ULBRA
CENTRO UNIVERSITÁRIO LUTERANO DE PALMAS
Recredenciado pela Portaria Ministerial nº 1.162, de 13/10/16, D.O.U. nº 198, de 14/10/2016
AELBRA EDUCAÇÃO SUPERIOR - GRADUAÇÃO E PÓS-GRADUAÇÃO S.A.

Thamyres de Souza Silva

PROPOSTA DE INCLUSÃO DE TÉCNICAS DO BDD (BEHAVIOR DRIVEN TEST) NO PROCESSO DE DESENVOLVIMENTO DA FÁBRICA DE SOFTWARE CEULP/ULBRA

Palmas – TO

2021

Thamyres de Souza Silva

PROPOSTA DE INCLUSÃO DE TÉCNICAS DO BDD (BEHAVIOR DRIVEN TEST) NO PROCESSO DE DESENVOLVIMENTO DA FÁBRICA DE SOFTWARE CEULP/ULBRA

Trabalho de Conclusão de Curso (TCC) II elaborado e apresentado como requisito parcial para obtenção do título de bacharel em Ciência da Computação pelo Centro Universitário Luterano de Palmas (CEULP/ULBRA).

Orientador: Prof. Esp. Fábio Castro Araújo.

Palmas – TO

2021

Thamyres de Souza Silva

PROPOSTA DE INCLUSÃO DE TÉCNICAS DO BDD (BEHAVIOR DRIVEN TEST) NO PROCESSO DE DESENVOLVIMENTO DA FÁBRICA DE SOFTWARE CEULP/ULBRA

Trabalho de Conclusão de Curso (TCC) II elaborado e apresentado como requisito parcial para obtenção do título de bacharel em Ciência da Computação pelo Centro Universitário Luterano de Palmas (CEULP/ULBRA).

Orientador: Prof. Esp. Fábio Castro Araújo.

Aprovado em: _____/_____/_____

BANCA EXAMINADORA

_____

Prof. Esp. Fabio Castro

Orientador

Centro Universitário Luterano de Palmas – CEULP

_____

Prof. M.e Fabiano Fagundes

Centro Universitário Luterano de Palmas – CEULP

_____

Prof. M.e Jackson Gomes de Souza

Centro Universitário Luterano de Palmas – CEULP

Palmas – TO

2021

AGRADECIMENTOS

Agradeço a Deus e aos guias espirituais por me ajudarem e iluminarem meus caminhos até aqui, me dando forças para sempre seguir em frente.

Ao CEULP/ULBRA e aos professores pelo excelente ensino e dedicação a todos os alunos.

Ao Prof. Esp. Fábio Castro, que aceitou ser meu orientador. Muito obrigada por confiar em mim, por me ensinar e me fazer acreditar na minha capacidade e nunca permitir que eu cedesse à tentação de desistir.

Agradeço aos meus amigos Alexandre, Augusto, Larisse e Muriel por todo o apoio durante o desenvolvimento deste trabalho.

Quero agradecer também a todos os funcionários do Restaurante Meninas Gerais, a dona Maria Cristina dos Reis, Ana Lúcia dos Reis e Aparecida de Fátima, mulheres com um enorme coração gentil e acolhedor. Agradeço pelo café e os pães de queijo, por todo o apoio, por todos os momentos de motivação, que saudade daquelas conversas no restaurante que animavam o meu dia. O meu muito obrigado!

Dedico este trabalho de conclusão de curso aos meus pais, Rubens Júnior e Marineira, à minha tia Maria José (*in memorian*), meus irmãos, Rubens Neto e Andrey Gabriel por sempre me apoiarem e me amarem incondicionalmente, e a todos os meus amigos e familiares que, direta ou indiretamente, me ajudaram a caminhar até aqui.

A todos que, de alguma forma, colaboraram para a realização deste trabalho.

RESUMO

SILVA, Thamyres de Souza. **Proposta de inclusão de técnicas BDD (Behavior Driven Test) no processo de desenvolvimento da Fábrica de Software CEULP/ULBRA**. 2021. 42 f. Trabalho de Conclusão de Curso (Graduação) – Curso de Ciência da Computação, Centro Universitário Luterano de Palmas, Palmas/TO, 2021[1].

O BDD (*Behavior Driven Development*) é um método de desenvolvimento ágil que surgiu como evolução do TDD (*Test Driven Development* - Desenvolvimento Guiado a Testes) , tendo como característica unir os membros da equipe de desenvolvimento, no intuito de atingir um entendimento compartilhado da forma como o *software* deve ser desenvolvido, podendo ser aplicado tanto em equipes de pequeno porte, quanto em times de maior número. Este trabalho propõe o reuso de cenários de teste automatizados como uma alternativa de somar ainda mais o processo de desenvolvimento da Fábrica de *Software* CEULP/ULBRA, por meio da inserção de técnicas do BDD (*Behavior Driven Test*). Para esse fim, propôs-se uma metodologia para o reúso de cenários de teste suportada por um *framework* de automação de testes, o *Behave*, onde os cenários BDD serão escritos na linguagem *Gherkin*. Através do levantamento de requisitos as histórias são extraídas e as funcionalidades são escritas no formato de cenários, possuindo sua própria estrutura (dado, quando e então), para que os testes possam ser realizados de acordo com os comportamentos previstos das funcionalidades. Como resultado tem-se uma nova metodologia para desenvolvimento da Fábrica que agrega elementos de BDD sem fazer grandes alterações a forma atual de desenvolvimento. Dentre os novos artefatos adaptados tem-se a adição de histórias e cenários do usuário no *Product Backlog* e *Sprint Backlog*, respectivamente.

Palavras-chaves: BDD, Testes de Software, Metodologias Ágeis.

1

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

BDD - *Behavior Driven Development*

TDD - *Test Driven Development*

CEULP - Centro Universitário Luterano de Palmas

HTML - *Hypertext Markup Language*

CSS - *Cascading Style Sheets*

PO - *Product Owner*

SUMÁRIO

1 INTRODUÇÃO

O *Behavior Driven Development* (BDD), ou Desenvolvimento Guiado por Comportamento, é uma prática de desenvolvimento ágil de *software* que evoluiu com o passar do tempo para alcançar um cenário mais abundante da análise ágil e teste de aceitação automatizada (NORTH, 2016, Tradução de Gabriel Olivério). É uma prática que envolve os membros da equipe para abordar o comportamento esperado do sistema ou funcionalidade, buscando o entendimento compartilhado. O BDD é a evolução de *Test Driven Development* (TDD) ou Desenvolvimento Guiado por Testes, onde primeiro os testes são elaborados e depois o código é escrito (SOARES, 2011). Essa prática pode ser aplicada tanto em ambientes de desenvolvimento com poucos integrantes - como a Fábrica de *Software* do Centro Universitário Luterano de Palmas (CEULP/ULBRA) -, quanto em ambientes de desenvolvimento com muitos integrantes.

A Fábrica de *Software* é o centro de desenvolvimento de soluções tecnológicas do CEULP/ULBRA. Nela são desenvolvidas soluções *web* para as demandas da instituição, como *softwares*, páginas *web* etc. E a Fábrica de *Software* utiliza a metodologia ágil *SCRUM*.

O *SCRUM* é um *framework* para o gerenciamento e desenvolvimento de produtos que propõe um conjunto de conceitos e práticas que se adequam no desenvolvimento de produtos e foca numa entrega de valor mais rápida e desde o início (CRUZ, 2013). Vem sendo utilizado em diferentes mercados, de multinacionais a startups, como por exemplo empresas de *marketing* e de desenvolvimento de *hardware* (SABBAGH, 2014).

Entretanto, mesmo que essas metodologias ágeis sejam aplicadas, a fase de testes acaba ficando omissa e/ou não é realizada. Para isso, pode-se utilizar práticas ágeis de testes como BDD que por ser um processo que reduz custos, foca na entrega de funcionalidades, e permite que mudanças sejam feitas de forma mais práticas e seguras, proporciona melhor comunicação, aproxima os membros dos testadores e faz com que a equipe mantenha o mesmo nível de informação durante o desenvolvimento. Isso diminui a chance de gerar um *software* mal testado que pode acarretar em elevação de custos e falhas no sistema. Com base nisso, o presente trabalho teve como objetivo elaborar uma proposta para a utilização de técnicas do BDD (*Behavior Driven Test*) para o processo de desenvolvimento da Fábrica de *Software* do CEULP/ULBRA.

No contexto da Fábrica de *Software*, o BDD irá gerar testes antes de desenvolver o sistema, esses testes serão escritos no formato de história de usuário e cada cenário será escrito no seguinte formato: dado, quando e então. Além disso, por meio da inserção de técnicas BDD na Fábrica de *Software*, será possível reutilizar cenários de testes automatizados

como uma alternativa de somar ainda mais o processo de desenvolvimento da Fábrica, estabelecendo os comportamentos previstos das funcionalidade, de acordo com os cenários identificados.

Portanto, realizou-se uma análise do ambiente de desenvolvimento da Fábrica de *Software* do CEULP/ULBRA, elaborou-se um relatório com a descrição do processo de desenvolvimento atual da Fábrica, propôs-se aplicação de Técnicas do BDD para o processo de desenvolvimento da Fábrica e desenvolveu-se um relatório com a descrição da aplicação das técnicas propostas disponibilizada em forma de páginas *Web* . A conclusão de todos esses objetivos possibilita a Fábrica de Software uma nova metodologia para desenvolvimento de software que agrega as principais características do *SCRUM* com elementos do BDD.

2 REFERENCIAL TEÓRICO

Esta seção apresenta a base teórica utilizada para elaboração deste trabalho, abordando assuntos como Testes de *Software,* Processo de Desenvolvimento, Metodologias Ágeis, Teste Guiado pelo Comportamento e Trabalhos Relacionados.

2.1 Testes de software

Testes de *software* é o processo que avalia a qualidade do sistema, indicando se as especificações determinadas pelo usuário final foram atingidas (DIAS NETO, 2007). Através dos testes, os erros/falhas no sistema podem ser apontados, identificados e corrigidos. Segundo Crespo et al. (2004, p. 02) "ainda que as técnicas de teste de *software* mais utilizadas foram criadas por volta dos anos 70, as empresas têm uma grande dificuldade com a atividade de teste". Isso representa a situação atual que as empresas têm para implantar/melhorar o processo de teste utilizado, representa uma possível falta de profissionais especializados na área de teste de *software* (CRESPO et al., 2004).

Durante as décadas de 60 e 70 os desenvolvedores dedicavam cerca de 80% dos seus esforços nas atividades de codificação e nos testes unitários e somente na década de 80 passaram a dar maior importância à análise de requisitos, ao desenho funcional e técnico dos novos sistemas (RIOS; MOREIRA, 2013). De acordo com Dias Neto (2007) é preciso entender a diferença entre defeito, erro e falha, conforme a norma *Institute of Electrical and Electronics Engineers* (IEEE 610, 1990), que determina o erro como um comportamento do *software* diferente do previsto, podendo ser resultado de uma ação humana incorreta/inesperada, este pode resultar em falhas, e por fim, o defeito que é o *software* fora dos padrões de requisitos/especificações.

Rios e Moreira (2013, p.10) afirmam que não se pode garantir que um programa ficará livre de bugs, já que é quase impossível testar todas as possibilidades de entrada de dados e que é no processo de testes que os investimentos para a qualidade do *software* são realizados. Testar *software* é um processo caro que exige que os profissionais sejam especializados, conheçam técnicas de teste adequadas, possuam facilidade na implantação do processo de teste, entre outras. Crespo (2004) afirma que a ausência de algumas dessas características citadas dificulta a testagem de *softwares*.

