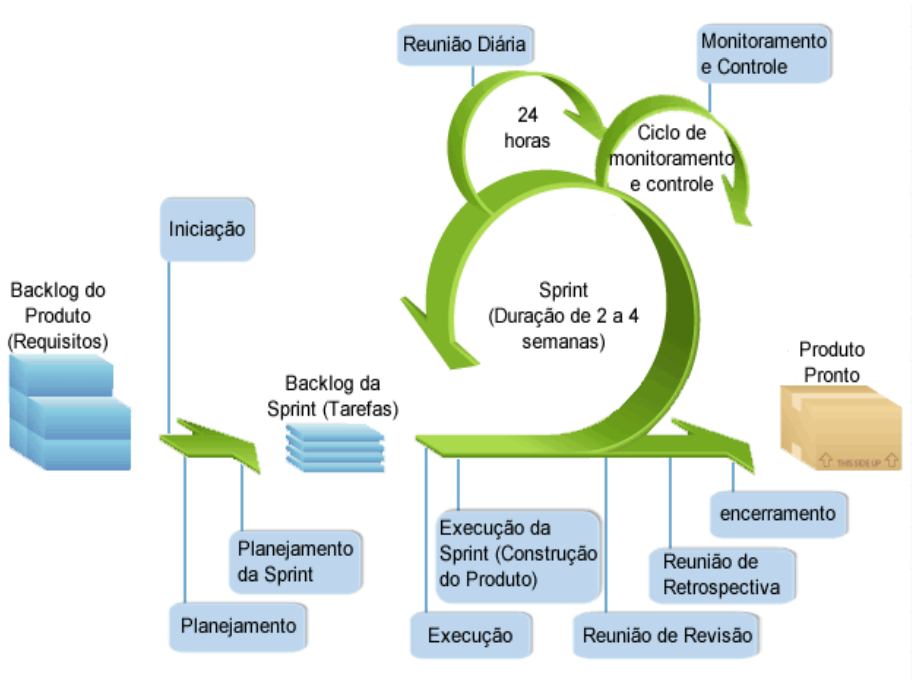
A próxima fase é a do desenvolvimento das tarefas priorizadas pelo cliente. Durante a execução da sprint, o Time Scrum realiza reuniões diárias (*Daily Meeting*) com duração de no máximo 15 minutos para verificar e acompanhar o progresso do desenvolvimento através do gráfico *Burndown Chart*, bastante usado para acompanhar as tarefas a realizar, que estão em andamento e que já foram realizadas.

A terceira e última etapa da sprint começa com uma reunião para a validação da entrega (*Sprint Review*), é onde o *Product Owner*e a quem interessar o produto podem verificar se o objetivo da Sprint foi alcançada. Em seguida, o Time Scrum se reúne para uma realizar uma retrospectiva (*Sprint Retrospective*) para verificar o que houve de positivo e negativo durante a execução da sprint com a finalidade de encontrar formas de melhorar o processo para as próximas iterações.

Na figura a seguir, podemos observar como funciona o desenvolvimento do scrum:

.

Figura 5: Ciclo de desenvolvimento do scrum



Fonte: AELIAN

3.5.3 Artefatos do Scrum

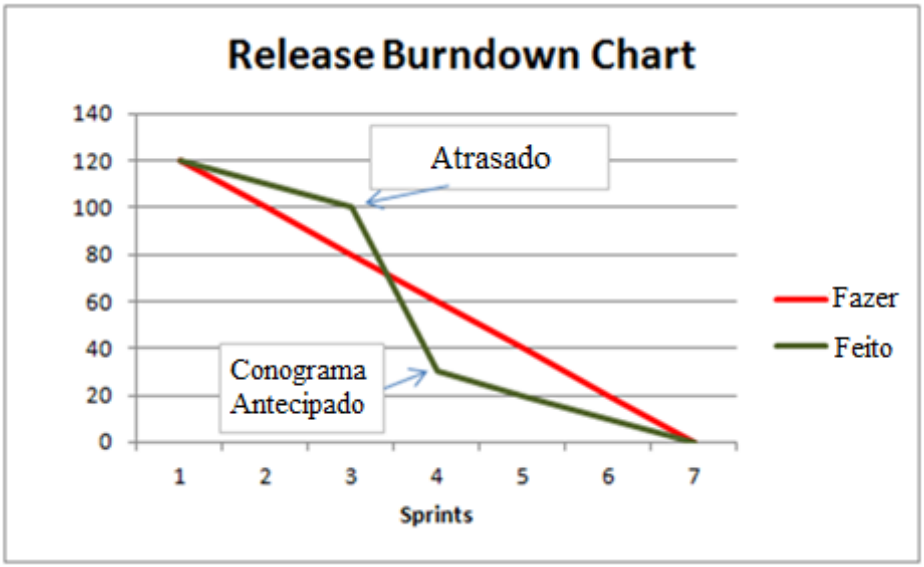
- Product Backlog: de responsabilidade do Product Owner, o Backlog do Produto é onde as listas de funcionalidades, requisitos do projeto e erros encontrados no sistema são armazenadas e organizadas conforme seu grau de prioridade. É importante ressaltar que o *Product Backlog* pode estar sempre em constante evolução, já que, novas funcionalidades e requisitos podem surgir, seja por manutenção no software ou refatoração;
- Planning Poker: ferramenta de estimativa, geralmente, utilizada pelo *Time Scrum* para avaliar a complexidade de cada tarefa. Por isso, é importante que as estimativas de cada tarefa do *Product Backlog* esteja sempre revisada e

atualizada a cada sprint para que o *Planning Poker* possa ser usado de maneira eficaz.

- Sprint Backlog: lista de funcionalidades retiradas do *Product Backlog* durante o *Sprint Planning* que farão parte do *Sprint Backlog* para que sejam desenvolvidas durante a iteração. Caso todas as funcionalidades sejam desenvolvidas antes do término da sprint, o *Time Scrum*, com o consentimento do *Product Owner*, pode adicionar mais estórias do *Product Backlog* ao *Sprint Backlog*;
- Impediment Backlog: lista de impedimentos que podem atrapalhar a entrega do produto, seja na sprint ou no projeto inteiro. Geralmente, o *Impediment Backlog* é composto por tarefas que não puderam ser realizadas por conta de fatores externos. É papel do *Scrum Master* encontrar soluções para que os impedimentos possam ser solucionados.
- Burndown Chart: gráfico que auxilia o *Time Scrum* a monitorar as tarefas diariamente, possibilitando identificar se o projeto está sendo desenvolvido conforme o planejado e estimado no *Sprint Planning*.

A Figura 6 mostra um gráfico onde o eixo horizontal informa o número de dias em que a sprint é realizada; No eixo vertical, o total de trabalho que precisa ser realizado durante a iteração. Com base nos resultados é possível visualizar se o trabalho planejado foi realizado em sua plenitude ou não.

Figura 6: Burndown Chart



Fonte: ADAPTADO DE EFFECTIVE

3.5.4 Papéis e responsabilidades do Scrum

OTime Scrum é composto por três papéis (*Scrum Master, Product Owner e Time de Desenvolvimento*) muito bem definidos. De acordo com o Guide Scrum (2013), os Times Scrum entregam produtos de forma iterativa e incremental, maximizando as oportunidades de realimentação. As responsabilidades dos três papéis que compõem o Time Scrum são:

- Scrum Master: é um facilitador responsável por fazer o Scrum ser entendido e aplicado pelo Time Scrum. Acompanha o desempenho e andamento do projeto diariamente através de reuniões.

  Segundo RIBEIRO (2015), algumas responsabilidades do Scrum Master são:

  - Educar o Time e Stakeholders sobre o processo;
  - Assegurar que a equipe faz o Daily Scrum no horário certo e de modo produtivo;
  - Resolver os impedimentos da melhor maneira possível;
  - Manter o foco das reuniões;
  - Indicar pontos de melhoria no processo e no ferramental;
- Product Owner: é o representante do cliente dentro do Time Scrum e principal responsável por entender de todas a regra de negócio que envolve o projeto, pois é seu papel repassar as informações necessária para o Time Scrum. Segundo RIBEIRO (2015), algumas responsabilidades do Product Owner são:

  - Participar do Daily Scrum;
  - Tirar as dúvidas dos desenvolvedores ou indicar quem poderia respondê-las;
  - Prover uma meta clara para cada Sprint;
  - Obter feedback e expectativas dos clientes e extrair deles as necessidades;
  - Manter o Product Backlog atualizado;
- Time de Desenvolvimento: é o time responsável por desenvolver e entregar uma versão usável do produto a cada final de Sprint.

  Segundo o Guide Scrum (2013), os Times de Desenvolvimento tem as seguintes características:

  - Autogerenciáveis, onde nem mesmo o Scrum Master diz ao Time como fazer o Backlog se tornar um incremento na versão final do produto;
  - Multifuncionais, possuindo todas as habilidades necessárias para criar o incremento da versão final do produto;

3.5.5 Etapas da Sprint

De acordo com KEN SCHWABER, referenciado por VARASCHIM (2009), sprint é o ciclo de desenvolvimento do Scrum, caracterizado por ter um curto período onde a equipe foca no atingimento de uma meta específica proposta pelo Product Owner. Para atingir essas metas durante a iteração, o Time Scrum precisa realizar algumas reuniões que fazem parte da Sprint. Tais etapas são:

- Sprint Planning Meeting: reunião que acontece no primeiro dia de cada iteração com a participação do Product Owner, Scrum Master e o Time Scrum, com o objetivo planejar as metas dessa Sprint e desenvolver o Sprint Backlog.
- Daily Scrum: reunião realizada diariamente com o objetivo de manter a equipe informada sobre o que foi realizado no dia anterior, identificar possíveis impedimentos para o presente dia e priorizar o que deve ser desenvolvido durante o dia. Por se tratar de uma reunião que acontece todos os dias durante a iteração, a Daily deve durar no máximo 15 minutos com a presença do Scrum Master e o Time Scrum.
- Sprint Review: reunião realizada no final de cada Sprint com a presença do Product Owner e o Time Scrum com a finalidade de verificar se o que foi planejado na Sprint Planning Meeting foram satisfeitas.
- Sprint Retrospective: reunião realizada no final de cada Sprint com a presença do Scrum Master e o Time Scrum com o foco no que foi produtivo, negativo e o que pode ser melhorado para as próximas iterações.

3.6 EXTREME PROGRAMMING (XP)

Segundo SATO (2013), o *XP* foi uma das primeiras metodologias ágeis que revolucionou como o software era desenvolvido, e é ideal para pequenas e médias equipes de desenvolvimento de software com requisitos vagos e em constante mudança. Criada por Kent Beck e publicada em seu livro *Extreme Programming Explained* em 1999 onde definiu os valores, princípios e práticas desta metodologia com a finalidade de obter alta qualidade no desenvolvimento de softwares.

3.6.1 Valores do eXtreme Programming

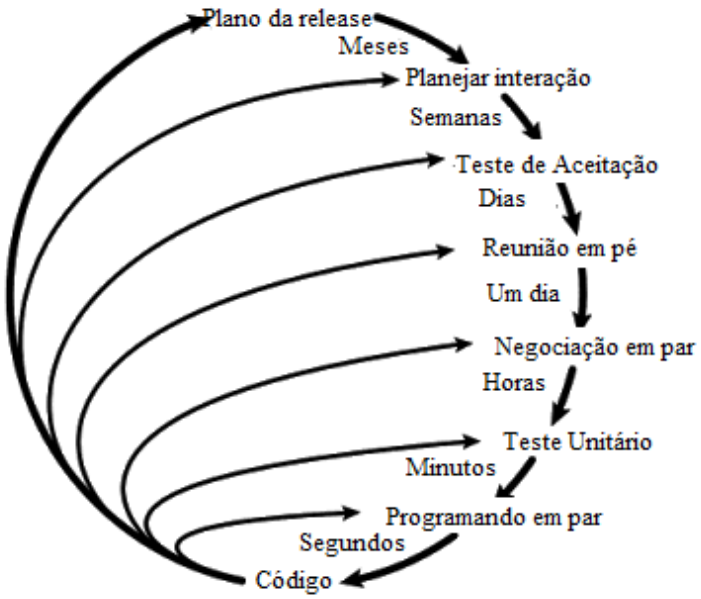
Os valores do XP são conceitos não tangíveis que se acredita fazer uma grande diferença na qualidade final do produto e na motivação dos times (RIBEIRO, 2015). Conforme TELES

(2004), para guiar o desenvolvimento, o *eXtreme Programming* é constituído de cinco valores fundamentais. Esses valores são:

- Comunicação: deve ser tratada de forma Direta, Eficaz e Esclarecedora entre o cliente e a equipe para que os detalhes do projeto sejam tratados com mais atenção e agilidade;
- Feedback: envolve uma relação entre o cliente e a equipe com a finalidade de, principalmente, identificar erros rapidamente e definir prioridades. Porém, o feedback pode ser conseguido através do próprio código do projeto. Segundo RIBEIRO (2015), Para que o feedback do código funcione bem, são necessários testes automatizados de unidade e um servidor de integração contínua para que os testes mais longos sejam rodados com frequência e, se quebrarem, sinalizem uma inconsistência;

Na Figura 7 temos uma visão de como funcionam os ciclos de feedback do XP.

