UNIVERSIDADE FEDERAL DO PARÁ
INSTITUTO DE CIÊNCIAS EXATAS E NATURAIS
FACULDADE DE COMPUTAÇÃO
CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

LUCAS TAVARES MONTEIRO

UMA MELHORIA NO PROCESSO ÁGIL DA COTIC-PROEG DA UFPA PARA ATENDIMENTO DO PROCESSO DE DESENVOLVIMENTO DE REQUISITOS DO MR-MPS-SW

Belém
2018

LUCAS TAVARES MONTEIRO

UMA MELHORIA NO PROCESSO ÁGIL DA COTIC-PROEG DA UFPA PARA ATENDIMENTO DO PROCESSO DE DESENVOLVIMENTO DE REQUISITOS DO MR-MPS-SW

Trabalho de Conclusão de Curso apresentado como um dos requisitos para a obtenção do título de Bacharel em Ciência da Computação pelo Curso de Bacharelado em Ciência da Computação da Universidade Federal do Pará.

Orientador: Prof. Dr. Sandro Ronaldo Bezerra Oliveira.

.

Belém – PA
2018

LUCAS TAVARES MONTEIRO

UMA MELHORIA NO PROCESSO ÁGIL DA COTIC-PROEG DA UFPA PARA ATENDIMENTO DO PROCESSO DE DESENVOLVIMENTO DE REQUISITOS DO MR-MPS-SW

Trabalho de Conclusão de Curso apresentado como um dos requisitos para a obtenção do título de Bacharel em Ciência da Computação, pelo Curso de Bacharelado em Ciência da Computação da Universidade Federal do Pará.

Aprovado em ___/___/___

BANCA EXAMINADORA

_____

MSc. Isaac Souza Elgrably
Aluno de Doutorado do PPGCC/UFPA

_____

MSc. Lennon Sales Furtado
Aluno de Doutorado do PPGCC/UFPA

_____

Prof. Dr. Sandro Ronaldo Bezerra Oliveira – Orientador
Faculdade de Computação – ICEN – UFPA

Belém – PA
2018

Para a minha mãe, Maria Iolanda Monteiro Tavares, e a todos meus familiares, dedico este trabalho com muito amor e carinho.

AGRADECIMENTOS

Agradeço imensamente à minha mãe por estar sempre ao meu lado, incentivando a busca pelos meus objetivos tanto na vida pessoal como na profissional, e por ser a principal responsável por ter me transformado na pessoa que sou.

A meus familiares pelo apoio que me foi oferecido, principalmente no que diz respeito à minha formação educacional.

Aos meus amigos, por estarem sempre ao meu lado, tornando a caminhada em busca da vida profissional mais suave, e cheia de lembranças agradáveis.

Aos meus colegas de turma, que tive a satisfação de compartilhar muitas experiências durante a minha graduação.

Aos meus amigos da COTIC-PROEG, antiga AIT-PROEG, que me forneceram grande conhecimento na área de Engenharia e Desenvolvimento de *Software*, e transformaram o ambiente de trabalho em um lugar agradável e acolhedor.

Ao meu professor orientador Sandro Oliveira, que além de contribuir para a minha formação acadêmica, forneceu todo suporte e ajuda que precisei para o desenvolvimento deste trabalho.

RESUMO

Atualmente, muitas empresas buscam aprimorar seu processo de desenvolvimento de *software* utilizando modelos de qualidade como base. Nesse contexto, o programa de Melhoria de Processo de Software Brasileiro (MPS.BR) foi desenvolvido, assim como o Modelo MPS de Software (MR-MPS-SW). Frequentemente, as organizações enfrentam dificuldades na etapa de desenvolvimento dos requisitos do produto de *software*, fator crucial para a garantia da criação de *software* com qualidade. Esse trabalho propõe uma melhoria do processo ágil de desenvolvimento da COTIC-PROEG, utilizando como base o processo de Desenvolvimento de Requisitos (DRE) do modelo de qualidade MR-MPS-SW, que tem como objetivo o estabelecimento de requisitos que atendam as expectativas, necessidades e restrições do cliente. Para tal, foi realizada a modelagem do processo atual de desenvolvimento da COTIC, sob o qual foi realizada uma avaliação com base em evidências de atendimento aos resultados esperados do DRE, presentes no Guia de Implementação de *Software* do nível de maturidade D do MR-MPS-SW, e então proposto um novo processo de desenvolvimento, de forma que todos os resultados esperados sejam completamente atendidos.

Palavras-chave: Desenvolvimento de Requisitos, MR-MPS-SW, Processo de Desenvolvimento de Software, Melhoria Contínua do Processo.

ABSTRACT

Nowadays, many companies seek to improve their software process development, using quality models as its foundation. In this context, the Brazilian Software Process Improvement (MPS.BR), as well as the Software MPS Model (MR-MPS-SW), were created. Frequently, organizations face difficulties on developing software requirements for a software product, which is a crucial factor to guarantee the creation of high-quality software. This work proposes an improvement on the agile development process of COTIC-PROEG, using the Requirements Development (DRE) process, from the MR-MPS-SW quality model, as its foundation. To do so, was made the modeling of the COTIC present process, which was used to make an evaluation based on the evidences of it attending the DRE expected results, found in the Software Implementation Guide of maturity level D of MR-MPS-SW, and then proposed a new development process, in a way that all expected results are fulfilled.

Keywords: Requirements Development, MR-MPS-SW, Software Development Process, Continuous Process Improvement.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE ABREVIAÇÕES

AIT – Assessoria de Informação e Tecnologia

AP – Atributos de Processo

BPD – Business Process Diagram

BPM CBOK – Business Process Management Body of Knowledge

BPMI – Business Process Modeling Initiative

BPMN – Business Process Management Notation

CMMI – Capability Maturity Model Integration

COTIC – Coordenadoria de Tecnologia da Informação e Comunicação

DRE – Desenvolvimento de Requisitos

MPS.BR – Melhoria do Processo de Software Brasileiro

MR-MPS-SW – Modelo de Referência MPS para Software

OMG – Object Management Group

PO – Product Owner

PROEG – Pró-Reitoria de Ensino e Graduação

UFPA – Universidade Federal do Pará

XP – Extreme Programming

SUMÁRIO

1 INTRODUÇÃO

Nesta seção, será realizada uma introdução ao trabalho, contendo o contexto de sua elaboração, as motivações e justificativas do mesmo, seus objetivos, sendo eles gerais ou específicos, e a forma como o trabalho foi estruturada.

1.1 CONTEXTO

Software computacional é, atualmente, um dos produtos mais importantes que compõem o mercado mundial. Segundo Sommerville (2011), o mundo moderno não poderia existir sem os *softwares*. A grande maioria dos serviços nacionais, distribuições industriais, produtos elétricos, sistemas financeiros, indústria de música, jogos de computador, enfim, uma infinidade de áreas que compõem uma sociedade, nacional e internacional, dependem e fazem uso intensivo de *software*.

Verifica-se, portanto, a importância da Engenharia de *Software* para o mundo contemporâneo, tendo como objetivo apoiar o desenvolvimento profissional de *software*, com foco no custo-benefício. Essa engenharia inclui técnicas que apoiam especificação, projeto e evolução de programas, tendo extrema importância na mitigação de problemas, envolvendo desenvolvimento de *software*. Atualmente, empresas tem construído um processo padrão de desenvolvimento, que combinam um conjunto de práticas de engenharia de *software* para auxiliar na criação de *software* de qualidade. Trata-se, portanto, de um conjunto de atividades e metodologias que irão guiar a produção de um produto de *software* (SOMMERVILLE, 2011).

Entretanto, ainda existem diversos desafios a serem enfrentados. Um desses desafios diz respeito à criação de requisitos para o desenvolvimento de *software* dentro de um projeto. Segundo o modelo de referência MPS de Software (SOFTEX, 2016a), o processo de Desenvolvimento de Requisitos (DRE) tem como objetivo a criação dos requisitos que dizem respeito ao cliente, ao produto e aos componentes do produto, e envolve atividades como a coleta das expectativas e restrições do cliente, definição de requisitos funcionais e não-funcionais e desenvolvimento de conceitos operacionais.

Dado o contexto, este trabalho tem como objetivo principal realizar a avaliação de um processo de desenvolvimento de *software*, utilizando como referência o atendimento aos resultados esperados do processo de Desenvolvimento de Requisitos (DRE), presente no Guia de Implementação de Software nível D, do Modelo de Referência MPS de Software (MR-MPS-SW) (SOFTEX, 2016b). Para tal, foi utilizado como objeto de estudo o processo de

desenvolvimento da Coordenadoria de Tecnologia da Informação e Comunicação (COTIC), realizando a modelagem do mesmo a partir da linguagem BPMN e, posteriormente, foi realizada a modelagem do processo incluindo sugestões de melhoria para a aderência aos resultados esperados do DRE.

1.2 JUSTIFICATIVA

É verificada a dificuldade na criação de requisitos de *software* apresentada pelos desenvolvedores da Coordenadoria de Tecnologia da Informação e Comunicação (COTIC). Devido à falta de informação e pouco refinamento dos requisitos do produto da coordenadoria, a organização apresenta problemas na etapa de desenvolvimento do produto, pois muitas vezes esses requisitos não apresentam informações suficientes para o desenvolvimento, ou até mesmo possuem detalhamentos que não dizem respeito às expectativas, necessidades e restrições do cliente.

Portanto, a finalidade deste trabalho é promover uma melhoria no processo de desenvolvimento da COTIC, afim de otimizar o desenvolvimento de requisitos no mesmo, para que a organização consiga estabelecer a criação desses requisitos de forma que auxilie o desenvolvimento de *software*, atendendo as expectativas do cliente.

Em decorrência da grande disseminação do modelo de referência para *software* (MR-MPS-SW) no país, o mesmo foi utilizado como base para avaliação do processo de desenvolvimento ágil da COTIC, com o intuito de verificar quais são as possíveis falhas que estão ocorrendo no mesmo, sugerindo uma melhoria de processo que atenda aos resultados esperados do processo de DRE do MR-MPS-SW por completo.

1.3 MOTIVAÇÃO

Um grande motivador para a realização deste trabalho foi o fato de que o processo de desenvolvimento da COTIC é considerado um processo ágil de desenvolvimento, fazendo com que a adequação do mesmo para o DRE representasse um grande diferencial.

Outra grande motivação foi poder contribuir para a melhoria do processo da COTIC, uma organização que busca a melhoria contínua do seu processo de desenvolvimento, e que, frequentemente, utiliza de suas experiências para disseminar conhecimento e informações acerca de um ambiente de trabalho ágil.

1.4 OBJETIVOS

Para realizar a confecção deste trabalho, foram estabelecidos objetivos a serem cumpridos com o mesmo, que foram divididos em objetivo geral e objetivos específicos.

1.4.1 Objetivo Geral

Este trabalho tem como objetivo geral avaliar o processo de desenvolvimento da equipe de Tecnologia da Informação da PROEG-UFPA, por meio da modelagem desse processo, utilizando como base os resultados esperados presentes no processo de DRE do nível D do MR-MPS-SW. Feita a avaliação, é sugerida uma melhoria para esse processo, de forma que todos os resultados esperados sejam atendidos totalmente.

1.4.2 Objetivos Específicos

- Estudar as metodologias utilizadas pela COTIC;
- Realizar o mapeamento das etapas do processo de desenvolvimento da coordenadoria;
- Realizar a modelagem do processo;
- Estudar o nível D do modelo MR-MPS-SW, com aprofundamento no processo de Desenvolvimento de Requisitos (DRE);
- Avaliar o processo modelado com base nos resultados esperados do DRE;
- Sugerir melhorias para o atendimento dos resultados esperados por completo; e
- Realizar a modelagem do processo de desenvolvimento melhorado.

1.5 METODOLOGIA

Este trabalho teve início a partir do mapeamento dos processos de desenvolvimento da COTIC. Para tal, foi realizada uma pesquisa bibliográfica com o intuito de formar a base teórica relacionada às metodologias ágeis presentes no processo da instituição, utilizando como material de apoio trabalhos relacionados a esse processo. Além disso, foram realizadas entrevistas com o coordenador da COTIC para coletar informações sobre o processo e auxiliar no mapeamento.

A partir da análise efetuada, foi realizada a escolha da ferramenta e da linguagem de modelagem, que seriam utilizadas para modelar o processo de desenvolvimento ágil da COTIC. Feita essa escolha, o processo foi devidamente modelado.

Após a modelagem do processo, foi realizado um estudo do conteúdo relacionado ao desenvolvimento de requisitos, utilizando o modelo MR-MPS-SW para o entendimento dos resultados esperados referentes ao processo de Desenvolvimento de Requisitos (DRE), e como realizar o atendimento completo desses resultados.

Posteriormente, foi realizada a avaliação do processo modelado com base no estudo realizado acerca do DRE. Feita a avaliação, foi possível detectar as possíveis melhorias que foram sugeridas para o processo. De acordo com o que foi sugerido, foi realizada a modelagem do processo de desenvolvimento melhorado da COTIC.

1.6 TRABALHOS CORRELATOS

Esta seção tem como objetivo apresentar o processo de revisão e análise de alguns trabalhos relacionados, 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], e os dois trabalhos mais importantes que serviram como base para este foram trabalhos de conclusão de curso que envolviam o processo de desenvolvimento da COTIC.

Um dos trabalhos utilizados como referência para modelagem foi o trabalho de conclusão de curso da autora FREITAS (2016), que utiliza o processo de desenvolvimento da COTIC como objeto de estudo para a realização de uma modelagem por meio do *software* SPIDER-ML. Por ser um trabalho de 2016, a modelagem presente já estava defasada, pois o modelo atual da COTIC já passou por algumas mudanças desde então. Entretanto, o trabalho de FREITAS foi de importante ajuda para realização deste, pois apresentava a modelagem de etapas importantes do processo, como o desenvolvimento do produto, de forma bem detalhada.

O segundo trabalho, do autor BARATA (2018), realiza a modelagem do processo de desenvolvimento da COTIC com o objetivo de propor melhorias para o mesmo, com foco em *Test Driven Development*. Por ser um trabalho mais recente, estava mais atualizado com o processo atual da COTIC, sendo a principal referência para mapeamento de etapas e atividades que ocorrem dentro do processo.

[1] Disponível em: scholar.google.com

1.7 ESTRUTURA DO TRABALHO

Este trabalho está estruturado da seguinte forma:

- Capítulo 1 – Consiste na introdução do trabalho, utilizado para realizar a contextualização do mesmo, assim como apresentar sua justificativa, motivação, objetivos e a metodologia utilizada;
- Capítulo 2 – Apresenta a fundamentação teórica utilizada para este trabalho, sendo dividido em seis partes: Engenharia de Software, Engenharia de Requisitos, Processo de Software, Melhoria de Processo, MPS.BR e Metodologia Ágil;
- Capítulo 3 – Intitulado de Relato da Melhoria, tem como objetivo relatar os procedimentos ocorridos para a realização da melhoria do processo analisado, dividido em quatro partes: Contexto, Processo Ágil Antes da Melhoria, Avaliação do Processo e Melhoria no Processo Ágil Pós-Avaliação; e
- Capítulo 4 – É constituído pelas considerações finais do trabalho, divido em quatro etapas: Visão Geral, Contribuições, Limitações e Trabalhos Futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Nesta seção, serão apresentados os conceitos necessários para a contextualização da metodologia utilizada neste trabalho, assim como fornecer a fundamentação teórica fundamental para o entendimento da melhoria realizada no processo da COTIC-PROEG, abordando temas como: Engenharia de Software; Engenharia de Requisitos; Processo de Software; Melhoria de Processo; MPS.BR; a definição do Desenvolvimento de Requisitos no MR-MPS-SW – Modelo de Referência para Melhoria de Processo do Software Brasileiro (SOFTEX, 2016a); e Métodos Ágeis.

2.1 ENGENHARIA DE SOFTWARE

Segundo Sommerville (2011), a Engenharia de Software é uma disciplina cujo foco está em todo o processo de criação de um produto de *software*, desde a especificação do sistema, em seu estágio inicial, até a manutenção do sistema, estágio final da produção, onde o mesmo já se encontra em uso. A engenharia de software não se preocupa apenas com os processos técnicos, também incluindo atividades de gerenciamento do projeto e desenvolvimento de estratégias e teorias para auxiliar a produção de *software*.

De acordo com Pressman (2016), a pedra fundamental que sustenta a Engenharia de Software é o foco na qualidade, pois qualquer abordagem de engenharia deve estar fundamentada em um comprometimento organizacional com a qualidade. A engenharia de software pode ser definida como uma tecnologia em camadas, que são sustentadas pelo foco na qualidade, sendo elas:

- Processos: camada definida como o elo que mantém as camadas de tecnologia coesas, possibilitando o desenvolvimento de *software* a partir de uma metodologia estabelecida para realizar efetivamente a entrega de tecnologia de engenharia de software;
- Métodos: camada que fornece informações técnicas para o desenvolvimento do *software*; e
- Ferramentas: camada responsável por oferecer suporte automatizado ou semiautomatizado para o processo.

A Figura 1 ilustra uma representação visual dessas camadas.

Figura 1. Camadas da Engenharia de Software, Adaptado de Pressman (2016).

Este trabalho tem como foco principal a Engenharia de Requisitos, que está contida na disciplina de engenharia de software, e será abordado no tópico a seguir.

2.2 ENGENHARIA DE REQUISITOS

Segundo Pressman (2016), a Engenharia de Requisito é um conjunto de técnicas e tarefas que levam a um entendimento dos requisitos necessários para desenvolver um produto de *software*, sendo responsável por construir uma ponte entre o projeto e sua construção, permitindo a observação do contexto do trabalho de *software* a ser desenvolvido, as necessidades específicas a que o projeto e a construção devem atender, as prioridades que guiam a ordem na qual o trabalho deve ser concluído e funções e comportamentos que terão impacto no projeto.

De acordo com Sommerville (2011), requisitos de um sistema são descrições do que esse sistema deve fazer, os serviços oferecidos e suas restrições de funcionamento. Os requisitos refletem as necessidades, restrições e objetivos do cliente com o sistema que irá ser desenvolvido, e a engenharia de requisitos é o processo de descoberta, análise e documentação dos mesmos.

Os requisitos de *software* são frequentemente classificados como funcionais e não funcionais. Os requisitos funcionais são declarações de serviços que o sistema deve oferecer, como ele irá reagir a determinadas situações, como deve se comportar e, em alguns casos, o que esse sistema não deve fazer. Os requisitos não-funcionais, por sua vez, são restrições aos serviços ou funções do sistema como um todo, estando relacionados às propriedades emergentes do mesmo, como confiabilidade, tempo de resposta, desempenho e proteção (SOMMERVILLE, 2011).

2.3 PROCESSO DE SOFTWARE

Processo de software pode ser definido como um conjunto coerente de políticas, estruturas organizacionais, tecnologias, técnicas e artefatos necessários para projetar, desenvolver, implantar e manter um produto de *software* (FUGGETTA, 2000). De acordo com Humphrey (1989), Processo de Software pode ser definido como o conjunto de atividades essenciais para transformar os requisitos do usuário em *software*. De forma simplificada, o processo de software trata-se de um guia que estabelece tarefas, regras, ações e restrições que, caso seja seguido de forma correta, irá produzir, como resultado, um produto de *software*, levando em consideração todos os requisitos do cliente e seus objetivos.

De acordo com Pressman (2016), o processo de software pode ser definido como um *framework* de atividades, tarefas e ações necessárias para a construção de *software* de alta qualidade. Uma definição genérica desse *framework* institui cinco atividades metodológicas básicas:

- Comunicação: antes do trabalho técnico começar, é importante estabelecer uma comunicação com o cliente e os *stakeholders*, para entender seus objetivos e seus requisitos para o projeto;
- Planejamento: é de suma importância realizar o planejamento para o projeto, definindo as tarefas que serão conduzidas, os riscos que poderão ocorrer, os recursos necessários e um cronograma de trabalho;
- Modelagem: para o entendimento e aperfeiçoamento do projeto, faz-se necessário realizar a modelagem do mesmo. Trata-se de um esboço, ou desenho, que será montado para possibilitar a visualização e o entendimento do produto em sua forma final. Se necessário, deve-se refinar esse modelo, detalhando-o cada vez mais e facilitando o entendimento do mesmo;
- Construção: tudo que foi planejado será construído. Essa etapa envolve o desenvolvimento de código e testes;
- Entrega: tudo aquilo que foi criado será entregue ao cliente, que avalia o material e responde com seu *feedback*.

Cada atividade metodológica é constituída por um conjunto de operações de engenharia de software, que por sua vez são definidas pelas tarefas de trabalho que precisam ser concluídas, os artefatos que serão produzidos, os fatores que irão ser requeridos para assegurar a qualidade

e os marcos que serão utilizados para indicar progresso (PRESSMAN, 2016). A Figura 2 representa esquematicamente essa definição.

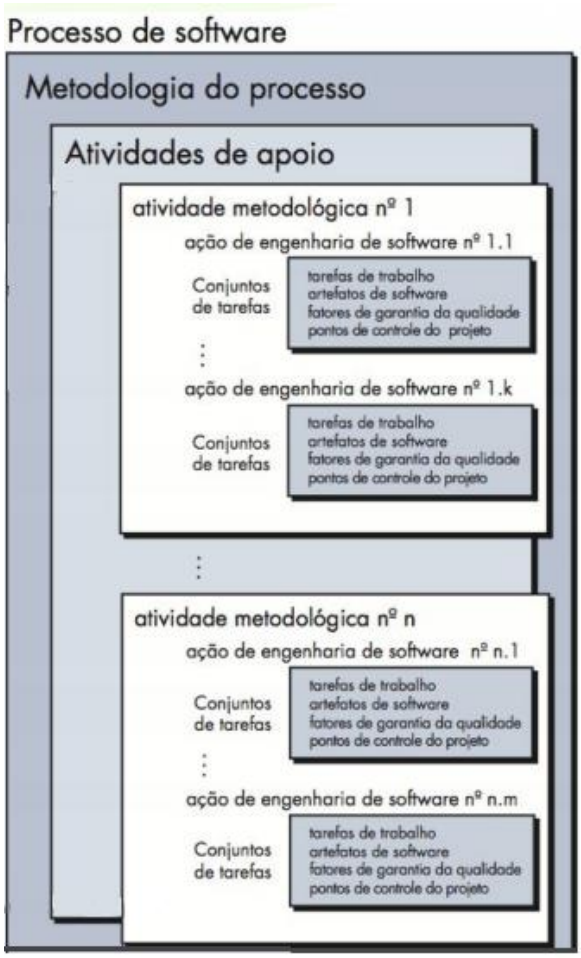


Figura 2. Processo de Software, adaptado de Pressman (2016).

De acordo com Sommerville (2011), descrições do processo e de suas atividades podem também incluir papéis, que indicam as responsabilidades das pessoas envolvidas no processo, e pré e pós-condições, que podem ser definidas antes e depois de uma atividade do processo ou da produção de um produto.

Para definir a relação das atividades metodológicas umas com as outras, é necessário estabelecer um fluxo de processo, que irá configurar a ordem em que essas atividades serão executadas e as condições necessárias para o avanço na execução. Na Figura 3 são apresentados dois exemplos de fluxo de processo: linear e iterativo. No fluxo linear, a execução das atividades é dada de forma sequencial, começando na Comunicação e terminando na Entrega. Já no fluxo iterativo, existem ciclos que repetem uma ou mais atividades, de acordo com condições definidas dentro do processo.