2.2 Processos de Desenvolvimento

Para Sommerville (2019) um processo de *software* ou processo de desenvolvimento de software, é uma sequência de atividades necessárias para criar um sistema, e existem quatro atividades fundamentais que devem ser incluídas no processo de desenvolvimento, são elas:

- especificação de *software*: os requisitos e restrições são definidos pelos clientes e engenheiros;
- projeto e implementação de *software*: o software é projetado e programado;
- validação de *software*: o *software* é verificado e testado para garantir que esteja fazendo o que o cliente solicitou;
- evolução de *software*: o *software* precisa evoluir para atender os requisitos do cliente e do mercado.

Várias organizações pequenas e médias não utilizam nenhum processo de desenvolvimento por não possuírem recursos suficientes para adotar o uso de processos mais adequados para a produção do *software*, acarretando em um produto final de baixa qualidade, além de prejudicar a entrega do sistema no prazo definido e sua evolução futura (SOARES, 2004). Segundo Sommerville (2019) por serem atividades complexas, elas incluem subatividades como validação dos requisitos e projeto de arquitetura, e quando os processos estão sendo discutidos é importante descrever quem está envolvido, as atividades e a sequência das atividades.

Soares (2004) afirma que é comum que empresas criem/adaptem os processos à sua realidade, já que existem vários definidos na Engenharia de Software, entre esses processos, há as metodologias tradicionais (orientadas a documentação) e as metodologias ágeis (desenvolve *softwares* com o mínimo de documentação).

2.3 Metodologias ágeis

Foi através do surgimento de propostas para que houvesse um foco mais ampliado nas pessoas do que em processos de desenvolvimento que as metodologias ágeis surgiram, e vêm sendo indicadas como uma alternativa ao modelo de gestão tradicional de projetos (SOARES, 2004). Para Santos, Soares e Caldeira (2013) metodologias ágeis têm como uma de suas características a adaptabilidade, ou seja, se adaptam às mudanças que ocorrem no desenvolvimento do projeto, e trabalham com *feedback* contínuo por meio de entregas rápidas e frequentes das partes do *software*, que é um ponto positivo.

Sommerville (2019) afirma que os métodos ágeis são constituídos por diferentes processos e compartilham o mesmo conjunto de princípios baseado no manifesto ágil. Abaixo tem a representação desses princípios na Tabela 1.

Tabela 1 - Princípios dos Métodos Ágeis

Princípio	Descrição

Envolvimento do cliente	Os clientes devem ser envolvidos em todo o processo de desenvolvimento. Seu papel é fornecer e priorizar novos requisitos de sistema e avaliar as iterações do sistema.
Acolher as mudanças	Tenha em mente que os requisitos do sistema mudam e, portanto, deve-se projetar o sistema para acomodar essas mudanças.
Entrega incremental	O software é desenvolvido em incrementos, e o cliente especifica os requisitos incluídos em cada um deles.
Manter a simplicidade	Deve-se ter como foco a simplicidade, tanto do software que está sendo desenvolvido quanto do processo de desenvolvimento. Sempre que possível, trabalhe ativamente para eliminar a complexidade do sistema.
Pessoas, não processos	As habilidades do time de desenvolvimento devem ser reconhecidas e aproveitadas da melhor maneira possível. Seus membros devem ter liberdade para desenvolver seu modo próprio de trabalhar sem se prender a processos determinados.

Fonte: Sommerville (2019)

A Tabela 1 apresenta uma tabela com os princípios dos métodos ágeis e a descrição de cada um. Para Sommerville (2019) os métodos ágeis funcionam bem em situações como, o desenvolvimento de um produto pequeno ou médio (por uma empresa de *software*), e o desenvolvimento de sistemas personalizados para uma organização, já que promove uma constante comunicação entre o gerente do produto/cliente do sistema e o time de desenvolvimento.

2.4 Teste guiado pelo comportamento (BDD)

Segundo Soares (2011, p. xx) “Kent Beck apresentou ao mundo em 2003, através do seu livro “*Test-Driven Development*”, uma técnica para criar sistemas baseados em testes que visa garantir a qualidade e a funcionalidade do *software*”. Ou seja, que tem como objetivo descobrir se os requisitos do sistema descritos através do comportamento atendem as

necessidades do cliente. De acordo com Soares (2011) o *Behaviour Driven Development* (BDD) é a evolução do TDD.

Smart (2014, tradução nossa) afirma que o BDD auxilia as equipes a dedicar seu empenho na identificação, compreensão e construção de recursos valiosos para as empresas e assegura que esses recursos sejam bem projetados e implementados. É por meio do uso de uma linguagem ubíqua (linguagem comum) que os testes são escritos e entendidos com facilidade, o que permite que os desenvolvedores evitem focar em detalhes técnicos e foquem em por que o código deve ser criado (ROCHA, 2013).

O BDD é uma prática bastante colaborativa, tanto entre os usuários e a equipe de desenvolvimento, quanto dentro da própria equipe (SMART, 2014, Tradução nossa). O BDD incentiva a interação entre negócio, desenvolvedores, qualidade e usuários finais em um projeto. Smart (2014, Tradução nossa) afirma que analistas de negócios, desenvolvedores e testadores trabalham junto com os usuários finais para definir e especificar recursos, e os membros da equipe extraem ideias de sua experiência e conhecimento individuais.

Segundo ENGEL, RICE e JONES [entre 2012 e 2019] é através do diálogo com as partes interessadas no projeto que o BDD é capaz de obter uma compreensão clara do comportamento desejado do *software*. Chelimsky e Astels (2010, Tradução nossa) afirmam que o BDD visa ajudar a comunicação, simplificando a linguagem utilizada para descrever os cenários nos quais o *software* será usado:

- dado (*given*) algum contexto;
- quando (*when*) algum evento ocorre;
- então (*then*) visa mostrar algum resultado.

Dado, quando e então são as estruturas básicas utilizadas pela linguagem do BDD. São palavras simples, que podem ser utilizadas para falar sobre o comportamento do aplicativo ou comportamento do objeto e, são facilmente compreendidos por analistas de negócios, testadores e desenvolvedores (CHELIMSKY; ASTELS, 2010, Tradução nossa).

As ferramentas BDD podem ajudar a transformar os requisitos em testes automatizados que ajudam a orientar o desenvolvedor, verificar o recurso e documentar o que o aplicativo faz (SMART, 2014, tradução nossa). Soares (2011) afirma que durante o levantamento de requisitos as histórias são extraídas, gerando a linguagem de negócios usada em BDD. Os exemplos para que as funcionalidades do sistema sejam desenvolvidas são criados através da colaboração do usuário e são escritos em formato de cenários, possuindo sua própria estrutura para que os testes possam ser realizados (ROQUETTE, 2018).

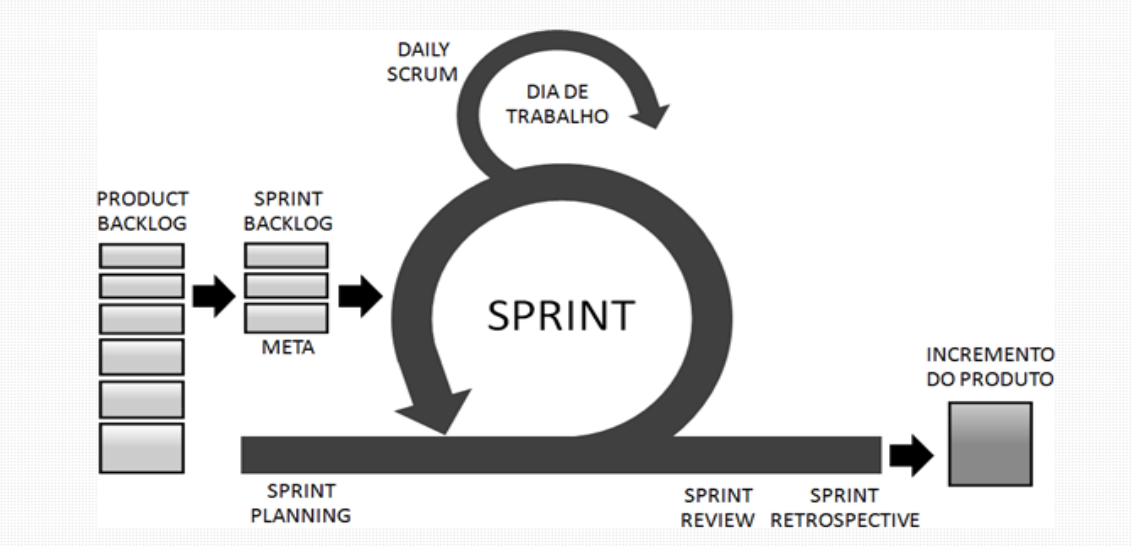
2.5 SCRUM

De acordo com Sabbagh (2014) o *Scrum* teve início nos anos 1990, mas só se tornou popular nos anos 2000, superando os métodos tradicionais e se tornando a forma mais usada em projetos de desenvolvimento de *software*. *Scrum* é uma teoria de controle de processo empírica que usa uma abordagem iterativa e incremental para otimizar a previsibilidade e controlar o risco, e o controle do processo experimental consiste em três valores centrais: transparência, inspeção e adaptabilidade (SUTHERLAND e SCHUWABER, 2013).

O *Scrum* é um *framework* que se ajusta a diferentes circunstâncias se utilizado junto com outras técnicas e práticas, e seus benefícios incluem: entregas frequentes de partes do produto, redução dos riscos do projeto, maior qualidade no produto gerado, mudanças utilizadas como vantagem competitiva, visibilidade do progresso do projeto, redução do desperdício e aumento de produtividade (SABBAGH, 2014). De acordo com as afirmações de Sutherland e Schuwaber (2011) o *framework Scrum* é simples de implementar e descompacta automaticamente e incentiva equipes de desenvolvimento de *software* a implantar as melhores práticas documentadas em *Organizational Patterns of Agile Development* (Padrões Organizacionais de Desenvolvimento Ágil).

É necessário que o início do planejamento do produto de *software* e a elicitação de requisitos estejam bem definidos e direcionados, e que o cliente esteja presente nas etapas iniciais dando *feedback* constante ou definindo novos requisitos a cada ciclo, e para isso o Scrum possui etapas bem definidas que auxiliam na mudança de cada ciclo conforme mostra a Figura 1 (DEVMEDIA, 2015).

