ProDOC: Uma Proposta de Processo de Desenvolvimento Orientado a Comportamento

Yury Alencar Lima[1], **Juliana Mareco Medeiros**[1], **Luciano Marchezan**[1], **Elder Rodrigues**[1], **Maicon Bernardino**[1]

[1]Universidade Federal do Pampa (UNIPAMPA)
Código Postal 97.546-550 – Alegrete – RS – Brasil

{yuryalencar19,julianamareco18,lucianomarchezan94,eldermr}@gmail.com
bernardino@acm.org

Abstract. *Due to the difficulties of understanding the requirements in the Software Engineering scenario, this work addresses a proposed development process, which aims to improve the understanding of the system requirements, impacting the implementation and creation of the test cases, according to behavior of the software. The Behavior Oriented Development Process (ProDOC) uses specifications for the elicitation and elaboration of use scenarios for each functionality, in order to increase understanding of the requirements and improve the validation of the final system. To evaluate ProDOC, a questionnaire was carried out with researchers and / or professionals of the area, thus showing the acceptance and relevance of the proposal.*

Resumo. *Devido as dificuldades de compreensão dos requisitos no cenário da Engenharia de Software, este trabalho aborda uma proposta de processo de desenvolvimento, o qual tem por objetivo melhorar o entendimento dos requisitos do sistema, impactando na implementação e na criação dos casos de testes, de acordo com o comportamento do software. O Processo de Desenvolvimento Orientado a Comportamento (ProDOC), utiliza especificações para a elicitação e elaboração de cenários de uso para cada funcionalidade. A fim de aumentar a compreensão dos requisitos e melhorar a validação do sistema final. Com o intuito de avaliar o processo ProDoc foi enviado um questionário para pesquisadores e/ou profissionais da área. Os resultados do questionário mostraram a aceitação e relevância da proposta.*

1. Introdução

Com o objetivo de melhorar a produção e satisfazer as necessidades dos clientes são desenvolvidos processos, focando na agilidade no desenvolvimento e qualidade do produto final [Beck et al. 2001]. Dentre estes processos é possível citar as metodologias ágeis, que possuem um grande crescimento decorrente do surgimento de várias *startups* [Souza et al. 2015]. Isto acontece, devido a estas empresas disporem de grande concorrência e uma alta demanda, que resulta na necessidade de realizarem entregas rápidas e funcionais para o cliente, garantindo uma alta qualidade e confiabilidade [Souza et al. 2015]. Assim, com o uso destas metodologias, a empresa consegue desenvolver o sistema de forma rápida, adaptando-se as necessidades encontradas durante o desenvolvimento, além de obter um *feedback* frequente do cliente. Entretanto, este processo

tende a diminuir o formalismo e a documentação em comparação aos demais processos tradicionais.

Existem várias metodologias ágeis, onde muitas vezes são instanciadas do *framework* SCRUM que, por sua vez, tem como objetivo a gestão e o planejamento de processos de software [Schwaber and Sutherland 2011]. Este *framework* ganhou bastante espaço entre as *startups* devido a constante comunicação com o cliente e as entregas rápidas e iterativas, que possibilitam um constante *feedback* das partes interessadas, além da facilidade de adaptação dos requisitos do sistema em desenvolvimento. A utilização deste *framework* também não possui limitação relacionada ao uso de novas práticas de um projeto para outro [Schwaber and Sutherland 2011]. Sabendo disto, foram desenvolvidas tecnologias de apoio ao desenvolvimento, as quais podem ser utilizadas de acordo com o projeto.

Uma destas técnicas que podem ser utilizadas em conjunto com a metodologia ágil é o *Behavior-Driven Development* (BDD) [North 2006]. Esta técnica tem a finalidade de aumentar o entendimento da equipe em relação ao comportamento do sistema que será desenvolvido. A mesma também possibilita práticas como o *Test-Driven Development* (TDD) [Astels 2003], em que os testes serão criados antes e durante a implementação do software, neste caso, focando em métodos e classes. No caso de BDD estes testes são implementados baseados em contextos de uso descritos pelo próprio cliente. Para relatar estes cenários comumente é utilizado o Gherkin, que é uma linguagem descritiva de funcionalidades, cenários e exemplos [Cucumber 2018b]. Esta linguagem também pode ser usada em conjunto com algum *framework* possibilitando a automatização dos testes de sistema.