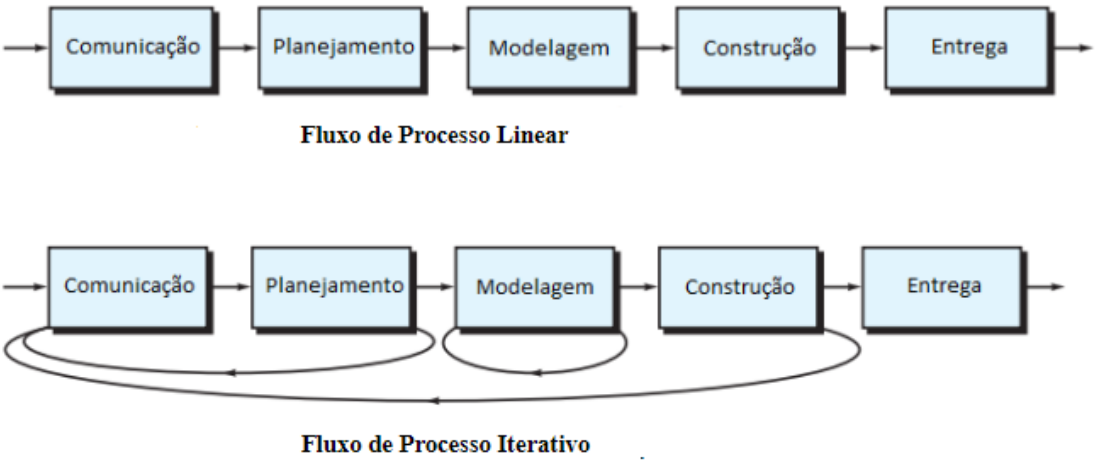


Figura 3. Fluxos de Processo, adaptado de Pressman (2016).

Definir um processo de software, tal qual o fluxo que será utilizado, é uma decisão complexa que envolve vários aspectos de uma organização, entre eles: as pessoas envolvidas, os projetos a serem realizados e os recursos disponíveis para a implementação do processo. Processos de software são complexos, e dependem de pessoas para tomar decisões e realizar julgamentos. Não existe, de fato, um processo ideal para todas as organizações, fazendo com que cada empresa acabe desenvolvendo os seus próprios. Para alguns sistemas críticos, por exemplo, observa-se a necessidade de implementar um processo que seja extremamente bem estruturado, com etapas bem definidas e pouco maleáveis; para sistemas de negócios, em que o cliente esteja indeciso e pode haver muitas modificações nos requisitos, há a necessidade de se implementar um processo mais flexível (SOMMERVILLE, 2011).

De acordo com Falbo (2005), no entanto, existe a possibilidade de estabelecer um conjunto de ativos de processo (subprocessos, atividades, subatividades, artefatos, recursos e procedimentos), que serão utilizados para definir o processo de software de uma organização, constituindo o que é chamado de Processo Padrão. Segundo Humphrey (1989), os principais motivos para padronizar um processo são:

- Reduzir problemas relacionados a treinamento, revisão e utilização de ferramentas;
- Utilizar experiências de processos específicos, com o intuito de melhorar o processo geral;
- Proporcionar uma base para medições de processo e qualidade, a partir dos padrões de processo; e

- A impraticabilidade de criar um novo processo para cada projeto que irá ser realizado.

Diante dos pontos citados, observa-se que a padronização de um processo fornece diversas vantagens para a organização que irá criá-la. É importante ressaltar que estabelecer um processo padrão de desenvolvimento de *software* bem definido não irá, necessariamente, garantir a qualidade do produto desenvolvido. Aumenta, no entanto, a probabilidade de atender, ao final da execução do processo, os requisitos levantados. Além disso, facilita a medição e coleta de dados de execução do processo, dando visibilidade aos gerentes do andamento dos projetos (BERTOLLO, 2006). Há, portanto, a necessidade de criar um processo padrão para a instituição. Para tal, pode ser realizada a Modelagem de Processo.

2.3.1 Modelagem de Processo

A modelagem de processo de negócio, segundo Josuttis (2008), é o conjunto de práticas e atividades que podem ser executadas para descrever visualmente os aspectos de processos de negócios, incluindo seu trajeto, pontos de decisão, condições para a execução das atividades, o contexto em que uma atividade é executada e os recursos associados.

Para realizar a modelagem, pode ser utilizada uma linguagem, ou notação. De acordo com Storolli, Zanolla e Borsoi (2009), linguagens de modelagem são utilizadas para descrever os resultados das etapas do procedimento. As descrições obtidas com essas linguagens cumprem os critérios sintáticos e semânticos dos conceitos notacionais. Esses conceitos compõem o vocabulário básico para exprimir a arquitetura de processo.

Segundo o *Guide to the Business Process Management Body of Knowledge* (BPM CBOK), notação de modelagem de processos é um conjunto padronizado de símbolos e regras que determinam o significado desses símbolos. Uma notação de modelagem de processos contém ícones e conectores utilizados para representar o relacionamento entre componentes do processo de negócio. Para a realização da modelagem do processo da COTIC-PROEG foi utilizada a notação BPMN.

2.3.2 *Business Process Management Notation* (BMPN)

Desenvolvido pelo *Business Process Modeling Initiative* (BPMI) e, atualmente, mantida pelo *Object Management Group* (OMG) após a união das duas organizações em 2005, o BPMN é uma notação para modelagem de processos. Em março de 2011 foi lançada sua versão 2.0.

O BPMN é uma notação de modelagem de processos que utiliza um conjunto de ícones padrões que são utilizados para realizar a modelagem do processo de forma que o usuário entenda facilmente (PIZZA, 2012). Ainda no que diz respeito a Pizza (2012), o BPMN é utilizado para modelar um processo em seu cenário atual, chamado de *As Is*, que irá ser analisado e discutido para, posteriormente, ser realizada a modelagem de uma versão melhorada desse processo, ou seja, o cenário considerado ideal, chamado de *To Be*.

Para a modelagem do processo da empresa ser realizado de forma simples, com elementos gráficos de fácil entendimento, é criado um *Business Process Diagram* (BPD). De acordo com a OMG (2011), um BPD é constituído de três elementos principais: Eventos; Atividades; e *Gateway*. Um evento é representado por um círculo, que pode ser dividido em três tipos, como visto na Figura 4:

- Evento Inicial: primeiro elemento de qualquer processo, onde o fluxo é iniciado;
- Evento Intermediário: acontece no meio do fluxo de processo, possuindo variações para cada ocasião; e
- Evento Final: último elemento de qualquer processo, onde o fluxo é finalizado.

Figura 4. Eventos do BPMN, adaptado de OMG (2011).

Atividades são representadas por retângulos, e podem ser divididas em dois tipos: atividade e subprocesso, como pode ser visto na Figura 5. Atividade é uma tarefa a ser realizada no processo, contendo seus requisitos para seguir no fluxo. Já subprocesso, é uma atividade que é composta por várias outras atividades que necessitam ser executadas. Ou seja, é uma espécie de processo menor, que ocorre dentro do processo geral.

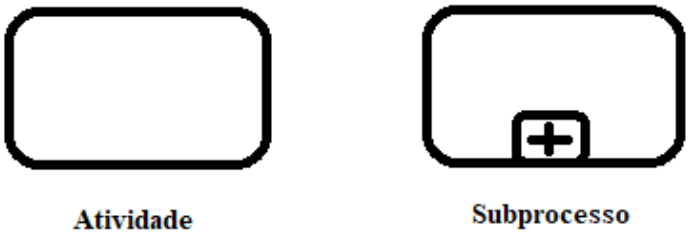


Figura 5. Atividades do BPMN, adaptado de OMG (2011).

Gateways são representadas por losângos, e podem ser divididos em diversos tipos, pois servem para expressar uma condição que deverá ocorrer para a continuação do fluxo, podendo também oferecer outros caminhos do fluxo dependendo da condição verificada. Os mais importantes para a modelagem do processo da COTIC-PROEG, no entanto, são dois: exclusivo e paralelo, como visto na Figura 6. O *gateway* exclusivo é utilizado para verificar se uma condição está sendo atendida. Caso esteja, o fluxo continuará normalmente, e caso contrário, poderá ser oferecido outro caminho a ser seguido no fluxo do processo. Já no *gateway* paralelo, há separação do fluxo em dois ou mais caminhos, que serão executados paralelamente e deverão finalizar juntos.

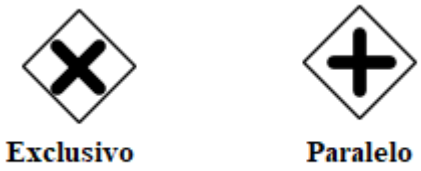


Figura 6. *Gateways* do BPMN, adaptado de OMG (2011).

Para realizar a modelagem do processo da COTIC-PROEG foram utilizados os seguintes elementos, como visto na Figura 7, além dos três principais:

- Fluxo de Sequência: utilizado para identificar a ordem de execução das atividades;
- *Pool*: utilizado para caracterizar, graficamente, um participante na colaboração de um processo;
- Associação: utilizado para associar elementos às tarefas, como artefatos ou papéis responsáveis por realizá-la; e
- Objeto de dados: utilizado para representar quaisquer dados que a tarefa irá produzir ou terá como insumo.

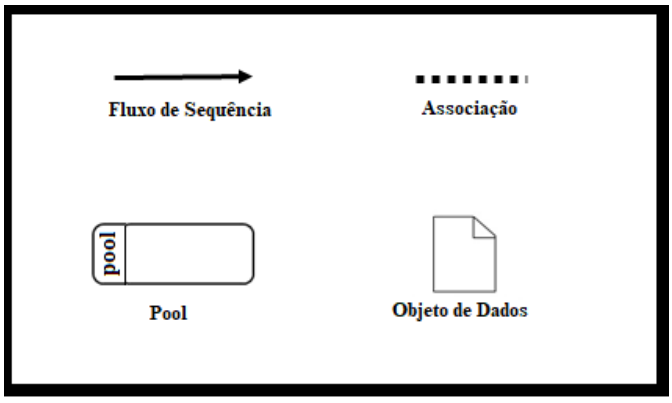


Figura 7. Outros elementos do BPMN, adaptado de OMG (2011).

2.4 MELHORIA DE PROCESSO DE SOFTWARE

As mudanças que estão ocorrendo nos ambientes de negócios têm motivado as empresas a modificar estruturas organizacionais e processos produtivos, saindo da visão tradicional baseada em áreas funcionais em direção a redes de processos centrados no cliente. A competitividade vem dependendo cada vez mais das conexões estabelecidas nestas redes, criando ligações essenciais nas cadeias produtivas. Alcançar competitividade pela qualidade resulta na melhoria da qualidade dos produtos de *software* e dos processos de produção e distribuição do mesmo (SOFTEX, 2016a).

Segundo Oliveira (2007), à medida que aumenta a preocupação com a qualidade de software, há também maior procura das organizações por seguirem orientações de modelos e padrões de qualidade de processo. Com o surgimento de diversos padrões internacionais, as organizações passaram a adotar os referidos modelos para definir seus processos de software, buscando melhorar a qualidade de seu produto, obter maior produtividade de sua equipe, e reduzir riscos e custos referentes ao desenvolvimento de *software*.

Entre os modelos de melhoria de processo de software são exemplos bem conhecidos: CMMI (*Capability Maturity Model Integration* ou Modelo Integrado de Maturidade e Capacidade); ISO/IEC 12207; e MPS.Br (Melhoria do Processo de Software Brasileiro). A seguir, será apresentado o modelo de melhoria de processo de software brasileiro MPS.Br (SOFTEX, 2016a), que foi utilizado como base para a melhoria do processo da COTIC-PROEG.

2.5 MPS.BR

De acordo com SOFTEX (2016a), o programa MPS.Br (Melhoria do Processo de Software Brasileiro) é um programa mobilizador, de longo prazo, criado em dezembro de 2003, coordenado pela Associação para Promoção da Excelência do Software Brasileiro (SOFTEX), com apoio do Ministério da Ciência, Tecnologia e Inovação (MCTI), Financiadora de Estudos e Projetos (FINEP), Serviço Brasileiro de Apoio às Micro e Pequenas Empresas (SEBRAE) e Banco Interamericano de Desenvolvimento (BID/FUMIN). O objetivo do MPS.Br é aumentar a competitividade no mercado de *software*, guiando organizações a atingirem qualidade a partir da melhoria de processos.

Ainda no que diz respeito a SOFTEX (2016a), o MPS possui cinco componentes: Modelo de Referência MPS para Software (MR-MPS-SW); Modelo de Referência MPS para

Serviço (MR-MPS-SV); Modelo de Referência MPS para Gestão de Pessoas (MR-MPS-RH); Método de Avaliação (MA-MPS); e Modelo de Negócio (MN-MPS). Neste trabalho, será abordado o Modelo de Referência MPS para Software (MR-MPS-SW), pois trata-se de um modelo com foco na implementação da melhoria de processo de software em organizações. A Figura 8 representa de forma esquematizada a estrutura atual do modelo MPS, assim como suas bases técnicas.

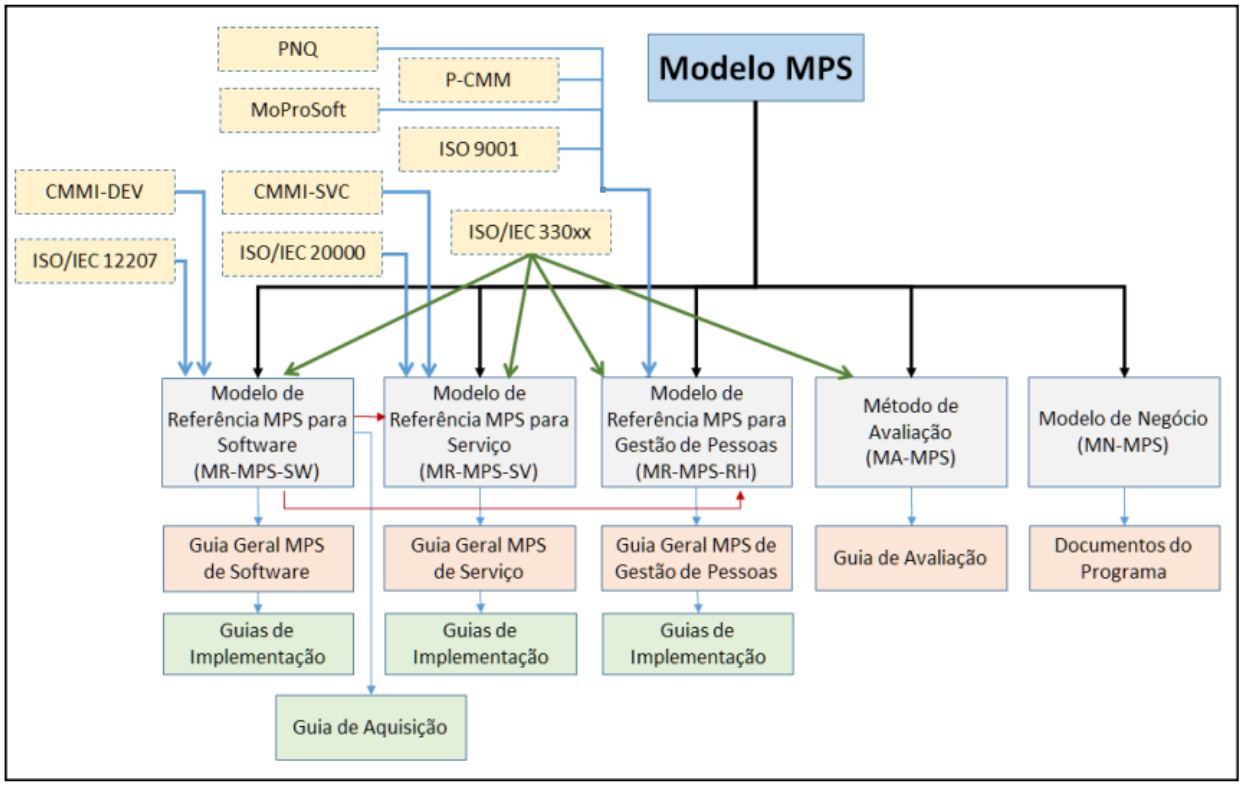


Figura 8. Componentes do MPS (SOFTEX, 2016a).

Como pode ser visto na Figura 8, o Modelo de Referência MPS para Software (MR-MPS-SW) tem como base técnica a NBR ISO/IEC 12207 (ISO/IEC, 2008) e o CMMI-DEV (SEI, 2010). Assim como os outros modelos do MPS, o MR-MPS-SW possui um guia para facilitar a utilização do mesmo, chamado Guia de Implementação. Esse guia descreve sugestões de implementação para cada nível dos modelos MPS, porém não constituindo requisitos do modelo, sendo consideradas apenas informativas (SOFTEX, 2016a).

De acordo com a SOFTEX (2016a), a capacidade do processo expressa o grau de refinamento e institucionalização com que o processo é executado na organização, sendo representada por um grupo de atributos de processo (AP). À medida que a organização evolui nos níveis de maturidade, um maior nível de capacidade para desempenhar o processo deve ser atingido.

Existem nove atributos de processo, cujas descrições detalhadas podem ser encontradas no Guia Geral de Software (SOFTEX, 2016a), sendo eles:

- AP 1.1: O processo é executado;
- AP 2.1: A execução do processo é gerenciada;
- AP 2.2: Os produtos de trabalho do processo são gerenciados;
- AP 3.1: O processo é definido;
- AP 3.2: O processo está implementado;
- AP 4.1: O processo é objeto de análise quantitativa;
- AP 4.2: O processo é controlado quantitativamente;
- AP 5.1: O processo é objeto de melhorias incrementais e inovações; e
- AP 5.2: O processo é objeto de implementação de melhorias inovadoras e incrementais.

Os atributos de processo AP 4.1, AP 4.2, AP 5.1 e AP 5.2 devem ser implementados apenas para processos críticos da organização, selecionados para análise de desempenho. Os demais AP devem ser implementados para todos os processos (SOFTEX, 2016a).

Ainda no que diz respeito a SOFTEX (2016a), no MR-MPS-SW são definidos níveis de maturidade que, por sua vez, são uma combinação entre processos e sua capacidade. A definição dos processos é realizada declarando o seu propósito e os devidos resultados esperados, permitindo avaliar e atribuir graus de efetividade na execução dos processos. Entretanto, o guia de implementação não contém as atividades necessárias para o atendimento dos resultados esperados, ficando a cargo da empresa.

Existem, ao todo, sete níveis de maturidade definidos no MR-MPS-SW, começando no nível G e finalizando no nível A. Esses níveis requerem o atendimento dos atributos de processo, embora não seja detalhado dentro de cada processo, e são acumulativos, ou seja, se uma empresa possui o nível A, por exemplo, ela também possui os processos de todos os outros níveis inferiores implementados, os quais serão executados de acordo com o grau de capacidade exigido no nível atual (SOFTEX, 2016a). A obtenção de um nível de maturidade ocorre quando todos os resultados, propósitos dos processos e dos atributos de processos relacionados ao nível que está sendo implementado e aos anteriores são atendidos (BERTOLLO, 2006). O Quadro 1 apresenta cada nível de maturidade, seus processos e os atributos de processo correspondentes.

NÍVEL	PROCESSOS	ATRIBUTOS DE PROCESSO
A		AP 1.1, AP 2.1, AP 2.2, AP 3.1, AP 3.2, AP 4.1, AP 4.2, AP 5.1 e AP 5.2
B	Gerência de Projetos – GPR (evolução)	AP 1.1, AP 2.1, AP 2.2, AP 3.1 e AP 3.2, AP 4.1 e AP 4.2
C	Gerência de Riscos – GRI Desenvolvimento para Reutilização – DRU Gerência de Decisões – GDE	AP 1.1, AP 2.1, AP 2.2, AP 3.1 e AP 3.2
D	Verificação – VER Validação – VAL Projeto e Construção do Produto – PCP Integração do Produto – ITP Desenvolvimento de Requisitos – DRE	AP 1.1, AP 2.1, AP 2.2, AP 3.1 e AP 3.2
E	Gerência de Projetos – GPR (evolução) Gerência de Reutilização – GRU Gerência de Recursos Humanos – GRH Definição do Processo Organizacional – DFP Avaliação e Melhoria do Processo Organizacional – AMP	AP 1.1, AP 2.1, AP 2.2, AP 3.1 e AP 3.2
F	Medição – MED Garantia da Qualidade – GQA Gerência de Portfólio de Projetos – GPP Gerência de Configuração – GCO Aquisição – AQU	AP 1.1, AP 2.1 e AP 2.2
G	Gerência de Requisitos – GRE Gerência de Projetos – GPR	AP 1.1 e AP 2.1

Quadro 1. Níveis de Maturidade e Processos do MR-MPS-SW (SOFTEX, 2016a).

No MR-MPS-SW, existem dois processos que envolvem requisitos: Gerência de Requisitos (GRE), presente no nível de maturidade G, e Desenvolvimento de Requisitos (DRE), presente no nível D. Este trabalho tem como foco somente o processo Desenvolvimento de Requisitos (DRE). No entanto, já existe um trabalho de conclusão de curso desenvolvido no contexto da COTIC, apresentado no período 2017.4, na UFPA, cujo objetivo é atender os resultados esperados do GRE, e pode ser encontrado em (MENEZES, 2017).

O nível de maturidade D é o quarto nível do MR-MPS-SW. Como consta no Quadro 1, esse nível é composto por cinco processos: Verificação (VER); Validação (VAL); Projeto e Construção do Produto (PCP); Integração do Produto (ITP); e Desenvolvimento de Requisitos (DRE). Além disso, é também composto por cinco atributos de processo: AP 1.1; AP 2.1; AP 2.2; AP 3.1; e AP 3.2.

2.5.1 Desenvolvimento de Requisitos (DRE)

De acordo com a SOFTEX (2016b), "o propósito do processo Desenvolvimento de Requisitos é definir os requisitos do cliente, do produto e dos componentes do produto". Inicialmente, as necessidades, expectativas, restrições e interfaces do cliente são levantadas para, em sequência, serem transformadas em requisitos do cliente. Posteriormente, os requisitos do cliente serão refinados e descritos em termos técnicos, caracterizando os requisitos funcionais e não-funcionais do produto. Além disso, deve ser elaborada uma definição desses requisitos, assim como conceitos operacionais e cenários, detalhados de forma que permita a criação de projetos (*design)* técnicos e a construir a solução do *software* para resolver o problema em questão. Os requisitos devem, também, ser analisados, validados e gerenciados ao longo do ciclo de desenvolvimento ou de manutenção do produto (SOFTEX, 2016b).

Ao todo, o processo DRE do MR-MPS-SW descreve oito resultados esperados, que estão detalhados a seguir.

2.5.1.1 DRE1

No primeiro resultado esperado, DRE1, as necessidades, expectativas e restrições do cliente, tanto do produto quanto de suas interfaces, são identificadas. O alcance deste resultado esperado envolve a utilização de métodos adequados para identificar necessidades, expectativas, restrições e interfaces do cliente. Deve-se buscar o envolvimento de representantes do cliente e utilizar técnicas de elicitação de requisitos para identificar de forma proativa requisitos

adicionais não discutidos explicitamente pelos clientes (SOFTEX, 2016b). De acordo com Boria (2013), para esse resultado esperado ser atendido deve ser seguido um processo sistemático para identificar o que o cliente necessita, para que consiga obter o que deseja, de acordo com o que expressa.

2.5.1.2 DRE2

No segundo resultado esperado, DRE2, um conjunto definido de requisitos do cliente é especificado e priorizado a partir das necessidades, expectativas e restrições identificadas. As necessidades, expectativas e restrições do cliente identificadas anteriormente são traduzidas em requisitos do cliente. Para que isso ocorra, pode ser necessária a resolução de conflitos entre os fornecedores de requisitos e demais envolvidos no projeto relacionados à especificação de requisitos. Além disso, podem surgir questões relevantes a serem verificadas e/ou validadas. A priorização dos requisitos auxilia na determinação do escopo do projeto, iteração ou incremento, e garante que os requisitos funcionais e não funcionais que sejam críticos sejam tratados mais rapidamente (SOFTEX, 2016b). De acordo com Boria (2013), para o atendimento do DRE2 deve ser construído um documento que permita priorizar os requisitos e especificá-los, tornando possível sua revisão e análise.