Figura 1 – Ciclo do *Scrum*



Fonte: DEVMEDIA (2015)

Na Equipe Scrum existem três funções: *The Product Owner* (O Dono do Produto), *The Team* (A Equipe) e o *Scrum Master* (SUTHERLAND e SCHUWABER, 2011). A Figura 1 apresenta as etapas do processo utilizado pelo Scrum. A seguir as etapas serão especificadas conforme Sutherland e Schuwaber (2007):

- *Product Backlog*: uma lista de requisitos refinados e priorizados do produto no qual a prioridade dos requisitos são definidas pelo *Product Owner*;
- *Sprint Backlog*: as tarefas do projeto são divididas e registradas;
- *Sprint*: ocorre o desenvolvimento do incremento de um produto ao final de cada *Sprint,* e as tarefas são executadas de acordo com o tempo estipulado;
- *Daily Meet*: reunião curta com a equipe (mais ou menos 15 minutos), que acontece todos os dias em um horário determinado, onde cada membro da equipe relata três coisas que fizeram:
    - O que eles foram capazes de fazer desde a última reunião;
    - O que eles planejam determinar desde a última reunião;
    - Quais impedimentos que estão em seu caminho.
- Dia de trabalho:
- *Sprint Planning*: ocorre a reunião de planejamento da sprint (*Sprint Planning Meeting*), que são divididas em duas sub-reuniões distintas:
    - *Sprint Planning Part One*: o *Product Owner* e a Equipe revisam os itens de alta prioridade, discutem os objetivos, revisam a "Definição do Produto" (que foi estabelecida anteriormente) que todos os itens devem atender. A Parte Um enfoca a compreensão do que o *Product Owner* deseja;
    - *Sprint Planning Part Two*: planejamento detalhado de tarefas sobre como implementar os itens que a equipe decide assumir, começando com os itens (retirados do *Product Backlog*) de maior prioridade para o *Product Owner*.
- *Sprint Review*: no final de cada *Sprint*, há a *Sprint Review* (Revisão da *Sprint*), onde a Equipe e o *Product Owner* revisam a *Sprint*. *Sprint Review* é uma atividade de inspeção e adaptação para o produto. É o momento para o *Product Owner* aprender o que está acontecendo com o produto e com a Equipe e para a equipe aprender o que está acontecendo com o *Product Owner* e o mercado.

- *Sprint Retrospective*: onde a equipe discute sobre o que está funcionando e o que não está funcionando para chegar a um acordo sobre quais mudanças deverão ser feitas. A equipe e o *Scrum Master* comparecerão, e o *Product Owner* é bem-vindo, mas não é obrigado a comparecer.

Deemer et al. (2010) afirmam que *Scrum* não resolve os problemas de desenvolvimento, os torna visíveis e fornece uma estrutura para a equipe explorar maneiras de resolver problemas em ciclos curtos e com pequenas tentativas de melhoria, esse é o mecanismo básico que produz os benefícios mais significativos que as equipes que usam o Scrum experimentam.

2.6 Trabalhos Relacionados

Nesta subseção serão apresentados trabalhos que utilizaram o *Behavior Driven Development* (BDD) ou Teste Guiado pelo Comportamento. Em Oliveira Neto (2019) é analisado como fazer uso das práticas de desenvolvimento de jogos utilizando o BDD, já que até o momento atual existem poucos estudos apontando os benefícios e limitações durante o uso do BDD no ciclo de desenvolvimento de jogos.

Foi realizada uma pesquisa para saber a opinião dos desenvolvedores de jogos independentes, de modo a identificar o uso dessas práticas durante o seu dia a dia, entender como esses profissionais utilizam testes de *software* em jogos e quais tópicos/pontos ponderam ao realizar os testes, constatando assim as particularidades que irão beneficiar os desenvolvedores de jogos independentes e que são fornecidas através da utilização do BDD (OLIVEIRA NETO, 2019).

Em Matte (2011) é apresentado um estudo sobre teste de *software* em metodologias ágeis através da abordagem da técnica BDD, apresentando tópicos que intercalam temas de qualidade de *software* com o objetivo de mostrar que testar o produto é uma atividade essencial ao processo de desenvolvimento. Nesse sentido, Anderle (2015) apresenta um estudo a introdução de BDD como melhoria de processo de desenvolvimento ágil, no qual os resultados e conclusões da introdução dessa técnica são apontados, como melhoria de um processo em que o projeto utiliza a metodologia ágil *SCRUM*.

Já em Reali (2015) é apresentado um estudo de caso do desenvolvimento de uma aplicação *web*, e através desse estudo é criado um *framework* de avaliação baseado em mensuração, que tem como objetivo possibilitar a comparação entre o desenvolvimento segundo a metodologia BDD e a uma estrutura adaptada do Processo Unificado.

No trabalho de Albiero (2017) é proposta uma abordagem para a utilização de arquivos de *layout* de aplicações com objetivo de extrair informações sobre os componentes da interface de aplicativos nativos para *Android*.

3 METODOLOGIA

3.1 MATERIAIS

Nesta seção serão apresentados todos os materiais utilizados no presente trabalho.

3.1.1 GHERKIN

Gherkin é a linguagem que o BDD utiliza através da ferramenta *Behave* e que, por meio da criação de histórias e cenários, descreve testes de aceitação (ARAUJO, 2017). Conforme *Gherkin* (2020, Tradução nossa), atualmente essa linguagem é implementada para as plataformas mencionadas abaixo:

- *JavaScript*
- *Java*
- *Python*
- *Go*
- *Perl*
- *.NET*
- *Ruby*
- *Objective-C*

O *Gherkin* pode ser usado como uma biblioteca ou através de uma interface de linha de comando (CLI) já que foi projetado para ser utilizado composto por outras ferramentas como o *Gherkin-Lint* ou *Behave* (GHERKIN, 2020, Tradução nossa). Para Cezerino e Nascimento (2016) a linguagem serve para documentação e automação de testes e entre algumas definições da estrutura do seu arquivo inclui as seguintes palavras-chave:

- *scenario*: palavras que identificam o comportamento de um cenário são, *given, the*, *when, and e but*;
- *feature*: deve ser seguida por um dos termos *scenario, background* ou *scenario outline*;
- *background;*
- *given;*
- *when;*
- *then;*
- *and;*
- *but*

A utilização desta linguagem na Fábrica de *Software* poderá auxiliar na comunicação da equipe através da sua notação simples e fácil de entender, e através da descrição do

comportamento que uma *feature* deve ter. O uso do arquivo em *Gherkin*, além de auxiliar na documentação, foi uma fase da automatização de testes dentro do *framework Behave*.

3.1.2 HTML

Hypertext Markup Language (HTML) ou Linguagem de Marcação de Texto permite inserir documentos como títulos, parágrafos, *links* e formulários, e é uma linguagem escrita na forma de *tags* (que também são chamados de elementos), outras linguagens podem ser usadas junto com HTML, como o *JavaScript* e PHP (FLATSCHART, 2011). O HTML foi inspirado em uma linguagem que codificava documentos industriais complexos utilizados por empresas gigantescas e organizações governamentais, a linguagem é denominada *Standard Generalized Markup Language* (SGML) ou Linguagem de Marcação Padrão Generalizada da IBM (LEWIS; MOSCOVITZ, 2010).

Segundo Torres (2018) o HTML possibilita através da definição de um conjunto de códigos determinar onde cada elemento deverá aparecer na página *web*. Manzano (2010) afirma que o texto da página é escrito através de comandos específicos chamados *tags* e que essas *tags* frequentemente são utilizadas em dupla, por exemplo, <*tag*> (abertura) <*/tag*> (fechamento), dentro do par de *tags* fica o conteúdo da página como o texto e imagem.

O HTML foi usado na criação da estrutura do conteúdo web, como os cabeçalhos, as seções, parágrafos, aplicações do sistema iterativo para a demonstração das técnicas BDD que serão utilizadas na Fábrica de *Software* CEULP/ULBRA.

3.1.3 CSS

Cascading Style Sheets (CSS) ou Folha de Estilo em Cascata é uma linguagem usada para descrever a aparência de um documento que utiliza linguagem de marcação como o HTML, essa formatação pode ser na fonte e/ou na cor da letra e/ou da página, por exemplo (FLATSCHART, 2011). Gavazoni e Romani (2013) afirmam que a utilização do CSS proporciona várias vantagens, como estilos de páginas *web* mais refinadas e diferentes, e que a causa da redução no tempo de manutenção de uma página *web* e no trabalho com a sua criação e formação ocorreu devido ao uso desses estilos para a definição de *layout* de documentos HTML.

O CSS foi utilizado para descrever a página *web* do site que foi criado para apresentar as técnicas BDD que serão usadas na Fábrica de *Software*, definindo como serão exibidos os elementos da documentação como, por exemplo, menus em cascata.

3.1.4 JavaScript

JavaScript é uma linguagem de programação que pode agregar mais interatividade e controle em páginas HTML e, tem como uma das suas características validar as informações

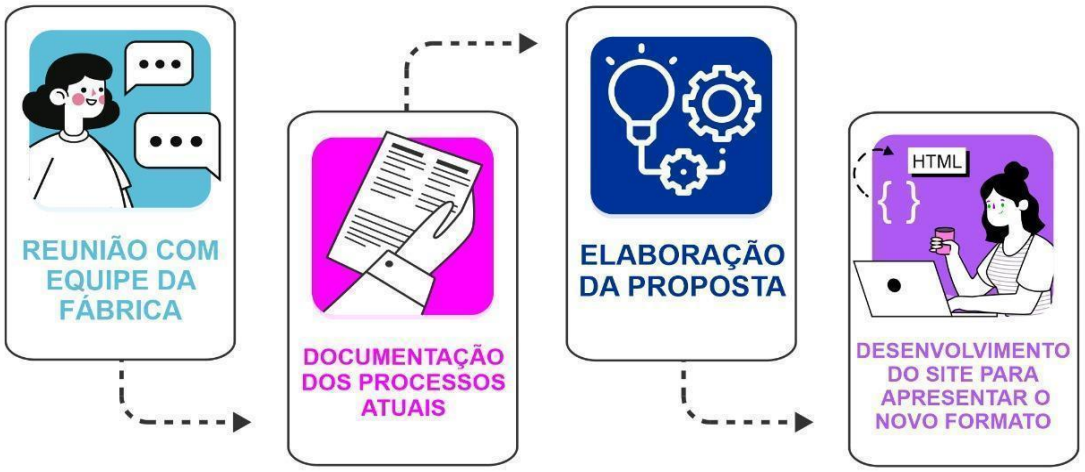
do formulário antes dele ser enviado (FLATSCHART, 2011). Para Bortolossi (2012) além do *JavaScript* proporcionar incorporar e adaptar a apresentação e o conjunto de conceitos e ideias dos diversos elementos que constitui o documento, vários recursos são disponibilizados pela linguagem, tais como: campos de entrada, botões e seletores.

O *JavaScript* é uma linguagem de programação *web client-side* (lado do cliente), ou seja, que comparada a outras linguagens como C++ e Java, tem uma sintaxe menos complexa e comportamento dinâmico (SILVA e SOBRAL, 2017). Silva e Sobral (2017) afirmam que em maio de 2017 o *JavaScript* foi considerado uma das linguagens de programação mais populares, de acordo com o ranking PYPL, alcançando o Top 5 (apud PYPL, 2017).

3.2 Métodos

Para o desenvolvimento deste trabalho, os processos foram divididos em etapas. A composição da metodologia de desenvolvimento é apresentada na Figura 2.

Figura 2 - Metodologia do trabalho



A Figura 2 apresenta as etapas realizadas para o desenvolvimento deste trabalho e para garantir que os objetivos definidos sejam realizados corretamente. As etapas da metodologia são detalhadas abaixo:

1. **Reunião com a equipe da Fábrica:** nesta etapa foi realizado reuniões com a equipe da Fábrica de *Software* CEULP/ULBRA para o levantamento do processo de desenvolvimento da Fábrica;
2. **Documentação dos processos atuais:** após o levantamento dos dados, foi elaborado uma documentação com os processos atuais da Fábrica de *Software*;
3. **Elaboração da proposta:** em seguida, foi elaborada uma proposta com as técnicas do BDD que serão aplicadas na Fábrica de *Software*.