Tendo em vista que a extração e compreensão dos requisitos do cliente é um ponto essencial no desenvolvimento de qualquer sistema este estudo propõe um novo processo de desenvolvimento, focando na elicitação de requisitos e implementação das reais necessidades do cliente. O presente trabalho foi estruturado conforme segue. A Seção 2 apresenta o referencial teórico utilizado para a criação do Processo de Desenvolvimento Orientado a Comportamento (ProDOC). A Seção 3 apresenta a proposta de processo e exemplificação do mesmo. A Seção 4 apresenta uma avaliação da proposta de processo. A Seção 5 apresenta as considerações finais e trabalhos futuros.

2. Fundamentação Teórica

2.1. Scrum

O Scrum é definido como um *framework* para o planejamento e gerência de projetos complexos [Schwaber and Beedle 2002]. Pode ser considerado uma alternativa para a utilização da metodologia ágil no desenvolvimento de sistemas. Desta maneira, é possível resolver problemas e se adequar a novos requisitos, enquanto são realizadas entregas de partes do produto final até a conclusão do software. As responsabilidades do gerenciamento e planejamento do projeto são divididas em três papéis:

- ***Product Owner (PO)*:** Responsável por fornecer os requisitos do sistema, definir seus níveis de prioridade e ordem de implementação;
- ***Scrum Team:*** Responsável pelo desenvolvimento do sistema, ou seja, modelagem, codificação, testes e documentação;

- ***Scrum Master*:** Responsável pelo apoio técnico a equipe de desenvolvimento. Realiza a organização do grupo e participar efetivamente da proteção da equipe assegurando que a mesma não se comprometa demais. Ele certifica que cada pessoa envolvida no projeto está seguindo seu papel e seguindo as regras do Scrum.

O Scrum pode ser separado em três fases: Planejamento, *Sprints* e Encerramento. Na fase de planejamento tem-se a criação do *Product Backlog*, o qual é um conjunto de funcionalidades que deverão ser implementadas durante todo o processo de desenvolvimento e são definidos os papéis de cada membro da equipe. Após esta etapa é definido o *Sprint Backlog* que é composto de um subconjunto de funcionalidades inseridas no *Product Backlog*. Devido ao Scrum realizar entregas incrementais, o *Sprint Backlog* tem o objetivo de definir quais requisitos serão implementados naquele respectivo *Sprint*. Os *Sprints* são ciclos de desenvolvimento dentro do projeto que possuem duração máxima e fixa. O *Sprint Backlog* é criado pelo PO e o *Scrum Master* e é realizado a priorização das funcionalidades críticas para o sistema. Cada integrante do *Scrum Team* escolhe uma das atividades necessárias para a conclusão do *Sprint* e inicia-se todo o desenvolvimento.

Durante o decorrer do *Sprint* são realizadas reuniões diárias denominadas de *Scrum Daily Meeting*. Estas reuniões diárias tem por objetivo disseminar o conhecimento do que foi feito no dia anterior, identificar os impedimento encontrados para a concretização das atividades, além de enfatizar os próximos passos de cada integrante da equipe. Após o tempo determinado de um *Sprint* é gerado um incremento funcional do produto final baseado no *Sprint Backlog*. Ao final desta etapa, é realizado um *Sprint Retrospective*, onde é realizado uma análise do desenvolvimento no *Sprint*. O encerramento do Scrum acontece depois de um ou vários *Sprints* e é marcado pela conclusão do projeto, em que são realizados os testes de integração, testes de sistema, documentação do usuário e preparação do material de treinamento e marketing.