2.5.1.3 DRE3

No que tange o terceiro resultado esperado, DRE3, um conjunto de requisitos funcionais e não-funcionais, do produto e dos componentes do produto, que descrevem a solução do problema a ser resolvido, é definido e mantido a partir dos requisitos do cliente. De acordo com a SOFTEX (2016b), o alcance deste resultado esperado compreende a consolidação das necessidades, expectativas e restrições do cliente em um conjunto de requisitos funcionais e não-funcionais do produto e dos componentes do produto. Ao especificar os requisitos funcionais e não-funcionais é possível perceber falta de informações, inconsistências e erros. Nessas situações, é necessário buscar informações complementares e resolver as inconsistências detectadas. A especificação dos requisitos deve permitir identificar funções, assim como os atributos de qualidades não-funcionais do produto (BORIA, 2013).

2.5.1.4 DRE4

De acordo com o DRE4, quarto resultado esperado do DRE, os requisitos funcionais e não-funcionais de cada componente do produto são refinados, elaborados e alocados. Segundo a SOFTEX (2016b), alcançar este resultado esperado significa elaborar os requisitos funcionais

e não-funcionais de cada componente do produto nos termos técnicos necessários para o desenvolvimento do produto e dos componentes do produto. Como indicador de alcance deste resultado, deve-se evidenciar que o conjunto de requisitos funcionais e não-funcionais foi refinado, detalhado e documentado ao longo do ciclo de vida para o desenvolvimento do produto e dos componentes do produto. De acordo com Boria (2013), para alcançar o atendimento ao DRE4 devem ser definidos os detalhes das funcionalidades para serem incluídos no desenvolvimento.

2.5.1.5 DRE5

No que tange o quinto resultado esperado do DRE, chamado de DRE5, interfaces internas e externas do produto e de cada componente do produto são definidas. As interfaces internas e externas do produto e de cada componente do produto devem ser especificadas e documentadas de acordo com a arquitetura definida do produto. As definições dessas interfaces são úteis para projetar e construir as unidades de código dos componentes do produto, bem como para servir de base para verificar a integração entre cada componente do produto e para verificar a integração do produto com outros elementos externos. As definições das interfaces geralmente são definidas em termos de tipos e formatos de dados de entrada e saída entre os componentes do produto e entre elementos do sistema, especificações de protocolos de comunicação, entre outros (SOFTEX, 2016b). Parte da especificação deve ser dirigida expressamente ao ambiente de funcionamento do produto, sejam interfaces internas ou externas (BORIA, 2013).

2.5.1.6 DRE6

No que diz respeito ao DRE6, sexto resultado esperado, conceitos operacionais e cenários são desenvolvidos. De acordo com a SOFTEX (2016b), O alcance deste resultado esperado exige o desenvolvimento de conceitos operacionais e cenários para o produto e os componentes do produto. O alcance deste resultado esperado pode abranger: definir o ambiente no qual o produto operará, incluindo limites e restrições; elaborar um conceito operacional detalhado para cada produto ou componente do produto que defina a interação do produto, do usuário final, do ambiente e que satisfaça as necessidades de operação, manutenção e apoio; e revisar conceitos operacionais e cenários para refinar e descobrir novos requisitos. Segundo Boria (2013), para o atender o DRE6 é necessária a construção de estórias de usuário, narrativas e/ ou casos de uso que expliquem o uso do produto.

2.5.1.7 DRE7

No sétimo resultado esperado, DRE7, os requisitos são analisados, usando critérios definidos, para balancear as necessidades dos interessados com as restrições existentes. Segundo a SOFTEX (2016b), este resultado visa garantir que os requisitos, em seus diferentes níveis, sejam analisados de forma a balancear as necessidades dos interessados com as restrições de projeto existentes. Os requisitos podem ser analisados juntamente com cenários, conceitos operacionais e definições detalhadas dos requisitos, para determinar se eles são necessários, corretos, testáveis e suficientes para atingir os objetivos e requisitos de alto nível (requisitos do cliente).

2.5.1.8 DRE8

De acordo com o oitavo resultado esperado do DRE, o DRE8, os requisitos são validados. Este resultado esperado visa garantir que os requisitos sejam validados utilizando-se de técnicas adequadas, de forma a garantir que o produto terá o desempenho adequado quando instalado no seu ambiente alvo. A validação aumenta a confiança de que os requisitos definidos são capazes de guiar o desenvolvimento satisfatoriamente. Quanto mais cedo problemas forem identificados, menos retrabalho e custo serão necessários para adequar os requisitos às expectativas do cliente (SOFTEX, 2016b).

2.6 METODOLOGIA ÁGIL

Segundo Sommerville (2011), na década de 1980 e início da de 1990, havia uma visão generalizada de que, para conseguir desenvolver um produto de *software* da melhor forma possível, precisa-se de um planejamento rigoroso e cuidadoso do projeto, utilizando abordagens pesadas de engenharia de software que, por sua vez, eram comuns para sistemas críticos e que possuíam a necessidade de ser extremamente bem planejados e executados, como sistemas governamentais e aeroespaciais. O problema, no entanto, dava-se na execução dessas mesmas abordagens em sistemas de médio e pequeno porte, nos quais observava-se um enorme gasto de tempo na análise e planejamento do projeto, maior até do que o tempo utilizado para seu desenvolvimento, além do fato de que, por não serem sistemas críticos, os requisitos mudavam com muita frequência, ocasionando no retrabalho de planejamento e especificação do sistema.

Insatisfeitos com essa abordagem pesada de engenharia de software, desenvolvedores de *software* propuseram, na década de 1990, novos ‘métodos ágeis’, cujo objetivo era permitir

que a equipe de desenvolvimento focasse no *software* em si, e não na sua concepção e documentação. Com o propósito de ser adequado a projetos cujos requisitos mudam rapidamente durante o processo de desenvolvimento, o método ágil é, universalmente, baseado em uma abordagem incremental para a especificação, desenvolvimento e entrega do *software*, realizando entrega contínua de *software* funcional em curtos períodos de tempo e, logo em seguida, propondo alterações e novos requisitos a serem incluídos nas iterações posteriores do projeto (SOMMERVILLE, 2011).

Nesse contexto, em uma reunião composta por um grupo de dezessete desenvolvedores, em fevereiro de 2001, deu-se origem ao que foi chamado de manifesto ágil. Esse manifesto trata de juntar princípios compartilhados pelas pessoas presentes na reunião, as quais se autodenominaram "*Agile Alliance"* – em tradução livre, "Aliança Ágil" (HIGHSMITH, 2001). Segundo a Agile Alliance (2016), no manifesto ágil podem ser encontrados os seguintes valores:

- Indivíduos e interações mais que processos e ferramentas;
- *Software* funcional mais que documentação abrangente;
- Colaboração com o cliente mais que negociação de contratos; e
- Responder a mudanças mais que seguir um plano.

Além dos valores apresentados, a Agile Alliance também entrega doze princípios que compõem o manifesto ágil:

1. A prioridade é satisfazer o cliente com entrega adiantada e contínua de *software* funcional;
2. Mudanças são bem-vindas, mesmo tardiamente no desenvolvimento;
3. Entregar *software* funcional frequentemente, em curtos períodos de tempo;
4. Pessoas de negócio e desenvolvedores devem trabalhar diariamente em conjunto durante todo o projeto;
5. Construir projetos em torno de indivíduos motivados, fornecendo ambiente e suporte necessário;
6. Preferência por comunicação face a face;
7. *Software* funcional é a medida primária do progresso;
8. Processos ágeis promovem desenvolvimento sustentável;
9. Contínua atenção à excelência técnica e bom design aumenta a agilidade;
10. Manter a simplicidade, maximizando a quantidade de trabalho não feito, é essencial;

11. As melhores arquiteturas, requisitos e designs emergem de equipes auto organizáveis; e
12. Periodicamente, a equipe reflete sobre como se tornar mais eficaz.

Existem diversos métodos e *frameworks* que se baseiam nos valores e princípios contidos no manifesto ágil. Para o escopo deste trabalho, serão abordadas as metodologias *Scrum*, *Kanban* e *Extreme Programming* (XP), pois são a base para o processo de desenvolvimento atual da COTIC-PROEG.

2.6.1 Scrum

O Scrum é uma metodologia de desenvolvimento ágil criada por Jeff Sutherland e sua equipe de desenvolvimento, no início dos anos 1990. Tem seus princípios baseados no manifesto ágil, e os utiliza para orientar atividades de desenvolvimento dentro de um processo que incorpora cinco atividades metodológicas, sendo elas: requisitos, análise, projeto, evolução e entrega. Além disso, possui um conjunto de padrões de processo que provaram sua eficácia para projetos com curtos prazos de entrega e modificação constante de requisitos (PRESSMAN, 2016).

Segundo Pressman (2016), na metodologia de desenvolvimento Scrum, em cada uma das atividades metodológicas anteriormente descritas, existe um padrão de processo chamado iteração, ou *Sprint*. Trata-se de um ciclo realizado em curtos períodos de tempo, geralmente quinze dias, no qual é estabelecido um conjunto de atividades que deverá ser implementado. Os papéis envolvidos nesse processo são: PO (*Product Owner*), que representa o cliente do produto; *Scrum Team*, representando a equipe de desenvolvimento; e *Scrum Master*, que serve como um líder do *Scrum Team*, sendo responsável pro assegurar que as práticas do Scrum estão sendo respeitadas.

No início de cada *Sprint* existe uma reunião chamada de *Sprint Planning Meeting*, ou reunião de planejamento, na qual devem estar presentes o PO, *Scrum Master* e *Scrum Team*. Nessa reunião, o *Scrum Team* negocia com o PO, tendo o *Scrum Master* como facilitador dessa negociação, um conjunto de requisitos presentes no *Product Backlog* (lista de requisitos ou funcionalidades do projeto, priorizados de acordo com o valor comercial que apresenta para o cliente) que serão movidos para o *Sprint Backlog* (lista de requisitos ou funcionalidades do projeto que serão desenvolvidos nessa *Sprint*).

Durante a execução de uma *Sprint* é realizada, diariamente, uma reunião chamada de *Daily Scrum*, onde participam o *Scrum Team* e o *Scrum Master*, com o propósito de relatar tudo

o que foi feito desde a última *Daily Scrum*, e planejar o que será executado no dia seguinte. No final da iteração, há uma reunião com o PO, *Scrum Master* e o *Scrum Team*, chamada de *Sprint Review*, na qual o cliente irá validar as funcionalidades implementadas durante a *Sprint*. Ao final, é realizada a última reunião da *Sprint*, chamada de *Sprint Retrospective*, na qual serão identificados aspectos positivos que ocorreram durante a *Sprint*, quais problemas surgiram e como resolvê-los caso voltem a repercutir, além de rastrear possíveis riscos que poderão ocorrer na próxima *Sprint* e como evitá-los.

2.6.2 Kanban

De acordo com Godoy (2014), o Kanban é um sistema que pode ser representado por um quadro, *software*, ou até mesmo uma folha de papel, onde cartões que representam o trabalho, ou uma tarefa, seguem um fluxo de estágios pré-estabelecidos. Na medida em que a tarefa vai progredindo, os cartões vão mudando de estágio até chegar à conclusão da mesma. É uma forma simples de gestão à vista, que possibilita a rápida verificação da situação do trabalho. Além disso, oferece transparência, pois todas as informações necessárias estão dispostas a todos, permitindo melhor comunicação e maior integração.

O Kanban não impõe regras quanto à quantidade ou natureza das etapas que serão colocadas no quadro, fazendo com que cada organização implemente um quadro adaptado ao seu processo de desenvolvimento. Existe, entretanto, um modelo básico desse quadro, que pode ser visto na Figura 9, onde são apresentas três etapas fundamentais a serem marcadas no progresso e uma tarefa: a fazer, fazendo e feito.

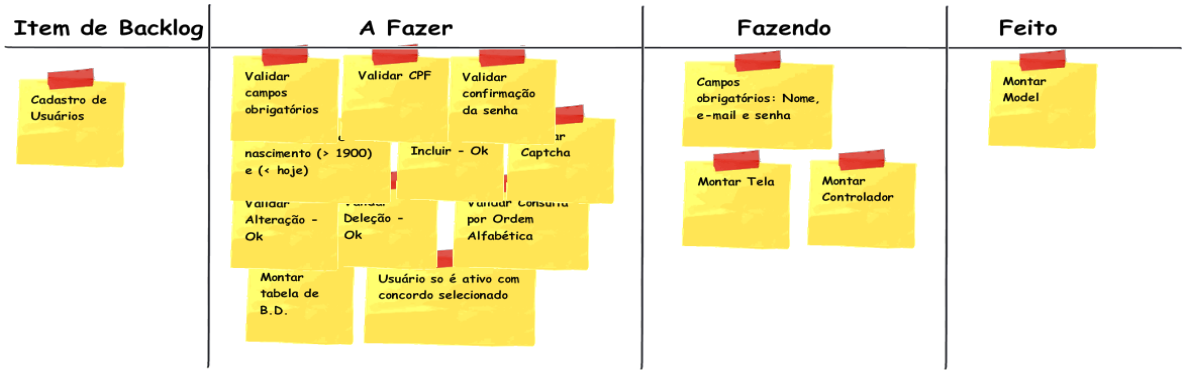


Figura 9. Quadro Kanban (RAHAL JUNIOR, 2011[2])

2.6.3 *Extreme Programming* (XP)

[2] Disponível em: http://blogdoabu.blogspot.com/2010/09/melhorando-o-entendimento-como-fazer.html. Acesso em: out 2018.

De acordo com Pressman (2011), o *Extreme Programming* (XP) é o mais conhecido e utilizado dos métodos ágeis. O nome foi dado por Kent Beck, pois a abordagem foi desenvolvida para impulsionar boas práticas, já conhecidas, ao nível extremo.

No XP os requisitos são expressos como cenários (chamados de *User Stories*, ou Estórias de Usuário), que são implementadas diretamente como um conjunto de tarefas. As *User Stories* que não foram implementadas podem mudar ou serem descartadas, à medida que os requisitos do projeto mudam.

Beck (2004) define um conjunto de valores que formam a base para as atividades do XP, são eles:

- Comunicação: valoriza-se o diálogo presencial para estabelecer comunicação entre as partes interessadas;
- Simplicidade: o time deve-se concentrar no que é realmente necessário e se mostrou essencial;
- *Feedback*: os desenvolvedores devem procurar entregar o quanto antes as funcionalidades, enquanto o cliente deve fornecer todas as informações necessárias para a equipe de desenvolvimento conseguir identificar acertos e desvios do projeto;
- Coragem: projetar e desenvolver o projeto sabendo que irão ocorrer imprevistos e mudanças durante sua execução, e essas mudanças irão exigir retrabalho; e
- Respeito: para que o projeto seja bem sucedido, é essencial o respeito entre os membros do time.

Segundo Medeiros (2013), surge um conjunto de treze práticas, cujo objetivo é reforçar os valores do XP, sendo elas:

- Jogo do planejamento: tem como objetivo definir rapidamente as estimativas, o escopo e as prioridades;
- Testes: serve como parâmetro para verificar o que está de acordo com as especificações e o que não está;
- Programação em pares: trata-se de duas pessoas trabalhando em uma única máquina onde um codifica, e o outro critica ou dá sugestões;
- Projeto simples: quanto mais simples o projeto, menor o custo de modificações;
- Refatoração: melhorar o código sem alterar sua funcionalidade;
- Propriedade coletiva: todos podem modificar o código a qualquer momento;

- Integração contínua: códigos devem ser integrados diariamente e testes devem passar antes e depois da integração;
- Cliente presente: o cliente deve estar presente para definir prioridades e participar de testes de aceitação.
- Semana de 40 horas: manter ritmo sustentável de trabalho;
- Padrões de código: todos devem entender facilmente o código e ter a capacidade de trabalhar nele;
- Metáfora: linguagem comum para facilitar o entendimento de todos; e
- Reunião diária: tem como objetivo discutir o que foi feito no dia anterior, e definir o que será feito no dia atual.

3 RELATO DA MELHORIA

Esta seção tem como objetivo documentar os passos realizados para a modelagem do processo da COTIC-PROEG, a análise realizada com base no DRE do MR-MPS-SW e a modelagem da melhoria do processo.

3.1 CONTEXTO

Para a realização do relato de melhoria deste trabalho foi utilizado como objeto de estudo o processo de desenvolvimento da Coordenadoria de Tecnologia da Informação e Comunicação (COTIC) da Pró-reitoria de Ensino e Graduação (PROEG) da Universidade Federal do Pará (UFPA).

3.1.1 Caracterização da COTIC

A COTIC, anteriormente chamada de AIT (Assessoria de Informação e Tecnologia), foi fundada em meados de agosto de 2009. Passou, posteriormente, a ser uma coordenadoria da PROEG e adotou o nome atual da organização apenas no ano de 2017. O principal objetivo da COTIC é, além da criação de novos sistemas de *software* requisitados pela PROEG, realizar a manutenção, melhoria e suporte aos sistemas já existentes em seu portfólio.

3.1.2 Configuração da Equipe de Trabalho

Atualmente, existem nove pessoas que atuam como colaboradores para a COTIC, sendo dois servidores públicos e sete bolsistas. Os servidores públicos possuem carga horária de trabalho fixa de quarenta horas semanais, enquanto que os bolsistas têm carga horária fixa de vinte horas semanais. Os colaboradores da COTIC não possuem papéis fixos dentro do processo de desenvolvimento. A cada *sprint*, ocorre a rotatividade do papel de *Scrum Master*, e todos os outros membros formam o *Scrum Team*. O papel de *Scrum Master* serve para auxiliar na gerência da *Sprint*, além de facilitar a conversa entre o PO e os outros membros do time. No contexto da organização, todos os membros que compõem o *Scrum Team* podem atuar como desenvolvedores, testadores ou suporte técnico. O principal motivo para não haver uma divisão de papéis dentro do time é estimular a participação e colaboração em todas as atividades do processo de desenvolvimento.

3.1.3 Finalidade do Processo

Para cumprir seus objetivos, foi criado um processo padrão de desenvolvimento próprio da COTIC, com o intuito de otimizar a forma como os produtos requisitados são

desenvolvidos, visando atender as necessidades e expectativas da PROEG, assim como da própria coordenadoria. Atualmente, o processo de desenvolvimento utilizado na COTIC está sendo seguido para a criação de novos sistemas, desde a concepção do novo projeto, seu desenvolvimento, e a entrega de funcionalidades.

O processo da COTIC já sofreu diversas mudanças ao decorrer de sua existência, buscando a utilização de técnicas presentes em várias metodologias ágeis, visando adaptar seu processo de forma a otimizar o desenvolvimento de *software*. Inicialmente, para a criação do processo, foram utilizadas metodologias do Scrum e do XP, e após ser verificada a necessidade, foi introduzido o uso do Kanban nesse processo.

3.2 PROCESSO ÁGIL ANTES DA MELHORIA

3.2.1 Identificação e mapeamento dos processos

Tendo como objetivo identificar as etapas e atividades que compõem o processo de desenvolvimento da COTIC, foram realizadas reuniões com os membros da organização, principalmente com o servidor público responsável pela coordenadoria, visando executar entrevistas acerca do processo e verificar a fidelidade da modelagem realizada. Durante a pesquisa, foram disponibilizados dois trabalhos realizados anteriormente, que envolvem o processo da COTIC. O primeiro, da autoria de Jaily Vanzeler Freitas, tem como objetivo melhorar o processo da empresa utilizando a linguagem SPIDER-ML. O segundo, da autoria de Juan Carlos Barata, também visa a melhoria do processo, utilizando o *software Draw*.[3]

3.2.2 Modelagem do Processo Atual da COTIC-PROEG

Após os procedimentos mencionados na Seção 3.2.1, foi realizada a modelagem das atividades e dos subprocessos que compõem o processo padrão de desenvolvimento utilizado pela COTIC. Foi utilizada a linguagem BPMN para a modelagem a partir da utilização do *software Bizagi Modeler*[4]. Entre os componentes mencionados em 2.3.2, foi utilizado o **Objeto de Dados** para representar os artefatos que são gerados e que servem de insumo para cada atividade. Os componentes **Evento Inicial** e **Evento Final** utilizados no *software* são diferenciados pelas cores verde e vermelho, respectivamente. Além disso, foi adicionado um componente para corresponder aos papéis participantes em cada tarefa, demonstrado pela Figura 10.

[3] https://www.draw.io/
[4] https://www.bizagi.com/en/products/process-modeling

Figura 10. Componente representante dos papéis do processo, retirado de *iconshock*[5].

De acordo com o mapeamento realizado, foram identificados um processo geral de desenvolvimento, chamado de Processo Principal, e três subprocessos que podem ser encontrados dentro do Processo Principal, que são: Planejar *Release*, Executar *Sprint* e Realizar Desenvolvimento. A seguir, esses processos serão descritos de forma detalhada.

3.2.2.1 Processo Principal

O **Processo Principal**, representado pela Figura 11, tem como base o modelo clássico do Scrum, contendo algumas modificações necessárias para adaptar o processo ao contexto da COTIC. Tem seu início com o artefato *Escopo do Produto*, que se trata do escopo inicial do projeto, servindo de insumo para a primeira atividade do processo, chamada **Construir Modelo Abrangente**. Nessa atividade há a participação do *Product Owner* (PO), *Scrum Master* e *Scrum Team*. A Figura 12 mostra as tarefas que constituem uma atividade quando a mesma é selecionada no Bizagi Modeler. Devido ao fato de que o foco desta seção é demonstrar o fluxo do processo da COTIC, não serão mostradas as tarefas que constituem todas as atividades desse processo, entretanto o produto modelado encontra-se no Apêndice A, bem como o detalhamento da execução de cada uma das atividades a partir de suas tarefas.

[5]Disponível em: https://www.iconshock.com/icons-pack/general-icons/user-icon/. Acessado em Outubro de 2018.