Figura 7: Ciclos de Feedback do XP



Fonte: ADAPTADO DE RIBEIRO (2015)

- Coragem: buscar melhorias e refatoração de código sempre que necessário, mesmo que seja preciso apagar uma funcionalidade inteira para que possa refazê-la de maneira mais eficaz. Além de ter coragem para focar naquilo que realmente é prioritário e de maior importância para o andamento do projeto;

- Simplicidade: apenas desenvolver aquilo que for necessário para atender as necessidades do cliente e não antecipar funcionalidades que não foram solicitadas;

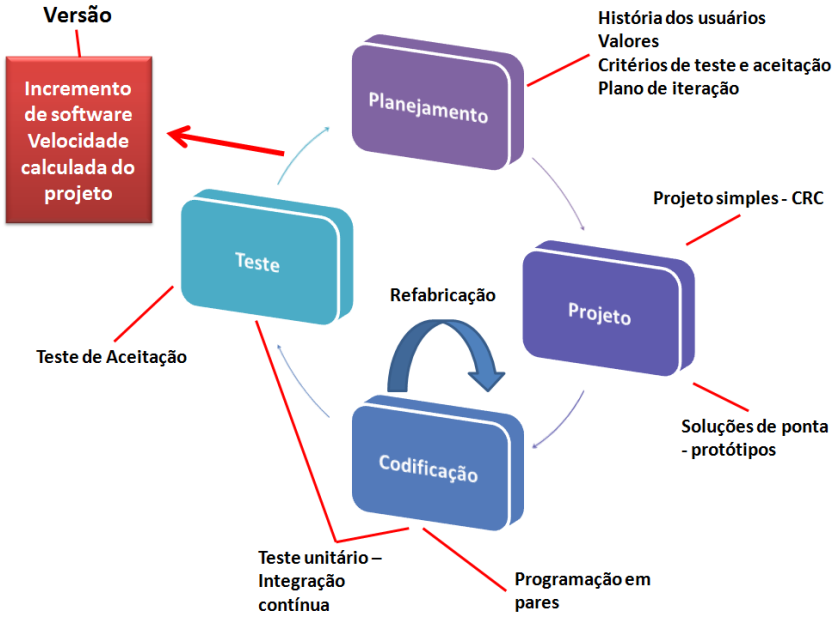
3.6.2 Ciclo do eXtremeProgramming

Por em andamento de forma ordenada as práticas e as estratégias do *eXtreme Programming* consiste em desenvolver o ciclo de vida do XP. Segundo PRESSMAN (2016), as práticas mais importantes do ciclo do XP são:

- Planejamento: momento em que o *Jogo do Planejamento* entra em ação. Inicialmente, há um levantamento de requisitos do software para que haja a criação das *estórias de usuários.* Em seguida, a equipe realiza uma reunião para que as *US* (User Stories) sejam colocadas em sua ordem de prioridade para ser implementada uma versão do produto que traga valor de negócio ao cliente;
- Projeto: o XP encoraja a equipe a usar os cards CRC (Classes-Responsabilidades-Colaboradores), prezando sempre pela simplicidade do projeto (Keep it simple). É realizada a organização dos cards e estórias conforme seu grau de prioridade e, antes que comecem as codificações do software, é criada uma série de testes de unidade e o exercício de cada *estórias* que farão parte do incremento de software para a criação de um protótipo;
- Codificação: o desenvolvimento do código do software é realizado com a *Programação em par*, gerando uma eficácia maior no código, já que há uma revisão instantânea do que está sendo produzido, evitando bugs no produto a ser entregue. Vale ressaltar que há a disseminação de conhecimento entre os membros que estão realizando a codificação por conta da programação pareada, aumentando o fortalecimento da equipe. A refatoração de códigos mal desenvolvidos e que necessitam de uma melhoria também faz parte desta etapa do ciclo;
- Teste: momento em que os testes de unidade, criado na etapa inicial do projeto, são desenvolvidos de forma que comecem a ser automatizados. Diariamente, podem ser realizados os testes de *integração* e *validação,* diferentemente do teste de *aceitação,* que é feita por um cliente e que fazem parte de uma versão do produto;

Na Figura 8 temos uma visão de como funciona de forma ordenada o ciclo do XP.

Figura 8: Ciclo do XP



Fonte: PRESSMAN (2010)

3.6.3 Papéis e Responsabilidades da Equipe eXtreme Programming

Segundo PAMPLONA (2004), em uma equipe do XP existem papéis bem e responsabilidades, os quais serão descritos a seguir.

- Gerente de projetos: responsável pela comunicação direta com o cliente por ser encarregado da parte da administrativa do projeto;
- Coach: responsável pela parte técnica do projeto. É de grande importância que haja um conhecimento elevado do processo de desenvolvimento, dos valores e práticas do XP, a ponto de orientar a equipe de forma correta caso haja erros cometidos, podendo assumir o papel de desenvolvedor também;
- Analista de testes: responsável pela qualidade do software através dos testes que foram escritos durante a etapa de projeto do ciclo do XP. Ajuda o cliente a escrever os casos de testes e, ao final de cada iteração, deve verificar se o que foi desenvolvido atende a todos os casos de testes;
- Redator técnico: responsável por toda a documentação do sistema, tirando essa responsabilidade da equipe de desenvolvimento, proporcionando que foquem somente na implementação dos códigos. Essa pessoa deve estar em plena sintonia com a equipe de desenvolvimento e o cliente, para que toda a documentação esteja clara e que atenda a todas as funcionalidades do produto;

- Desenvolvedor: responsável em analisar, projetar e codificar o software. Vale ressaltar que é natural que tenha desenvolvedores de diversos níveis de desenvolvedores em uma equipe, porém, com a *programação em par* a tendência é que, com o tempo, haja um equilíbrio técnico entre eles;

3.6.4 Práticas do eXtreme Programming

De acordo com TELES (2004), as práticas do XP podem ser descritas das seguintes formas:

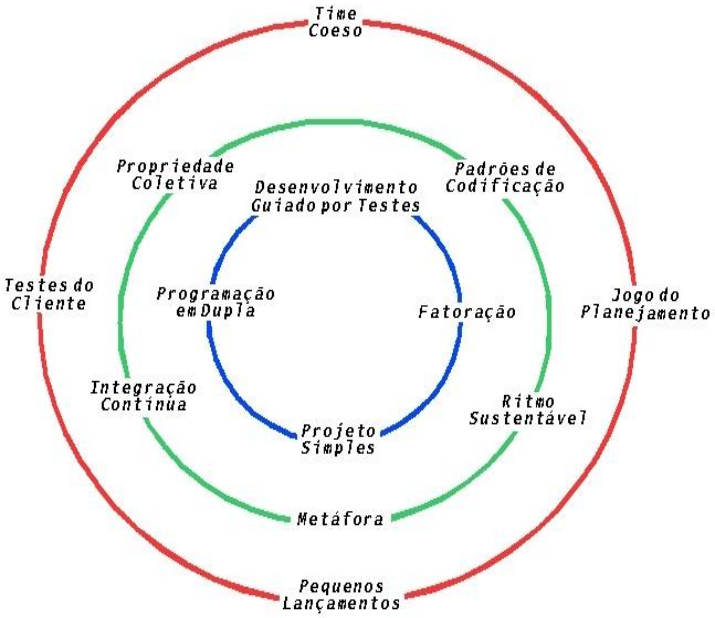
- Cliente Presente: o cliente deve conduzir o desenvolvimento do produto, participando ativamente do projeto junto à equipe de desenvolvimento. Sua presença é de suma importância para facilitar a comunicação entre os membros do projeto;
- Jogo de Planejamento: reunião onde os membros da equipe e o cliente visam o que há de mais importante para ser entregue. Em seguida, o time de desenvolvimento estima o esforço, tempo e valores necessários para desenvolver aquilo que foi priorizado;
- Releases Curtos: entrega de uma versão simples e funcional, onde o sistema desenvolvido tenha um conjunto de funcionalidades capazes de agregar algum valor para que o cliente possa começar a utilizar o sistema desenvolvido. Dessa forma, é possível realizar entregas contínuas, realizar edições e melhorias no software em tempo hábil;
- Ritmo Sustentável: o XP recomenda que a carga horária diária de trabalho de cada desenvolvedor seja de, no máximo, oito horas por dia, para que haja tempo o suficiente de descanso do trabalhador, para que, no dia seguinte, possa realizar seu trabalho de maneira produtiva.
- Metáfora: criação de um vocabulário comum para que todos os envolvidos do projeto possam compreender uma ideia complexa de forma simples. Assim, os elementos básicos do projeto serão entendidos por todos com mais clareza;
- Projeto Simples: o projeto deve ser projetado e desenvolvido da maneira mais simples possível, de maneira que possa atender as necessidades de cada funcionalidade;
- Desenvolvimento Guiado Por Testes: uma prática de validação para a construção de softwares de qualidade. Prática desenvolvida antes da codificação com a

criação de cenários de testes, também conhecida como TDD (Test Driven Development), onde é abordada com mais detalhes na seção 3.8;

- Refatoração: é o ato de modificar códigos com a finalidade de melhorá-lo sem afetar a funcionalidade do sistema, permitindo otimizá-lo;
- Propriedade Coletiva do Código: todos que fazem parte do time de desenvolvimento são responsáveis por todo o código, possibilitando uma liberdade para qualquer integrante do time realizar modificações em qualquer parte do sistema a qualquer momento;
- Programação Pareada (Pair Programming): dois programadores realizam a codificação em apenas um computador, de forma que, enquanto um codifica, outro faz a revisão do que está sendo desenvolvido, permitindo uma implementação mais eficaz e evitando bugs em no sistema;
- Integração Contínua: integração constante e diariamente do código a cada momento que for modificado ou um novo seja desenvolvido. Essa prática tem como objetivo evitar quebra do sistema com um código novo.
- Código Padronizado: o time deve possuir um padrão de código fonte para que todos os integrantes possam estar familiarizados com os códigos desenvolvidos, possibilitando que todos possam editar ou realizar alterações a qualquer momento no sistema;

A Figura 9 abaixo permite uma visualização do agrupamento das práticas do XP, onde as práticas do alinhamento exterior são relacionadas com os requisitos do sistema, na linha intermediária, são práticas relacionadas à qualidade do software e, na linha mais internas, são práticas relacionadas à codificação do produto.

Figura 9: Agrupamento das práticas do XP



Fonte: ADAPTADO DE AGILITY LADDER

3.7 TESTE DE SOFTWARE

Teste de software é um processo de execução de um programa com a intenção de encontrar erros e buscando proporcionar assim uma melhor qualidade ao software (FERREIRA, 2011). Segundo PRESSMAN (2006), testar o software é a forma mais precisa que temos para encontrar erros. Ao testar um software podemos encontrar as grandes diferenças entre os resultados que obtivemos e os que são esperados. Portanto, é uma fonte importante de feedback.

Os testes de software tornaram-se indispensáveis para detectar os defeitos que escapam das revisões e para avaliar o grau de qualidade de um produto juntamente com seus componentes.

De acordo com FERREIRA (2011), os principais objetivos dos testes são:

- Foco na prevenção de erros;
- Descobrir sintomas causados por erros;
- Fornecer diagnósticos para que os erros sejam facilmente corrigidos;
- Mostrar que o software tem erro;
- Segurança;

Este capítulo está destinado para a apresentação dos principais tipos de testes de softwares e os métodos de teste.