2.2. Gherkin

O Gherkin [Cucumber 2018b] é uma linguagem representativa que tem como finalidade a criação de especificações que podem ser executáveis ou não. Esta forma de caracterização possui um conjunto de palavras-chave e utiliza a indentação com o objetivo de realizar uma definição da sua estrutura e significado. As palavras-chave possuem suporte em diversos idiomas e possuem uma fácil compreensão tanto do cliente quanto do desenvolvedor, possibilitando sua utilização como artefato de documentação de requisitos, além da descoberta de vários cenários de uso de uma respectiva funcionalidade, um exemplo de uso é apresentado no trecho de código da especificação Gherkin na Figura 1.

```
Funcionalidade: Autenticar no site "X"
    Cenario: Realizar login no site "X".
        Dado o usuario estar com o site aberto
        Quando o mesmo inserir login
        E inserir a senha
        E clicar no botao "login"
        Entao o site redireciona a tela "home"
```

Figura 1. Exemplo de cenário de uso Gherkin

A primeira linha de um arquivo Gherkin é definida a linguagem da especificação seguindo é possível encontrar a palavra-chave primária "*Funcionalidade*" que representa

o nome de um requisito do sistema, que pode ser seguido ou não de uma breve descrição. No arquivo é descrito em alto nível um ou vários cenários de uso da respectiva funcionalidade que são ilustrados na linguagem por meio da palavra-chave "*Cenário*". A Figura 1 apresenta a palavra-chave "*Dado*" que é responsável por descrever alguma pré-condição para que seja executada a funcionalidade. Adicionalmente, outra denominação é o "*Quando*", o qual tem como função explicitar uma respectiva ação, caso necessite realizar mais do que uma é possível realizar o agrupamento utilizando o conectivo "*E*", o mesmo também pode ser usado em conjunto com outras palavras-chave. Por fim, um cenário é composto da última especificação que é o "*Então*", o qual representa os quais resultados das ações realizadas.

2.3. *Behavior Driven Development*

O *Behavior Driven Development* (BDD) [North 2006] é um conjunto de práticas que visam reduzir algumas atividades que acarretam o desperdício no desenvolvimento, como é o caso do retrabalho devido a requisitos mal compreendidos ou vagos, resistência em refatorar o código ou os ciclos de *feedback* lentos devido as transições. Além disso, é considerado uma técnica para desenvolvimento ágil que encoraja a colaboração entre os desenvolvedores, setores de qualidade e pessoas não-técnicas ou de negócios em um projeto de software. O objetivo do BDD é a integração das regras de negócio com a linguagem de programação, mantendo seu foco no comportamento do software com a finalidade de desenvolver e realizar a entrega do produto no melhor tempo possível com alta qualidade. Esta técnica pode ajudar a mitigar problemas como ambiguidade nas descrições do funcionamento do sistema. Isto é possível devido a descoberta deliberativa que acontece na utilização do BDD, que visa descrever o uso do software por meio de cenários ou exemplos de uso juntamente com os clientes [North 2006]. Assim, descobrindo novos requisitos e minimizando a possibilidade de por algum motivo estes serem ignorados pela equipe.

Com a finalidade de descrever o sistema e descobrir novas funcionalidades, são realizadas interações entre os principais interessados no projeto do sistema, como, por exemplo, os proprietários do produto, analistas de negócio, especialistas de domínio, programadores, testadores, entre outros. Isto é necessário devido os *stakeholders* possuírem papéis diferentes dentro do mesmo domínio além de possuírem visões distintas da mesma funcionalidade. Sabendo disto, a definição de exemplos concretos, muitas vezes, acarreta na descoberta de novos problemas e colabora com a equipe de desenvolvimento para a compreensão do domínio. Quando esses exemplos são criados colaborativamente são denominados de Mapeamento de Exemplo e podem se tornar testes de aceitação automatizados ou até mesmo documentação [Cucumber 2018a].

3. Processo de Desenvolvimento Orientado a Comportamento

O Processo de Desenvolvimento Orientado a Comportamento é baseado no *framework* Scrum e utiliza os mesmos papéis envolvidos, sendo eles: *Scrum Master*, *Product Owner* (PO) e o *Scrum Team*. Entretanto, aumenta o foco no quesito comunicação com o PO, compreensão e elicitação dos requisitos do produto, isso acontece devido ao uso de práticas do BDD. Além de melhorar na documentação do sistema, o mesmo mantém as três fases do Scrum: Planejamento, *Sprints* e Encerramento como pode ser visualizado na Figura 2.