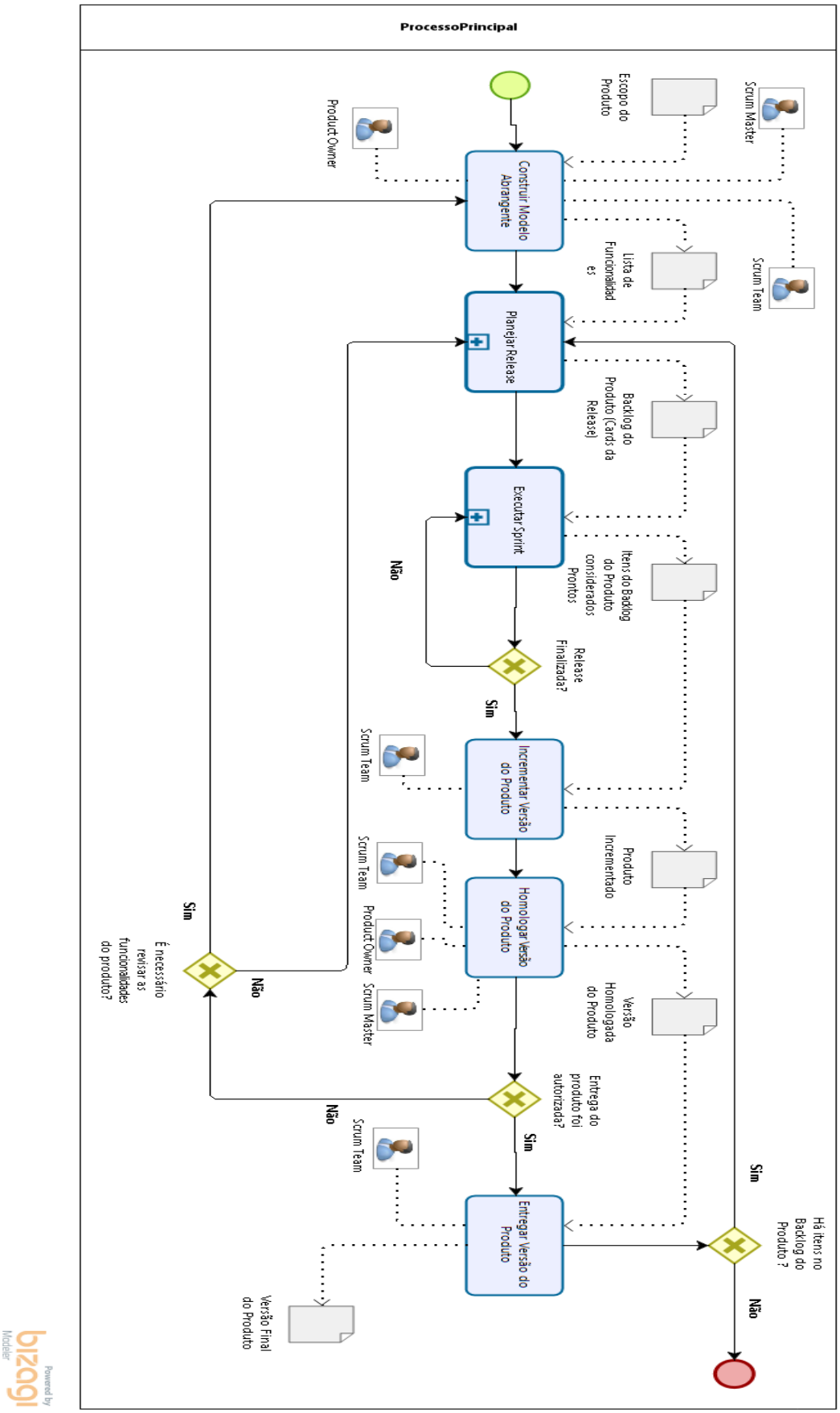

ProcessoPrincipal
Escopo do Produto
Scrum Master
Product Owner
Construir Modelo Abrangente
Lista de Funcionalidades
Scrum Team
Planejar Release
Backlog do Produto (Cards da Release)
Executar Sprint
Itens do Backlog do Produto considerados Prontos
Não
Release Finalizada?
Sim
Incrementar Versão do Produto
Scrum Team
Produto Incrementado
Homologar Versão do Produto
Scrum Team
Product Owner
Scrum Master
Versão Homologada do Produto
Entrega do produto foi autorizada?
Não
Sim
Entregar Versão do Produto
Scrum Team
Versão Final do Produto
Há itens no Backlog do Produto ?
Sim
Não
É necessário revisar as funcionalidades do produto?
Sim
Não

Figura 11. Processo Principal (Autor, 2018).

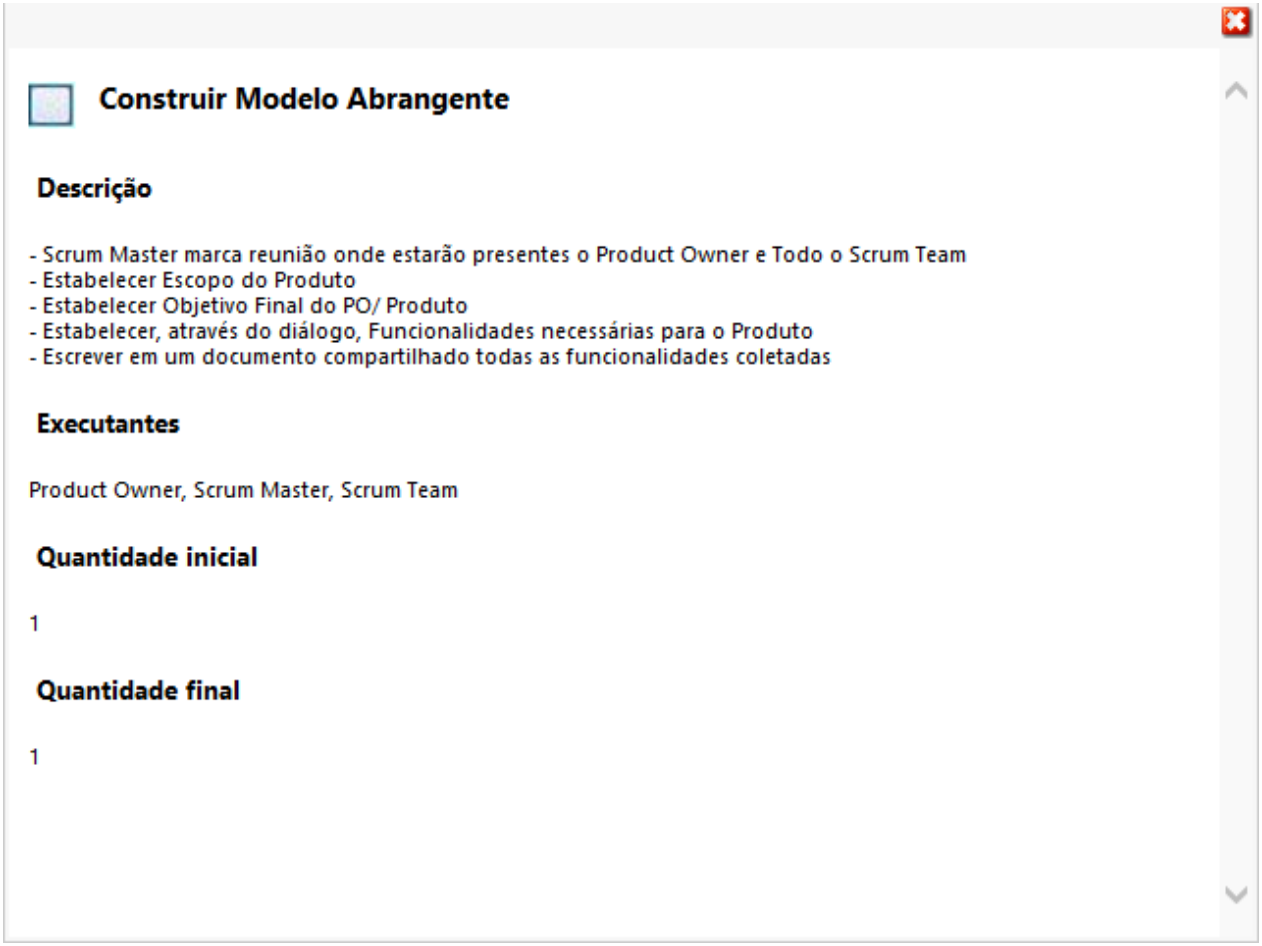


Figura 12. Descrição da atividade "Construir Modelo Abrangente" (Autor, 2018).

Após serem realizadas as tarefas, é gerado o artefato contendo a lista de funcionalidades do produto. Esse artefato servirá de insumo para a próxima atividade do fluxo, chamada **Planejar *Release***, que por ser um subprocesso, será detalhado na Seção 3.2.2.2. Após finalizar esse subprocesso, é gerado o artefato contendo o Backlog do Produto, que servirá de insumo para a próxima atividade, **Executar *Sprint***. Essa atividade também é um subprocesso, e será detalhado na seção 3.2.2.3.

Depois de finalizada a atividade **Executar *Sprint***, é gerado o artefato contendo os itens do Backlog do Produto que foram considerados finalizados. É verificado, então, se a *release* foi finalizada, ou seja, verifica-se, a partir do artefato de insumo, se todos os itens planejados para serem entregues na *release* foram terminados. Caso não, é realizada a atividade **Executar *Sprint*** novamente, e, caso contrário, o fluxo seguirá para a próxima atividade, chamada **Incrementar Versão do Produto**. Para realizar essa atividade o *Scrum Team* adiciona ao produto as funcionalidades que foram desenvolvidas na *release*, gerando como artefato uma versão do produto incrementado.

Finalizando o incremento do produto, o fluxo passa para a atividade **Homologar Versão do Produto**, onde o *Scrum Team* e o *Scrum Master* reúnem-se com o PO para apresentá-lo a nova versão do produto. Após analisar as funcionalidades implementadas, o PO decide se está tudo de acordo com suas necessidades, podendo autorizar ou não a entrega dessa versão do produto. Caso não seja autorizado, verifica-se a necessidade de revisar as funcionalidades do produto. Caso confirmada a necessidade, o fluxo volta à atividade **Construir Modelo Abrangente**, e, caso contrário, o fluxo voltará para a atividade **Planejar *Release***. Caso a entrega da versão do produto seja autorizada, é gerado o artefato contendo a Versão Homologada do Produto, que servirá de insumo para a última atividade do processo, chamada **Entregar Versão do Produto**.

Para realizar a última atividade do **Processo Principal**, o *Scrum Team* irá realizar a entrega da versão homologada do produto para o servidor de produção, enviando um e-mail para todos os envolvidos no projeto acerca da entrega. É gerada, portanto, a versão final do produto, que termina o processo de desenvolvimento da COTIC.

3.2.2.2 Planejar *Release*

Planejar *Release* é um subprocesso encontrado no **Processo Principal** da COTIC. Sendo iniciado após a construção do modelo abrangente, seu objetivo é obter o planejamento da próxima *release* (atividade de entrega do *Scrum*) que será realizada, oferecendo ao *Scrum Master* e ao *Scrum Team* um cronograma do que deverá ser entregue e qual data prevista, como visto na Figura 13. Tendo como insumo a lista de funcionalidades gerada na atividade anterior, o fluxo desse subprocesso tem início na atividade **Escrever *User Stories***, na qual o *Scrum Team* irá escrever em um conjunto de *Cards*, ou cartões, cenários chamados de *User Stories ou* Estórias de Usuário, contendo as informações acerca de quem irá utilizar essa funcionalidade, qual seu objetivo e o motivo da funcionalidade estar presente no produto, além de conter uma descrição mais detalhada de como deve ser realizada a implementação da estória.

Após finalizada a primeira atividade do subprocesso, o fluxo segue para a atividade **Calcular Esforço**, tendo como insumo o conjunto de estórias que foi previamente definido. Nessa atividade, o *Scrum Team* irá calcular, através do *Planning Poker*, o esforço necessário para a implementação de cada estória. Imediatamente após finalizar o cálculo dos esforços é discutida a necessidade de redefinir alguma estória, verificando a existência de estórias com esforço muito grande, caracterizando Épicos. Caso haja a necessidade de fragmentação, o fluxo do processo segue para a atividade **Fragmentar Épicos em Estórias Menores**, tendo como

insumo os cartões já com esforço calculado, onde tem-se a definição de quais épicos serão divididos em duas ou mais estórias. Após a finalização dessa tarefa, os cartões que foram selecionados para serem fragmentados servirão de insumo para a atividade **Reescrever Estórias**. Nessa atividade, cada cartão selecionado será fragmentado em dois ou mais cartões, configurando novas estórias de usuário, que por sua vez tem esforço menor.

Após a finalização da atividade **Reescrever Estórias**, o fluxo do processo volta para **Calcular Esforço**, onde será calculado o esforço necessário para a implementação de cada estória que foi reescrita. Com todos os cartões devidamente calculados faz-se novamente a pergunta: “Existem Estórias Muito Grandes (Épicos)?”. Caso a resposta seja não, tem-se o seguimento do fluxo para a próxima atividade, chamada **Priorizar Estórias**.

Em **Priorizar Estórias** o *Scrum Master* marca uma reunião com ambos PO e *Scrum Team* para ser discutida a priorização das estórias escritas. O PO, então, esclarece suas prioridades e informa quais estórias oferecem-lhe mais valor. Com o valor e o esforço de cada estória definidos, o *Scrum Team* irá criar uma lista contendo todas as estórias, já ordenadas de acordo com a prioridade do PO, chamada de *Backlog* do Produto. Todas as estórias presentes no *Backlog* são representadas no Kanban da COTIC, contendo um *checklist* de aceitação para a estória ser considerada pronta. A partir do *Backlog* do Produto, o fluxo do processo segue para a atividade **Definir Quantidade de *Sprints***. Nessa atividade, o *Scrum Team* e o *Scrum Master* irão definir quantas *Sprints* (períodos de duas semanas) são necessárias para finalizar a *release*, utilizando o esforço das estórias do *Backlog* como base.

Após definida a quantidade de *Sprints*, verifica-se a viabilidade de finalizar a *release* com base na quantidade nas estórias que serão implementadas em cada *Sprint*. Caso não haja viabilidade, o fluxo do processo volta para a atividade **Priorizar Estórias**. Caso contrário, questiona-se sobre a necessidade da criação de uma nova *User Story*. Se for confirmada a necessidade, o fluxo voltará para a atividade **Escrever Estórias de Usuário**, e, caso contrário, o subprocesso **Planejar *Release*** é finalizado, tendo como produto a quantidade de *Sprints* definidas para a *release* e os cartões que serão desenvolvidos em cada uma delas, retornando ao **Processo Principal**.

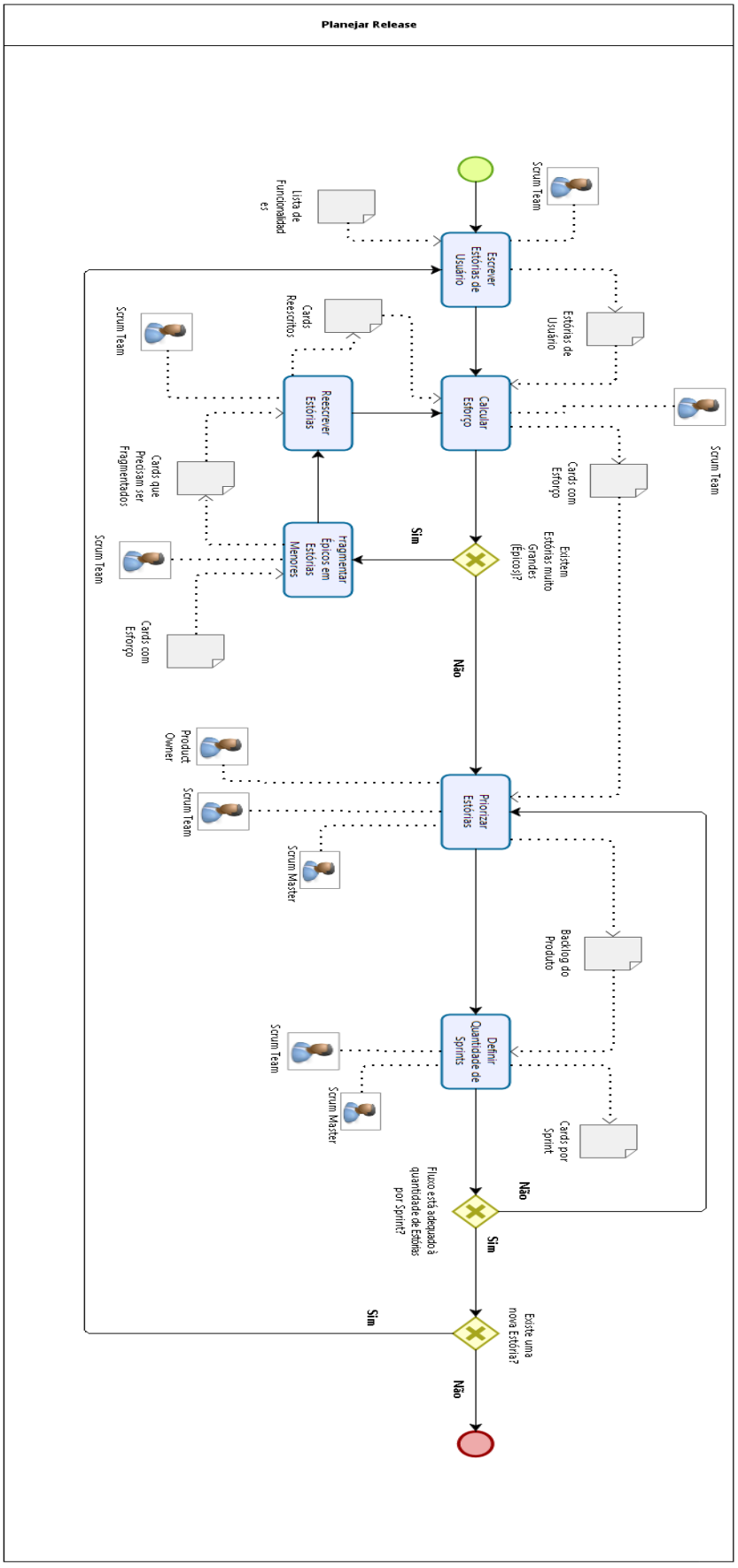


Figura 13. Planejar *Release* (Autor, 2018).

3.2.2.3 Executar *Sprint*

Executar *Sprint* é um subprocesso do fluxo do **Processo Principal**, que se inicia logo após a finalização do planejamento da *release*, e tem como objetivo executar a iteração do Scrum chamada *Sprint*, realizada na COTIC em um período de duas semanas, como visto na Figura 14. Tendo como insumo o *Backlog* do Produto já com os cartões priorizados, quais *cards* desse *Backlog* já foram finalizados e, em caso de já ter sido realizada alguma *Sprint* anteriormente, a lista de melhorias que foram planejadas ao final da *sprint* anterior, dá-se início a primeira atividade do fluxo, chamada de **Planejar Sprint**. Nessa atividade, o *Scrum Master* e o *Scrum Team* reúnem-se para definir o *Backlog* da *Sprint*, ou seja, quais estórias deverão ser desenvolvidas até o final dessa *Sprint*.

Finalizada a primeira atividade do fluxo, o *Backlog* da *Sprint* está definido, e serve como insumo para a próxima atividade do fluxo, chamada **Realizar Desenvolvimento do produto**, o qual, por se tratar de um subprocesso, será detalhado na Seção 3.2.2.4. Finalizado o Desenvolvimento do Produto, obtém-se uma lista de itens que foram desenvolvidos pelo *Scrum Team*, que será utilizada na próxima atividade do fluxo, chamada de ***Sprint Review***, que se trata de uma das etapas do Scrum. Nessa atividade, que acontece no último dia da *Sprint*, o *Scrum Master* e o *Scrum Team* verificam o progresso atual das tarefas planejadas para essa *Sprint*, atualizam o *Backlog* do Produto e marcam uma reunião com o PO para mostrá-lo do progresso realizado, o qual irá, então, dar seu parecer acerca desse progresso e das funcionalidades desenvolvidas.

Após a finalização da ***Sprint Review***, o fluxo do subprocesso segue para sua última atividade, chamada **Realizar Reunião de Retrospectiva da *Sprint***. Tendo como insumo o *Backlog* do Produto atualizado, o *Scrum Master* e o *Scrum Team* reúnem-se para discutir tudo que aconteceu durante a *Sprint*. São levantados os pontos positivos e negativos que ocorreram durante o período de duas semanas, além de verificar o desempenho apresentado para desenvolver as estórias planejadas. Ao final da retrospectiva, são apontadas possíveis melhorias para a próxima *Sprint*.

Terminada a atividade, são produzidos quatro artefatos que serão utilizados para o planejamento da próxima *Sprint*, que são: *Feedback* do Scrum *Team*, *Feedback* do Scrum Master, Avaliação do desempenho do Scrum *Team* e Melhorias para a próxima *Sprint*. Finaliza-se, portanto, a *Sprint* atual, sinalizando o final do subprocesso **Executar *Sprint***, voltando ao **Processo Principal**.

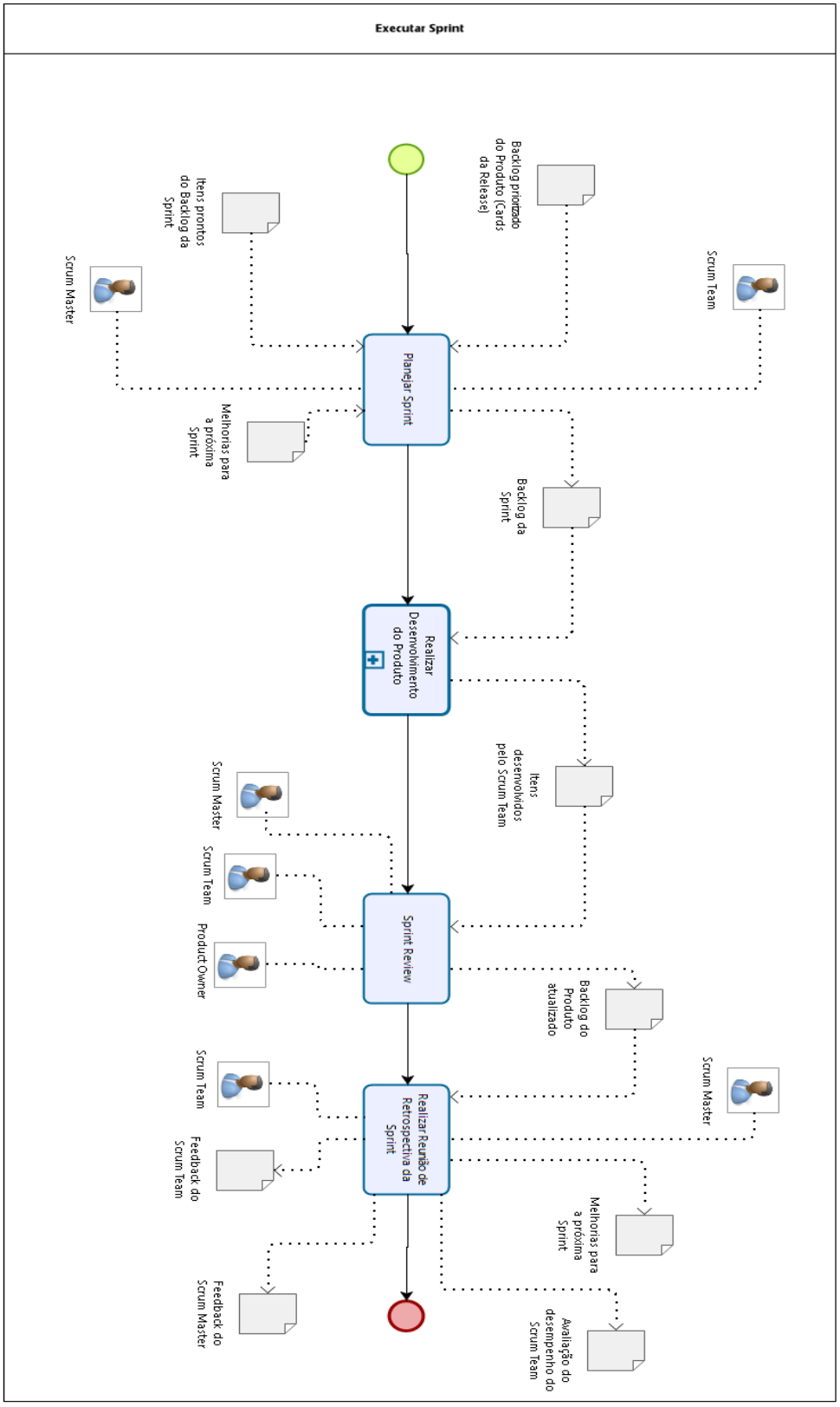


Figura 14. Executar *Sprint* (Autor, 2018).

3.2.2.4 Realizar Desenvolvimento do Produto

O subprocesso **Realizar Desenvolvimento do Produto**, representado pela Figura 15, tem início após a finalização da atividade **Planejar *Sprint***, presente no subprocesso **Executar *Sprint***.