3.7.1 Métodos de Teste

Segundo FERREIRA (2011), atualmente, existe dois tipos de métodos de testes de software, os quais serão descritos abaixo:

3.7.1.1 Teste de Caixa-preta

O *Teste de Caixa-preta* é um método focado diretamente nas entradas e saídas dos sistemas, pouco se preocupando com forma estrutural. O foco nos resultados e se ele entrega as saídas corretas é o ponto forte desse método, porém, isso pode ser considerado um ponto negativo também, pois a forma que o software foi implantado é desconsiderado, o que proporciona não ser analisado se o sistema está se portando de forma mais eficiente conforme as boas práticas de programação.

Segundo PRESSMAN (2006), o *Teste de Caixa-preta* refere-se a testes que são conduzidos na interface do software. Um *Teste de Caixa-preta* examina um aspecto fundamental do sistema, pouco se preocupando com a estrutura lógica interna do software.

3.7.1.2 Teste de Caixa-branca

Os *Testes de Caixa-branca* são testes destinados aos códigos de um software, com a finalidade de avaliar os possíveis cases lógicos que podem ser executados pelo sistema.

Um grande exemplo desse método de teste é o *Teste Unitário,* já que esse tipo de teste é capaz de verificar uma parte específica do código, validando os dados, sejam eles de entrada ou saída, gerados a partir daquele pedaço de código que está sendo testado.

Segundo PRESSMAN (2006), o *Teste de Caixa-branca* de software é baseado em um exame rigoroso do detalhe procedimental. Caminho lógicos internos ao software e colaborações entre componentes são testados, definindo-se casos de teste que exercitam conjunto de condições e/ou ciclos.

3.7.2 Principais Tipos de Teste de Software

3.7.2.1 Testes Unitários

Teste utilizado na prática do *TDD* (Test Driven Development), o *Teste Unitário ou Teste de Unidade* avalia uma pequena parte de um sistema, um componente ou uma classe, podendo ser escritos e desenvolvidos pelos analistas de testes e pelos próprios desenvolvedores dos códigos. O *Teste Unitário* tem a finalidade de analisar e validar as entradas e saídas geradas pelo sistema.

3.7.2.2 Testes de Aceitação

Teste realizado no sistema antes de sua implantação, e é feito pelo cliente ou por uma pessoa que não fez parte do desenvolvimento do produto, com a finalidade de avaliar se as entradas e saídas geradas pelo sistema estão de acordo com os requisitos.

3.7.2.3 Testes de Integração

Teste realizado para avaliar a integração entre os componentes desenvolvidos, principalmente se o sistema for desenvolvido em partes, já que são necessárias várias integrações. Esse teste é fundamental para empresas que utilizam a metodologia ágil, mais precisamente o e*Xtreme Programming,* já que essa metodologia trabalha com integração contínua, possibilitando até mesmo várias integrações durante um mesmo dia.

3.7.2.4 Testes de Regressão

Teste realizado após a integração de novos componentes no sistema. Tem a finalidade de verificar se as outras partes do sistema não foram afetadas com a integração da nova funcionalidade.

> "Cada vez que um novo módulo é adicionado como parte do teste de integração, o software se modifica. Novos caminhos de fluxo de dados são estabelecidos, novas E/S pode ocorrer e nova lógica de controle é adicionada. Essas modificações podem causar problemas com funções que previamente funcionavam impecavelmente." (PRESSMAN, 2006, p. 300).

3.8 DESENVOLVIMENTO GUIADO POR TESTE

O *Test Driven* Development (TDD) "é uma metodologia ágil de desenvolvimento de software que surgiu da derivação do *eXtreme Programming"* (BECK, 2000). Essa metodologia é usada na fase de implementação do software, onde a equipe de desenvolvimento usam casos de testes de um determinado objeto para dar início e seguimento no projeto, seguida de um conjunto de práticas e valores, a fim de agregar valor ao software e um bom retorno ao cliente.

Segundo FERREIRA (2011), os desenvolvedores usam testes para guiar o projeto do sistema durante o desenvolvimento. Eles automatizam os testes para que sejam executados repetidamente. Através dos resultados (falhas ou sucessos) julgam o progresso do desenvolvimento. Os programadores fazem continuamente pequenas decisões aumentando as funcionalidades do software a uma taxa relativamente constante. Todos estes casos de teste

devem ser realizados com sucesso antes do novo código ser considerado totalmente implementado.

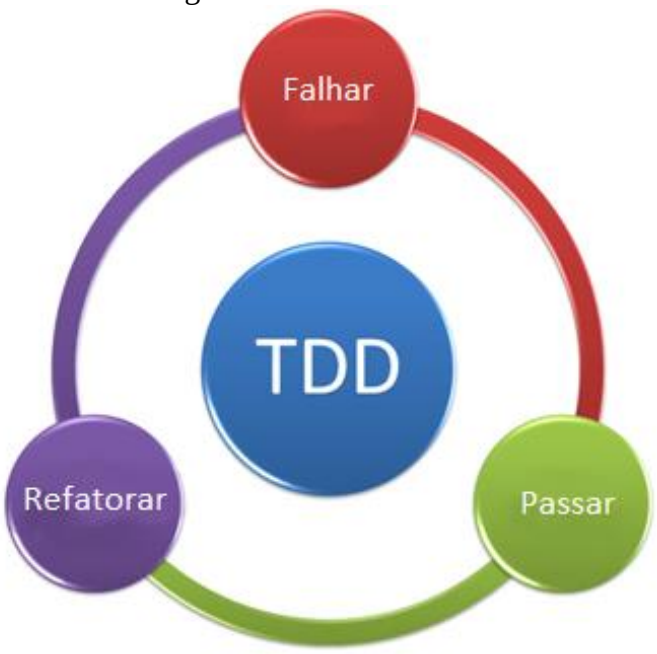
De acordo com BECK (2000), o *TDD* possui um conjunto de iterações realizadas para poder completar uma tarefa. Essas iterações são compostas pelos seguintes passos:

- Escolher a área ou requisito para guiar o desenvolvimento do projeto;
- Projetar o teste de maneira mais simples o possível para que o resultado obtido seja o desejado;
- Refatorar o código para satisfazer o teste desenvolvido e os novos possíveis;
- Refatorar o sistema para eliminar redundâncias (se houver), sem quebrar os testes.

3.8.1 Ciclo do TDD

De acordo com BECK (2010), o Test Driven Development segue um ciclo de três regras básicas para sua implementação, os quais estão ilustrados na Figura 10:

Figura 10: Ciclo do TDD



Fonte: ADAPATADA DE CODE4CODERS

- Vermelho (Falhar): Escrever um teste que, a princípio, falhe. Não se faz necessário a compilação deste código.
- Verde (Passar): Escrever um código que o teste passe de qualquer forma, mesmo que isso implique em não utilizar as melhores soluções. O importante é fazer o código passar.

- Refatorar: Este é o momento em que o desenvolvedor deve se preocupar em eliminar as redundâncias no código, simplificá-lo e, até mesmo, reescrevê-lo com as melhores soluções.

> O TDD é um processo iterativo, portanto estes três passos anteriormente citados, devem ser seguidos até que o desenvolvedor esteja satisfeito com o novo código gerado. A forma que o TDD trabalha é decompondo os requisitos do sistema em um conjunto de comportamentos necessários para cumprir estes requisitos, sendo que para comportamento do sistema a primeira coisa a ser feita é escrever um teste de unidade para testá-lo. (SANTIN, Carlos Eduardo. Desenvolvimento guiado por testes e Ferramentas XUnit.)

3.8.2 Lista de Testes

"Uma boa prática do TDD é iniciar construindo uma lista de testes, essa lista vai sendo modificada incrementalmente com novos testes, conforme as necessidades forem aparecendo", SILVA (2012).

De acordo com BECK (2000), antes de iniciar a codificação, deve ser feita uma lista com todos os testes que o desenvolvedor acredita que será necessário escrever. Assim, quando o desenvolvedor começar a codificar, ele terá um guia do que é necessário fazer, e não corre o risco de esquecer algo.

3.8.3 Assertivas

São as formas utilizadas para a realização dos testes ao passar as entradas, saídas esperadas e os métodos para testar os resultados. Frameworks como o PHPUnit possui uma vasta lista de assertivas que podem ser utilizadas pelo desenvolvedor quando for criar os testes.

4. ESTUDO DE CASO: MODELAGEM DO PROCESSO DE SOFTWARE DA COTIC

Este capítulo tem como objetivo documentar os passos necessários para a modelagem do processo da Coordenadoria de Tecnologia da Informação e Comunicação, desde a sua identificação até a sua implantação. A ferramenta utilizada para realizar a modelagem na linguagem BPMN 2.0 foi o Draw[3].

4.1 ETAPAS DO ESTUDO REALIZADO NA COTIC

4.1.1 Identificação e mapeamento dos processos

Com a finalidade de identificar os processos da organização, foram realizadas entrevistas com os membros da organização, principalmente, com o servidor público responsável pela empresa. Durante a pesquisa exploratória, foi disponibilizado um trabalho realizado anteriormente com a finalidade de melhorar o processo da empresa utilizando a linguagem SPIDER-ML que tem como autora Jaily Vanzeler Freitas. Estes dados foram separados para serem analisados e utilizados na próxima etapa.

4.1.2 Levantamento dos dados detalhados para a modelagem

Como dito na etapa 4.1.1, atualmente, a COTIC possui um documento que lista os processos utilizado pela organização. A partir deste documento, foi realizado um estudo dos processos de negócio, utilizando os requisitos necessários para a modelagem dos processos, estes requisitos são: complexidade, impacto, número de atividades e participantes no processo.

Ao todo, foram contabilizados 4 processos, entre processos e subprocessos (Processo Principal, Planejar Release, Executar Sprint e Realizar Desenvolvimento), com cerca de 21 atividades e 4 grupos de envolvidos (Product Owner, Scrum master, Time de desenvolvimento, Time Scrum). Com o levantamento dos dados de todos os processos, a próxima etapa é modelá-lo de acordo com a notação escolhida.

[3]https://www.draw.io/

4.1.3 Modelagem do processo atual (As Is) utilizado na COTIC

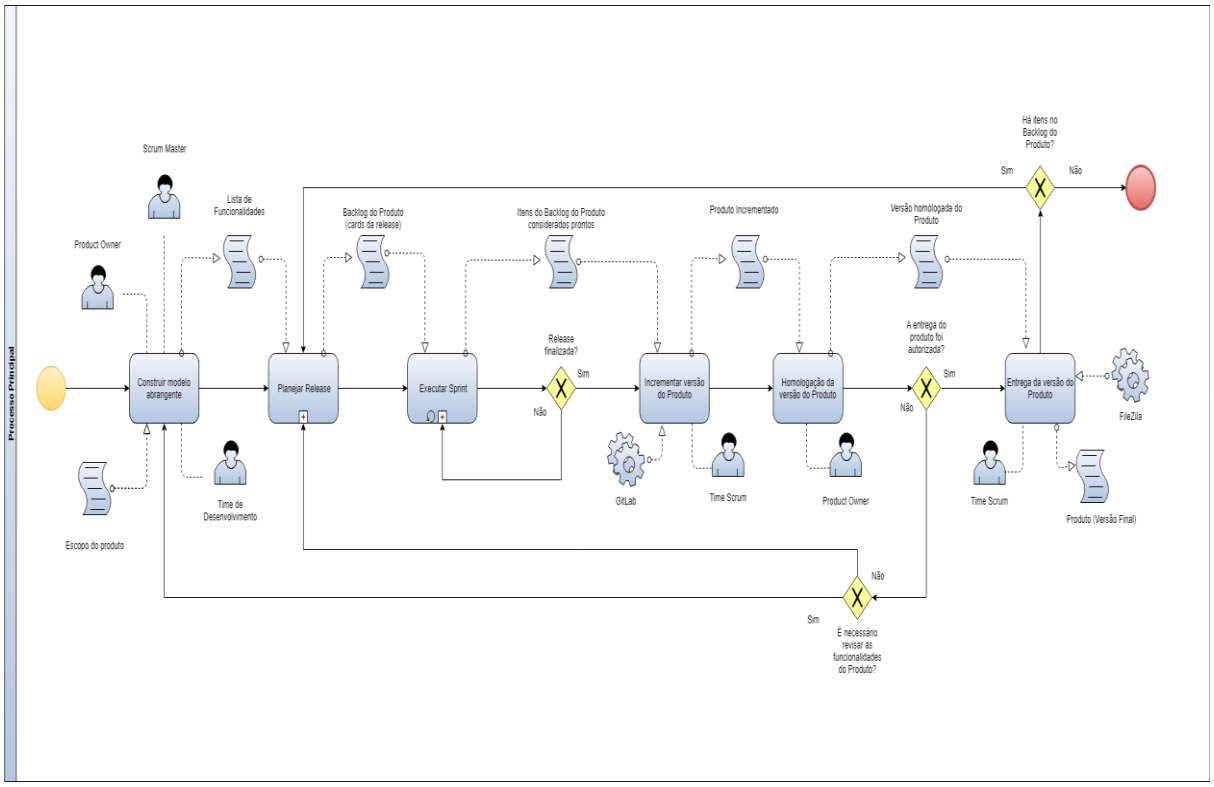
Após a realização do levantamento dos dados feito na etapa 4.1.2, foi realizada a modelagem dos processos e subprocessos utilizado pela COTIC. A seguir, esses processos serão descritos detalhadamente.