4. **Desenvolvimento do site para apresentar o novo formato:** a última etapa consiste no desenvolvimento de um site com a descrição das técnicas do BDD que serão aplicadas na Fábrica de *Software*. O site foi desenvolvido com a utilização das seguintes linguagens: Gherkin (Linguagem que o BDD utiliza), *JavaScript*, CSS e HTML.

As etapas para o desenvolvimento do presente trabalho foram pensadas para a proposta de inclusão de técnicas BDD na Fábrica de *Software* CEULP/ULBRA, passando pela reunião da equipe, documentação dos processos atuais da Fábrica, elaboração da proposta e por fim o desenvolvimento do site para apresentar o novo formato.

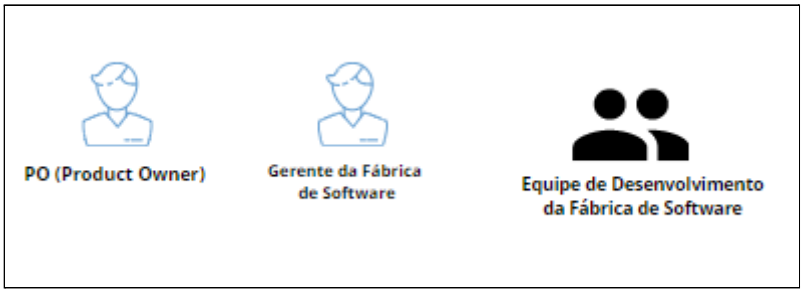
4 RESULTADOS

Esta seção tem por objetivo descrever e apresentar o fluxo de trabalho realizado para modelagem dos processos propostos.

4.1 Papéis, Fluxos de Trabalho e Estrutura Atual da Fábrica

A Fábrica de *Software* do CEULP/ULBRA é constituída pelo *Scrum Master*, *Product Owner* e por uma equipe de desenvolvimento composta de dois membros, conforme a Figura 4.

Figura 3 - Papéis da Estrutura Atual da Fábrica



Os Papéis da Fábrica de *Software* são detalhados a seguir:

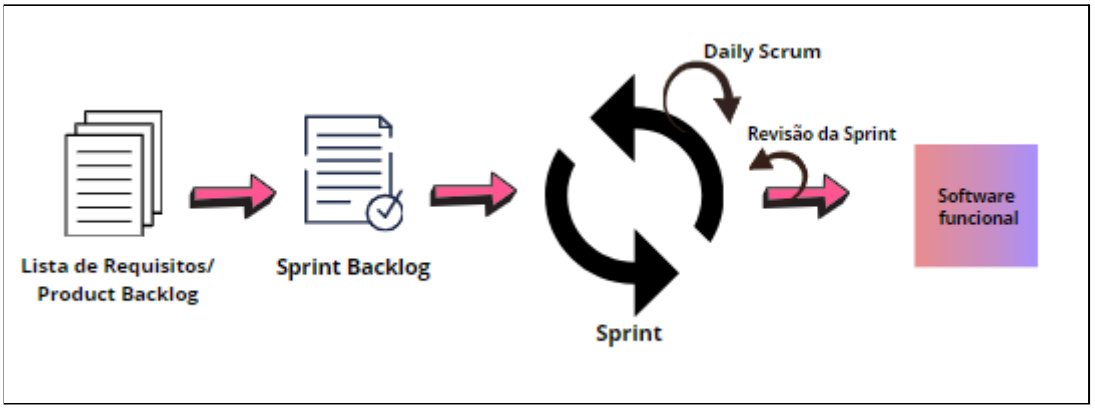
- *Product Owner* é responsável por trabalhar com gerenciamento de produto;
- *Scrum Master* (Gerente da Fábrica), responsável pelo gerenciamento da Fábrica;
- Equipe de desenvolvimento composta por dois membros responsáveis por desenvolver o produto/resolver bugs, são uma equipe multifuncional com um conjunto de habilidades diferentes.

A equipe da Fábrica de *Software* do CEULP/ULBRA é responsável por alguns sistemas da instituição, dos quais os principais são o Conecta e o SIG. O SIG é o sistema de informações gerenciais que tem informações do sistema de eventos e de infraestrutura. O Conecta está ligado à parte acadêmica, que está em constante mudança, e gera mais alterações no sistema devido a parte acadêmica ter constantes alterações a cada semestre, por exemplo, a alteração na carga horária das disciplinas e o aumento das *web* atividades. Na maior parte do tempo, o trabalho da equipe de desenvolvimento se concentra em alterações para o Conecta, entretanto, atualmente esse esforço tem sido dividido para que o SIG possa atender algumas das funções acadêmicas que hoje estão apenas no Conecta.

Além das atividades de desenvolvimento dos sistemas, a equipe também dá suporte, ou seja, implementa novas funcionalidades e/ou resolvem *bugs*, e também trabalham no atendimento de suporte aos usuários. Essas são as tarefas mais executadas pela equipe durante o dia-a-dia. A equipe faz uma escala no qual sempre terão uma pessoa responsável pelo

atendimento de suporte aos usuários, quando a equipe tem uma demanda maior nos atendimentos colocam duas pessoas por dia. No momento a equipe enfrenta problemas para iniciar novos projetos pelo número de integrantes, que está reduzido a dois membros e antes eram quatro. Sempre tem uma funcionalidade nova que a equipe está trabalhando, e se tem um membro da equipe no atendimento o outro fica focado em melhorar ou continuar o desenvolvimento. Durante a reunião com a Fábrica de *Software*, a equipe apresentou o processo de desenvolvimento utilizado por eles, que pode ser representado na Figura 3.

Figura 4 - Metodologia Atual da Fábrica



O processo de desenvolvimento atual da Fábrica de *Software* representado acima é detalhado a seguir:

1. ***Product Backlog*:** na primeira etapa o PO (*Product Owner*) gerencia uma lista ordenada de requisitos e de tudo o que é necessário para o produto.
2. ***Sprint Backlog*:** na segunda etapa o gerente da Fábrica de *Software* (*Scrum Master*) seleciona quais funcionalidades precisam ser executadas durante uma *sprint* e essa ação indica todas as entregas a serem realizadas pois controlam o andamento do projeto em cada *sprint*.
3. ***Sprint*:** é estipulado o tempo para desenvolvimento do produto.
4. ***Daily Scrum*:** durante a *sprint* acontece a reunião diária, na qual a equipe de desenvolvimento da Fábrica se reúne com o gerente com o objetivo de discutir o que foi feito, o que será feito e se há algum impedimento para realizar as atividades.
5. **Revisão da *Sprint*:** é realizado ao final de cada *sprint* com o objetivo de que um incremento de software funcionando seja entregue e que ele tenha sido testado e que esteja pronto para ser colocado em produção.

A Fábrica de *Software* segue o *Scrum* como metodologia de desenvolvimento, então nas reuniões que fazem diariamente, são definidas as *issues*, que são pequenas

funcionalidades que deverão ser entregues durante a semana e, diariamente são apresentadas as *issues* que foram desenvolvidas desde a reunião anterior. Geralmente eles definem um *backlog* menor para a semana e vão entregando aos poucos. Atualmente a equipe está diminuindo um pouco essa demanda de entrega de produtos porque a equipe reduziu de tamanho. A equipe não segue o *Scrum* perfeito dia-a-dia, porque se o *Scrum Master* não puder reunir, a equipe de desenvolvimento não para pra reunir e, geralmente o *Scrum Master* vai organizando as coisas relacionadas a reunião/decisão, eles também utilizam um pouco da metodologia XP na hora da implementação, realizando a implementação em pares sempre que possível. Pensando em tudo o que eles já desenvolveram e tudo que funcionou, o *Scrum* foi o principal.

O *Scrum Master* incentiva que todos da equipe tenham conhecimento sobre as tecnologias, *back-end* e *front-end*, embora alguém se destaque em determinadas áreas. E algumas divisões de tarefas acontecem por conveniência, como a manutenção de *hardware*, no qual um membro da equipe se tornou responsável pela manutenção do servidor local do CEULP/ULBRA.

Desde que a equipe teve que passar a trabalhar remotamente, devido a pandemia do COVID-19 que teve início no Brasil no ano de 2020, a equipe não tem mantido tanta documentação, por isso, dentro do processo se trabalha mais com as *issues* do *GitLab*, que dependendo do que estão trabalhando, pode gerar um tipo de relatório/documento. O *GitLab* é uma plataforma de hospedagem de código-fonte e as *issues* são uma funcionalidade importante dessa plataforma, onde os problemas encontrados no projeto são descritos e podem ser resolvidos por quem quiser contribuir. E para alteração no sistema ou para algo que vai acontecer a equipe trabalha nas *issues*, mantendo no *GitLab* o que eles estão trabalhando no momento e se tem algo que eles ainda vão fazer, organizando para manter esse histórico no *GitLab*.

As demandas de correção de software chegam de várias formas, o principal é o *e-mail* por já existir há mais tempo. No *e-mail* há mais demandas específicas da instituição e, por isso, eles recebem coisas voltadas para algo que pode virar uma nova funcionalidade. Recentemente foi iniciado um novo tipo de atendimento, por meio de um aplicativo de mensagens, que inclusive exige uma maior parte do tempo e é onde eles recebem mais mensagens dos alunos querendo tirar dúvidas ou informar problemas que ocorreram naquele momento. Um exemplo é de um acesso a uma turma que não está diretamente relacionado a uma funcionalidade do sistema, e sim a um processo de matrícula ou de algo que possa estar impedindo-o de ter recebido o acesso a tudo.

Além dos contatos de *e-mail* e aplicativo de mensagens, a equipe pode expandir o contato pelo *meet* ou alguma outra forma de reunião, por exemplo. E trabalhando nesse formato remoto a equipe da Fábrica está sempre em contato com a direção acadêmica por meio de videoconferência para discutirem sobre alguma alteração da organização acadêmica que pode gerar uma demanda para alteração no sistema.

Quando ocorre algum problema no sistema e os usuários enviam mensagens por *e-mail* ou aplicativo de mensagens, é a equipe da Fábrica que responde, então geralmente quando tem um erro ou um problema no sistema eles acabam recebendo de um professor ou um aluno que percebeu algo que não funcionou como deveria. E a partir disso, primeiro a equipe estuda o sistema, roda a versão do sistema no local para testar e ver o que está acontecendo, e assim pode identificar um erro menor, corrigir e alterar na produção para o usuário poder utilizar, e se for algo maior, a equipe faz um estudo e leva para a reunião para discutir com o *Scrum Master* sobre o problema e sugerir as alterações.

Na reunião a equipe define quem vai fazer, como vai ser e se for algo que já foi estudado é definido um prazo. Então dependendo do *bug* a equipe define a melhor maneira dele ser resolvido e pode ser algo muito simples que não gera muita alteração e eles resolvem no mesmo dia, ou pode ser algo maior e/ou mais complexo que a equipe discute e se planeja para resolver da melhor forma possível; esses são os dois formatos principais de resolver os *bugs* que encontram. Sempre que tem uma nova demanda/funcionalidade no sistema, a equipe planeja junto com o *Scrum Master*, quando as novas demandas são divididas para que todos os membros consigam desenvolver o mais rápido possível.