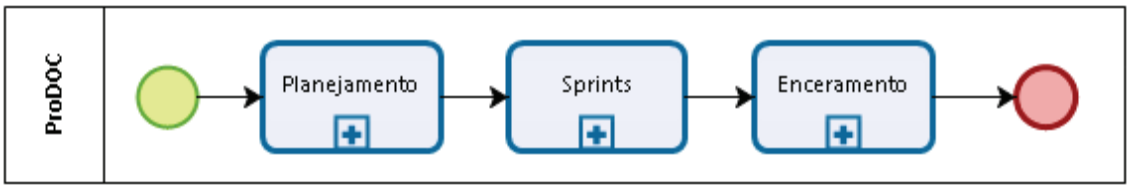


Figura 2. Modelo alto nível do ProDOC

A etapa de Planejamento que pode ser visualizada na Figura 3, é adaptada no ProDOC é onde acontece a reunião entre os interessados e os envolvidos no desenvolvimento. Nesta reunião, o PO tem como um dos seus objetivos elicitar requisitos para a equipe (Figura 3), e o mesmo é responsável pela estruturação e criação de arquivos de funcionalidades, e formatar requisitos usando a sintaxe Gherkin. As perguntas do PO terão como intuito extrair cenários de uso para cada funcionalidade descrita. Após isto, é recomendado priorizar os requisitos, com a finalidade de descobrir quais funcionalidades são mais importantes. Ao final desta etapa, é gerado o *Product Backlog* ordenado e estruturado em arquivos no formato Gherkin.

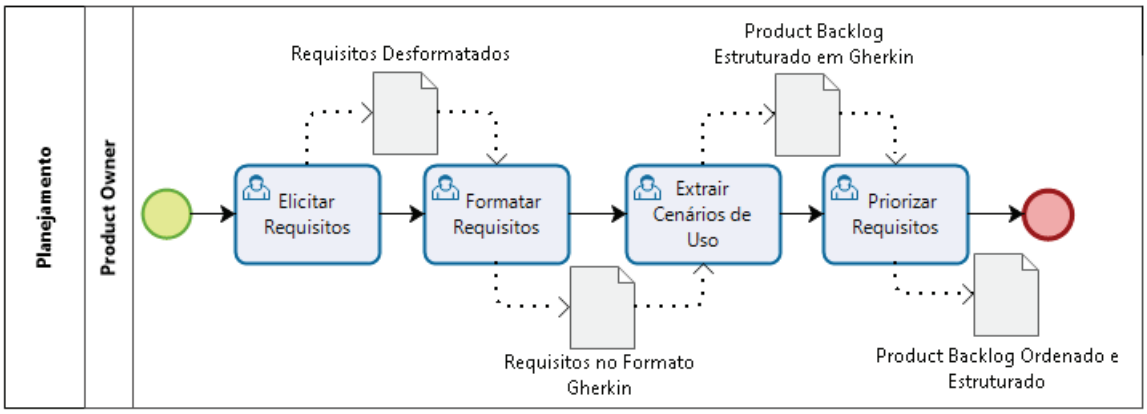


Figura 3. Fluxo do subprocesso Planejamento utilizando ProDOC

Concluída a etapa de planejamento, é dado início aos *Sprints* (Figura 4), tendo como primeira atividade a escolha dos requisitos que serão desenvolvidos, para isso, deve ser considerada a priorização e dependência entre as funcionalidades. O *Scrum Master* supervisiona a equipe em relação ao comprometimento com as funcionalidades. Como no Scrum, a escolha das atividades é por responsabilidade do integrante do *Scrum Team*. A atividade de implementação é orientada aos cenários descritos nos arquivos Gherkin, focando no comportamento da funcionalidade. Com o objetivo de verificar o andamento do projeto são realizadas diariamente as *Scrum Daily Meeting*, que são as reuniões em que cada membro da equipe descreve as atividades exercidas no dia anterior, dificuldades, impedimentos e atividades que serão exercidas.