- Processo principal:

De acordo com o levantamento dos dados mostrados na etapa anterior, foram identificados 4 processos, e o primeiro deles foi chamado de Processo Principal, tendo como base o modelo clássico do Scrum, sendo modificado e adaptado para abranger as práticas das outras metodologias utilizadas pela organização.

Essas práticas são originadas das seguintes metodologias ágeis de desenvolvimento de software: *Feature Driven Development* (Construir Modelo abrangente), *eXtreme Programming* (Planejar Release e Realizar Desenvolvimento) *e Scrum* (Executar Sprint). A Figura 11 mostra com detalhes como esse processo funciona.

Figura 11: Processo Principal



Fonte: Autor (2018).

O processo inicia-se com o artefato de entrada que possui o *Escopo do Produto*. Uma vez montado o escopo inicial do produto é necessário organizar as ideias que surgiram a partir deste

artefato, e a partir dele é realizada a primeira atividade do fluxo, denominado de ***Construir Modelo Abrangente***.

Nesta etapa, é feita uma reunião entre o ***Product Owner, Scrum Master e Time de desenvolvimento*** para desenvolver um protótipo daquilo que pretendem construir. Com a finalização desta etapa, o ***Scrum Master e o Time de desenvolvimento*** realizam um estudo de negócio para verificar se o produto a ser desenvolvido está de forma clara, caso contrário, estuda-se novamente sobre o domínio de negócio até que seja entendido por todos os membros que irão fazer parte do desenvolvimento do produto.

Então, o ***Time de desenvolvimento*** começa a produzir a modelagem do produto, originando os Diagramas de Classe inicial, o qual deve passar pelas seguintes validações com todos os integrantes do desenvolvimento: Apresentação dos Modelos elaborados. Se houver um consenso, segue-se a construção do modelo abrangente, caso contrário, refatora-se os Modelos.

Após a validação dos Modelos, cria-se o protótipo de tela e o *BPM* do produto, e são repassados ao ***Product Owner*** para serem inspecionados e validados. Caso não estejam de acordo com o que foi solicitado, retorna-se para a composição dos modelos sobre o domínio de negócio, mas se estiver de acordo, o ***Time de desenvolvimento*** elabora uma ***Lista de Funcionalidades*** que será o artefato de saída da tarefa de construção do modelo abrangente e passa para o subprocesso de planejamento da release.

- Planejar Release:

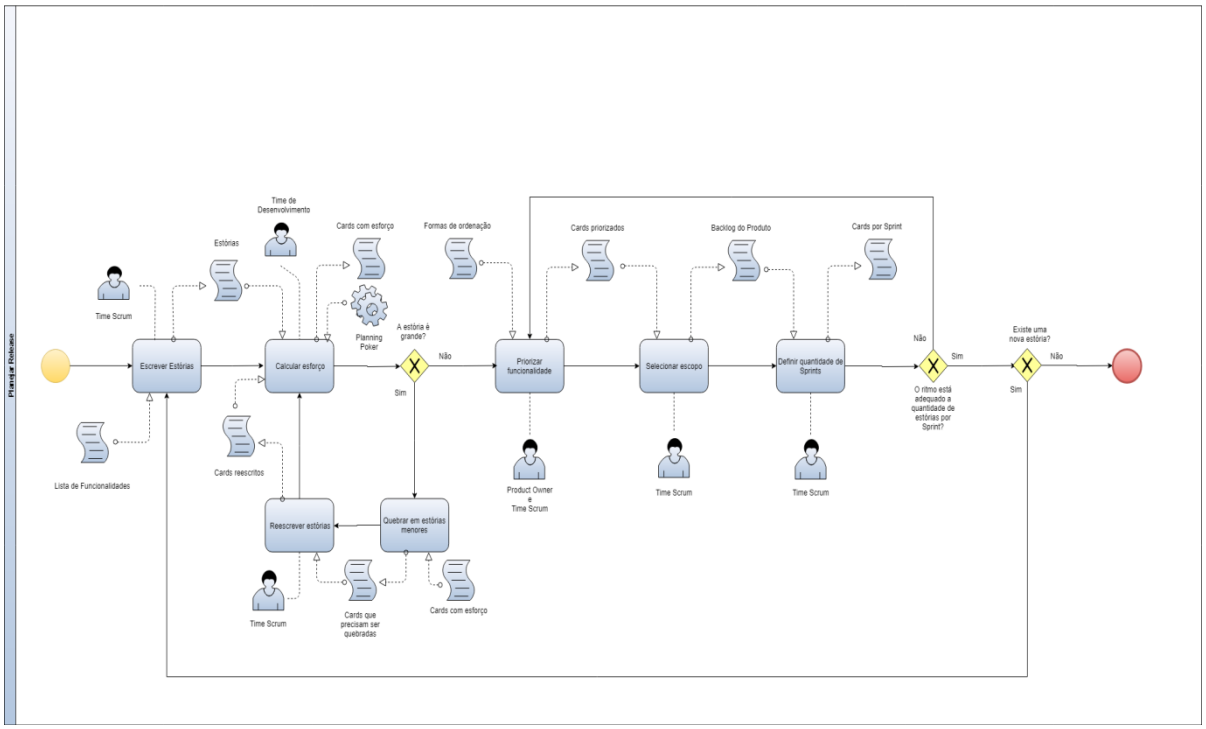
Tem como objetivo criar um planejamento da release, permitindo que o ***Time Scrum*** tenha uma visão geral desse cronograma e entrega do produto, para que possam ajustar-se de acordo com os pedidos do cliente.

O subprocesso de planejamento da release, mostrado na Figura 12, inicia-se com a atividade ***Escrever Estórias***, tendo como entrada a ***Lista de Funcionalidades***, gerada na etapa anterior, e tem como saída as *Estórias* de Usuários. Em seguida, é feito o cálculo do esforço que aquela estória exigirá do ***Time de desenvolvimento***, o que originará os ***Cards*** *com* ***esforço***. Caso as estórias forem grandes demais para serem desenvolvidas, é feito a quebra em estórias menores e em seguida vai para o processo de Reescrever Estórias, então volta ao processo de Calcular Esforço. Então, o fluxo segue para a priorização das funcionalidades de acordo com a importância de cada uma delas, este grau de prioridade é denominado em reunião com o ***Product Owner*** *e* ***o Time Scrum***, tendo como entrada as *Formas de ordenação* e saída os ***Cards priorizados***.

O processo segue para a seleção do escopo, o qual é de inteira responsabilidade do ***Time Scrum***, que tem como entrada os ***Cards priorizados*** e saída *o* ***Backlog do produto***, que por sua

vez é a entrada da próxima atividade, ***Definir quantidade de Sprints***. Este último processo é de responsabilidade do ***Time Scrum*** e tem como saída os ***Cards por Sprint***. Em seguida, o processo conta com uma condicional: "O ritmo está adequado à quantidade de estórias da Sprint?". Caso não esteja, ele volta para a atividade ***Priorizar Funcionalidade*** e segue o fluxo a partir deste processo, caso esteja, entra em uma outra condicional com a seguinte pergunta: Existe uma nova estória? Caso não exista, o fluxo é encerrado, caso exista, o fluxo retorna para a atividade ***Escrever Estórias***.

Figura 12: Planejar Release

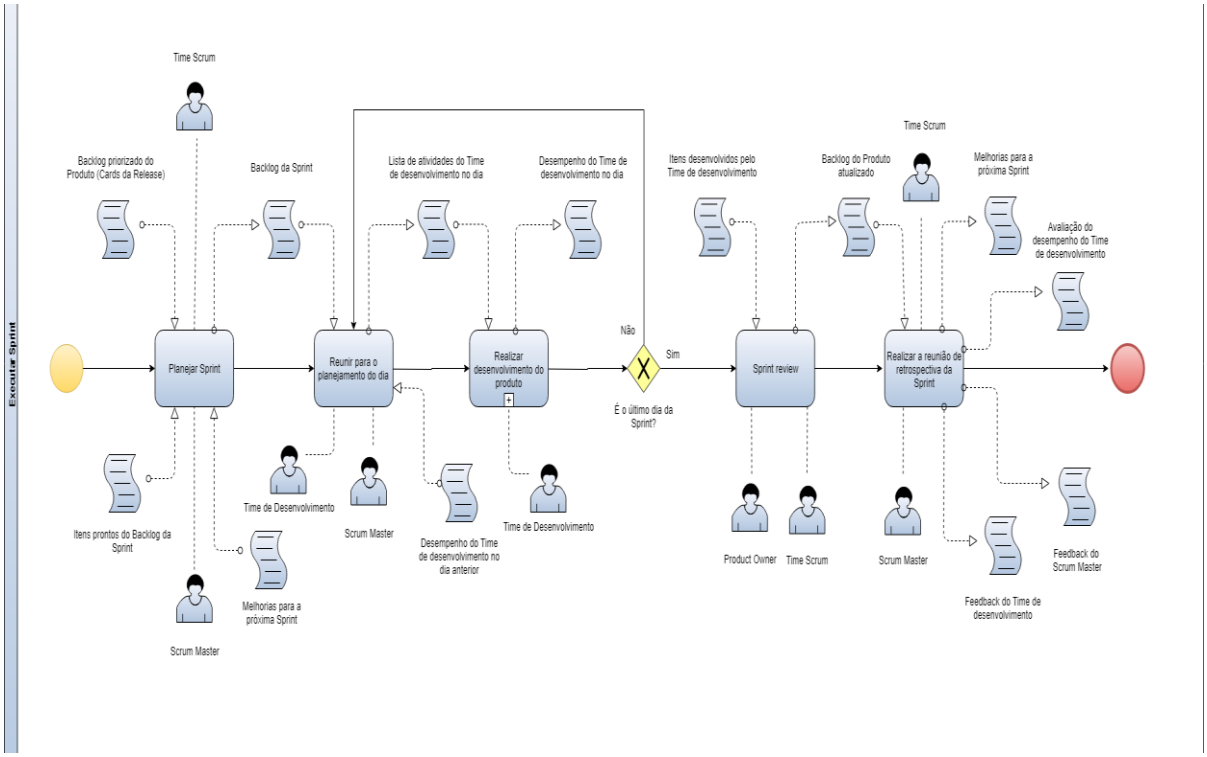


Fonte: Autor (2018).

- Executar Sprint:

O subprocesso ***Executar Sprint***, ilustrado na Figura 13, começa com o artefato ***Backlog priorizado do produto***, originado na etapa anterior, o qual servirá para que o ***Time Scrum*** possa definir o tempo de cada iteração e quais são os itens do Backlog farão parte desta sprint, originalizando o ***Backlog da Sprint***.