Para dar início a um novo projeto a equipe começa estudando as ferramentas caso seja algo que ainda não dominem e, após isso, as funcionalidades são divididas e, caso haja alguém que as domine mais, essa pessoa ficará responsável pelo projeto. Mas a equipe da Fábrica sempre procurou envolver todos os membros para que cada um entendesse o funcionamento do sistema e para ficar mais fácil para todos testarem depois.

A equipe de desenvolvimento da Fábrica estuda adicionar novas etapas no processo de desenvolvimento, principalmente na parte de documentação, já que é importante que algumas partes do sistema possuam documentação. Eles estudaram o método *Conventional Commits* (*Commits* Convencionais) para descrever melhor as *issues* no *git,* e estudaram a possibilidade de gerar ata a cada reunião.

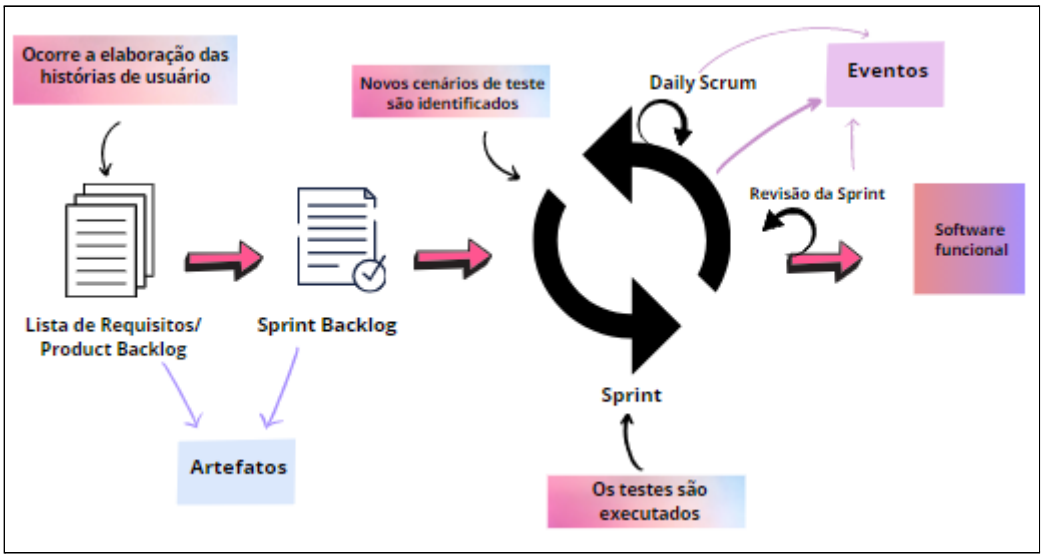
A equipe não tem um padrão de testes definido, não tem uma metodologia de testes bem especificada. A equipe da Fábrica de *Software* faz um *review* de uma funcionalidade e, ao receber uma demanda, é feita a descrição da funcionalidade, de como irá funcionar e, ao

finalizar, antes de mandar para produção, é feita uma reunião mais detalhada com o *Scrum Master* para mostrar a aplicação funcionando e testarem juntos.

4.2 Modelo Proposto

O levantamento de requisitos da Fábrica será mantido porque já é baseado em metodologia ágil, na sequência propõe-se a utilização da escrita de casos de teste e que o processo de desenvolvimento utilizado pela equipe da Fábrica funcione conforme apresentado na Figura 5.

Figura 5 - Proposta da metodologia da Fábrica com elementos do BDD



O novo processo de desenvolvimento da Fábrica ficou da seguinte forma:

01. Artefatos
    a. *Product Backlog* é uma lista de requisitos ordenada por prioridade e que está em constante mudança e atualização e, durante esse processo de levantamento de requisitos, foi adicionada a escrita das histórias de usuário utilizando o BDD. As histórias seriam escritas pelo desenvolvedor e validadas pelo *PO*.
    b. *Sprint Backlog* é uma lista de atividades que serão trabalhadas durante a *Sprint* e que vai evoluindo ao longo do tempo. Essa lista é definida pela equipe de desenvolvimento e as atividades são selecionadas com base nas prioridades definidas durante o *Product Backlog,* validadas pelo *PO*. Após a seleção das atividades, os cenários serão escritos com as técnicas BDD (*Given, When, Then*). Contudo, existe um alinhamento entre as prioridades validadas pelo *PO* e as atividades necessárias para serem executadas naquele período de tempo, notadas pela equipe de

desenvolvimento. A escrita de cenários de testes na linguagem *Gherkin* facilitará o desenvolvimento e automatização dos testes, já que os testes ajudam a trazer mais qualidade para os sistemas de *software*. É nesta etapa que os requisitos são detalhados.

02. Eventos

a. Durante a iteração da *sprint* ocorre a execução dos testes de unidade e/ou comportamento utilizando o *framework* BDD, e a descoberta e escrita de novos cenários de testes.

03. Papéis

a. Como a equipe da Fábrica de *Software* está reduzida não foi adicionado testador. Propõe-se que um membro da equipe aplique as técnicas - testes de unidade e comportamento que são do BDD - e o outro revise.

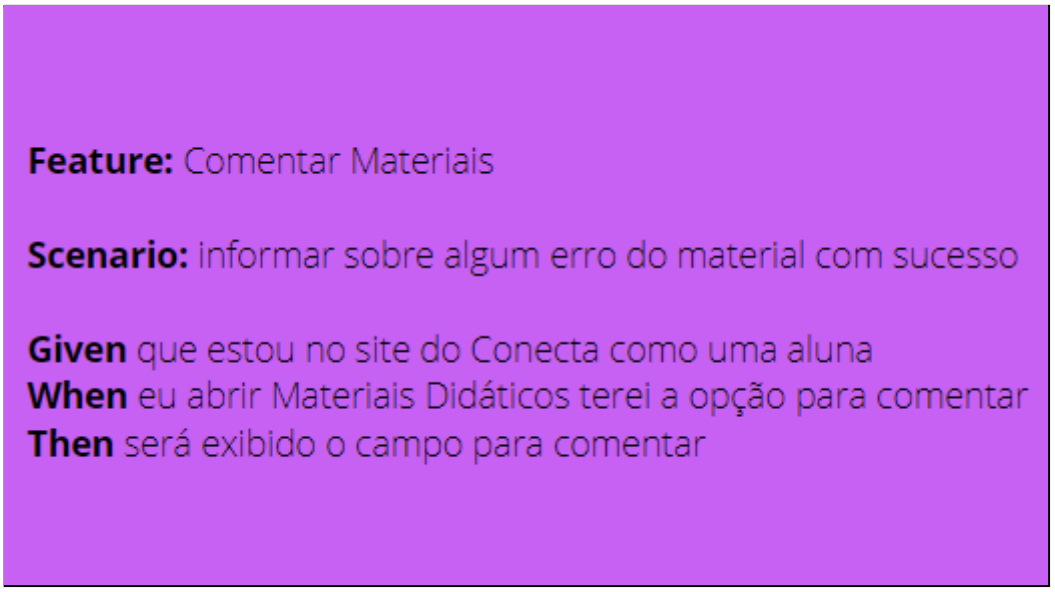
As histórias de usuário têm como objetivo explicar de forma geral e modo informal uma função do sistema e serão escritas no seguinte padrão:

As-I want-so

- *As* (como): representação do cliente;
- *I want* (eu quero): objetivo;
- *So* (para que): objetivo final.

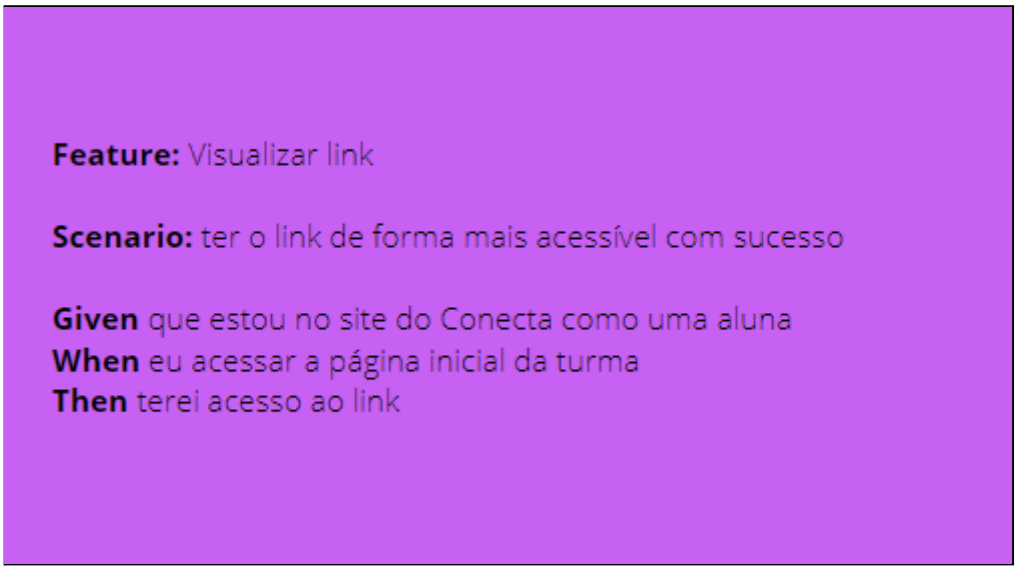
Para que os cenários sejam executados a equipe deverá instalar a biblioteca do *Python* chamada *Behave*. Na elaboração dos cenários primeiro são definidas as pré-condições de cada cenário (*Given*), depois é detalhada a interação que o sistema terá quando o cenário tiver sido executado (*When*) e após isso, caso o cenário tenha sido executado com sucesso, o resultado esperado é detalhado (*Then*). A Figura 6 exibe um exemplo de história de usuário e cenário BDD.

Figura 6 - Exemplo 1 de cenário BDD



A Figura 6 apresenta um exemplo de como o formato proposto da utilização de técnicas BDD ficará com os cenários, na linguagem *Gherkin*. A escrita dos cenários (*Scenarios*) é feita utilizando o padrão composto pelas seguintes palavras chave: *given, when, then*. E cada palavra é usada como mostra no exemplo. Cada cenário exemplifica um aspecto próprio do comportamento do sistema.

Figura 7 - Exemplo 2 de cenário BDD



A Figura 7 apresenta um exemplo de como as técnicas BDD ficarão com os cenários na linguagem *Gherkin*. É notório o quanto a leitura e a escrita se tornam mais fáceis utilizando essas técnicas, já que o BDD mostra o objetivo das funcionalidades de uma forma simples. A Figura 8 apresenta um exemplo do teste automatizado na linguagem *Python*.

Figura 8 - Exemplo do código

```
class VisualizarLink(object):

  def testar_disponibilidade_link(self):
     # Given
     aluno = request.user.dados()

     # When
     aluno.redirect.paginaInicial()
     aluno.link = requestLink()

     # Then
     assert_(link.isDisponivel())
```

Conforme a Figura 8, a implementação do teste fica a critério do testador, contanto que atenda a finalidade da *Feature*, mas recomenda-se que o nome do método deve descrever o comportamento. A Figura 8 é o resultado da transposição do cenário da Figura 7 escrito na linguagem *Gherkin* para linguagem de programação utilizada pela Fábrica de *Software*. Esses testes serão utilizados durante a *Sprint* para validação do que está sendo desenvolvido.