Em virtude da especificação Gherkin conter o comportamento que o sistema deve exercer é possível realizar a implementação dos testes em paralelo com a do sistema. Estes testes são utilizados para verificar a aceitação da funcionalidade, tendo em vista que

representam o padrão comportamental especificado pelo PO como pode ser visualizado na Figura 4. Seguindo o processo, são realizadas as integrações necessárias e a disponibilização das funcionalidades implementadas. Considerando que os *Sprints* são iterativos e incrementais, após cada integração são executados os testes de regressão, com o propósito de verificar as funcionalidades já implementadas no sistema. Os *Sprints* são realizados até que todas as funcionalidades estejam implementadas com o comportamento que foram descritas nas especificações Gherkin. Durante qualquer fase do *Sprint* podem ser encontrados alguns erros e devem ser realizadas suas refatorações.

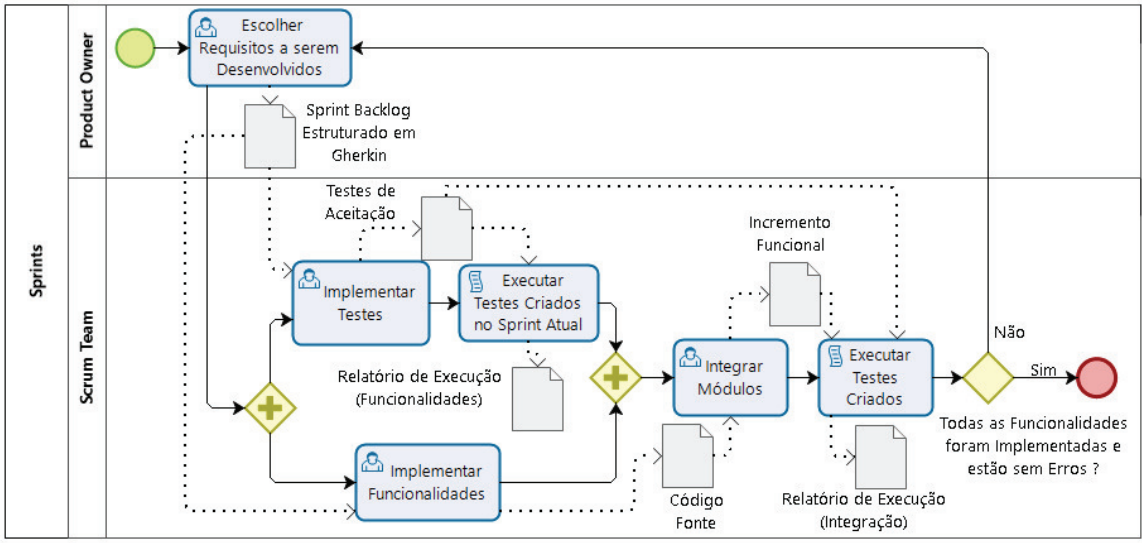


Figura 4. Fluxo do subprocesso *Sprints* utilizando ProDOC

Após a execução de todos os *Sprints* necessários, o ProDOC possui a fase de Encerramento, em que toda a documentação de usuário é entregue no formato Gherkin, inserindo apenas informações consideradas relevantes ao sistema. Como garantia, antes da entrega são executados novamente a suíte de testes relacionadas aos cenários cruciais do sistema. Nesta etapa é preparado o material de marketing e treinamento que pode também ser baseado na documentação Gherkin. Este subprocesso é representado na Figura 5. Ao final destas atividades, o produto final é entregue para o cliente. Com o intuito de analisar o débito técnico da equipe e projetar melhorias são analisados os relatórios de teste gerados a cada *Sprint*.