Figura 13: Executar Sprint



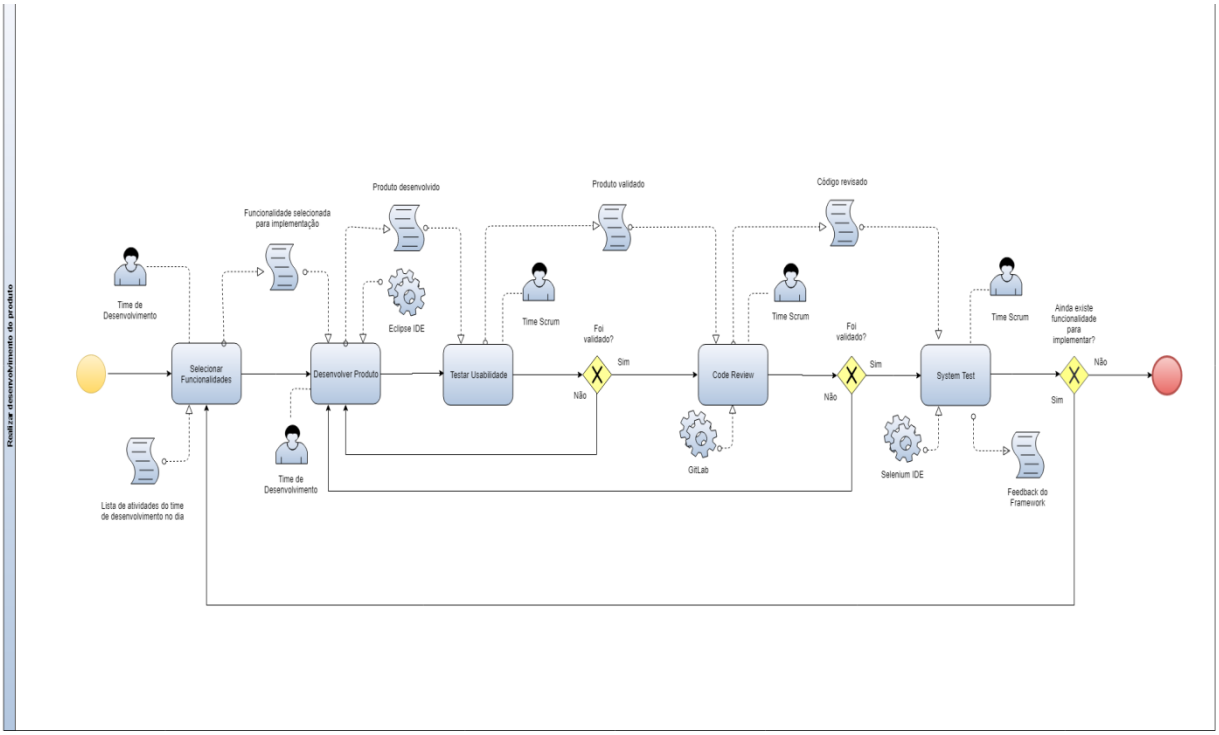
Fonte: Autor (2018).

Depois de realizar o planejamento da sprint, começa-se o serviço a partir da reunião de planejamento diário, a qual tem como facilitador o ***Scrum Master*** e com a presença do ***Time de desenvolvimento***. O principal objetivo dessa atividade é responder três perguntas básicas: “O que foi feito no dia anterior?”, “O que farei hoje?” e “Quais são os possíveis impedimentos para a realização da tarefa?”. Na ocasião, os planejamentos diários são feitos para atender apenas um turno de serviço, até o início do próximo turno, onde será realizado um novo planejamento para as próximas 4 horas, baseando-se sempre no turno anterior. A partir disso, é gerada a ***Lista de atividades do Time de desenvolvimento*** para aquele turno. Então, inicia-se o subprocesso ***Realizar desenvolvimento do produto***.

- Realizar desenvolvimento do produto:

O subprocesso ***Realizar desenvolvimento do Produto***, ilustrado na Figura 14, começa com o artefato ***Lista de funcionalidades do time de desenvolvimento no dia***, originado na reunião diária da Sprint, o qual servirá para que o ***Time de desenvolvimento*** possa selecionar as funcionalidades que serão desenvolvidas no dia. Após a seleção das funcionalidades, a equipe parte para a atividade de ***desenvolvimento do produto*** fazendo uso da ferramenta ***Eclipse IDE***, tendo como resultado o ***Produto desenvolvido***.

Figura 14: Realizar desenvolvimento do produto (As Is)



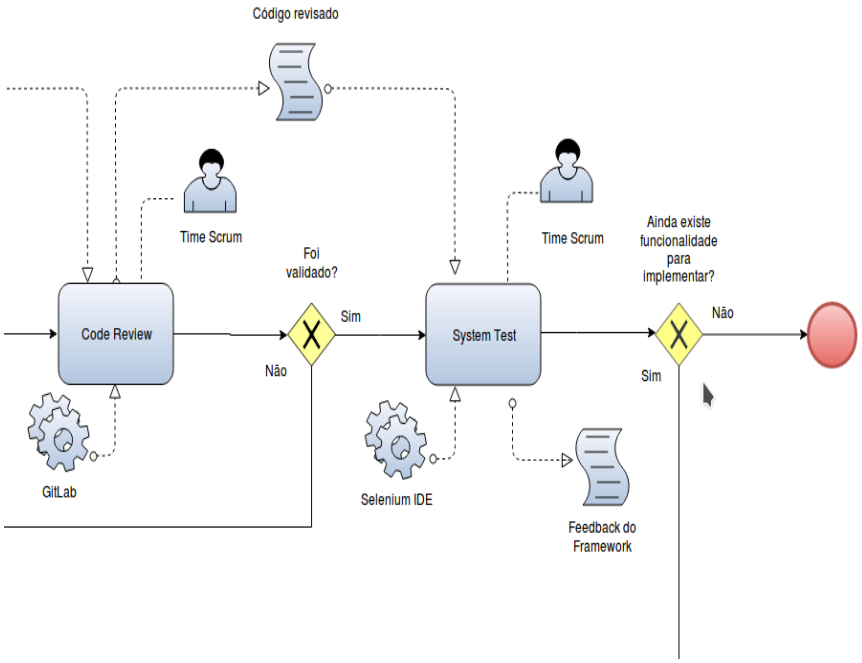
Fonte: Autor (2018).

O processo continua com a atividade ***Testar usabilidade***, realizada por membros do ***Time Scrum*** que participaram ou não do ***desenvolvimento do produto***. O ***teste de usabilidade*** tem o objetivo de verificar a funcionalidade desenvolvida pelo ***Time de desenvolvimento***, e então, é feito uma validação pelos membros que estão realizando o teste, caso a funcionalidade seja validada de forma positiva, o fluxo prossegue para a ***revisão do código (Code Review)***, caso contrário, o processo retorna para o ***desenvolvimento do produto***, até que a funcionalidade desenvolvida esteja de acordo com o solicitado e tenha uma boa usabilidade.

A próxima etapa, o ***Code Review***, tem como principal objetivo verificar se o código fonte e a arquitetura utilizada pelo ***Time de desenvolvimento***, na etapa de ***desenvolvimento do produto***, está de acordo com os padrões de codificação utilizado pela COTIC em seus sistemas. Esta etapa é executada por integrantes do ***Time Scrum*** que não fizeram parte do ***desenvolvimento do produto***, uma vez que serão revisados os códigos produzidos e precisam ser analisados por uma perspectiva diferente para que esta etapa seja realizada com maior efetividade. Após realizar esta revisão, os membros que estão encarregados pela tarefa irão validar o código. Caso a validação seja positiva, o fluxo segue para o ***teste do sistema (System Test)***, caso contrário, o processo retorna para o ***desenvolvimento do produto***, até que o código produzido esteja de acordo com os padrões da organização.

Em seguida é realizado o ***teste do sistema (System Test)*** com auxílio da ferramenta de teste automático ***Selenium IDE***, feito pelo ***Time Scrum***, podendo participar integrantes que fizeram parte do ***desenvolvimento do produto***, ***teste de usabilidade*** *ou* ***revisão do código***. Esta etapa tem o objetivo verificar se a integração da nova funcionalidade desenvolvida não trouxe problemas de execução do sistema nas funcionalidades que já fazem parte de releases anteriores. Após o feedback do framework (***Selenium IDE***), o fluxo segue para o questionamento: "Ainda existe funcionalidade para implementar?". Caso haja funcionalidade esperando para ser desenvolvida, o processo retorna para a primeira atividade (***Selecionar funcionalidade***) do subprocesso. Caso contrário, o subprocesso é finalizado como está exemplificado na Figura 15.

Figura 15: Fim do subprocesso Realizar desenvolvimento do produto (As Is).



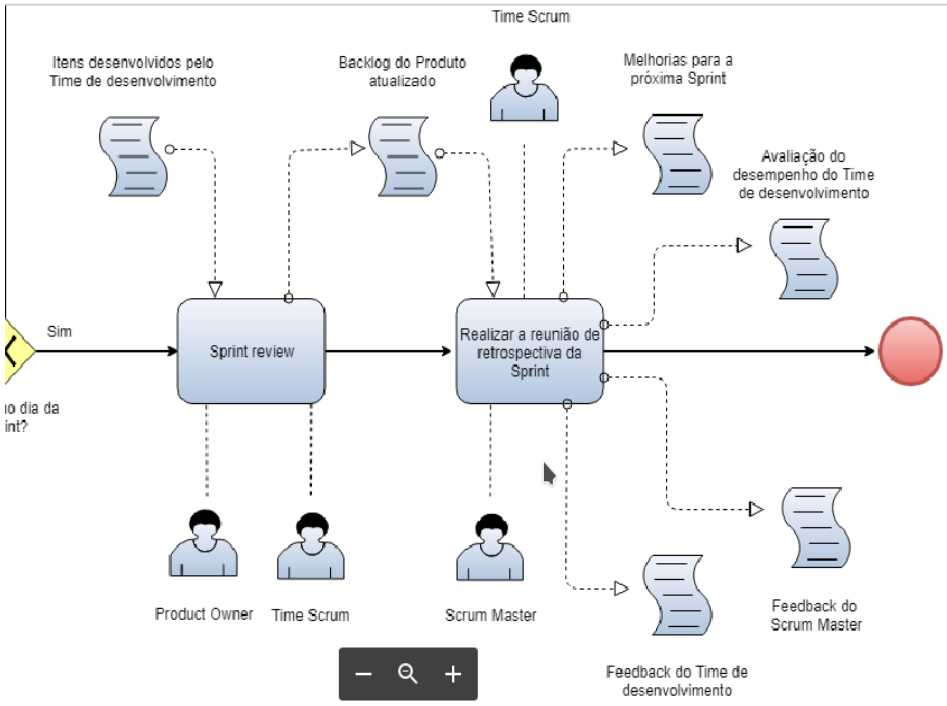
Fonte: Autor (2018).

Com a finalização da etapa dedesenvolvimento do produto, o fluxo retorna à reunião diária até que a *sprint* chegue em seu último dia. Então, o ***Time Scrum*** reúne-se com o ***Product Owner*** para a revisão da sprint (**Sprint Review**), a fim de que verifique-se quais são os itens desenvolvidos pelo Time de desenvolvimento durante a sprint, haja uma analisa o que foi completo do ***Backlog do Produto*** e o atualize.

Em seguida, é realizada a retrospectiva da ***sprint***, onde o ***Scrum Master*** e o ***Time Scrum*** se reune para discutirem a respeito do desempenho do time ao longo da iteração, ouvir o

feedback do ***Scrum Master*** e propor melhorias para a próxima ***sprint***, e então, a ***sprint*** é encerrada, como mostra a Figura 16.

Figura 16: Fim do subprocesso Executar Sprint.

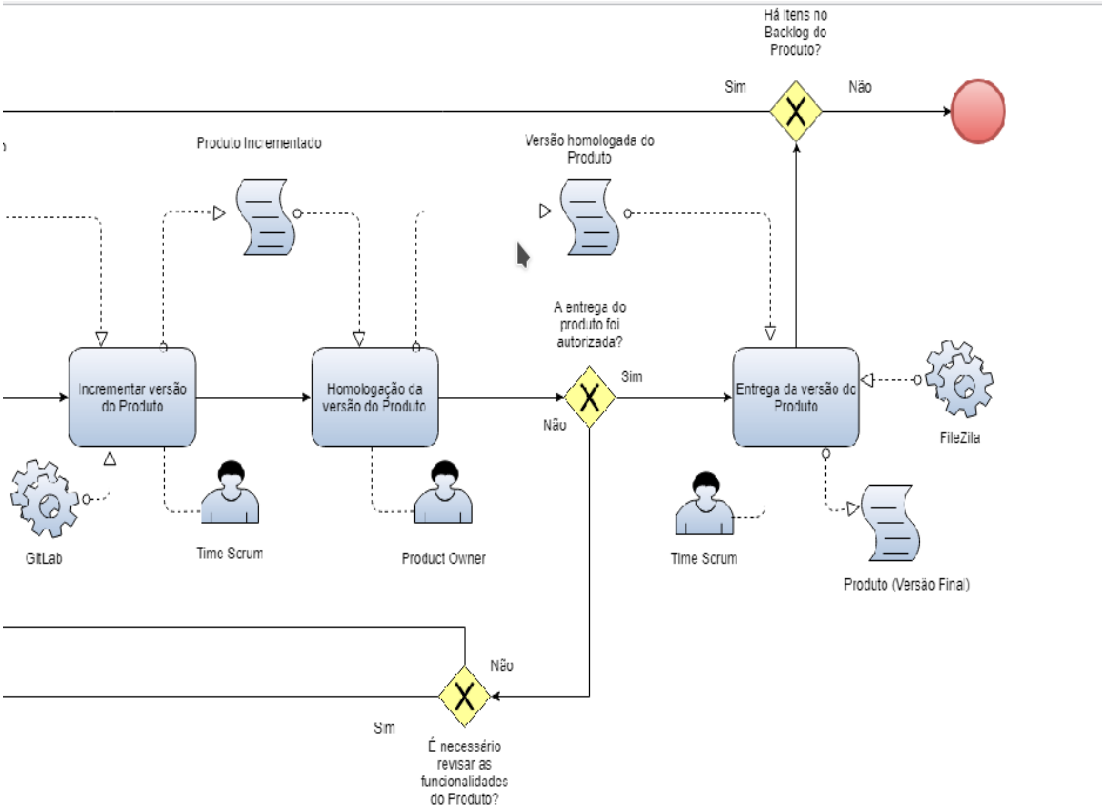


Fonte: Autor (2018).