4.3 Apresentação do Site com a Proposta

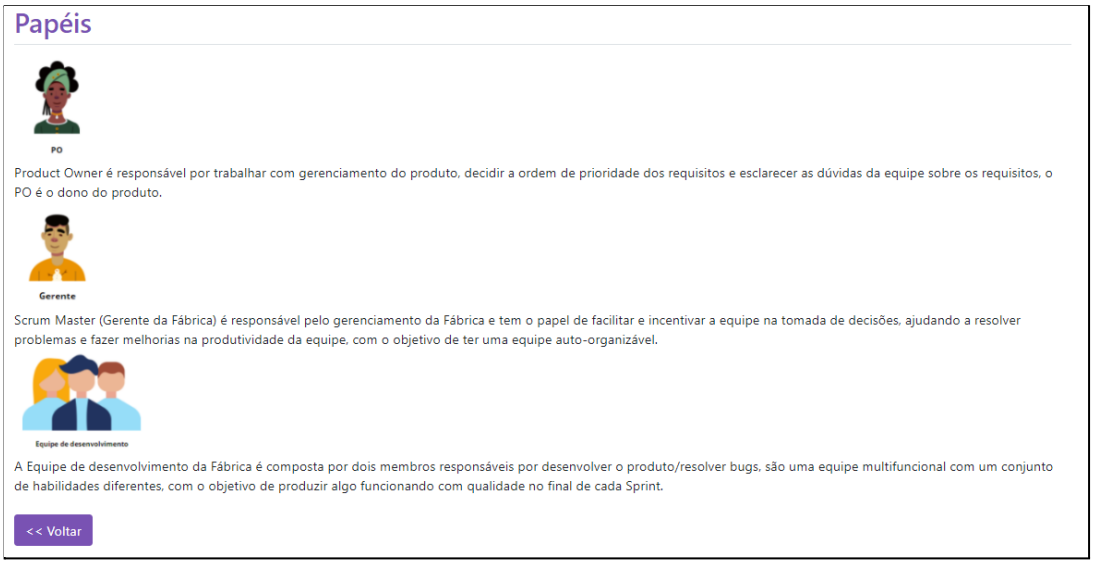
Foi desenvolvido um *site* para apresentar e auxiliar na implantação das técnicas do BDD no processo de desenvolvimento da Fábrica. O *site* foi desenvolvido com informações referentes ao processo de desenvolvimento atual da Fábrica e a proposta de inclusão das técnicas do BDD, que pode ser acessado através do seguinte *link* https://thamyresss.github.io/. O *site* tem como objetivo apresentar as principais áreas da estrutura atual da Fábrica e do modelo proposto, sendo eles: papéis, eventos e artefatos.

A Figura 9 apresenta a página inicial do *site* com uma ilustração do fluxo do processo *Scrum* utilizado pela Fábrica, uma breve descrição da estrutura atual e das suas principais áreas, com um *link* de acesso para as áreas mais importantes, papéis, eventos e artefatos.

Figura 9 - Página Inicial Estrutura Atual

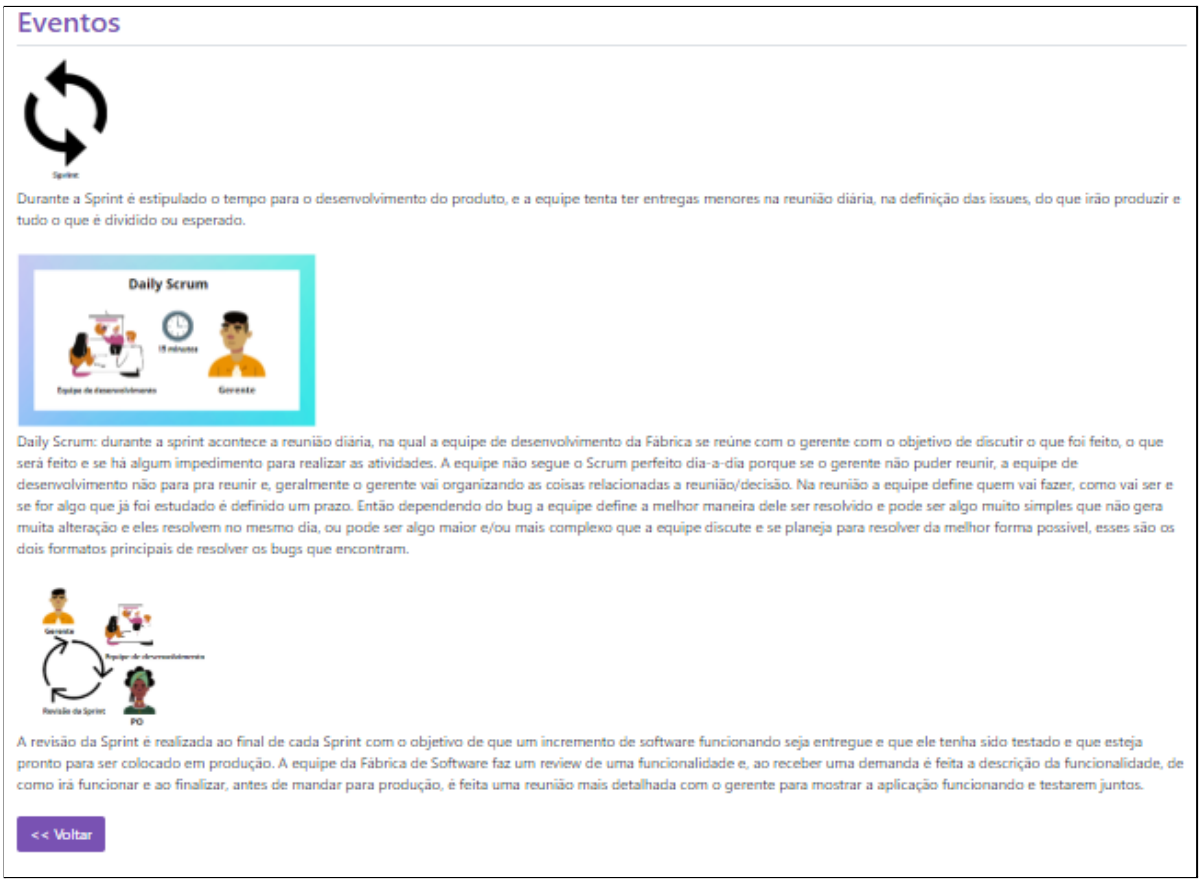
A Figura 10 apresenta a descrição dos papéis da estrutura atual da Fábrica, *Product Owner*, *Scrum Master* e equipe de desenvolvimento, uma ilustração desses papéis, descrição de como são desempenhados no processo de desenvolvimento utilizado pela Fábrica e também é apresentada a relação existente entre eles.

Figura 10 - Página de Apresentação dos Papéis



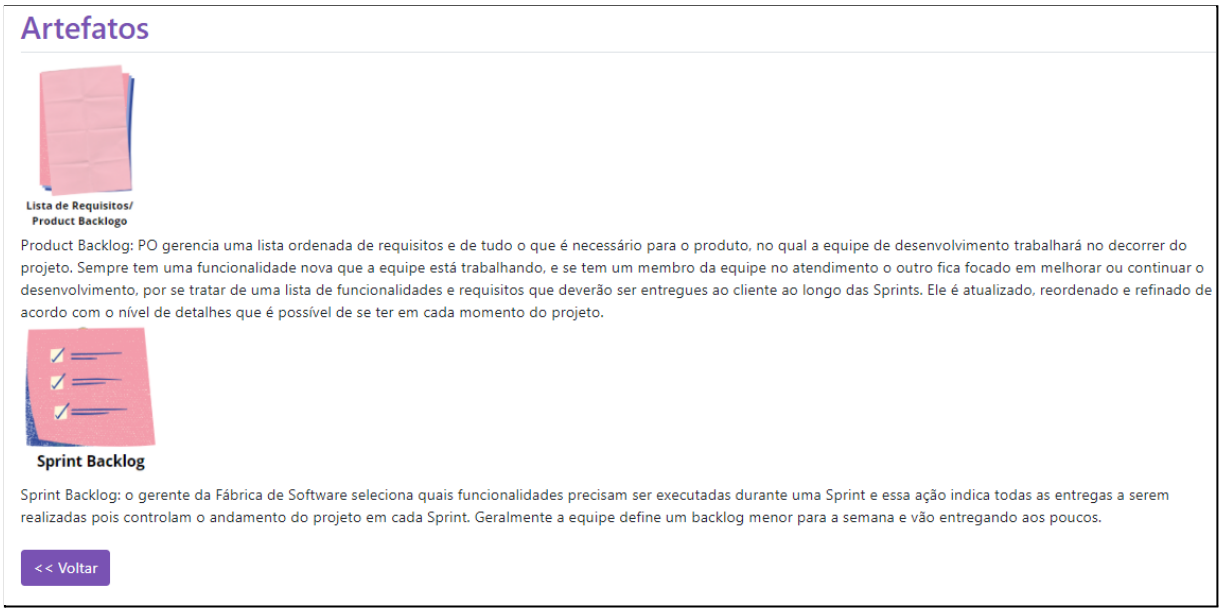
A Figura 11 apresenta a descrição dos eventos *Sprint*, *Daily Scrum* e revisão da *Sprint*, uma ilustração desses eventos, descrição de como são desempenhados no processo de desenvolvimento utilizado pela Fábrica e é apresentada a relação existente entre eles.

Figura 11 - Página de Apresentação dos Eventos



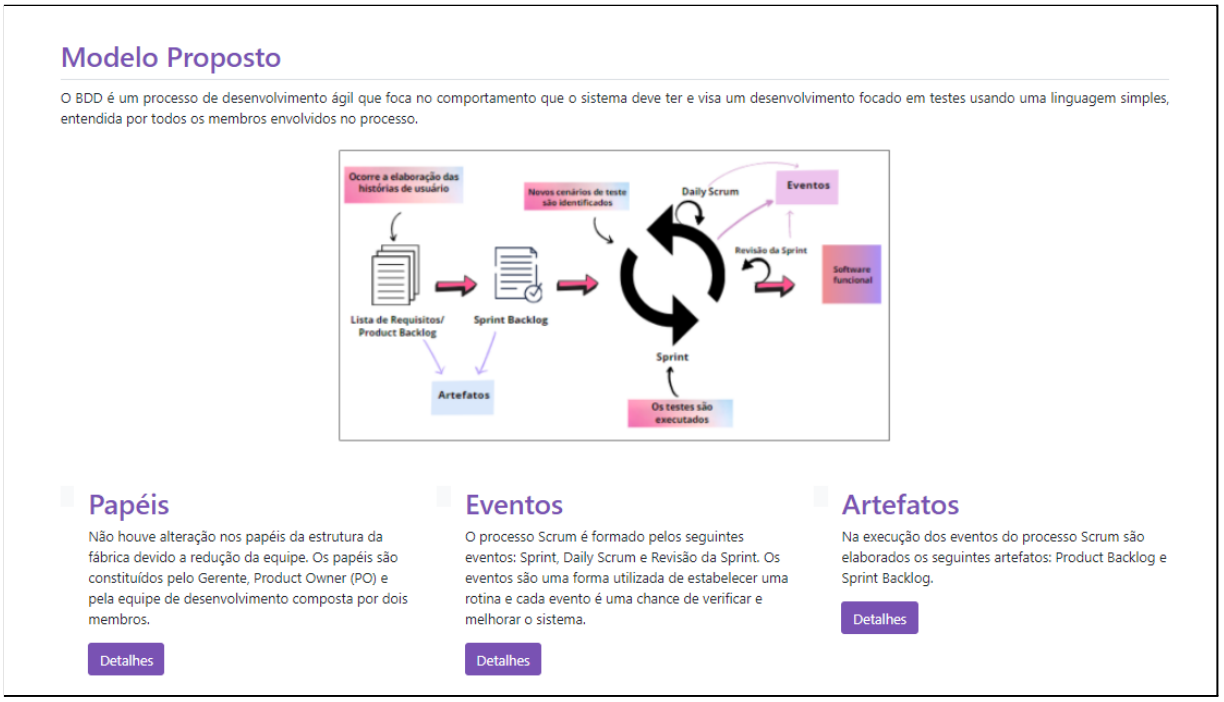
A Figura 12 apresenta a descrição dos artefatos *Product Backlog* e *Sprint Backlog*, - que são executados pelos papéis - uma ilustração desses artefatos, descrição de como são desempenhados no processo de desenvolvimento utilizado pela Fábrica e a relação existente entre eles.