4. Avaliando a Proposta

Para realizar uma avaliação preliminar do ProDOC foi aplicado um questionário[1] em dez pesquisadores e/ou profissionais que já trabalharam com metodologias ágeis, principalmente Scrum. Além de coletar dados de caracterização de perfil dos participantes, o questionário coletou informações referentes ao nível de conhecimento relacionados aos tópicos (metodologias ágeis e desenvolvimento dirigido a comportamento) associados ao ProDOC. Resultando que dentre os participantes envolvidos 40% trabalharam ou trabalham na indústria e/ou graduação com Scrum por mais de três anos, 50% possuem até três

[1]Questionário disponível em: `https://tinyurl.com/prodoceres`

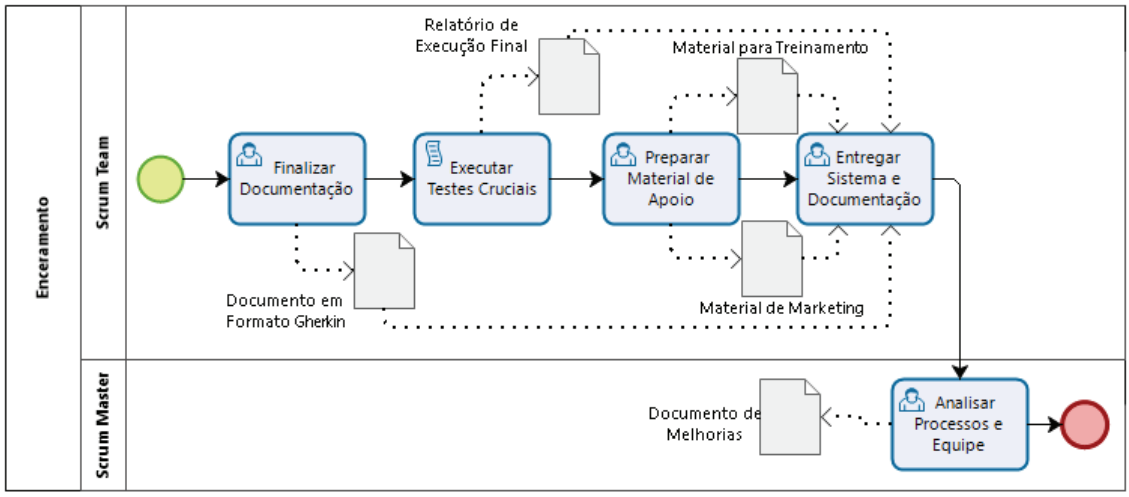
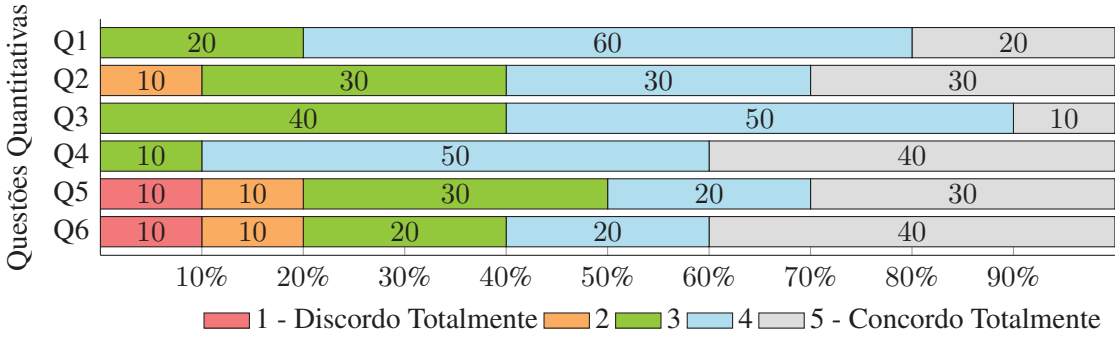


Figura 5. Fluxo do subprocesso Encerramento utilizando ProDOC

anos de experiência com metodologias ágeis e Scrum na graduação e/ou indústria e 10% não possuíam nenhuma experiência.