Com o encerramento da ***sprint***, o fluxo retorna ao ***Processo principal***, onde é feita a verificação do fim da *Release.* Caso não esteja finalizada, é executado um ciclo de uma nova iteração, até que a *Release* seja finalizada. Então, o ***Time Scrum*** realiza a incrementação da versão desenvolvida com o auxílio da ferramenta ***GitLab***. Após o incremento do produto, o ***Product Owner*** realiza a homologação da versão do produto. Caso a versão esteja de acordo com o que foi solicitado pelo ***Product Owner***, é feito o ***deploy*** (implantar a release desenvolvida) do produto através da ferramenta ***FileZila*** e o mesmo é entregue pelo ***TimeScrum***. Caso contrário, o fluxo do processo retorna para o planejamento da ***Release*** (***Planejar Release***) e caso seja necessário rever as funcionalidades, retorna-se para ***Construir Modelo Abrangente***.

Após a ***Entrega da versão do produto***, é feito uma nova verificação para saber se ainda há itens no ***Backlog do produto***. Se ainda contiver, é feito um novo planejamento para a ***Release*** que será implementada, caso contrário, o produto está completamente finalizado. A Figura 17 mostra com exatidão esta etapa final.

Figura 17: Fim do Processo principal.



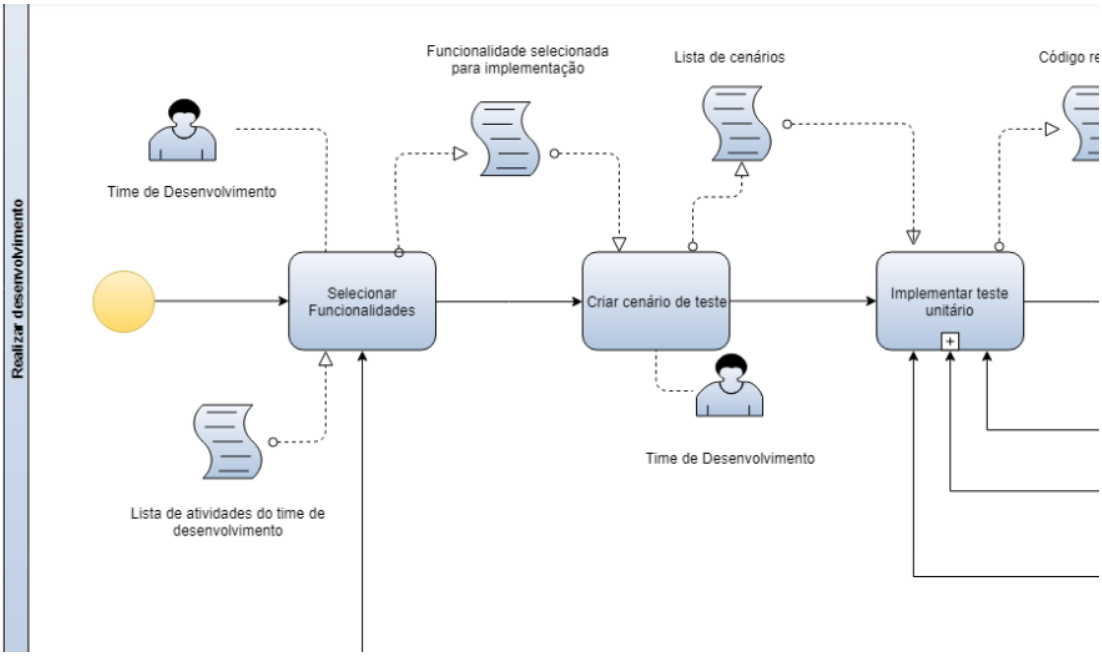
Fonte: Autor (2018).

4.1.4 Aplicação de melhorias e remodelagem do processo utilizado na COTIC (To be)

Com a finalização da modelagem, do processo atual (As Is), utilizado pela COTIC, estudos e análises mais críticas foram realizadas no processo mapeado, no intuito de promover mudanças de melhoria no processo de desenvolvimento.

Em seguida, uma remodelagem foi realizada, de acordo com as melhorias de processo identificadas. Uma nova metodologia (Test Driven Development) foi adicionada ao processo de desenvolvimento de software e a atividade "***Desenvolver produto***" (que encontra-se no subprocesso **Realizar Desenvolvimento do Produto**) sendo alteradapara atividades como "**Criar cenário de teste**" e "**Implementar teste unitário**", melhorando a qualidade do software produzido. As mudanças citadas para melhorar este modelo redesenhado, estão ilustradas na Figura 18.

Figura 18: Início do subprocesso Realizar Desenvolvimento do Produto remodelado (To be).



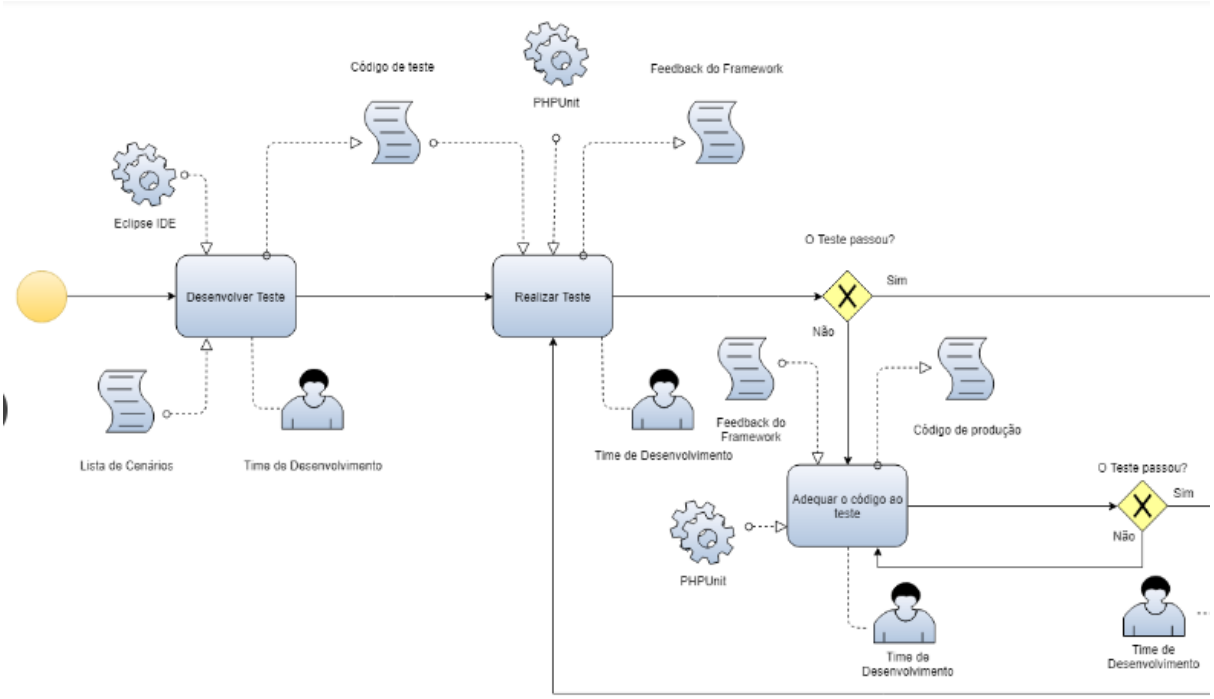
Fonte: Autor (2018).

Como podemos verificar na figura acima, o subprocesso ***Realizar desenvolvimento do Produto***, começa com o artefato ***Lista de funcionalidades do time de desenvolvimento***, originado na reunião diária da Sprint. Então, o processo sofre a primeira mudança de melhoria comparado ao que é utilizado atualmente. O fluxo segue para ***Criar cenário de teste***, onde o ***Time de desenvolvimento*** se reúne para montar os cenários em que a funcionalidade pode ser testada, originalizando a ***Lista de cenários*** que serão implementadas na atividade seguinte.

O próximo passo é o subprocesso ***Implementar teste unitário***, onde a equipe de desenvolvimento parte para a atividade de desenvolvimento das funcionalidades guiados pelos cenários de testes listados. Observa-se dentro do subprocesso a aplicação do *Ciclo do TDD* (fail-pass-refactor) através das atividades e de como o fluxo caminha. Inicialmente, tem-se a atividade ***Desenvolver Teste***, onde é escrito o código do teste unitário.

Em seguida, a atividade ***Realizar Teste*** é responsável por realizar os testes do código desenvolvido com o auxílio do ***framework PHPUnit*** para realizar as asserções no código. Como o *Ciclo do TDD* sugere, o primeiro teste deve trazer um *feedback* negativo, sendo assim, nesse momento observa-se a fase *Fail* do *TDD.* Então, é preciso escrever o código de produção para assegurar que o teste entrará na fase *Pass* do *Ciclo do TDD.* Essas duas fases do *Ciclo do TDD* (*fail-pass*) estão sendo mostradas com mais detalhes na Figura 19 a seguir.

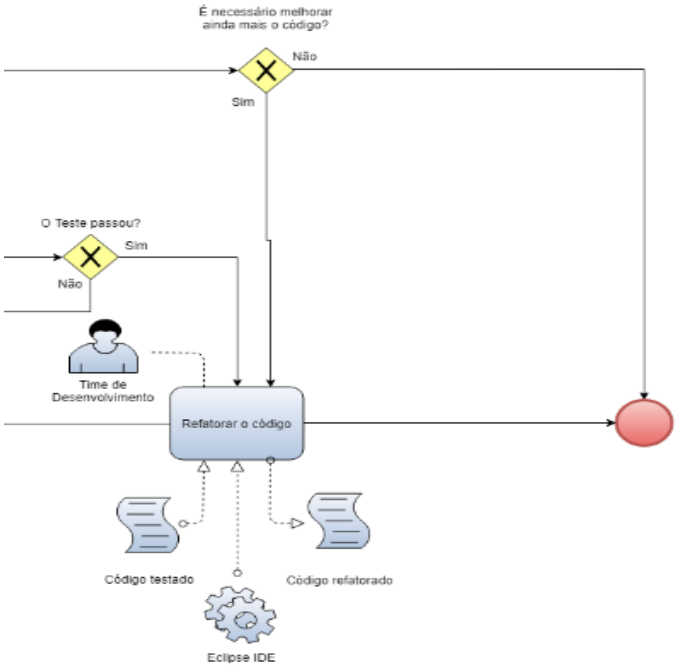
Figura 19: Fases fail-pass dentro do subprocesso Implementar Teste Unitário.



Fonte: Autor (2018).

Após o *feedback* positivo do *framework*, é verificado se o código precisa ser refatorado, então, são aplicadas as boas práticas de refatoração, por exemplo, remover duplicações no código, escrever o código de maneira simples e legível, dentre outras mais. Após a refatoração ser realizada, é necessário executar novamente o teste com o novo código produzido para verificar se o desempenho não regrediu, ou seja, se o que estava funcionando antes passou a apresentar algum erro com o novo código. Essa fase (*Refactor*) está sendo mostrada com mais detalhe na Figura 20 a seguir.

Figura 20: Fase refactor dentro do subprocesso Implementar Teste Unitário.



Fonte: Autor (2018).