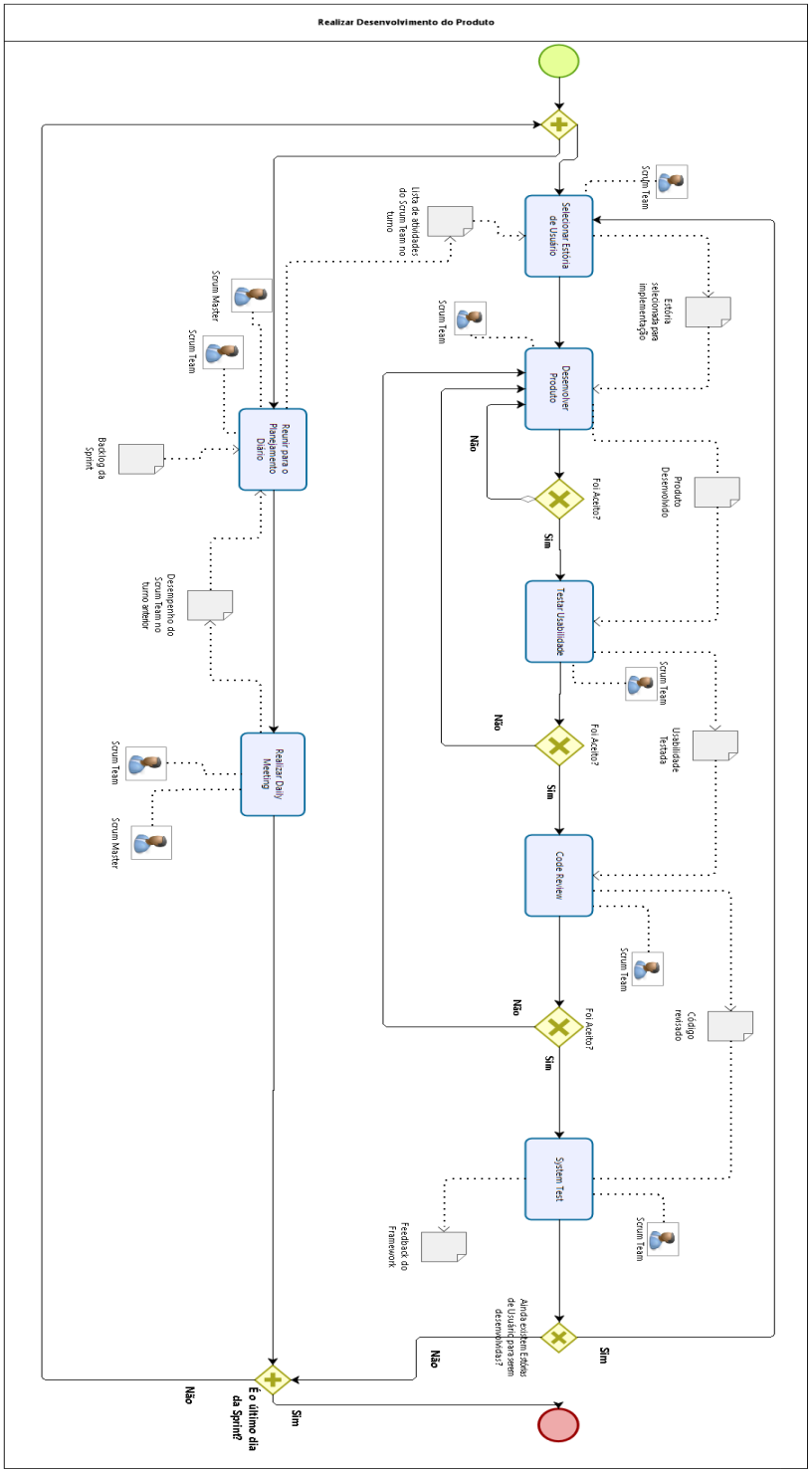


Figura 15. Realizar Desenvolvimento do Produto (Autor, 2018).

Ao iniciar esse subprocesso, o mesmo é dividido em dois fluxos que acontecem em paralelo. Um dos fluxos é referente às atividades que acontecem diariamente, que são: **Reunir para o Planejamento Diário** e **Realizar *Daily Meeting***. Ambas as atividades estão presentes em metodologias ágeis acontecendo apenas uma vez ao dia, entretanto, devido ao fato de a COTIC ser composta por uma maioria de membros bolsistas que trabalham apenas meio período, ou seja, apenas no turno matutino ou vespertino, essas atividades foram adaptadas para acontecer uma vez a cada turno.

O *Daily Meeting* é uma atividade presente no Scrum, que acontece ao final de ambos os turnos de trabalho da COTIC e tem como objetivo anotar tudo que foi realizado durante o turno, pendências para o próximo e sugestões, gerando o artefato de desempenho do *Scrum Team*. Os participantes dessa atividade são ambos *Scrum Master* e *Scrum Team*.

A reunião de planejamento diário é uma das atividades presentes na metodologia XP, acontecendo no início de ambos os turnos matutino e vespertino. Essa reunião, que conta com a participação de ambos *Scrum Team* e *Scrum Master*, tem como objetivo definir o que será realizado no atual turno de trabalho, tendo como insumo o *Backlog* da *Sprint* e, caso alguma *Daily Meeting* tenha sido realizada, o Desempenho do *Scrum Team* no dia anterior. Após finalizada a reunião, é gerada a lista de atividades do turno.

Em paralelo a esse primeiro fluxo existe outro fluxo referente ao desenvolvimento de estórias de usuário. Esse fluxo inicia-se com a atividade **Selecionar Estória de Usuário**, que por sua vez tem a lista de atividades do *Scrum Team* no turno como insumo. Para realizar essa atividade o *Scrum Team* organiza-se para escolher as estórias que serão desenvolvidas no turno e quais pessoas estarão responsáveis pelo desenvolvimento.

Após a atividade de seleção de estória de usuário, o fluxo atual segue para a atividade que representa a etapa em que a estória selecionada encontra-se. As atividades seguintes, portanto, representam as etapas do Kanban da COTIC, para o qual foi utilizada a plataforma *Trello*[6], pois a mesma permite a criação de *checklists* e anexos à estória, facilitando a organização do quadro. A primeira etapa do Kanban é representada pela atividade **Desenvolver Produto**. Nessa atividade, membros do *Scrum Team* realizam a programação em par e, de acordo com o padrão de código, atividades presentes na metodologia XP para desenvolver a estória de usuário. Caso o desenvolvimento esteja finalizado, são construídos testes unitários para o mesmo. Após a finalização dessa atividade, é realizada a verificação se todos os itens do *checklist* da

[6] Disponível em: https://trello.com/. Acessado em novembro de 2018.

estória foram concluídos. Caso seja confirmado, o fluxo segue para a próxima atividade, gerando o artefato Produto Desenvolvido, e, caso contrário, o fluxo permanece na atividade de desenvolvimento.

A próxima atividade do fluxo é chamada **Testar Usabilidade**, no qual os membros do *Scrum Team* que não realizaram o desenvolvimento do produto irão verificar se tudo aquilo que foi desenvolvido está de acordo com o *checklist* da estória, e se as funcionalidades implementadas comportam-se devidamente diante de cenários que simulam erros comuns de usuário. Caso seja verificado, o fluxo segue para a próxima atividade, gerando o artefato chamado Código Validado. Caso contrário, o fluxo voltará para a atividade **Desenvolver Produto**. A partir do código validado, a próxima atividade do fluxo é o ***Code Review***, em que o *Scrum Team* irá verificar se o código, que passou no teste anterior, está de acordo com o padrão estabelecido pela COTIC. Caso seja verificado, tem-se o seguimento do fluxo, gerando o Código Revisado como artefato, e caso contrário, há o retorno à atividade **Desenvolver Produto**.

A última atividade desse fluxo é o ***System Test***, na qual o *Scrum Team* irá utilizar um *software* de teste automatizado para adicionar o cenário da estória, que acabou de ser desenvolvida, ao teste geral do sistema. Trata-se, portanto, de um teste de integração automatizado, no qual o *software* utilizado irá retornar o *feedback* se a nova estória foi devidamente implementada no sistema, ou se as novas funcionalidades comprometeram outras partes do mesmo. Caso o *feedback* seja negativo, o fluxo retorna à atividade **Desenvolver Produto**, e, caso contrário, verifica-se a existência de outras estórias para serem implementadas. Caso hajam outras estórias, o fluxo retorna à atividade **Selecionar Estória de Usuário**, e, caso contrário, os fluxos paralelos juntam-se novamente.

O subprocesso **Realizar Desenvolvimento do Produto** continua repetindo-se até que se chegue ao último dia de *Sprint*. Ao chegar do último dia, finaliza-se esse subprocesso dando continuidade ao subprocesso **Executar *Sprint***, com a atividade ***Sprint Review***.

3.3 AVALIAÇÃO DO PROCESSO

Após a modelagem do processo ágil da COTIC, foi realizada a avaliação do mesmo, utilizando como base os resultados esperados do processo de DRE do MR-MPS-SW, detalhados na Seção 2.5. Para tal, foram utilizadas as tarefas que compõem cada atividade do processo como evidências do atendimento desses resultados esperados, de forma que qualquer atividade presente no processo que represente o atendimento de determinado resultado esperado configure uma evidência daquele resultado dentro do processo.

Após coletar as tarefas caracterizadas como evidência, foi verificado o grau de atendimento de cada resultado esperado, podendo ser: total, caso haja atendimento completo do mesmo; ou parcial, caso haja evidências do atendimento, porém não suficientes para o atendimento ser completo. No caso de não haver tarefas que evidenciem o resultado esperado, o mesmo foi caracterizado como não atendido. Os resultados dessa análise podem ser visualizados no Quadro 2.

RESULTADO ESPERADO	**EVIDÊNCIAS**	**GRAU DE ATENDIMENTO**
DRE1	- Reunião com o P.O - Diálogo acerca do escopo do produto	Parcial
DRE2	- Escrever Estórias de Usuário - Calcular Esforço das Estórias - Fragmentar Épicos em Estórias Menores - Reescrever Estórias de Usuário - Definir Estórias Prioritárias (Criar *Backlog* do Produto)	Total
DRE3	- Escrever *checklist* de aceitação no cartão da Estória	Total
DRE4	Não há evidências	Não Atende
DRE5	Não há evidências	Não Atende
DRE6	Não há evidências	Não Atende
DRE 7	- Teste de Usabilidade - Testes Unitários - System Test (*Smoke Test*) - *Checklist* de tarefas finalizadas da Estória	Total
DRE8	- *Sprint Review* - Homologação da Versão do Produto	Parcial

Quadro 2. Análise das evidências do DRE (Autor, 2018).

Os resultados esperados que estão sendo atendidos de forma total no processo da CO-TIC, como pode ser visto no Quadro 2, são: DRE2, DRE3 e DRE7. Portanto, não há necessidade de justificar o grau de atendimento dos mesmos. A seguir, serão detalhadas as justificativas para os resultados esperados que foram classificados com atendimento parcial e não atendimento.

3.3.1 Resultados Esperados Parcialmente Atendidos

De acordo com a análise realizada, os resultados esperados que foram classificados como tendo atendimento apenas parcial foram: DRE1 e DRE8. O motivo pelo qual o DRE1 foi classificado dessa forma dá-se pelo fato de que, para esse resultado ser completamente atendido, as necessidades, expectativas e restrições do cliente devem ser devidamente coletadas, que serão fundamentais para criação dos requisitos do produto. Embora existam evidências da conversa com o cliente, representadas pelas tarefas "Reunião com o PO" e Diálogo acerca do escopo do produto" da atividade **Construir Modelo Abrangente**, não existe nenhum método de entrevista ou diálogo específico para a coleta dos dados mencionados por completo, ficando totalmente dependente da conversa aberta que acontece com o PO e com os interessados no produto, o que caracteriza uma reunião informal que não gera nenhum documento, fazendo com que o atendimento desse resultado esperado seja apenas parcial.

No caso do DRE8, o motivo do mesmo ser classificado dessa forma dá-se pelo fato de que, atualmente, apesar de haver uma validação com o cliente do que foi desenvolvido, evidenciada pelas atividades ***Sprint Review*** e **Homologação da Versão do Produto,** não há nenhum artefato no processo que comprove essa validação. Portanto, há ausência de uma tarefa que comprove a ocorrência dessa validação.

3.3.2 Resultados Esperados Não Atendidos

De acordo com a análise realizada, os resultados esperados que foram classificados como não atendidos foram: DRE4, DRE5 e DRE6. O motivo pelo qual o DRE4 foi classificado dessa forma dá-se pelo fato de que, para esse resultado ser completamente atendido, os requisitos coletados no DRE3 devem ser refinados, aprimorando seu entendimento e facilitando o desenvolvimento dos mesmos. O processo de desenvolvimento antes da melhoria não apresentava nenhuma tarefa de refinamento dos requisitos funcionais e não-funcionais após suas definições. No fluxo antes da melhoria, as Estórias de Usuário são apenas detalhadas a ponto de ser criado um *checklist* que atende ao DRE3, não havendo um refinamento ou elaboração posterior.

No caso do DRE5, o motivo do mesmo ser classificado como não atendido dá-se pelo fato de que, para ser alcançado o grau de atendimento considerado total, devem ser definidas as interfaces internas e externas dos componentes do produto, facilitando a visualização da interação dos componentes tanto entre eles mesmos quanto entre aplicações externas ao produto, facilitando o desenvolvimento dos mesmos. No processo antes da melhoria, após ser realizada a criação dos cartões com as Estórias de Usuário, o fluxo segue sem refinamentos, especificações ou prototipagens das estórias. Após ser criado um *checklist* contendo os requisitos funcionais e não-funcionais da estória, a mesma não sofre nenhuma alteração, passando a ser desenvolvida e, posteriormente, implementada. Portanto, não há prática que evidencie o atendimento do DRE5.

Para ocorrer o atendimento total do DRE6, o processo deve apresentar tarefas e atividades que evidenciem a criação de conceitos operacionais e cenários envolvendo os requisitos funcionais e não-funcionais que foram anteriormente criados. No processo antes da melhoria, os conceitos operacionais, cenários ou fluxos de telas não estão sendo criados, dificultando o entendimento do fluxo do produto para realizar o desenvolvimento.

3.4 MELHORIA NO PROCESSO ÁGIL PÓS-AVALIAÇÃO

Após realizada a avaliação do processo ágil com base nos resultados esperados do DRE, foi realizada a modelagem da melhoria desse processo. Para tal, foram dadas sugestões de como atender, separadamente, cada resultado esperado que não estava sendo atendido por completo. As tarefas e atividades sugeridas levam em consideração o contexto ágil da organização.

3.4.1 Sugestões de Melhoria

Com base na avaliação realizada em cima do processo, presente na Seção 3.3, o DRE1 foi considerado como sendo atendido parcialmente. Foram encontradas, no processo antes da melhoria, atividades e tarefas relacionadas ao atendimento desse resultado esperado. Não há, portanto, necessidade da criação de novas atividades dentro do fluxo do processo para garantir o atendimento total desse resultado esperado. Para o DRE1, foi sugerida a criação de um roteiro de entrevista para realizar com PO, assim como um questionário fechado contendo perguntas relacionadas ao escopo do produto, finalidade, restrições, o que o interessado espera desse produto e quais suas restrições. Essas tarefas foram incluídas na atividade **Construir Modelo Abrangente**, como pode ser visto na Figura 16.

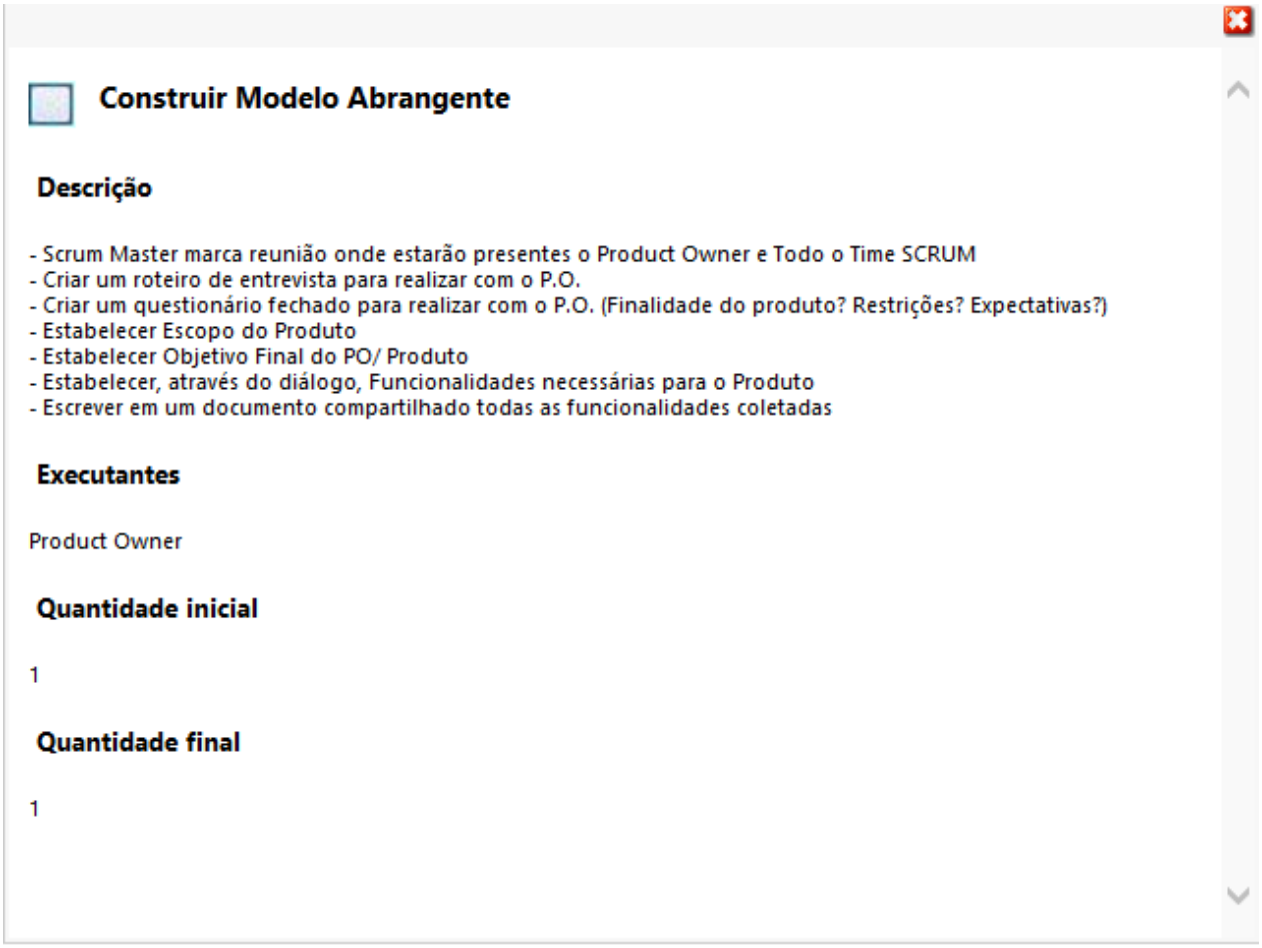


Figura 16. Construir Modelo Abrangente pós-melhoria (Autor, 2018).

Assim como na avaliação feita para o DRE1, existem evidências de atividades e tarefas relacionadas ao atendimento do DRE8. Não há, portanto, necessidade da criação de novas atividades dentro do fluxo, fazendo-se necessária apenas a implantação de tarefas nas atividades já existentes, desde que sejam suficientes para o atendimento total desse resultado. A sugestão, portanto, é adicionar no fluxo do Kanban uma raia para, após ter sido completamente desenvolvida e testada, a Estória de Usuário ter que ser validada com o PO. Após validado, será coletada a assinatura do cliente que comprova a concordância com o que está sendo entregue, e a estória passa para o *status* "Validado" no Kanban. Essas tarefas foram incluídas na atividade ***Sprint Review***, como pode ser visto na Figura 17.

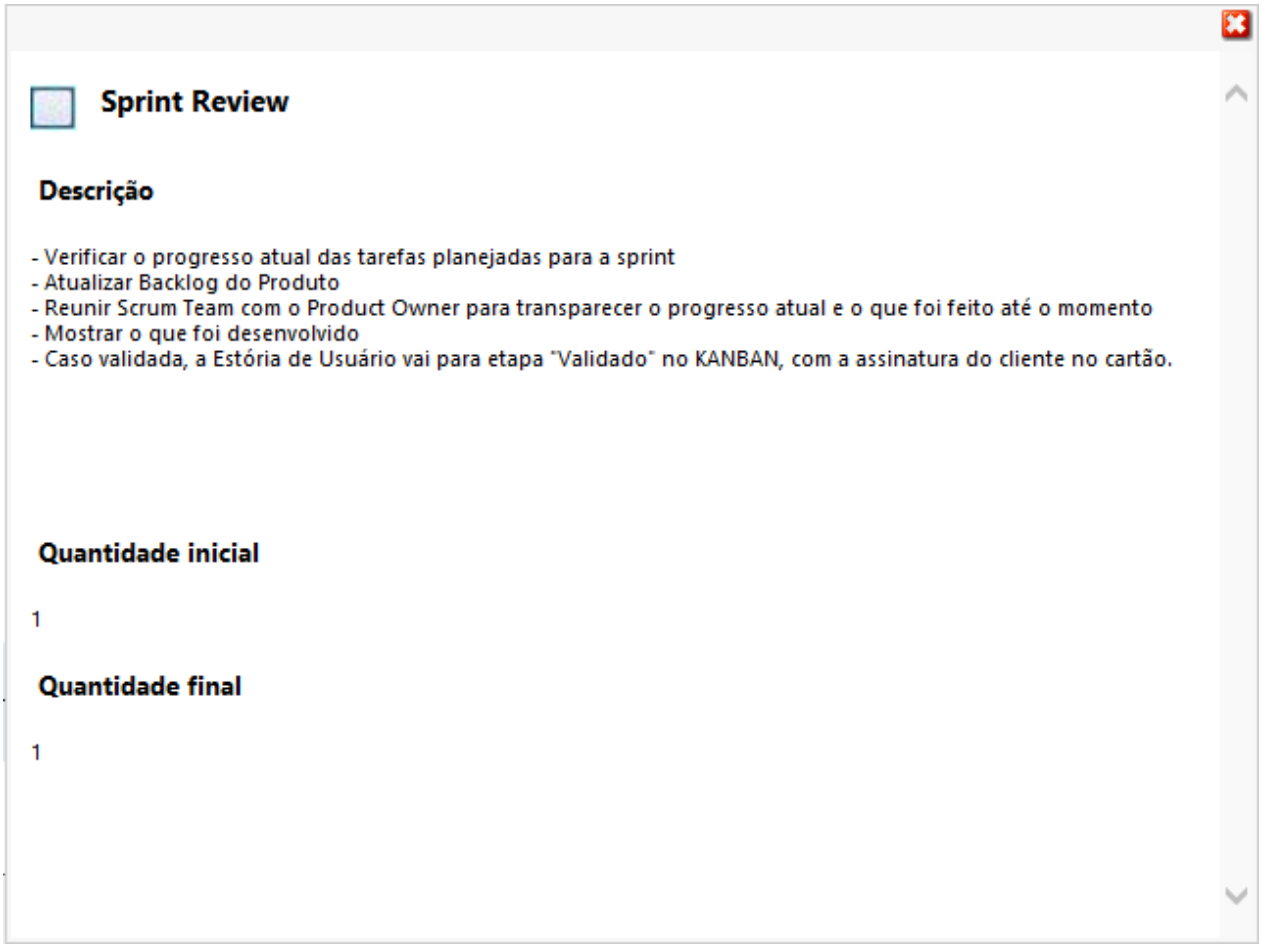


Figura 17. *Sprint Review* pós-melhoria (Autor, 2018).