Figura 12 - Página de Apresentação dos Artefatos



A Figura 13 apresenta a página inicial do *site* com uma ilustração do fluxo do processo *Scrum* com técnicas do BDD, uma breve descrição do modelo proposto e das suas principais áreas, com um *link* de acesso para as áreas mais importantes, papéis, eventos e artefatos.

Figura 13 - Página Inicial Modelo Proposto

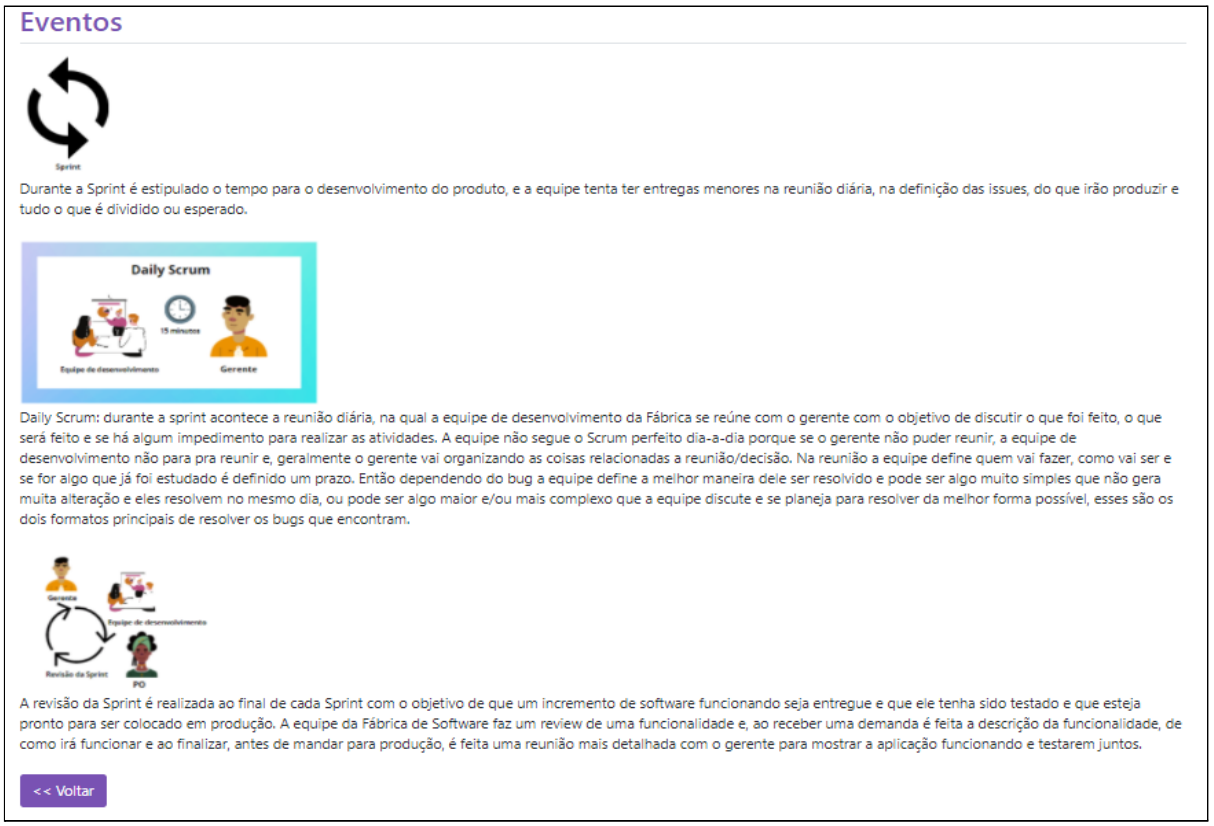


A Figura 14 apresenta a descrição dos papéis *Product Owner*, *Scrum Master* e equipe de desenvolvimento, - com as técnicas do BDD - uma ilustração desses papéis e descrição de como são desempenhados no modelo proposto.

Figura 14 - Página de Apresentação dos Papéis do Modelo Proposto

A Figura 15 apresenta a descrição dos eventos do modelo proposto, *Sprint*, *Daily Scrum* e revisão da *Sprint*, uma ilustração desses eventos, descrição de como são desempenhados no modelo proposto e é apresentado a relação existente entre eles.

Figura 15 - Página de Apresentação dos Eventos do Modelo Proposto



A Figura 16 apresenta a descrição dos artefatos *Product Backlog* e *Sprint Backlog*, - que são executados pelos papéis - uma ilustração desses artefatos, descrição de como são desempenhados no modelo proposto, é apresentado a relação existente entre eles e um exemplo de história de usuário e cenário BDD.

Figura 16 - Página de Apresentação dos Artefatos do Modelo Proposto

O principal objetivo do desenvolvimento do site foi para apresentar um guia sobre a estrutura atual da Fábrica e do modelo da proposta de inclusão de técnicas do BDD no processo de desenvolvimento da Fábrica.

5 CONSIDERAÇÕES FINAIS

O presente trabalho teve como objetivo propor a inclusão de técnicas do BDD no processo de desenvolvimento da Fábrica, propondo uma metodologia para o reúso de cenários de teste suportada por um *framework* de automação de testes, o *Behave*, onde os cenários BDD serão escritos na linguagem *Gherkin*. Por ser o centro de desenvolvimento de soluções tecnológicas do CEULP/ULBRA, na Fábrica de *Software* são desenvolvidas soluções *web* para as demandas da instituição, como *softwares*, páginas *web* e etc.

A proposta da inclusão de técnicas BDD poderá contribuir para a melhoria do processo de desenvolvimento da Fábrica, por ser uma prática que envolve os membros da equipe para abordar o comportamento esperado do sistema/funcionalidade, buscando o entendimento compartilhado. A partir da proposta fornecida foi possível identificar que a aplicação de uma linguagem natural, neste caso o padrão “Dado-Quando-Então” - aspecto da técnica BDD - é aplicável para a criação e execução de testes utilizando o *framework Behave*.

Tendo em vista que a Fábrica de *Software* do CEULP/ULBRA utiliza a metodologia *Scrum* e que o mesmo também foca em iteração, pessoas e no contato com o cliente, conclui-se que a linguagem natural contribuirá neste aspecto. Então, de forma geral, a aplicação de técnicas BDD no *Scrum* agrega grande valor durante todo o processo de desenvolvimento.

Conclui-se que a proposta feita tem como resultado uma nova metodologia para desenvolvimento da Fábrica que agrega elementos do BDD sem modificar de forma exagerada a forma atual de desenvolvimento. Com o objetivo de apoiar a Fábrica de *Software* a adotar estratégia para automação de teste, melhorando a comunicação dentro dos projetos, apoiando a qualidade do produto e dessa forma contribuir na redução dos custos de teste..

Trabalhos futuros incluem a validação da metodologia para o cenário apresentado, a implantação dessa metodologia proposta no processo de desenvolvimento da Fábrica de *Software* do CEULP/ULBRA. E em seguida, que as etapas do processo BDD sejam testadas para que as abordagens definidas sejam verificadas para saber se as necessidades da Fábrica de *Software* foram atendidas, e realizar uma análise da aplicação do processo.

REFERÊNCIAS

ALBIERO, Fernando Weber. **Uma abordagem de teste para aplicativos android utilizando os cenários do behavior driven development**. 2017. 75 f. Dissertação (Mestrado) - Curso de Ciência da Computação, Instituto de Informática, Universidade Federal do Rio Grande do Sul, Porto Alegre, Rs, 2017. Disponível em: https://www.devmedia.com.br/desenvolvimento-orientado-por-comportamento-bdd/21127. Acesso em: 07 set. 2020.

ANDERLE, Angelita. **Introdução de BDD (Behavior Driven Development) como Melhoria de Processo de Desenvolvimento Ágil de Software**. 2015. 44 f. Monografia (Especialização) - Curso de Especialista em Qualidade de Software, Universidade do Vale do Rio dos Sinos - Unisinos Unidade, São Leopoldo, 2015. Disponível em: http://www.repositorio.jesuita.org.br/bitstream/handle/UNISINOS/5314/Angelita+Anderle-Monografia_.pdf?sequence=1. Acesso em: 08 set. 2020.

ARAUJO, Mike Christian de Sousa. **Uma Abordagem Orientada A Aspecto Para Escrita De História Do Usuário Com Gherkin**. 2017. 88 f. Dissertação (Mestrado) - Curso de Ciência da Computação, Centro de Informática, Centro de Informática da Universidade Federal de Pernambuco, Recife, Pe, 2017. Disponível em: https://attena.ufpe.br/bitstream/123456789/30871/1/DISSERTAÇÃO%20Mike%20Christian%20de%20Sousa%20Araujo.pdf. Acesso em: 29 set. 2020.

ENGEL, Jens; RICE, Benno; JONES, Richard. **Welcome to behave!** c2019. Disponível em: https://behave.readthedocs.io/en/latest/. Acesso em: 01 nov. 2020.

BORTOLOSSI, Humberto José. Criando conteúdos educacionais digitais interativos em matemática e estatística com o uso integrado de tecnologias: GeoGebra, JavaView, HTML, CSS, MathML e JavaScript. **Revista do Instituto GeoGebra Internacional de São Paulo. ISSN 2237-9657**, v. 1, n. 1, 2012. Disponível em:https://revistas.pucsp.br/index.php/IGISP/article/view/8823/6595. Acesso em: 01 out. 2020.

CEZERINO, Aparecida; NASCIMENTO, Fernando Paes. Utilização Da Técnica De Desenvolvimento Orientado Por Comportamento (BDD) No Levantamento De Requisitos. **Revista Interdisciplinar Científica Aplicada**, Blumenau, v. 10, n. 3, p. 40-51, mar. 2016. Disponível em: https://rica.unibes.com.br/rica/article/view/728/609. Acesso em: 04 set. 2020.

CHELIMSKY, David; ASTELS, Dave. **The RSpec Book: Behaviour-Driven Development with RSpec, Cucumber, and Friends**. Tradução nossa. 2.1 Dallas, Texas: Pragmatic Bookshelf, 2010. 420 p. Disponível em: http://barbra-coco.dyndns.org/student/rspec/the-rspec-book_p2_1.pdf. Acesso em: 20 out. 2020.

CRESPO, Adalberto Nobiato et al. **Uma Metodologia para Teste de Software no Contexto da Melhoria de Processo**. 2004. Disponível em: https://www.researchgate.net/publication/237497188_Uma_Metodologia_para_Teste_de_Software_no_Contexto_da_Melhoria_de_Processo. Acesso em: 09 set. 2020.