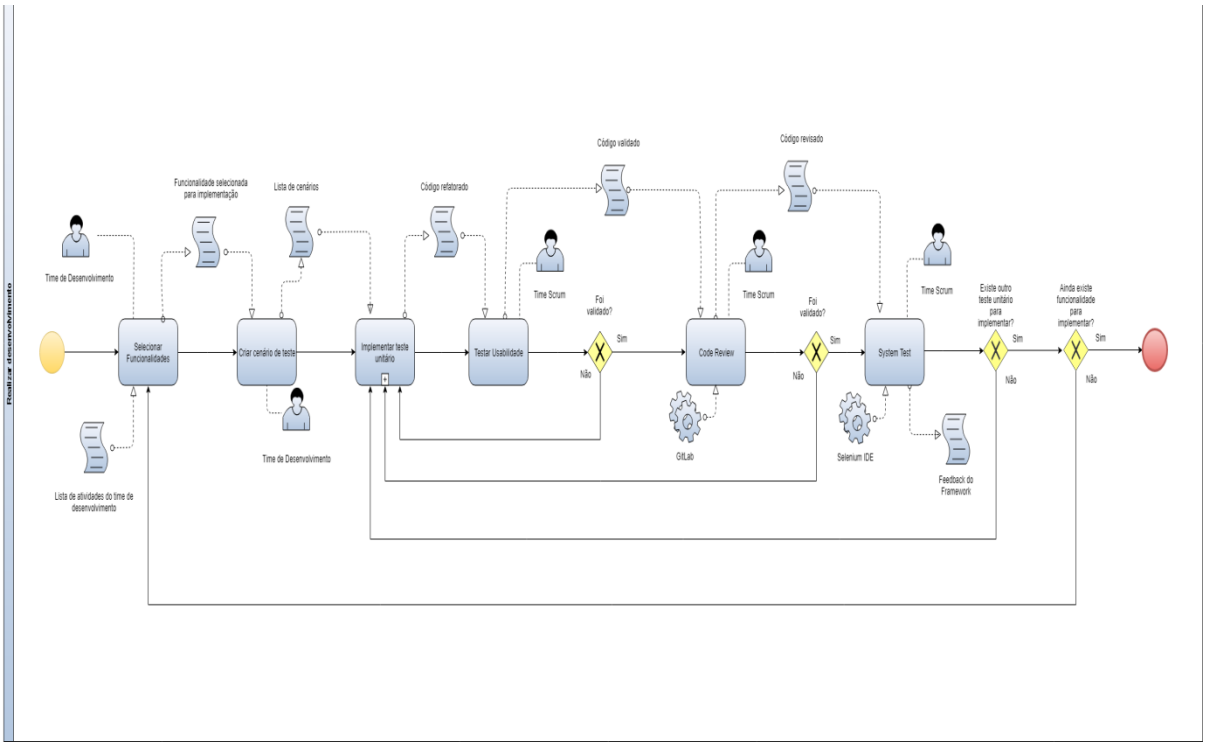
Ao finalizar o subprocesso ***Implementar teste unitário***, o fluxo volta para o ***Realizar desenvolvimento do produto*** e prossegue com a atividade ***Testar usabilidade***, realizada por membros do ***Time Scrum*** que participaram ou não do ***desenvolvimento do produto***, e então, é feito uma validação, caso a funcionalidade seja validada de forma positiva, o fluxo prossegue para a ***revisão do código*** (**Code Review**), caso contrário, o processo retorna para o *Implementar teste unitário* até que a funcionalidade desenvolvida esteja de acordo com o solicitado e tenha uma boa usabilidade.

A próxima etapa, o ***Code Review***, é executada por integrantes do ***Time Scrum*** que não fizeram parte do desenvolvimento do teste unitário, uma vez que serão revisados os códigos produzidos e precisam ser analisados por uma perspectiva diferente para que esta etapa seja realizada com maior efetividade. Após realizar esta revisão, os membros irão validar o código. Caso a validação seja positiva, o fluxo segue para o ***teste do sistema (System Test)***, caso contrário, o processo retorna para o desenvolvimento do teste unitário, até que o código produzido esteja de acordo com os padrões utilizados pela COTIC.

Em seguida é realizado o ***teste do sistema*** (**System Test**) com auxílio da ferramenta de teste automático ***Selenium IDE***, feito pelo ***Time Scrum***, podendo participar integrantes que fizeram parte do desenvolvimento do teste unitário, teste de usabilidade ou revisão do código.

Após o feedback do framework (***Selenium IDE***), o fluxo chega a sua terceira mudança comparado ao que está sendo usado atualmente pela COTIC, seguindo para um ***gateway*** com a finalidade deverificar se ainda existem outros testes unitário a serem desenvolvidos. Caso ainda haja outros testes a serem desenvolvidos, o fluxo volta para a atividade ***Implementar teste unitário***. Caso contrário, segue para um novo ***gateway*** com o questionamento: "Ainda existe funcionalidade para implementar?". Caso haja funcionalidade esperando para ser desenvolvida, o processo retorna para a primeira atividade (**Selecionar funcionalidade**) do subprocesso. Caso contrário, o subprocesso é finalizado como está exemplificado na Figura 21.

Figura 21: Realizar Desenvolvimento do Produto.



Fonte: Autor (2018).

4.2 AVALIAÇÃO DO ESTUDO REALIZADO NA COTIC

Para realizar a avaliação da modelagem desenvolvida, foi utilizada a ferramenta *SWOT* para analisar o cenário estratégico de uma empresa. O termo SWOT surgiu de uma mistura das iniciais de algumas palavras inglesas, as quais são: *strengths* (forças), *weakness* (fraquezas), *opportunities* (oportunidades) e *threats* (ameaças) (SANTOS *et al.*, 2010). A análise SWOT foi desenvolvida na década de 60, na Universidade de Harvard, a partir de uma estrutura resultante de discussões em sala de aula e posteriormente desenvolvido no intuito de ter uma estrutura para trabalhar com definições de estratégias de negócios (TAKAO *et al.*, 2006). Dessa forma, a Tabela 3 apresenta a matriz de *SWOT* usada para avaliar o processo modelado para a COTIC.

Tabela 3: Matriz SWOT do processo.

<table>
<tr><td rowspan="4">Ambiente Interno</td><td>Forças</td><td>Oportunidades</td><td rowspan="4">Ambiente Externo</td></tr>
<tr><td>● Cliente satisfeito;
● Modelo de escopo aberto;
● Maior gestão do projeto;
● Visualização do projeto.</td><td>● Avanço das demandas de projetos;
● Necessidade de entrega das demandas dentro do prazo.</td></tr>
<tr><td>Fraquezas</td><td>Ameaças</td></tr>
<tr><td>● Processo grande;
● Necessita de conhecimentos sobre metodologias ágeis.</td><td>● Projeto de escopo fechado.</td></tr>
</table>

Fonte: Autor (2018).

Diante das informações expostas e analisadas na matriz de *SWOT,* podemos perceber que o processo modelado da COTIC possui diversos benefícios no ambiente interno, os quais serão pontuados a seguir:

- Cliente satisfeito: o processo segue a risca o foco de acordo com a necessidade do cliente, buscando sempre compreender a necessidade do cliente, o que valoriza a cultura empresa de manter o cliente sempre por perto, aumentando seu grau de satisfação.
- Modelo de escopo aberto: possibilita adequações do projeto de acordo com as demandas do cliente, criando uma hierarquização das solicitações prioritárias de acordo com a necessidade do cliente.
- Maior gestão do projeto: maior controle do início ao fim do projeto, aumentando a possibilidade de identificar gargalos que possam surgir.
- Visualização do projeto: por se tratar de um processo que segue o incremento do projeto à cada iteração, o cliente recebe o que foi desenvolvido de acordo com sua priorização, podendo determinar qual funcionalidade é prioridade para a próxima entrega.

No entanto, vale ressaltar que pelo fato de ser um processo muito robusto, possuindo muitos fluxogramas, e que abrange diversas técnicas de metodologias ágeis, isso pode ser

considerado uma fraqueza internamente, pois todos os membros do time devem conhecer essas metodologias para compreender o processo.

Passando para a perspectiva externa, através da matriz de *SWOT* podemos identificar algumas oportunidades, tais como novas demandas, uma vez que as tecnologias vão avançando e, consequentemente, surgindo necessidades de utilizar sistemas que possam suprir as necessidades do cliente, proporcionando novas solicitações de projetos ou funcionalidades. Neste cenário, o processo desenvolvido destaca-se por ter seu fluxograma adequado para atender este cenário de mudanças frequentes em projetos.

É importante ressaltar que além de proporcionar uma melhor gestão das fases de execução do projeto, dando maior transparência para o cliente e melhor gestão por parte dos membros e, principalmente, para o *Scrum Master*, o processo desenvolvido influencia diretamente nos prazos estabelecidos de entrega do produto.

Uma ameaça identificada nesta ocasião são os projetos de escopo fechado, pois para este contexto, a utilização do processo modelado não é adequado por possuir um fluxograma grande e focar nos projetos de escopo aberto, onde o mesmo é utilizado quando não se conhece todas as necessidades do cliente.

5. CONSIDERAÇÕES FINAIS

Ao longo do desenvolvimento desta monografia, foi apresentado um estudo para propor melhorias nos processos utilizados pela Coordenadoria de Tecnologia da Informação e Comunicação da PROEG UFPA. Para isso foi sugerido a aplicação da modelagem de processo de software para visualizar o fluxo de trabalho utilizado e, além disso, realizar melhorias após a análise do processo existente. Como ponto de partida, foram analisadas técnicas de modelagem de processos, ferramentas de modelagem e algumas referências sobras as metodologias ágeis de desenvolvimento de software.

O guia de modelagem de processos de software resultante deste trabalho deverá auxiliar a automação de processos de negócio da COTIC PROEG, desde a identificação de novos requisitos até a sua implementação em um sistema real. A automação dos processos de negócio citados no estudo de caso da coordenadoria, permite reduzir consideravelmente o seu tempo de execução através do uso correto das metodologias ágeis apresentadas e aumentar seu grau de qualidade no produto desenvolvido através do uso do TDD em seu processo.

Durante o desenvolvimento deste trabalho, algumas tarefas necessitaram de uma atenção redobrada. Um bom exemplo está no tópico 4.1.2, o qual trata sobre a coleta dos dados detalhados do fluxo dos processos utilizados e organizá-los corretamente. Nesta etapa, é necessário possuir as informações de maneira bem explícita, correta e completa, pois a mesma servirá, posteriormente, para realizar o desenvolvimento da modelagem do processo utilizado pela COTIC PROEG, servindo de base para a melhoria a ser aplicada. O tópico 4.1.4, o qual trata sobre a aplicação de melhorias e remodelagem do processo utilizado na COTIC, também tem uma grande importância, pois é responsável por detalhar e mostrar de que forma os estudos e análises aprofundados do processo desenvolvido, resultaram e impactaram nas mudanças propostas neste trabalho.

Um dos pontos a ser destacado é o impacto positivo que o guia irá trazer na modelagem e automação dos processos de negócio da COTIC, haja vista que oferece uma fácil compreensão para pessoas que não estão familiarizadas com o tema, além de permitir um desenvolvimento rápido e objetivo. Outro ponto a ser enfatizado é a credibilidade dos produtos desenvolvidos e satisfação dos clientes da COTIC, uma vez que, muitos bugs deixaram de existir por conta da utilização do desenvolvimento guiado por testes, o qual, resultou na diminuição considerável das demandas de correções de funcionalidades desenvolvidas de maneira ineficiente.

É importante ressaltar, que a qualidade é fruto do uso de processos que estão em constante uso no mercado de trabalho e com resultados evidenciados na melhoria dos processos de

desenvolvimento de software. Já a agilidade é proveniente do uso das metodologias ágeis, proporcionando a conclusão das funcionalidades de maior prioridade para o cliente, diminuindo as chances de produzir funcionalidades inúteis ou dificilmente utilizadas.

5.1. DIFICULDADES ENCONTRADAS

Ao longo do desenvolvimento deste trabalho, foram encontradas algumas dificuldades pontuais. Entre elas, o pouco entendimento dos passos seguidos do início ao fim dos processos utilizados pela coordenadoria, desde a reunião com o cliente para a construção do modelo abrangente, resultando nos requisitos que serão desenvolvidos, passando pelo desenvolvimento destes requisitos, até a sua disponibilização ao solicitante.

Outra grande dificuldade foi adequar o processo utilizado pela COTIC para atender as práticas da metodologia de desenvolvimento guiado por teste, uma vez que, não foram encontrados exemplos de modelagens aplicando esta metodologia, portanto, o modelo apresentado como melhoria foi um dos primeiros a ser desenvolvido.