Diferente das melhorias sugeridas para os resultados esperados anteriores, os DRE4, DRE5 e DRE6 não apresentam evidência alguma de seu atendimento no processo de desenvolvimento da COTIC antes da melhoria. Houve, portanto, a necessidade da criação de novas atividades dentro do fluxo do processo, compostas por tarefas que atendem por completo esses resultados esperados. Devido ao fato de que as atividades referentes a esses três resultados devem ser introduzidas no fluxo após a criação dos requisitos funcionais e não-funcionais de cada estória e antes de ser iniciado o desenvolvimento das estórias, foi criado um novo subprocesso chamado de **Refinar Estória de Usuário**, com o intuito de agrupar as novas atividades que irão compor o processo.

O novo subprocesso representa uma nova etapa no Kanban da organização. Ao sair do *Backlog*, a estória de usuário deverá passar pela etapa de refinamento, para apenas posteriormente ser iniciado o desenvolvimento do produto. Esse subprocesso tem início logo após a finalização da atividade **Selecionar Estória de Usuário**, presente no subprocesso **Realizar Desenvolvimento do Produto**, e tem como artefato de insumo a estória que fora selecionada para implementação. A Figura 18 representa esse subprocesso implementado no fluxo.

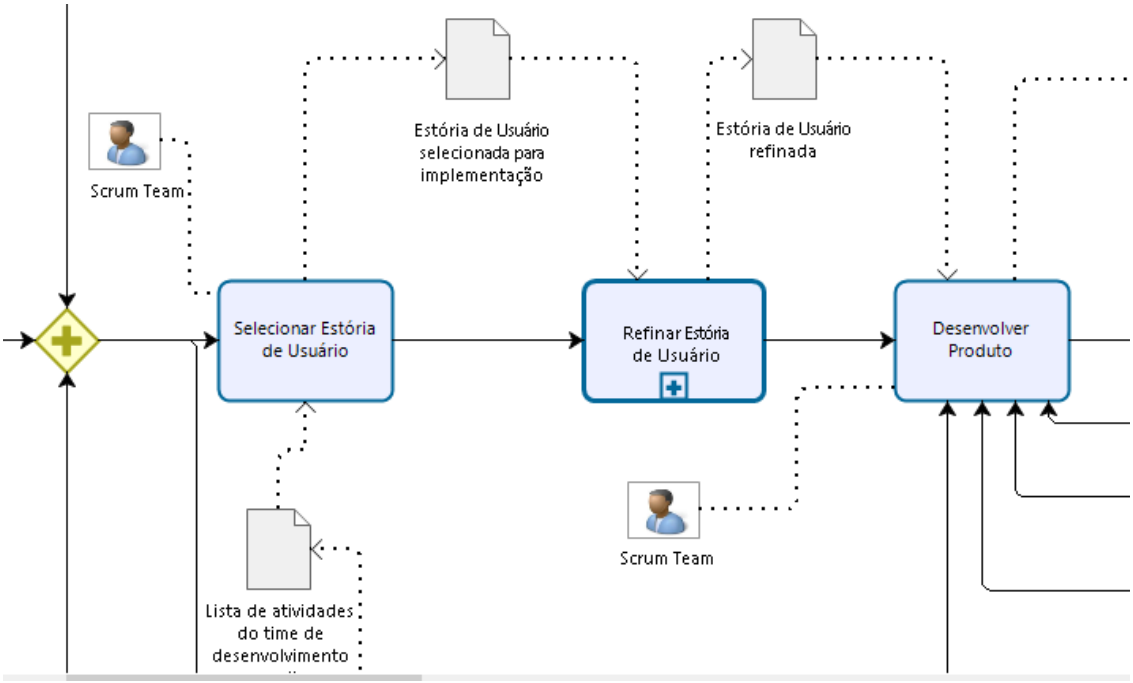


Figura 18. Refinar Estória de Usuário Adicionado ao Fluxo (Autor, 2018).

Dentro do subprocesso **Refinar Estória de Usuário**, foram criadas as atividades para atendimento aos resultados esperados restantes, como pode ser visualizado na Figura 19.

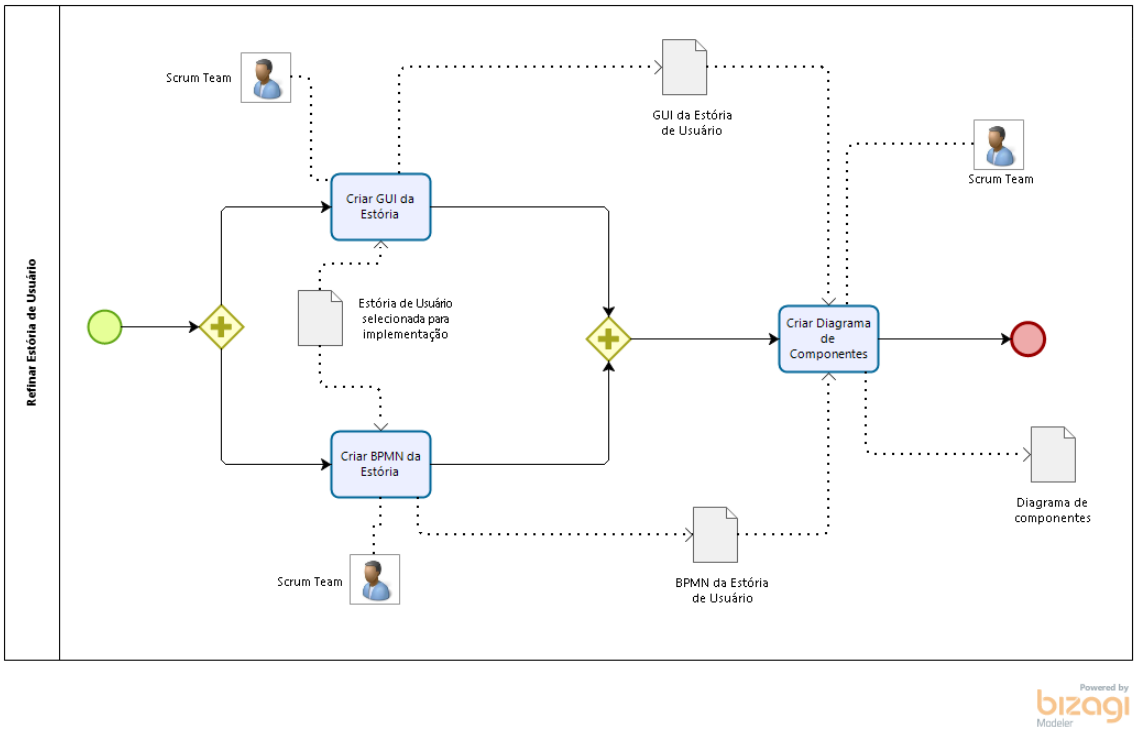


Figura 19. Subprocesso Refinar Estória de Usuário (Autor, 2018).

Ao iniciar esse subprocesso, o fluxo é dividido em duas atividades paralelas: **Criar GUI da Estória** e **Criar BPMN da Estória.** Para realizar a atividade **Criar GUI da Estória**, o *Scrum Team* irá utilizar como insumo a estória de usuário selecionada para a implementação na atividade anterior, com o objetivo de criação da *Graphical User Interface* (GUI) – Interface Gráfica de Usuário – referente a essa estória. Essa interface gráfica servirá como guia para a implementação da estória, pois mostra, visualmente, como as funcionalidades da mesma devem ser implementadas. Após realizada a atividade, tem-se o atendimento total do DRE4, dado que essa GUI servirá como refinamento para os requisitos já criados. As tarefas que compõem essa atividade estão representadas na Figura 20.

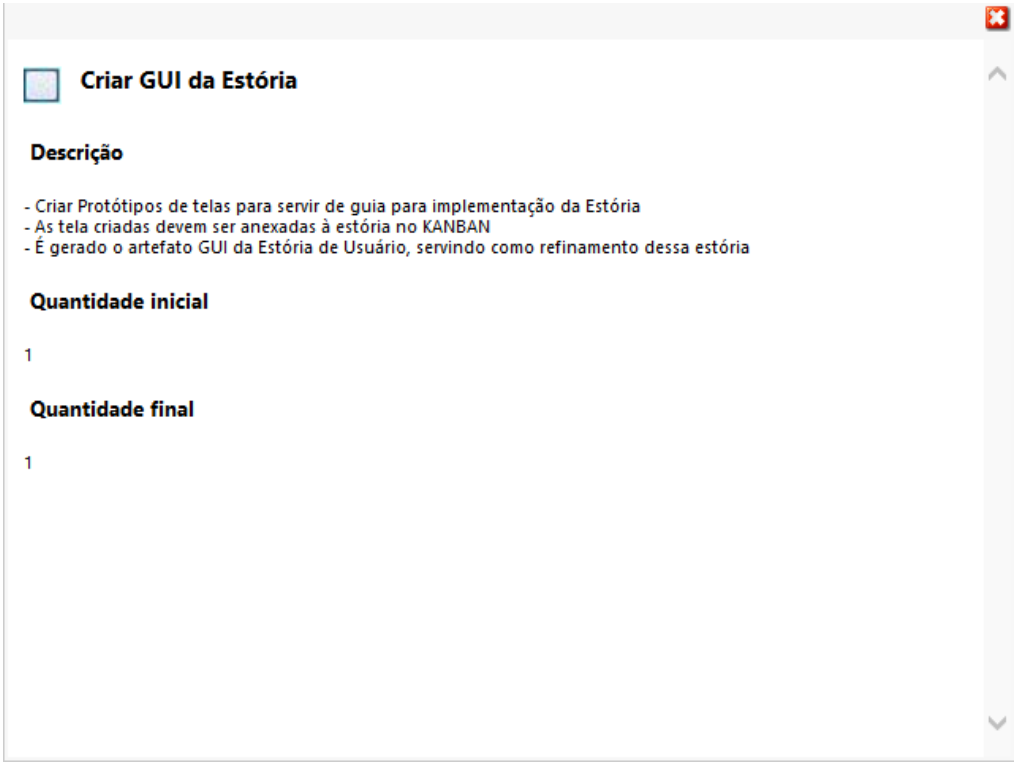


Figura 20. Atividade Criar GUI da Estória (Autor, 2018).

Paralelo à atividade **Criar GUI da Estória**, é realizada a atividade **Criar BPMN da Estória**. O *Scrum Team* irá utilizar como insumo a Estória de Usuário selecionada na atividade anterior ao subprocesso, com o objetivo de criar, para essa estória, um modelo que represente o fluxo do sistema, de forma que o desenvolvimento dos requisitos seja feito de acordo com esse modelo. A ferramenta que pode ser utilizada para atender esse requisito é o Bizagi Modeler, pois é sugerido que, para realizar essa modelagem, seja utilizada a notação BPMN.

Após finalizada essa atividade, é gerado o artefato BPMN da Estória de Usuário, que representa o fluxo que a estória deverá seguir em sua implementação. Por se tratar do desenvolvimento de um conceito operacional e representação de um cenário a partir da modelagem,

essa atividade é suficiente para o atendimento total do DRE6. As tarefas que compõem essa atividade podem ser visualizadas na Figura 21.

Após finalizadas as atividades **Criar BPMN da Estória** e **Criar GUI da Estória**, o fluxo junta-se novamente, utilizando os artefatos gerados como insumo para a última atividade do subprocesso.

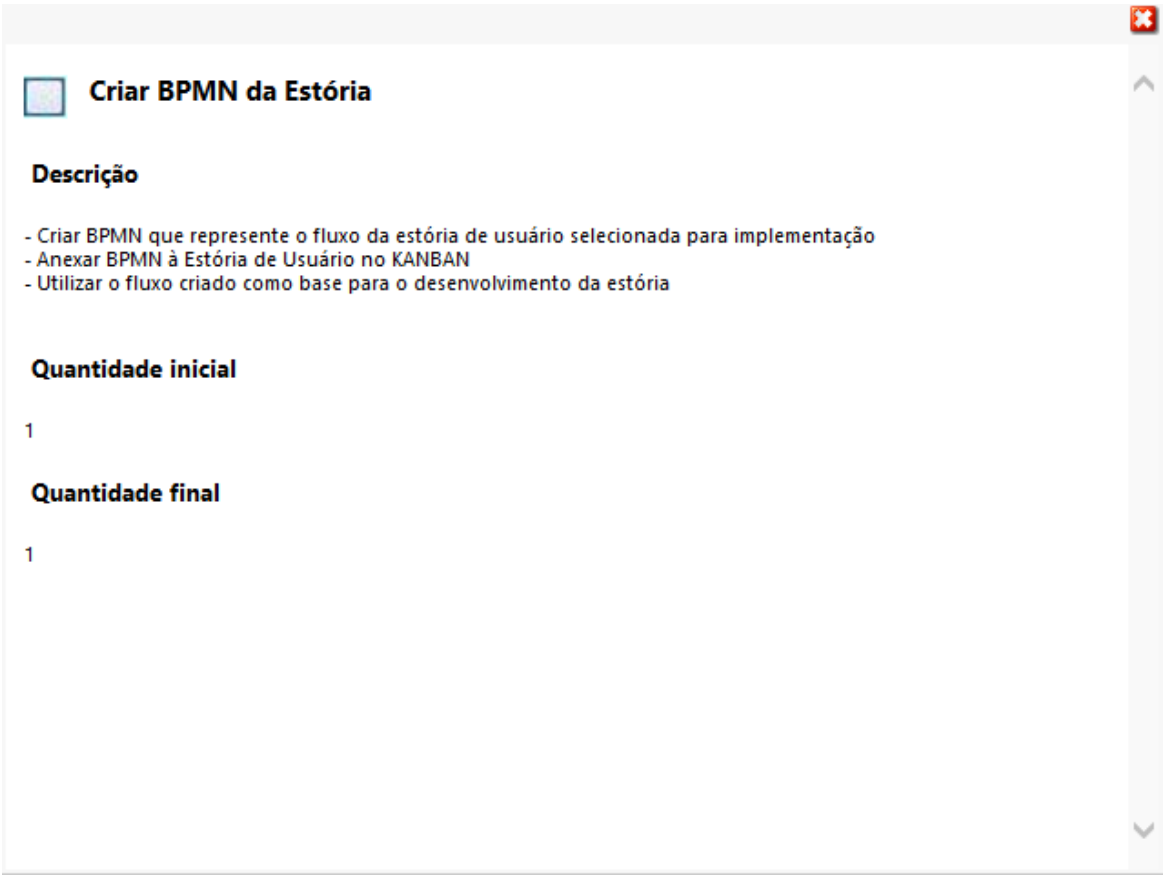


Figura 21. Atividade Criar BPMN da Estória (Autor, 2018).

A última atividade desse subprocesso é chamada de **Criar Diagrama de Componentes**. Tendo como insumo os artefatos gerados nas duas atividades anteriores, o *Scrum Team* irá realizar a criação de um diagrama de componentes, que foram anteriormente identificados, para traçar a relação desses componentes internos ao produto, além de verificar quais componentes externos ao produto também devem ser considerados.

Após a realização dessa atividade, observa-se o atendimento total ao resultado esperado 5 do DRE, visto que as interfaces internas e externas dos componentes e do produto foram identificadas e anexadas à estória. As tarefas que compõem essa atividade podem ser visualizadas na Figura 22. Após a finalização da atividade **Criar Diagrama de Componentes**, é

gerado o artefato Diagrama de Componentes, ocorrendo a finalização do subprocesso **Refinar Estória de Usuário** e o retorno ao subprocesso **Realizar Desenvolvimento do Produto**.

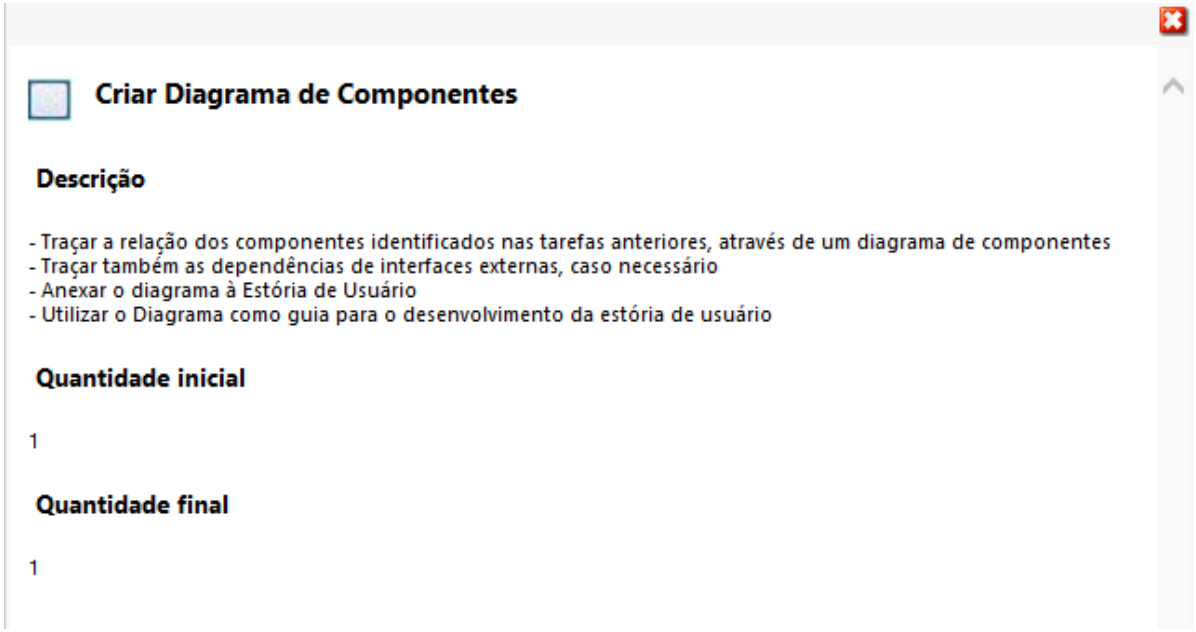


Figura 22. Atividade Criar Diagrama de Componentes (Autor, 2018).

Após a finalização desse subprocesso, todos os artefatos gerados dentro do mesmo foram resumidos no artefato Estória de Usuário Refinada. Esses artefatos servirão como insumo para o desenvolvimento da estória, portanto, foi necessária realizar a adição de novas tarefas que descrevem a execução da atividade **Desenvolver Produto**.

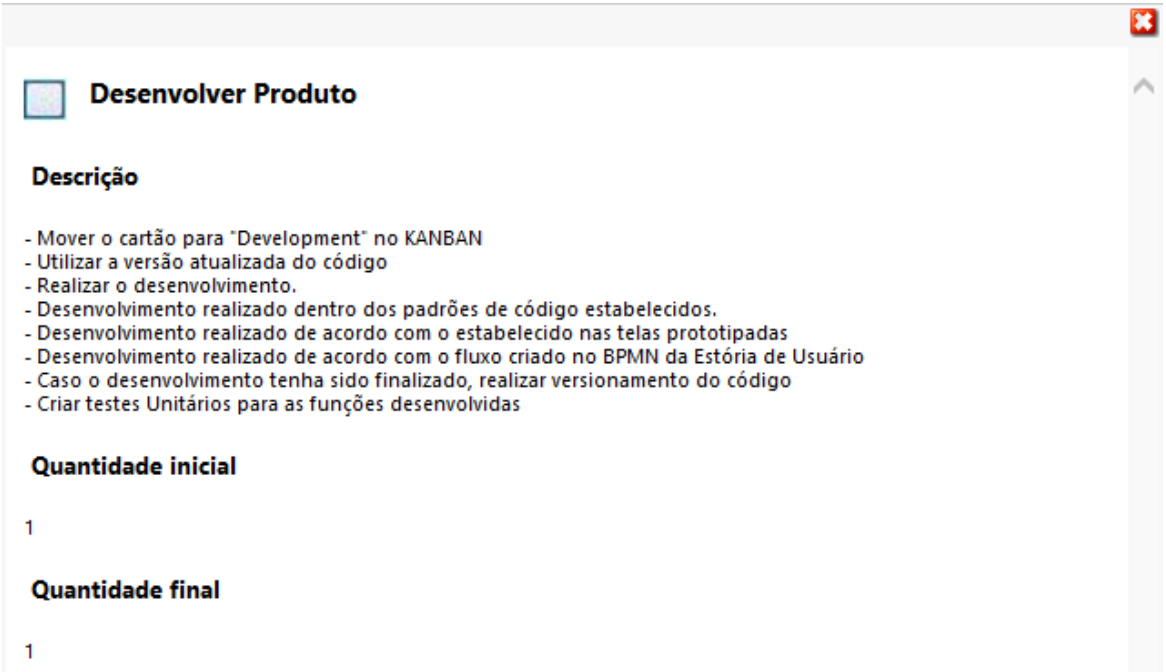


Figura 23. Atividade Desenvolver Produto (Autor, 2018).

Posterior à inserção das sugestões no processo de desenvolvimento, todos os resultados esperados presentes do processo de Desenvolvimento de Requisitos do MR-MPS-SW passam a ter evidências de seu atendimento por completo, finalizando a melhoria do processo ágil da COTIC.

4 CONCLUSÕES

Esta seção apresenta as conclusões do trabalho realizado, incluindo contribuições do mesmo para a área da Engenharia de Software no que tange o Desenvolvimento de Requisitos e para a instituição cuja melhoria de processo foi realizada. Desta forma, são apresentados a seguir uma visão geral sobre o trabalho realizado, suas contribuições, as limitações presenciadas ao decorrer de seu desenvolvimento e os trabalhos futuros.

4.1 VISÃO GERAL

Neste trabalho foi realizada uma melhoria no processo de desenvolvimento ágil da Coordenadoria de Tecnologia da Informação e Comunicação da Pró-Reitoria de Ensino e Graduação da Universidade Federal do Pará. Para tal, foi realizada a divisão do trabalho em três capítulos que antecedem este, de conclusão.

No capítulo introdutório, foi apresentada a contextualização deste trabalho, por meio de uma visão geral sobre o principal foco do trabalho, a justificativa de sua confecção, o que motivou a sua realização, além de seus objetivos e a metodologia aplicada para a realização do mesmo.

O segundo capítulo, cujo objetivo é o detalhamento do referencial teórico utilizado para este trabalho, introduz o conceito de Engenharia de Software juntamente com Engenharia de Requisitos, com o objetivo de apresentar os temas gerais que englobam este trabalho. Em seguida, foram detalhados os conceitos de Processo de Software e Melhoria de Processo, além da apresentação do modelo de qualidade de software MPS.BR, o processo de DRE do MR-MPS-SW e, por último, metodologias ágeis que envolvem o processo utilizado para modelagem neste trabalho.

O terceiro capítulo apresentou o contexto do processo de desenvolvimento ágil da COTIC, configurando os contribuintes da instituição e seus papéis dentro do processo. Posteriormente, foi realizada a modelagem desse processo, detalhando seu fluxo, seus subprocessos, as atividades que o compõem e uma descrição da execução das mesmas. Após a modelagem, foi realizada uma avaliação desse processo, utilizando como base os resultados esperados do DRE e verificando quais resultados foram atendidos por completo, de forma parcial ou não foram atendidos. Por fim, foi realizada a modelagem da melhoria desse processo, apresentando total atendimento aos resultados esperados anteriormente avaliados.

4.2 CONTRIBUIÇÕES

Ao realizar a avaliação do processo de desenvolvimento ágil da COTIC, foi possível perceber suas falhas em relação ao atendimento dos resultados esperados do Desenvolvimento de Requisitos (DRE) do MR-MPS-SW, contribuindo para a detecção de atividades e subprocessos que necessitavam de mudanças dentro desse processo.

Além disso, foi realizada a modelagem de uma melhoria para o processo de desenvolvimento da COTIC, demonstrando como pode ser realizada uma mudança nesse processo para o atendimento total dos resultados esperados do DRE. Este trabalho serve, portanto, como material de suporte para a implementação dessas mudanças, caso a instituição opte por segui-lo.