CRUZ, Fábio. **Scrum e PMBOK unidos no Gerenciamento de Projetos**. Rio de Janeiro, Rj: Brasport, 2013. 416 p. Disponível em: https://books.google.com.br/books?hl=pt-BR&lr=&id=SJA37S2QGR0C&oi=fnd&pg=PA1&dq=SCRUM+"cruz"&ots=lxG0x1yOuD&sig=giSeo8i7cO4xlTKtAJSd0OGglF0#v=onepage&q=SCRUM%20"cruz"&f=false. Acesso em: 19 set. 2020.

DEEMER, Pete, et al. **The Scrum Primer**. pdf, 2010. Disponível em: http://www.brianidavidson.com/agile/docs/scrumprimer121.pdf. Acesso em: 29 jan. 2021.

DIAS NETO, Arilo Cláudio. Introdução a Teste de Software. **Engenharia de Software Magazine**, Rio de Janeiro, Rj, v. 1, n. 1, p. 54-59, jan. 2007. Disponível em: https://www.researchgate.net/publication/266356473_Introducao_a_Teste_de_Software. Acesso em: 09 set. 2020.

FLATSCHART, Fábio. **HTML 5 - Embarque Imediato**. Rio de Janeiro, Rj: Brasport, 2011. 256 p. Disponível em: https://books.google.com.br/books?hl=pt-PT&lr=lang_pt&id=_cgsCgAAQBAJ&oi=fnd&pg=PA1&dq=html&ots=frxxT-7YV3&sig=DFBIeRRWXgw3L10lGYrOP5BYpiM#v=onepage&q=html&f=false. Acesso em: 29 set. 2020.

GAVAZONI, Vinicius; ROMANI, Luciana Alvim Santos. Benefícios e dificuldades do uso de CSS para criação de websites. In: **Embrapa Informática Agropecuária-Resumo em anais de congresso (ALICE)**. In: MOSTRA DE ESTAGIÁRIOS E BOLSISTAS DA EMBRAPA INFORMÁTICA AGROPECUÁRIA, 9., 2013, Campinas. Resumos... Brasília, DF: Embrapa, 2013., 2013. Acesso em: 01 out. 2020.

GHERKIN, Gherkin. **cucumber/gherkin**. GitHub. Tradução nossa. Disponível em: https://github.com/cucumber/cucumber/tree/master/gherkin. Acesso em: 09 set. 2020.

LEWIS, Joseph R.; MOSCOVITZ, Meitar. Marcação dá suporte ao CSS. Tradução de Edgard B. **CSS Avançado**. São Paulo, Sp: Novatec, 2010. p. 19-45. Disponível em: https://s3.novatec.com.br/capitulos/capitulo-9788575222201.pdf. Acesso em: 01 out. 2020.

MANZANO, Jose Augusto Navarro Garcia. **Guia de Orientação e Desenvolvimento de Sites: HTML, XHTML, CSS e Javascript/Jscript**. 2. ed. São Paulo, Sp: Saraiva Educação S.A., 2010. 451 p. Disponível em: https://books.google.com.br/books?hl=pt-PT&lr=lang_pt&id=1YuwDwAAQBAJ&oi=fnd&pg=PP35&dq=Linguagem+de+Marcação+de+Texto+HTML&ots=gLdqk-fMdI&sig=0HI2dfjh

mX2NxCCzYX5YoZGLjok#v=onepage&q=Linguagem%20de%20Marcação%20de%20Texto%20HTML&f=false. Acesso em: 01 out. 2020.

MATTÉ, Marcio Angelo. **Testes De Software: Uma Abordagem Da Atividade De Teste Software Em Metodologias Ágeis Aplicando A Técnica Behavior Driven Development Em Um Estudo Experimental**. 2011. 54 f. Monografia (Especialização) - Curso de Engenharia de Software, Universidade Tecnológica Federal do Paraná, Medianeira, Pr, 2011. Disponível em: http://repositorio.roca.utfpr.edu.br/jspui/bitstream/1/1111/3/MD_ENGESS_I_2012_17.pdf. Acesso em: 29 set. 2020.

NORTH, Dan. Introduzindo o BDD. Tradução de Gabriel Olivério. **BroncoDev**, 2016. Disponível em: https://dannorth.net/introducing-bdd/. Acesso em : 04 set. 2020.

OLIVEIRA NETO, José Nunes de. **BDD-I: Especificações de Uso de Behavior-Driven Developmente de Desenvolvimento de Jogos Independentes**. 2019. 131 f. Dissertação (Mestrado) - Curso de Ciência da Computação, Universidade Federal do Maranhão, São Luís, Ma, 2019. Disponível em: https://tedebc.ufma.br/jspui/bitstream/tede/2914/2/JOSÉ-NUNES.pdf. Acesso em: 15 set. 2020.

REALI, Diogo Giaretta. **Análise e experiência no desenvolvimento guiado ao comportamento**. 2015. 65 f. TCC (Graduação) - Curso de Ciência da Computação, Instituto de Informática, Universidade Federal do Rio Grande do Sul, Porto Alegre, Rs, 2015. Disponível em: https://www.lume.ufrgs.br/handle/10183/126080. Acesso em: 17 set. 2020.

RIOS, Emerson; MOREIRA, Trayahú. **Teste de Software**. 3. ed. Rio de Janeiro, Rj: Alta Books Editora, 2013. 304 p. Acesso em: 10 set. 2020.

ROCHA, Fabio Gomes. **Scrum e BDD: o casamento perfeito**. 2013. Disponível em: https://www.devmedia.com.br/scrum-e-bdd-o-casamento-perfeito/28174. Acesso em: 15 set. 2020.

ROQUETTE, José Henrique. **Uma Abordagem Utilizando Behavior Driven Development Para Geração De Casos De Teste: Um Estudo De Caso Na Área Automotiva**. 2018. 74 f. TCC (Graduação) - Curso de Ciência da Computação, Universidade Tecnológica Federal do Paraná, Ponta Grossa, Pr, 2018. Disponível em: http://repositorio.roca.utfpr.edu.br/jspui/bitstream/1/10417/1/PG_COCIC_2018_2_06.pdf. Acesso em: 15 set. 2020.

SANTOS, Gabriela; SOARES, Rodrigo Everton; CALDEIRA, Vagner. Metodologias Ágeis. 2013. Disponível em: https://d1wqtxts1xzle7.cloudfront.net/35266340/METODOLOGIA_AGIL.pdf. Acesso em: 12 set. 2020.

SOARES, Michel dos Santos. Metodologias Ágeis, Extreme Programming e Scrum para o desenvolvimento de Software. **Revista Eletrônica de Sistemas de Informação**,[s.l]. V. 3, N. 1, 2004.

SABBAGH, Rafael. **Scrum: Gestão ágil para projetos de sucesso. São Paulo, Sp: Editora Casa do Código**, 2014. 319 p. Disponível em: https://books.google.com.br/books?hl=pt-BR&lr=lang_pt&id=pG-CCwAAQBAJ&oi=fnd&pg=PT9&dq=scrum+&ots=ETLyvNIHxa&sig=sIDkqSLS6jOLZvCEiSoD66bRhrI#v=onepage&q=scrum&f=false. Acesso em: 19 set. 2020.

SILVA, Daniela Rocha; SOBRAL, Luis Felipe Bentin. **Um Estudo em Larga Escala sobre a Estrutura do Código-fonte de Pacotes JavaScript**. 2017. 58 f. TCC (Graduação) - Curso de Sistemas de Informação, Centro de Ciências Exatas e Tecnologia, Universidade Federal do Estado do Rio de Janeiro (Unirio), Rio de Janeiro, Rj, 2017. Disponível em: https://bsi.uniriotec.br/wp-content/uploads/sites/31/2020/05/201707DanielaRochaLuisSobral.pdf. Acesso em: 02 out. 2020.

SMART, John Ferguson. **BDD and unit testing. In: SMART, John Ferguson. BDD in Action: Behavior-Driven Development for the Whole Software**. Tradução nossa. New York, Ny: Manning Publications, 2014. p. 260-297. Disponível em: http://manning-content.s3.amazonaws.com/download/1/60da528-23c9-467a-946e-f55251878cc4/BDD_CH10.pdf. Acesso em: 09 set. 2020.

SOARES, Ismael. **Desenvolvimento orientado por comportamento (BDD)**. 2011. Disponível em: https://www.devmedia.com.br/desenvolvimento-orientado-por-comportamento-bdd/21127. Acesso em: 04 set. 2020.

SOARES, Michel dos Santos. Comparação entre Metodologias Ágeis e Tradicionais para o Desenvolvimento de Software. **INFOCOMP Journal Of Computer Science**, p. 8-13, 2004. Disponível em: file:///C:/Users/thamy/Dropbox/My%20PC%20(LAPTOP-KJS7B30B)/Downloads/68-Article%20Text-108-1-10-20140918.pdf. Acesso em: 29 set. 2020.

SOMMERVILLE, Ian. **Engenharia de Software**. 10. ed. São Paulo, Sp: Pearson Universidades, 2019. 768 p. Disponível em: https://plataforma.bvirtual.com.br/Leitor/Publicacao/168127/pdf/0?code=55xAfcnMm1SjfxhB+Mx7WltIvssP414rN+2yoEZ2DEUG4ojBrRXshXuzaVsKhXZ738vdB0wLyAWGhCgfKvzOdA==. Acesso em: 12 set. 2020.

SUTHERLAND, Jeff; SCHWABER, Ken. **The Scrum Papers:Nut, Bolts, and Origins of an Agile Framework**. Paris, Fr: Scrum Inc, 2011. Disponível em: https://d1wqtxts1xzle7.cloudfront.net/64196137/scrumpapers.pdf?1597609882=&response-content-disposition=inline%3B+filename%3DThe_Scrum_Papers_Nut_Bolts_and_Origins_o.p

df&Expires=1612027151&Signature=Viv0m8LtDz9BdJltDzPpLeLjryTpU6lV0K6grbbWQJsIco5T72ToWn2pDoR~-lyu1dWi937O3HbyQbUzm0ReZr7hhLoPBFkzKwHju2fXofzN-oAjARveDL5qqeqk4a4bmiEgpjcyVFGOHZqO-69lb6Z43tY862ug7VPK4JrwrnGCKHHK-3NI0FDCOrPTQYzJ~e-lwWHHN-9qFHf1KdVekCWyLNHDc4vILYN-8-He8WEyqZ5dcEkT84LX6sqW8OZcjvYXeZEXPQB4Lh8-NoJGzcpP9zeNmAaIols7NYW5lLtn-xzPF2nH-GNEvpJT4Fv5dKr4nPMb-1pMtUYNgALN7Q__&Key-Pair-Id=APKAJLOHF5GGSLRBV4ZA. Acesso em: 28 jan. 2021.

SUTHERLAND, Jeff; SCHWABER, Ken. **O guia do scrum**. O guia definitivo para o scrum: As regras do jogo. Scrum. org, 2013, 268. Disponível em: http://hanoiscrum.net/hnscrum/images/resource/scrum%20guide_vi_beta_tan.pdf Acesso em: 28 jan. 2021.

TORRES, V. M. HTML e seus Componente**s**. **Revista Ada Lovelace**, v. 2, p. 99-101, 20 dez. 2018. Disponível em: http://45.4.96.34/index.php/adalovelace/article/view/4652/2781. Acesso em: 01 out. 2020.