5.2. TRABALHOS FUTUROS

Como trabalhos futuros, uma das propostas seria a coleta das métricas do modelo atual (As Is), antes da utilização do novo modelo apresentado neste trabalho, e a coleta das métricas após a aplicação do modelo melhorado (To Be), para assim realizar a comparação entre os dois modelos e verificar o desempenho que se teve a partir do uso da melhoria desenvolvida.

Outra possível proposta, seria a adequação e melhoria deste novo processo para atender o Nível D do MPS.Br, afim de trazer maior qualidade e credibilidade ao desenvolvimento de requisitos dos projetos desenvolvidos na COTIC.

AGRADECIMENTOS

Agradeço imensamente a todos que fazem parte diretamente e indiretamente da equipe da Coordenadoria de Tecnologia da Informação e Comunicação da PROEG – UFPA, que ao longo de toda a minha trajetória e estadia dentro desta Pró-reitoria, me proporcionaram um ambiente de trabalho dinâmico e interativo, onde pude absorver um grande conhecimento técnico, prático e ético, o qual me tornou um profissional mais capacitado, ajudando-me na escolha do tema e desenvoltura deste trabalho de conclusão de curso.

REFERÊNCIAS

ABDALA, Martha. **Modelagem do Processo de Gerenciamento da Configuração de Software para um Ambiente Integrado.** Workcap, 2003.

ARAUJO, Luis César G. de; GARCIA, Adriana Amadeu; MARTINES, Simone. **Gestão de processos: melhores resultados e excelência organizacional**. São Paulo: Atlas, 2011.

ALBRECHT, Alan. **Measuring Application Development productivity**. Share/Guide IBM Application Development Symposium, 1979.

ALBRECHT, Alan; GAFFNEY, John. "Software Function, Lines of Code and Development Effort Prediction: a Software Science Validation". IEEE TransactionsOf Software Engineering, 1983.

AELIAN. Disponível em: http://www.sstecnologia.com.br. Acessado em: 21/09/2018

AGILITY LADDER. Disponível em: http://www.offshore-challenge.com/wp-content/uploads/2014/08/XP.png. Acessado em: 21/09/2018

BARBOSA, António. AZEVEDO, Bruno. PEREIRA, Bruno. CAMPOS, Pedro. SANTOS, Pedro. **Metodologia Ágil: FeatureDrivenDevelopment.**

BASILI, Victor; CALDIERA, Gianluigi; ROMBACK, H. Dieter. **Goal Question Metric Paradigm**. In: Encyclopedia of Software Engineering. [s.l.]: John Wiley& Sons, Inc., 1994.

BARVINSKI, Carla Adriana. CAGNIN, Maria Istela. LIMA FILHO, Nilson E. S. VERONEZI, Leandro. ***XP Tracking Tool:*** Uma Ferramenta de Acompanhamento de Projetos Ágeis. **In:** Anais do AgileBrazil 2010- Conferência Brasileira sobre Métodos Ágeis de Desenvolvimento de Software.

BALDAM, Roquemar, Valle Rogério, PEREIRA Humberto, HISLT Sérgio, ABREU Mauricio e SOBRAL, Valmir. **Gerenciamento de processos de negócio: BPM - BUSINESS PROCESS MANAGEMENT**. 2ª edição. São Paulo. 2008

BECK, Kent; Andres, Cynthia. **Extreme programming Explained**: Embrace chance, 2 edn. Addison-Wesley, 2004.

BECK, Kent. **TDD Desenvolvimento Guiado por Teste.** Porto Alegre: Bookman. 2010.

BECK, Kent. **Smalltalk Best Practice Patterns**. Upper Saddle River: Prentice Hall, 1996.

BERNARDO, Kleber. **Manifesto ágil, como tudo começou** https://www.culturaagil.com.br/manifesto-agil-como-tudo-comecou/. Acessado em: 15 de Abril de 2018.

BORGES, Eduardo. **Conceitos e benefícios do Test Driven Development.** http://www.inf.ufrgs.br/~cesantin/TDD-Eduardo.pdf.Acessado em: 03 de Agosto de 2018.

COHN, Mike. **Agile Estimating and Planning**. Upper Saddle River: Prentice Hall, 2005.

CRUZ, Tadeu. **BPM & BPMS BUSINESS PROCESS MANAGEMENT & BUSINESS MANAGEMENT** Systems. Rio de Janeiro 2009 2ª edição.

CASTRO, V.; CARDOSO, B; LISBOA, D.; **Aplicação de Abordagens Ágeis: Estudo de Caso da utilização do SCRUM- PROEG**. 2011

COMPARTILHANDO. Disponível em: http://jkolb.com.br/wp-content/uploads/2013/12/xp.png. Acessado em: 21/09/2018

CODE4CODERS. Disponível em: https://code4coders.files.wordpress.com/2016/03/tdd.png?w=680. Acessado em: 21/09/2018

DANTAS, V. F. **Uma metodologia para o desenvolvimento de aplicações Web num cenário global**. 2003.

DERNIAME, J. C.; KABA, B. A.; WASTELL, D.; **Software Process: Principles, Methodology, and Technology**. 1999

EBERT; CHRISTOF; DUMKE; REINER. **Software Measurement: Establish, Extract, Evaluate, Execute.** s.l. 2009.

EFFECTIVE, Project Management Consultancy. Disponível em: http://www.effectivepmc.com/_/rsrc/1525598842094/blog/agile/information-radiators/burn-down-chart/ReleaseBurnDown.png. Acessado em: 21/09/2018

FREITAS, Jaily. **Descrição de um processo de software utilizando SPIDER_ML: Um estudo de caso da AIT-PROEG.** 2016

FERREIRA, Allan. **Utilização da Metodologia TDD para Desenvolvimento de Software.** Marília, 2011.

FOWLER, Martin. **Refactoring Improvingthe design of existing code**. Addison-Wesley. 1999.

GitLab inc. GitLab. Disponível na internet em: http://www.gitlab.com.

GOMES, Andre. **Desenvolvimento de Software com entregas frequentes e foco no valor de negócio.** São Paulo: Casa do Código. 2013.

GONÇALVES, Hugo. **Guia para Modelagem e Automação de Processos de Negócios Acadêmicos: estudos de caso com processos da UFSC.** Florianópolis. 2016

GUERRINI, Fábio Müller ;et al. Modelagem da organização: Uma visão integrada. Porto Alegre: Bookman. 2014.

HIGHSMITH, Jim. **Agile Software Development Ecosystems**. Adisson-Wesley, 2002.

https://paginas.fe.up.pt/~aaguiar/es/artigos%20finais/es_final_22.pdf. Acessado em: 13 de Agosto de 2018.

International Function Point Users Group. Manual de Práticas de Contagem de Pontos de Função, Versão 4.3.1. 2010.

IPROCESS. **Exemplo de Notação BPMN.** Disponível em: http://blog.iprocess.com.br/wp-content/uploads/2012/10/inicio_fim.png. Acessado em: 21 de Setembro de 2018.

JEFFRIES, Let Ron. **What is Extreme Programming?** http://xprogramming.com/what-is-extreme-programming/#planning>. Acessado em: 07 de Agosto de 2018.

J. C. Derniame, B. A. Kaba, and D. Wastell. Software process: principles, methodology and technology. 1999.

Kitchenham, B.; Pfleeger, S. L.; Fenton, N. Towards a Framework for Software Measurement Validation. IEEE Transactions on Software Engineering, vol. 21, nº 12. December, 1995.

KUHN, G. R.; PAMPLONA, V.F. **Apresentando xp: Encante seus clientes com exterme programing**. 2004. Disponível em: http://javafree.uol.com.br/artigo/871447/.

KEENSKAUG, Trygve. **Working With Objects the Optam Software Engineering Method**.Manning, 1995.

MARQUES, André. **Metodologias ágeis de desenvolvimento: Processos e Comparações.** São Paulo, 2012. http://www.fatecsp.br/dti/tcc/tcc00064.pdf.Acessado em: 13 de Agosto de 2018.

Metrics Views. Vol5. Allan J. Albrecht Father of Function Points. Disponível em: <http://www.ifpug.org/metricviews/>.

MOORE, C. et al. **BPM CBOK**. [s.l.] Association of business process management professionals, 2013.

MPS.BR - Melhoria de Processo do Software Brasileiro: Guia Geral MPS de Software. SOFTEX: Rio de Janeiro, 2016.

Object Management Group. Disponível em: www.omg.org. Acessado em: 21/09/2018

PIZZA, William. **A metodologia Business Process Management (BPM) e sua importância para as organizações.** São Paulo, 2012.

PARK, Robert; GOETHERT, Wolfhart; FLORAC, William**. Goal-Driven Software Measurement- A Guide book.** Pittsburgh: Carnegie MellonUniversity, 1996

PRESSMAN, Roger. **Engenharia de Software**. 6° Edição, McGraw Hill. 2006.

PRESSMAN, R. S.; **Software Engineering: A Practitioner's Approach**. 7° Edição, McGraw Hill. 2010.

PRESSMAN, Roger; MAXIM, Bruce. Engenharia de Software: Uma Abordagem Profissional. 8ª edição. Porto Alegre: MCGraw-Hill , 2016.

PORTO, Gilberto. **BPM e CBOK.** Disponível em: http://igepp.com.br/uploads/arquivos/igepp_-_administracao_geral_-_gilberto_porto_-_cbok_24out14.pdf. Acessado em: 21 de Setembro de 2018.

REBOUÇAS, Djalma, **Planejamento estratégico: conceitos, metodologia e praticas.** 24. ed. São Paulo: Atlas, 2007.

RIBEIRO, Rafael Dias; RIBEIRO, Horácio da Cunha e Souza. **Métodos Ágeis em Gerenciamento de Projetos**. 1ª Edição, Rio de Janeiro, 2015

SANT'ANNA, Nilson; ABDALA, Martha. **Uma abordagem para a gerência das modificações e da configuração em um ambiente integrado para o desenvolvimento e gestão de projetos de software.** Instituto Nacional de Pesquisas Espaciais. 2004

SILVA, Everton. **Desenvolvimento Guiado Por Testes.** FATEC - SP, São Paulo, 2012.

SANTIN, Carlos Eduardo. **Desenvolvimento guiado por testes e ferramentas xUnit.** http://www.inf.ufrgs.br/~cesantin/TDDArtigo.pdf. Acessado em: 07 de Agosto de 2018.

Selenium inc. Selenium. Disponível na internet em: http//www.seleniumhq.org.

SOMMERVILLE, Ian. **Engenharia de Software**. 9ª edição. São Paulo: Pearson, 2011.

STOROLLI, A. L.; ZANOLLA, G. I.; GUIDINI, J. E.; BORSOI, B. T.; **Modelagem de processo de software**. 2009.

SANTOS, M. A.; GREGHI, J. G.; BERMEJO, P. H. S.. **Avaliação do Impacto do SCRUM no desenvolvimento de software utilizando a análise SWOT.** São Paulo. 2010

SUTHERLAND, Jeff. **Scrum: a arte de fazer o dobro do trabalho na metade do tempo**. São Paulo: LEYA, 2014.

SUTHERLAND, Jeff; SCHWABER, Ken**. Guia do Scrum. Um guia definitivo para o Scrum: As regras do jogo**. Disponível em:<http://www.scrumguides.org/download.html>.

TAKEUCHI, Hirotaka; NONAKA, Ikujiro; **The new new product development game**, Product Development Scrum, 1986.

TAKAO, E. L.; COPPPINI, N. L.; TOREGEANI, A. F. (2006). **Aplicação da Técnica SWOT na Viabilização de um Sistema de apoio para Executivo com Software Livre em uma Indústria de Embalagens Plásticas.** CADERNO DE ADMINISTRAÇÃO. v. 14, n.1, p. 9-17.

TELES, V. M. **Extreme Programming: Aprenda como encantar seus usuários desenvolvendo software com agilidade e alta qualidade**. Novatec, 2004.

Trello inc. Trello. Disponível na internet em: http://www.trello.com

VARASCHIM, Jacques Douglas; **Implantando o SCRUM em um Ambiente de Desenvolvimento de Produtos para Internet**. 2009.

WILLIAMS, Lauri e; MAXIMILIEN, E. Michael. **Assessing Test-Driven Development at IBM.** In: ICSE, 2003, Portland. Artigo. Washington: IEEE Computer Society, 2003. p. 564 - 569.

WIKIMEDIA COMMONS. Disponível em: https://upload.wikimedia.org/wikipedia/commons/2/2f/Fdd.png. Acessado em: 21/09/2018