O questionário era composto por uma seção de questões (**Q**) quantitativas que tinham como objetivo medir o impacto das vantagens de uso do Gherkin em relação a: levantamento de requisitos (**Q1**), entendimento relacionado ao sistema (**Q2**), entendimento do comportamento e maior facilidade no desenvolvimento (**Q3**), melhoria da documentação do sistema (**Q4**), possibilidade de implementar testes em paralelo (**Q5**), e auxiliar na criação de tutoriais e treinamentos para o usuário final (**Q6**). As respostas referentes ás questões quantitativas podem ser visualizadas através da Figura 6.



PS: Algumas porcentagens geram um decimal recorrente que não resulta em um total de 100%.

Figura 6. Diagrama de frequência das questões quantitativas

Resultados da Análise: Ao realizar uma análise dos resultados das questões quantitativas, é possível demonstrar que o processo está na direção correta, devido a maioria dos participantes concordarem com os princípios do processo. Pode-se notar que ainda há alguns problemas relacionados à implementação de testes em paralelo e criação de tutoriais para auxílio devido as respostas das questões **Q5** e **Q6**. Para estas problemáticas serão planejadas melhorias com o objetivo de suprir estas necessidades. Apesar desses problemas, obteve-se um alto grau de confiança relacionado ao impacto e relevância da proposta. A primeira afirmação, é interpretada a partir dos resultados das perguntas de

que o ProDOC teria impacto no desenvolvimento. Isso indica que a criação do processo é uma pesquisa significativa. Quanto à relevância da proposta, foram alcançados resultados satisfatórios, mostrando que o processo está sendo desenvolvido na direção correta. Dentre as últimas perguntas do questionário, foram inseridas quatro questões abertas, com o objetivo de proporcionar aos participantes a oportunidade de contribuir com comentários, sugestões e verificar se usariam o processo para o desenvolvimento. A maioria das sugestões estavam relacionadas a tornar o processo mais iterativo e incremental. Alguns participantes acham que a proposta tem potencial de proporcionar outras melhorias a fim de demonstrar maiores benefícios para o desenvolvimento. A maior parte dos participantes afirmaram que utilizariam o processo de desenvolvimento e outros que dependeria do contexto e do produto a ser implementado.

5. Considerações Finais

Com base nas respostas quantitativas da avaliação, o ProDOC se mostrou eficaz em diversas partes do desenvolvimento, mesmo que ainda possua melhorias a serem implementadas. O reaproveitamento de artefatos e entendimento do produto a ser desenvolvido são seus pontos fortes, isto acontece decorrente ao uso da sintaxe Gherkin para as especificações. Devido a documentação possuir um maior papel no desenvolvimento do sistema a mesma terá uma maior prioridade. Proporcionando, assim, uma melhor manutenção do sistema posteriormente. Além disso, o ProDOC permite a implementação dos testes em paralelo com o sistema.

Com o propósito de que o processo tenha êxito, é necessário uma equipe que consiga extrair os reais cenários de uso. Tendo em vista que é um ponto crucial para o sucesso do projeto. Como trabalho futuro serão realizadas melhorias no processo com base na avaliação apresentada e será realizada a aplicação do ProDOC em um cenário real, com o objetivo de medir o real impacto na equipe com o seu uso.

Referências

Astels, D. (2003). *Test driven development: A practical guide*. Prentice Hall Professional Technical Reference.

Beck, K., Beedle, M., Van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., et al. (2001). Manifesto for agile software development.

Cucumber (2018a). BDD Overview. Disponível em: `https://docs.cucumber.io/bdd/overview/`. Acesso em: 07.07.2018.

Cucumber (2018b). Gherkin Reference. Disponível em: `https://docs.cucumber.io/gherkin/reference/`, addendum. Acesso em: 07.07.2018.

North, D. (2006). Introducing behaviour driven development. *Better Software Magazine*.

Schwaber, K. and Beedle, M. (2002). *Agile software development with Scrum*, volume 1. Prentice Hall Upper Saddle River.

Schwaber, K. and Sutherland, J. (2011). The scrum guide. *Scrum Alliance*, 21.

Souza, G., Nunes, J., Oliveira, J., Araújo, N., Gorgônio, F., and Vale, K. (2015). Diretrizes para uma metodologia de desenvolvimento de software aplicada a startups de tecnologia da informação.