No que tange à área de Engenharia de Software e Melhoria de Processos, este trabalho propõe uma melhoria realizada em um processo cujas atividades são totalmente baseadas em metodologias ágeis de desenvolvimento de *software*, diferente de trabalhos que envolvem a etapa de desenvolvimento de requisitos de um produto cujos processos utilizados como objeto de estudo adotam metodologias tradicionais de desenvolvimento, também conhecido como modelo cascata, demonstrando o diferencial deste.

4.3 LIMITAÇÕES

Para realizar o desenvolvimento deste trabalho foi necessário mapear o fluxo e as atividades que constituem o processo de desenvolvimento da COTIC, bem como seus subprocessos e a descrição de cada atividade presente nos mesmos. Um dos obstáculos para a realização desse procedimento foi a ausência de documentos e artefatos que indicassem de forma clara as etapas presentes no processo, mostrando-se essencial a utilização de trabalhos que realizaram a modelagem do processo anteriormente. Entretanto, devido ao fato do processo da COTIC sofrer mudanças e adaptações frequentemente, também foram realizadas entrevistas com o atual responsável pela instituição, com o intuito de verificar diferenças entre os trabalhos anteriormente modelados e o processo atual.

Outra limitação é evidenciada pelo fato de que, por se tratar da modelagem de uma melhoria de processo, as mudanças que foram sugeridas neste trabalho não foram implementadas. Cabe aos colaboradores da COTIC realizarem a implementação das melhorias aqui sugeridas, verificando a viabilidade de adequação do seu processo às sugestões deste trabalho.

Por se tratar de uma modelagem de fluxo de processo, informações acerca de tecnologias utilizadas para desenvolvimento, *softwares* utilizados para testes e procedimentos técnicos

que envolvem o ambiente de trabalho da instituição precisaram ser abstraídos dessa modelagem.

4.4 TRABALHOS FUTUROS

Devido ao fato de que este trabalho utiliza apenas o processo de Desenvolvimento de Requisitos, deve ser considerada a realização de trabalhos que envolvam outros processos presentes no nível D do MR-MPS-SW, como os processos Integração do Produto (ITP) e Projeto e Construção do Produto (PCP).

Pode-se ponderar também a produção de uma análise das métricas envolvendo o desenvolvimento do processo antes e depois das melhorias implementadas, sugeridas neste trabalho.

REFERÊNCIAS

AGILE ALLIANCE. Disponível em: <https://www.agilealliance.org/>. Último acesso em: 20 out. 2018.

BARATA, J. C. N. **Uma Proposta de Melhoria do Processo e Qualidade do Software usando BPMN: Um Estudo de Caso da COTIC PROEG**. Trabalho de Conclusão de Curso – Universidade Federal do Pará: Castanhal, 2018.

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

BERTOLLO, Gleidson. **Definição de Processos Em Um Ambiente de Desenvolvimento De Software**. Universidade Federal do Espírito Santo: Vitória, 2006.

BENEDICT, T., et al.: **BPM CBOK Version 3.0**: **guide to the business process management common body of knowledge**. ABPMP International/Createspace (2013). Disponível em: <http://www.abpmp.org/>. Último acesso em: 15 out. 2018.

BORIA, J. L., Rubinstein, V. L., Rubinstein, A. **A História da Tahini-Tahini: Melhoria de processos de software com métodos ágeis e modelo MPS**. Brasília: Ministério da Ciência, Tecnologia e Inovação. Secretaria de Política e Informática, 2013.

DERNIAME, J. C., Kaba, B. A., Wastell, D. **Software Process**: **principles, methodology and technology**. Lecture Notes in Computer Science, Vol. 1500, Springer, 1999.

FALBO, Ricardo A. **Engenharia de Software – Notas de Aula**. Universidade Federal do Espírito Santo: Vitória, 2005.

FREITAS, J. V. **Descrição de Um Processo de Software Usando SPIDER_ML: Um Estudo de Caso da AIT-PROEG.** Trabalho de Conclusão de Curso – Universidade Federal do Pará: Belém, 2016.

FUGETTA, A. **Software process: A roadmap**. 22nd International Conference on Software Engineering. Limerick, Irlanda, 2000.

GODOY, R. **Kanban: 4 passos para implementar em uma equipe**. Disponível em: <https://www.devmedia.com.br/kanban-4-passos-para-implementar-em-uma-equipe/30218/>. Último acesso em: 25 out. 2018.

HIGHSMITH, J. **History: The Agile Manifesto.** Disponível em: https://agilemanifesto.org/history.html. Último acesso em: 24 out. 2018.

HUMPHREY, Watts S. **Managing the Software Process, The SEI Series in Software Engineering**. Addison-Wesley, 1989.

JOSUTTIS, N. M. **SOA na Prática –** A arte de modelagem de Sistemas Distribuídos. 1. Ed. Tradução de Ivan Bosnic. Rio de Janeiro: Alta Books, 2008.

MENEZES, I. **Proposta de um processo de Gerência de Requisitos baseada no nível G do MPS.Br: Estudo de caso da Ait-Proeg**. Trabalho de Conclusão de Curso – Universidade Federal do Pará: Belém, 2017.

OLIVEIRA, Sandro R. B. **ProDefiner: Uma Abordagem Progressiva para a Definição de Processos de Software no Contexto de um Ambiente Centrado no Processo**. Tese de Doutorado, CIN/UFPE. Recife, 2007.

PIZZA, William**. A metodologia Business Process Management (BPM) e sua importância para as organizações**. Trabalho de Conclusão de Curso – Faculdade de Tecnologia de São Paulo: São Paulo, 2012.

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

SOFTEX - SOCIEDADE PARA PROMOÇÃO DA EXCELÊNCIA DO SOFTWARE BRASILEIRO. **MPS.BR – Melhoria de Processo do Software Brasileiro, Guia Geral MPS de Software**. 2016a.

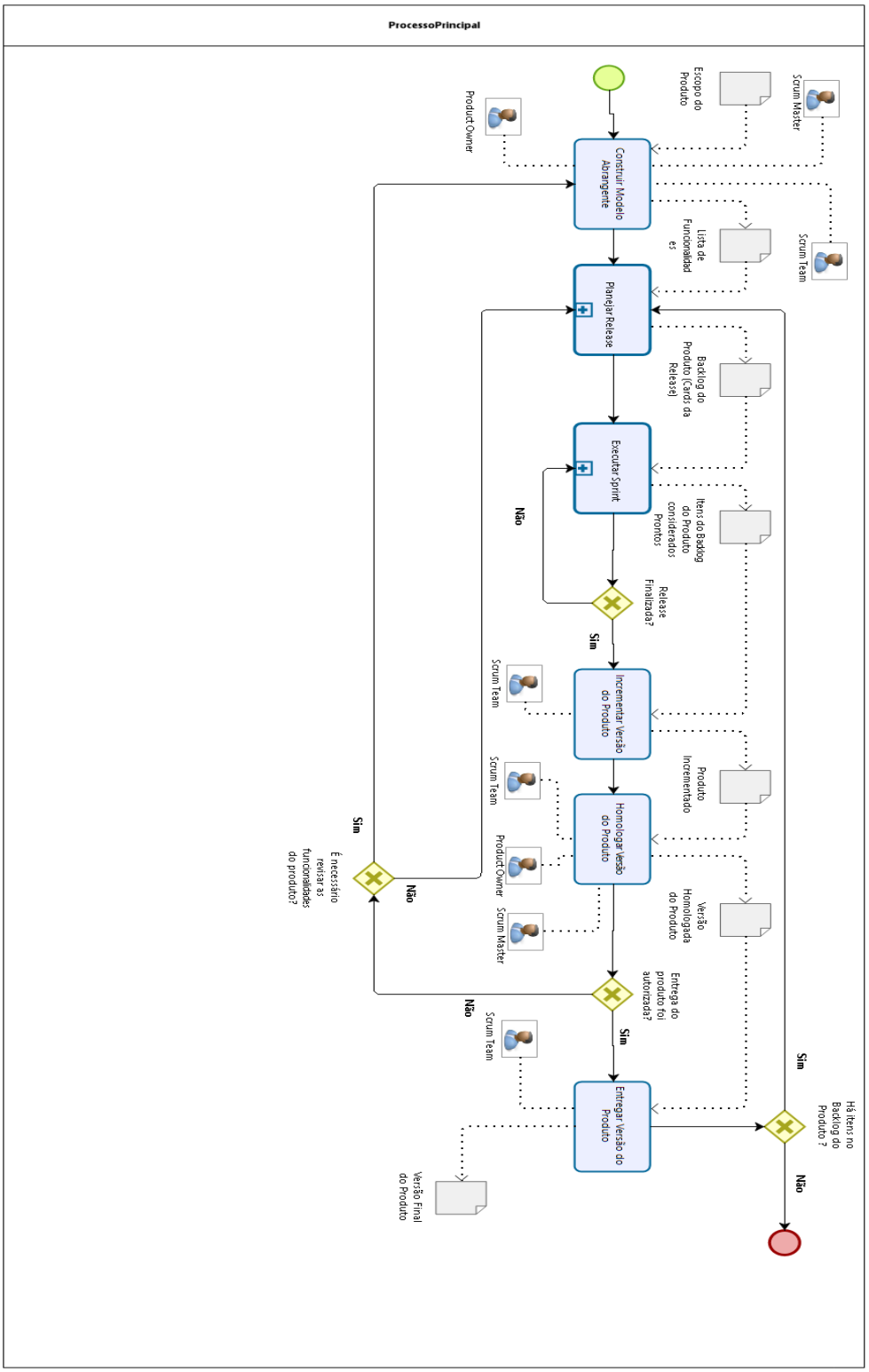
SOFTEX. SOCIEDADE PARA PROMOÇÃO DA EXCELÊNCIA DO SOFTWARE BRASILEIRO. **Melhoria de Processo de Software Brasileiro (MPS.BR). Guia de Implementação – Parte 4: Fundamentação para Implementação do Nível D do MR-MPS-SW:2016**. 2016b.

STOROLLI, A. L., Zanolla, G. I., Guidini, J. E., Borsoi, B. T. **Modelagem de processo de software**. 2009.

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

APÊNDICE A – Detalhamento das Atividades do Processo Melhorado

1 PROCESSO PRINCIPAL



Fonte: Autor, 2018.

1.1 CONSTRUIR MODELO ABRANGENTE

1.1.1 Descrição:

- Scrum Master marca reunião onde estarão presentes o Product Owner e Todo o Scrum Team;
- Criar um roteiro de entrevista para realizar com o P.O.;
- Criar um questionário fechado para realizar com o P.O. (Finalidade do produto? Restrições? Expectativas?);
- Estabelecer Escopo do Produto;
- Estabelecer Objetivo Final do PO/ Produto;
- Estabelecer, através do diálogo, Funcionalidades necessárias para o Produto; e
- Escrever em um documento compartilhado todas as funcionalidades coletadas

1.1.2 Executantes:

- Scrum Master, Scrum Team e Product Owner.

1.1.3 Artefato(s) de Insumo:

- Escopo do Produto.

1.1.4 Artefato(s) Produzido(s):

- Lista de Funcionalidades.

1.2 PLANEJAR RELEASE

1.2.1 Descrição:

- Por ser um subprocesso, será detalhado na etapa 2.

1.2.2 Artefato(s) de Insumo:

- Lista de Funcionalidades.

1.2.3 Artefato(s) Produzido(s):

- Backlog do Produto (Cards da Release).

1.3 EXECUTAR SPRINT

1.3.1 Descrição:

- Por ser um subprocesso, será detalhado na etapa 3.

1.3.2 Artefato(s) de Insumo:

- Backlog do Produto (Cards da Release).

1.3.3 Artefato(s) Produzido(s):

- Itens do Backlog do Produto considerados Prontos.

1.4 INCREMENTAR VERSÃO DO PRODUTO

1.4.1 Descrição:

- Criar nova versão do produto adicionando as funcionalidades implementadas; e
- Verificar possíveis conflitos no código utilizando o software de versionamento.

1.4.2 Executantes:

- Scrum Team.

1.4.3 Artefato(s) de Insumo:

- Itens do Backlog do Produto considerados Prontos.

1.4.4 Artefato(s) Produzido(s):

- Produto Incrementado.

1.5 HOMOLOGAR VERSÃO DO PRODUTO

1.5.1 Descrição:

- Reunir com o Product Owner e Scrum Team;
- Apresentar Nova versão do Produto;
- P.O. analisa as novas funcionalidades; e
- P.O decide se a nova versão do produto está autorizada para ser realizado o de-ploy para produção.

1.5.2 Executantes:

- Scrum Team, Scrum Master e Product Owner.

1.5.3 Artefato(s) de Insumo:

- Produto Incrementado.

1.5.4 Artefato(s) Produzido(s):

- Versão Homologada do Produto.

1.6 ENTREGAR VERSÃO DO PRODUTO

1.6.1 Descrição:

- Deploy do Produto para ambiente de produção utilizando FTP;
- Exportar o banco de dados e possíveis alterações para o servidor de Produção; e
- Enviar E-mail aos Envolvidos no produto sobre a entrega da nova versão.

1.6.2 Executantes:

- Scrum Team.

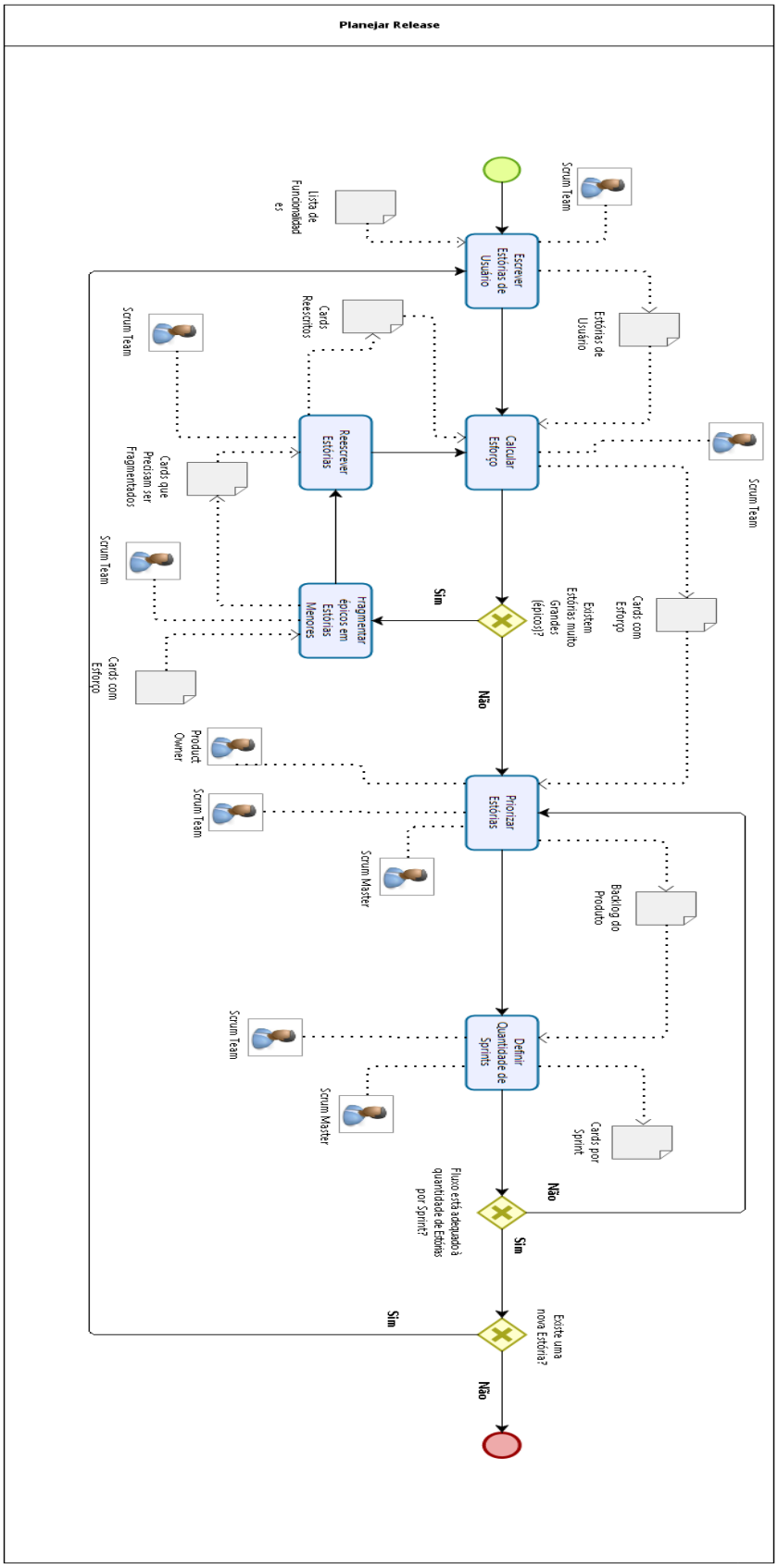
1.6.3 Artefato(s) de Insumo:

- Versão Homologada do Produto.

1.6.4 Artefato(s) Produzido(s):

- Versão Final do Produto.

2 PLANEJAR RELEASE



Fonte: Autor, 2018.

2.1 ESCREVER ESTÓRIAS DE USUÁRIO

2.1.1 Descrição:

- Utilizar Cartões de Papel para organizar as Funcionalidades;
- Na parte da frente do cartão, serão as funcionalidades serão escritas no seguinte formato:

    - Quem (Perfil/ Usuário final da funcionalidade)?

    - O que (Objetivo da funcionalidade)?

    - Por que (Motivo de a funcionalidade estar presente no produto)?

- No verso do Cartão, a User Story será descrita mais detalhadamente, contendo todos os aspectos necessários para sua implementação.

2.1.2 Executantes:

- Scrum Team.

2.1.3 Artefato(s) de Insumo:

- Lista de Funcionalidades.

2.1.4 Artefato(s) Produzido(s):

- Estórias de Usuário.

2.2 CALCULAR ESFORÇO

2.2.1 Descrição:

- Todos os membros do Scrum Team se reúnem para estimar o esforço necessário para implementar as User Stories;
- Realizar método para medição de esforço;
- Discutir sobre as estórias que ficaram com esforço muito alto;
- Escrever o esforço estimado nos cartões; e
- Estórias épicas que foram fragmentadas em outras estórias menores serão estimadas novamente.

2.2.2 Executantes:

- Scrum Team.

2.2.3 Artefato(s) de Insumo:

- Estórias de Usuário.

2.2.4 Artefato(s) Produzido(s):

- *Cards* com Esforço.

2.3 FRAGMENTAR ÉPICOS EM ESTÓRIAS MENORES

2.3.1 Descrição:

- Verificar quais Estórias estão exigindo um esforço muito alto para serem executadas;
- Time discute a possibilidade de fragmentar essa estória em outras menores; e
- Definir os cartões que serão fragmentados.

2.3.2 Executantes:

- Scrum Team.

2.3.3 Artefato(s) de Insumo:

- *Cards* com Esforço.

2.3.4 Artefato(s) Produzido(s):

- *Cards* que Precisam ser Fragmentados.

2.4 REESCREVER ESTÓRIAS

2.4.1 Descrição:

- Scrum Team reescreve em novos cartões as estórias que foram fragmentadas.

2.4.2 Executantes:

- Scrum Team.

2.4.3 Artefato(s) de Insumo:

- *Cards* que Precisam ser Fragmentados.

2.4.4 Artefato(s) Produzido(s):

- *Cards* Reescritos.

2.5 PRIORIZAR ESTÓRIAS

2.5.1 Descrição:

- Scrum Team marca reunião com PO, com as estórias de usuário devidamente escritas nos cartões;
- O PO irá determinar quais estórias apresentam mais valor para ele, estabelecendo uma ordem de prioridades; e
- Criar Backlog do Produto com as estórias já ordenadas por prioridade.

2.5.2 Executantes:

- Scrum Team, Scrum Master e Product Owner.

2.5.3 Artefato(s) de Insumo:

- *Cards* com Esforço.

2.5.4 Artefato(s) Produzido(s):

- *Backlog* do Produto.

2.6 DEFINIR QUANTIDADE DE SPRINTS

2.6.1 Descrição:

- Scrum Team se reúne com o Scrum Master;
- Time analisa o esforço necessário para finalizar as estórias que estão no Backlog; e
- Time discute quantas Sprints serão necessárias para finalizar as estórias e completar a Release.

2.6.2 Executantes:

- Scrum Team e Scrum Master.

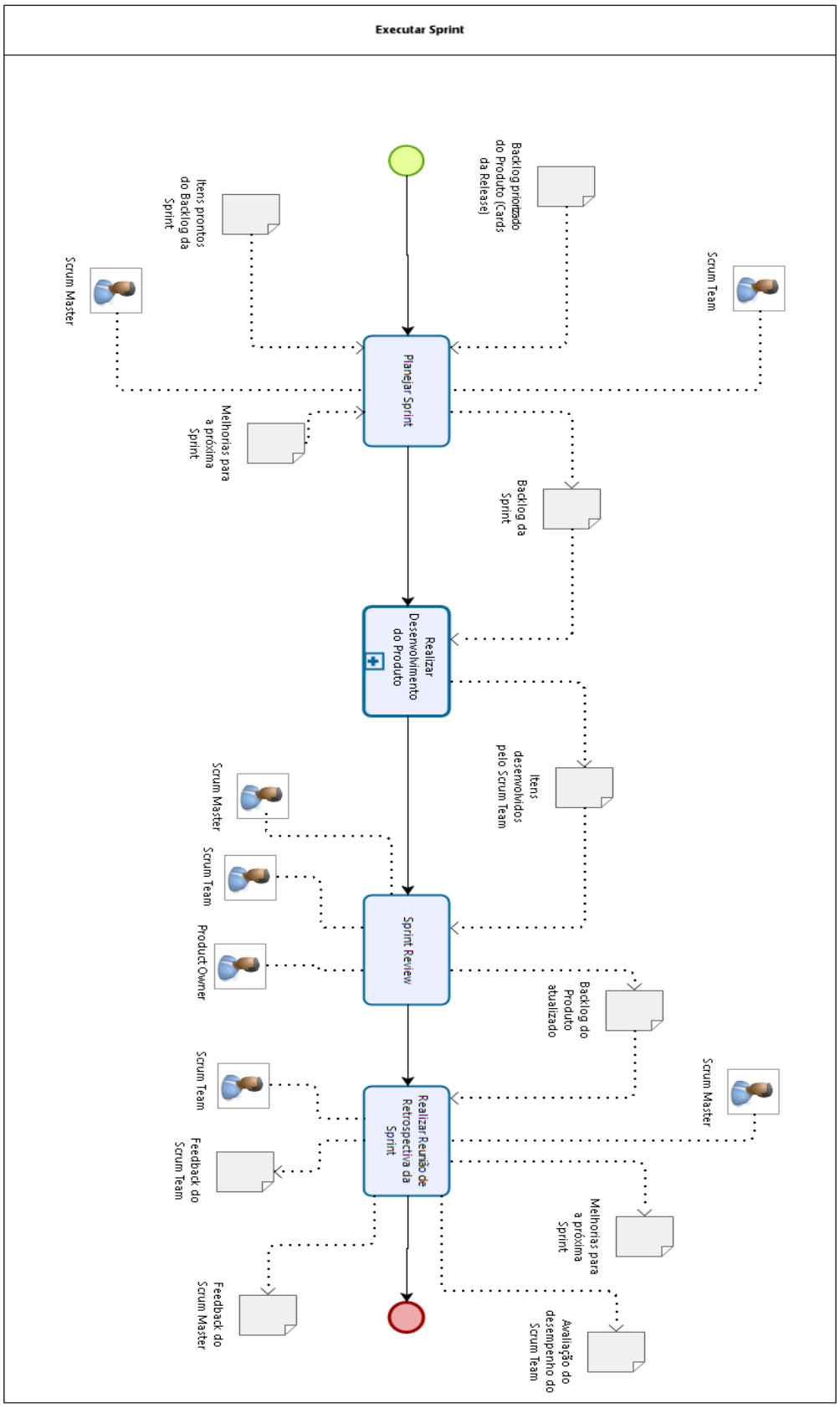
2.6.3 Artefato(s) de Insumo:

- *Backlog* do Produto.

2.6.4 Artefato(s) Produzido(s):

- *Cards* por *Sprint*.

3 EXECUTAR SPRINT



Fonte: Autor, 2018.

3.1 PLANEJAR SPRINT

3.1.1 Descrição:

- Caso não seja a primeira sprint da release, deverá ser coletado as possíveis melhorias apontadas na retrospectiva da Sprint anterior;
- Utilizar KANBAN para verificar as estórias pendentes (a fazer ou fazendo);
- Scrum Team se reúne para discutir quantas e quais estórias podem ser finalizadas no período da sprint atual, utilizando métodos de medição de valor/ esforço;
- Escrever no quadro de tarefas quais estórias foram escolhidas para serem feitas naquela sprint; e
- - Selecionar no KANBAN as estórias definidas para serem desenvolvidas nessa sprint, levando em consideração a prioridade estabelecida.

3.1.2 Executantes:

- Scrum Team e Scrum Master.

3.1.3 Artefato(s) de Insumo:

- *Backlog* Priorizado do Produto (*Cards* da *Release*).
- Itens Prontos do *Backlog* da *Sprint*.
- Melhorias para a próxima *Sprint*.

3.1.4 Artefato(s) Produzido(s):

- *Backlog* do Produto.

3.2 REALIZAR DESENVOLVIMENTO DO PRODUTO

3.2.1 Descrição:

- Por ser um subprocesso, será detalhado na etapa 4.

3.2.2 Artefato(s) de Insumo:

- *Backlog* da *Sprint*.

3.2.3 Artefato(s) Produzido(s):

- Itens Desenvolvidos pelo *Scrum Team*.

3.3 SPRINT REVIEW

3.3.1 Descrição:

- Verificar o progresso atual das tarefas planejadas para a sprint;
- Atualizar Backlog do Produto;
- Reunir Scrum Team com o Product Owner para transparecer o progresso atual e o que foi feito até o momento;
- Mostrar o que foi desenvolvido; e
- Caso validada, a Estória de Usuário vai para etapa "Validado" no KANBAN, com a assinatura do cliente no cartão.

3.3.2 Executantes:

- Scrum Team, Scrum Master e Product Owner.

3.3.3 Artefato(s) de Insumo:

- Itens Desenvolvidos pelo *Scrum Team*.

3.3.4 Artefato(s) Produzido(s):

- *Backlog* do Produto Atualizado.

3.4 REALIZAR A REUNIÃO DE RETROSPECTIVA DA SPRINT

3.4.1 Descrição:

- Scrum Team utiliza todos os dados coletados durante a sprint;
- Pontos positivos e sucessos durante a sprint são coletados;
- Pontos negativos e falhas durante a sprint são coletados;
- Sugestões de melhoria para a próxima SPRINT são coletados; e
- Todos os dados são escritos no Trello.

3.4.2 Executantes:

- Scrum Team e Scrum Master.

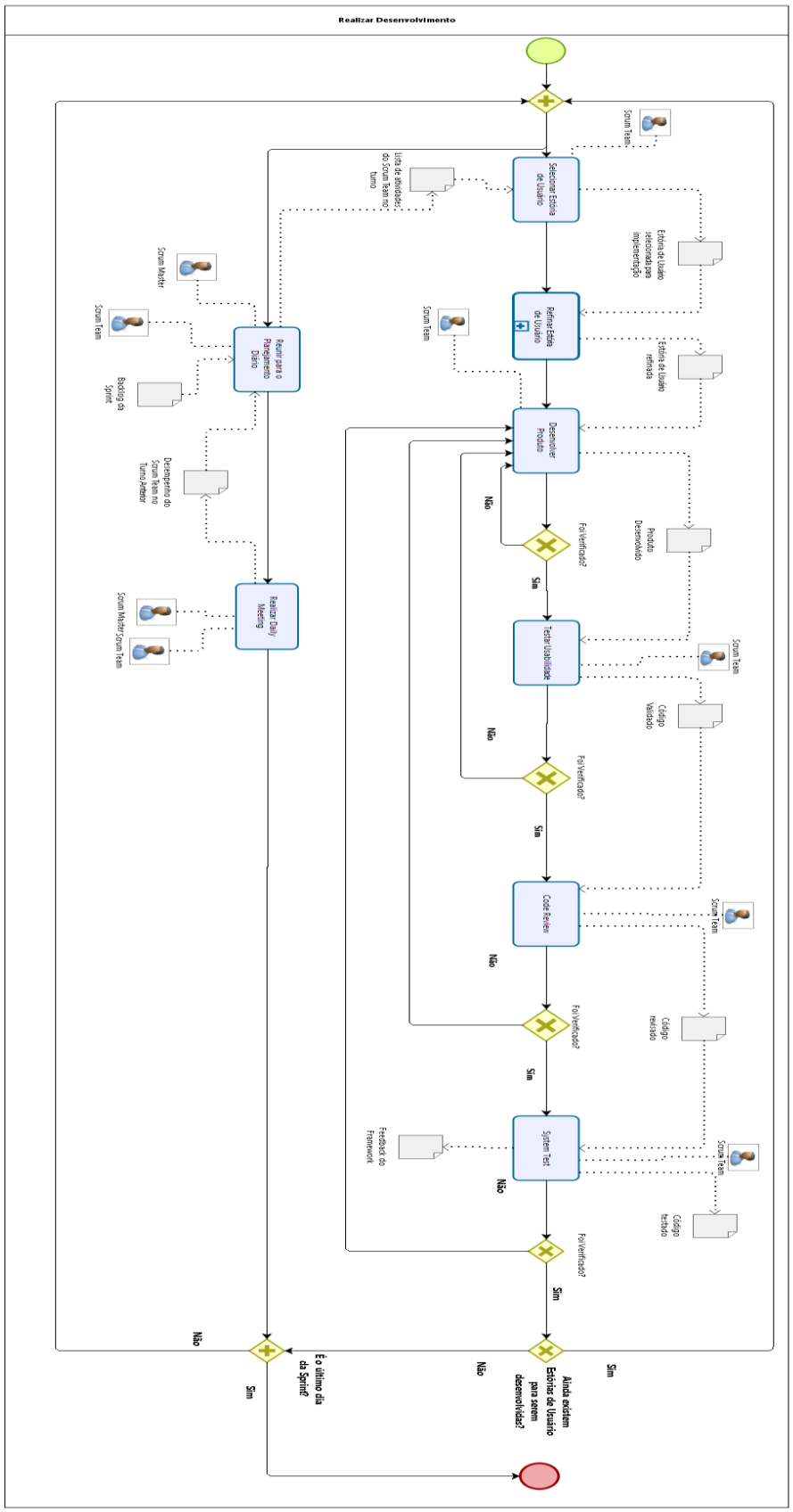
3.4.3 Artefato(s) de Insumo:

- *Backlog* do Produto Atualizado.

3.4.4 Artefato(s) Produzido(s):

- Avaliação do Desempenho do *Scrum Team*.
- *Feedback* do *Scrum Master*.
- *Feedback* do *Scrum Team*.
- Melhorias para a Próxima *Sprint*.

4 Realizar Desenvolvimento do Produto



Fonte: Autor, 2018.

4.1 REUNIR PARA O PLANEJAMENTO DIÁRIO

4.1.1 Descrição:

- Todos os membros presentes do time participam;
- Acontece no início do turno Matutino e do Vespertino;
- Definir o que será feito no turno, baseado no backlog da sprint e nas informações do KANBAN; e
- Priorizar tarefas e esforços baseado no desempenho e nas pendências descritas no dia anterior.

4.1.2 Executantes:

- Scrum Team e Scrum Master.

4.1.3 Artefato(s) de Insumo:

- Desempenho do *Scrum Team* no Turno Anterior.
- *Backlog* da *Sprint*.

4.1.4 Artefato(s) Produzido(s):

- Lista de Atividades do *Scrum Team* no Turno.

4.2 REALIZAR DAILY MEETING

4.2.1 Descrição:

- Tudo que foi realizado durante o dia é anotado, para servir de base para o planejamento diário seguinte; e
- Pendências e sugestões são anotadas.

4.2.2 Executantes:

- Scrum Team e Scrum Master.

4.2.3 Artefato(s) de Insumo:

- Por se tratar de um relatório das atividades no turno, não depende de artefatos de insumo.

4.2.4 Artefato(s) Produzido(s):

- Desempenho do *Scrum Team* no Turno Anterior.

4.3 SELECIONAR ESTÓRIA DE USUÁRIO

4.3.1 Descrição:

- Time de Desenvolvimento se organiza para realizar o desenvolvimento pareado nas estórias definidas como meta no Daily Meeting.

4.3.2 Executantes:

- Scrum Team.

4.3.3 Artefato(s) de Insumo:

- Lista de Atividades do *Scrum Team* no Turno.

4.3.4 Artefato(s) Produzido(s):

- Estória de Usuário Selecionada para Implementação.

4.4 REFINAR ESTÓRIA DE USUÁRIO

4.4.1 Descrição:

- Por ser um subprocesso, será detalhado na etapa 5.

4.4.2 Artefato(s) de Insumo:

- Estória de Usuário Selecionada para Implementação.

4.4.3 Artefato(s) Produzido(s):

- Estória de Usuário Refinada.

4.5 DESENVOLVER PRODUTO

4.5.1 Descrição:

- Mover o cartão para "Development" no KANBAN ;
- Utilizar a versão atualizada do código;

- Realizar o desenvolvimento;
- Desenvolvimento realizado dentro dos padrões de código estabelecidos;
- Desenvolvimento realizado de acordo com o estabelecido nas telas de protótipo;
- Desenvolvimento realizado de acordo com o fluxo criado no BPMN da Estória de Usuário;
- Caso o desenvolvimento tenha sido finalizado, realizar versionamento do código; e
- Criar testes Unitários para as funções desenvolvidas.

4.5.2 Executantes:

- Scrum Team.

4.5.3 Artefato(s) de Insumo:

- Estória de Usuário Refinada.

4.5.4 Artefato(s) Produzido(s):

- Produto Desenvolvido.

4.6 TESTAR USABILIDADE

4.6.1 Descrição:

- Mover o cartão para etapa "User Experience" no KANBAN;
- Verificar se as funcionalidades da estória foram devidamente implementadas;
- Testar a Affordance dos ícones e botões implementados; e
- Verificar se o que foi implementado está de acordo com as telas de prototipação e o fluxo desenvolvido.

4.6.2 Executantes:

- Scrum Team.

4.6.3 Artefato(s) de Insumo:

- Produto Desenvolvido.

4.6.4 Artefato(s) Produzido(s):

- Código Validado.

4.7 CODE REVIEW

4.7.1 Descrição:

- Mover o cartão para a etapa "Code Review" no KANBAN;
- Verificar se as linhas de código estão de acordo com o padrão de código utilizando o software de versionamento; e
- Realizar anotações em caso de sugestão de melhoria ou de regra infringida.

4.7.2 Executantes:

- Scrum Team.

4.7.3 Artefato(s) de Insumo:

- Código Validado.

4.7.4 Artefato(s) Produzido(s):

- Código Revisado.

4.8 SYSTEM TEST

4.8.1 Descrição:

- Mover o cartão para a etapa "System Test" no KANBAN;
- Gravar o fluxo das novas funcionalidades desenvolvidas; e
- Rodar o teste automatizado para verificar se a estória impactou em outras funcionalidades já existentes no sistema.

4.8.2 Executantes:

- Scrum Team.

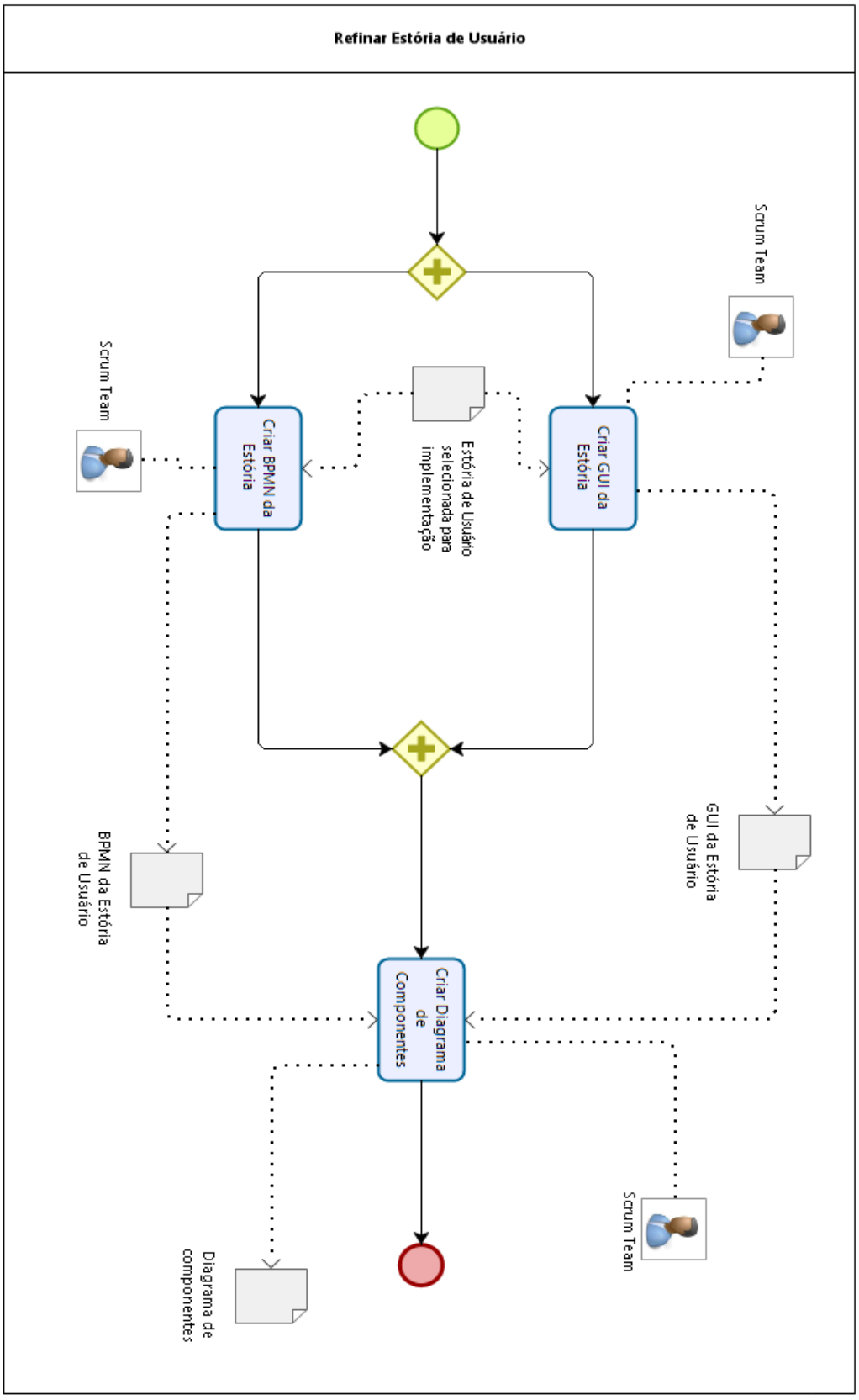
4.8.3 Artefato(s) de Insumo:

- Código Revisado.

4.8.4 Artefato(s) Produzido(s):

- Código Testado.
- *Feedback* do *Framework*.

5 REFINAR ESTÓRIA DE USUÁRIO



Fonte: Autor, 2018.

5.1 CRIAR GUI DA ESTÓRIA

5.1.1 Descrição:

- Criar Protótipos de telas para servir de guia para implementação da Estória;
- As telas criadas devem ser anexadas à estória no KANBAN; e
- É gerado o artefato *graphical user interface* (GUI) da Estória de Usuário, servindo como refinamento dessa estória.

5.1.2 Executantes:

- Scrum Team.

5.1.3 Artefato(s) de Insumo:

- Estória de Usuário Selecionada para Implementação.

5.1.4 Artefato(s) Produzido(s):

- GUI da Estória de Usuário.

5.2 CRIAR BPMN DA ESTÓRIA

5.2.1 Descrição:

- Criar BPMN que represente o fluxo da estória de usuário selecionada para implementação;
- Anexar BPMN à Estória de Usuário no KANBAN; e
- Utilizar o fluxo criado como base para o desenvolvimento da estória.

5.2.2 Executantes:

- Scrum Team.

5.2.3 Artefato(s) de Insumo:

- Estória de Usuário Selecionada para Implementação.

5.2.4 Artefato(s) Produzido(s):

- BPMN da Estória de Usuário.

5.3 CRIAR DIAGRAMA DE COMPONENTES

5.3.1 Descrição:

- Traçar a relação dos componentes identificados nas tarefas anteriores, através de um diagrama de componentes;
- Traçar também as dependências de interfaces externas, caso necessário;
- Anexar o diagrama à Estória de Usuário; e
- Utilizar o Diagrama como guia para o desenvolvimento da estória de usuário.

5.3.2 Executantes:

- Scrum Team.

5.2.3 Artefato(s) de Insumo:

- GUI da Estória de Usuário.
- BPMN da Estória de Usuário.

5.2.4 Artefato(s) Produzido(s):

- Diagrama de Componentes

APÊNDICE B – ENTREVISTA INICIAL REALIZADA COM O COORDENADOR DA COTIC

Entrevistado(a): Diego Assis da Silva Lisbôa.

Data: 03/08/2018

Local: COTIC-PROEG, UFPA.

Lucas: Boa tarde, Diego. Estou realizando essa entrevista para retirar algumas dúvidas e tentar mapear as etapas do processo de desenvolvimento da COTIC, pois irei realizar a modelagem do mesmo para produção do meu TCC, e gostaria de saber se você pode responder algumas perguntas e autoriza a utilização do processo que vocês utilizam para esse fim.

Diego: Boa tarde, Lucas. Você pode utilizar nosso processo para seu trabalho sim, irei lhe ajudar na medida do possível.

Lucas: Então, vamos lá. Primeiramente, como atual bolsista da COTIC, tenho conhecimento sobre boa parte das etapas que correspondem ao desenvolvimento de produto, entretanto também percebo que há ausência de artefatos e documentos que evidenciem algumas etapas do processo. A minha primeira pergunta, então, é: quais documentos posso utilizar como base para a modelagem que irei realizar? Existe algum trabalho em específico que me auxilie nessa etapa?

Diego: Realmente, a grande maioria dos documentos que produzimos envolvem a fase de desenvolvimento de produto. Entretanto, existem alguns trabalhos que tu podes pegar como orientação para o mapeamento do nosso trabalho. Um deles é o TCC da Jaily, que realiza um estudo de caso utilizando a modelagem do nosso processo pelo SPIDER_ML. O outro é o TCC do Juan, que realiza uma proposta de melhoria de processo através da modelagem com o Draw.io. Ambos estão no nosso drive, e posso disponibilizar pra realizares a modelagem.

Lucas: Certo, irei dar uma olhada nesses trabalhos. Outra dúvida, por estar presente na COTIC há algum tempo, sei que nosso processo já sofreu muitas mudanças, inclusive nos últimos meses. Esses trabalhos que tu mencionaste, acredito que não estão completamente atualizados com o modelo atual, correto?

Diego: De fato, tu tens que dar uma analisada nos trabalhos e verificar o que já mudou no nosso processo e o que continua correto a partir das modelagens. Mas acredito que as mudanças sejam pequenas, visto que o TCC do Juan foi realizado esse ano, portanto é a versão mais recente.

Lucas: Beleza Diego, obrigado. Vou realizar uma modelagem inicial do processo, e depois vou fazer outra entrevista pra discutir o que necessita de mudanças, beleza?

Diego: De nada Lucas. Quando tiveres com a modelagem inicial pronta, volta aqui que a gente verifica se tem alguma coisa a ser mudada.

APÊNDICE C – ENTREVISTA REALIZADA COM O COORDENADOR DA COTIC PARA FINALIZAR A MODELAGEM

Entrevistado(a): Diego Assis da Silva Lisbôa.

Data: 15/08/2018

Local: COTIC-PROEG, UFPA.

Lucas: Boa tarde, Diego. Estou de volta depois de fazer a modelagem inicial do processo, que já te mostrei. Direto ao assunto, um dos pontos que eu queria te perguntar é relacionado à atividade logo após a priorização das funcionalidades, no subprocesso de planejar release, chamada Selecionar Escopo, encontrada no trabalho do Juan. Particularmente, não entendi muito bem o motivo dessa atividade estar presente, e acabei retirando da modelagem, e queria saber se a minha decisão foi correta, ou se essa atividade deveria estar presente.

Diego: Boa tarde, Lucas. Cara, acredito que essa atividade já não está mais presente no nosso processo. Era uma etapa que a gente costumava fazer antes, mas decidimos excluir ela do processo, porque não tinha muito sentido selecionar um escopo pra funcionalidade, então você pode tirar essa atividade, sem problema.

Lucas: Entendido. A segunda dúvida foi no subprocesso Executar Sprint. De acordo com a modelagem do TCC do Juan, depois do planejamento da Sprint, tem a reunião para o planejamento diário, e depois dela vem o subprocesso de desenvolvimento do produto. Entretanto, a forma como foi modelada confunde um pouco o fluxo pois, pelo que eu entendo, temos duas atividades realizadas em cada turno, a reunião pra planejamento diário, que acontece no início do turno, e a *Daily Meeting*, que acontece no final do turno. Portanto, realizei uma mudança na modelagem, jogando essas duas atividades que mencionei para dentro do subprocesso de desenvolvimento, e essas atividades vão acontecer em paralelo ao desenvolvimento do produto. Tu achas que algo deve ser mudado, ou está certo desse jeito?

Diego: Então, nós realmente temos duas atividades, como tu dissestes. A reunião para planejamento diário seria uma atividade do XP, e o *Daily Meeting* é do *Scrum*. O Juan acabou juntando essas duas atividades em uma só, mas está correto do jeito que tu estás fazendo também, achei

interessante a forma como tu colocastes em paralelo ao desenvolvimento, podes deixar assim mesmo.

Lucas: Entendi. Última dúvida, na modelagem do Juan, ele divide quatro papéis presentes no processo: *Product Owner*, *Scrum Master*, *Scrum Team* e Time de Desenvolvimento. O problema é justamente com Time de Desenvolvimento. Atualmente, não vejo uma diferença entre desenvolvedores e o "resto" do time. No meu entendimento, todo mundo que faz parte do *Scrum Team* pode ser desenvolvedor ou realizar outras atividades, dependendo do que tiver que ser feito, não fazendo uma diferença de papéis dentro do time. Por causa disso, excluí o Time de Desenvolvimento, e tratei todo mundo como *Scrum Team*, tem algum problema?

Diego: Não tem problema nenhum, na verdade o Juan especificou quem era Time de Desenvolvimento mais para dividir melhor quem realiza qual tarefa, acredito eu. Mas realmente, está correto colocar todo mundo como *Scrum Team*, já que todos podem tanto desenvolver quanto realizar outras atividades.

Lucas: Beleza Diego, muito obrigado por responder minhas dúvidas.

Diego: De nada, Lucas. Qualquer coisa podes marcar uma reunião, vou estar disponível para tirar tuas dúvidas.