Aplicação e análise de processo de desenvolvimento de software: um estudo de caso no GPES-IFPB

Samyra L. F. Almeida[1], Nadja N. Rodrigues[2], Heremita B. Lira[3], Carlos D. Q. Lima[4], Ana Karina de M. P. Dunning[5], Rayssa Mª S. de Freitas[6]

[1] samyralara@outlook.com. [2] nadja.rodrigues@ifpb.edu.br. [3] heremita@ifpb.edu.br. [4] diego.quirino@tjpb.jus.br. [5] ana.dunning@academico.ifpb.edu.br. [6] rayssa.freitas@academico.ifpb.edu.br. Instituto Federal da Paraíba – Campus João Pessoa.

RESUMO

O Grupo de Pesquisa em Engenharia de Software do Instituto Federal da Paraíba vem desenvolvendo uma pesquisa para identificar e disponibilizar na web as boas práticas em processos de desenvolvimento de software, buscando propagar seu uso entre profissionais e acadêmicos. Como uma das etapas para realização deste trabalho, o grupo vem trabalhando na conformação e refinamento do seu processo de desenvolvimento de software, a partir das boas práticas identificadas em pesquisas do grupo, visando à sistematização de suas atividades técnicas para produção de software. Este artigo se propõe a apresentar o cenário nesse ambiente de pesquisa e o processo de desenvolvimento de software proposto para o grupo de trabalho em questão. Por se tratar de uma pesquisa exploratória e aplicada, os principais procedimentos metodológicos foram: pesquisa bibliográfica, definição, aplicação e avaliação do processo de desenvolvimento de software. Os principais resultados alcançados são: proposta do processo de desenvolvimento de software, aplicação e análise dos resultados de sua utilização e sugestões de melhorias em suas recomendações.

Palavras-chave: Processo de Desenvolvimento de Software. Engenharia de Software. Grupo de Pesquisa.

ABSTRACT

The Software Engineering Research Group from the Federal Institute of Paraiba is developing a research to identify and publicize on the web the best practices in software development processes, seeking to propagate their use among professionals and undergraduates. As one of the stages to completion of this work, the team is working on the conformation and refinement of its software development process, based on the best practices identified in research activities, aiming to systematise their technical activities for software production. This article aims to present the software development scenario in a research environment and the proposed software development software for the focal research group . As this is an exploring and applied research, the main methodological procedures used were: bibliographical research, definition, application and evaluation of the software development process. Results indicate: the proposition of software development process, application and result analysis of its use, and improvement suggestions in its recommendations.

Keywords: *Software Development Process. Software Engineering. Research Group.*

SUBMETIDO 02/12/2017 > AVALIADO 20/04/2018

1 Introdução

Nos últimos anos, as tecnologias de informação e comunicação e os softwares passaram a fazer parte do dia-a-dia das pessoas e das instituições. À medida que as exigências dos *stakeholders*[1] crescem, a Engenharia de Software (ES) desenvolve abordagens para produzir software com mais qualidade. Para Sommerville (2011), a ES é uma área voltada à produção de software, desde os estágios iniciais do sistema até a sua manutenção.

Nesse cenário, as empresas precisam definir seus Processos de Desenvolvimento de Software (PDS), estruturas que especificam e relacionam práticas e ajudam as equipes ao longo das etapas de sua produção, sendo vistos como um dos fatores para softwares de qualidade. A indústria de software apresenta inúmeros modelos de PDS (SOMMERVILLE, 2011), para uso nas empresas, porém alguns desafios são enfrentados como, por exemplo: (i) a dificuldade de compreensão; e (ii) o uso parcial do modelo, devido a particularidades dos projetos.

Para satisfazer os clientes e oferecer alternativas para o trabalho técnico das equipes de software, a utilização de metodologias ágeis trouxe a oportunidade de "desenvolver softwares cada vez mais adaptáveis às inovações tecnológicas e ao mercado competitivo" (SOMMERVILLE, 2007). Nos últimos anos, percebe-se um interesse crescente da indústria e da academia na utilização de metodologias ágeis como estratégia para minimizar problemas no desenvolvimento de software (MEDEIROS *et al.*, 2015). Sendo assim, as boas práticas em ES surgem como aliadas na construção do software e são conceitos vitais para a academia e a indústria.

Considerando esse propósito, atualmente, o Grupo de Pesquisas em Engenharia de Software do IFPB[2] (GPES) possui três projetos relacionados em andamento. O primeiro deles, "Repositório de Práticas em Processos de Desenvolvimento de Software" (RPPDS) (SILVA *et al.*, 2017), vem sendo desenvolvido no sentido de identificar, avaliar e recomendar boas práticas em PDS, analisando as suas características e cenários de aplicação. O segundo, "Repositório de Práticas em Gerenciamento de Projetos de Software" (RPGPS) (MORAIS *et al.*, 2017a), objetiva fazer a identificação, análise e recomendação das boas práticas em Gerência de Projetos (GP) de software. O terceiro, "Ambiente de Boas Práticas Reconfiguráveis em Engenharia de Software" (ABPRES) (MORAIS *et al.*, 2016), propõe criar um ambiente web para disponibilizar as boas práticas da ES, ou seja, as boas práticas identificadas nos projetos anteriores.

Sentindo a necessidade de ter um PDS que apoiasse o desenvolvimento do ABPRES e fazendo uso dos resultados do levantamento de boas práticas de PDS, advindas do RPPDS, o GPES elaborou um PDS próprio. Sendo assim, este artigo apresenta uma versão PDS do GPES do IFPB, os resultados da sua aplicação e os refinamentos sugeridos a partir dessa experiência.

A seção de referencial teórico e a subseção de processos ágeis de software apresentarão os conceitos fundamentais e as fontes que deram suporte para a pesquisa em questão. Na seção de trabalhos relacionados, serão apontados alguns exemplos de PDS utilizados nos meios acadêmicos. Em materiais e métodos será apresentada a metodologia adotada para a construção do PDS. Posteriormente, serão descritos o contexto do projeto ABPRES, e, em seguida, o detalhamento do PDS proposto. Por fim, serão comentados os resultados e feita a conclusão.

2 Referencial teórico

Diferentemente de outros produtos, o software não é algo que pode ser construído através de processos de manufatura. Diante disso, a Engenharia de Software (ES) surge para organizar as etapas e processos envolvidos na produção de software. Se suas especificações são aplicadas de forma correta e eficaz, a ES auxilia a construção do software e diminui a possibilidade de erros em suas etapas, reduzindo os riscos nas atividades. Para que a ES possa trazer esses benefícios aos processos técnicos, é importante que a academia, empresas e profissionais de TI tenham clareza sobre seus conceitos.

De acordo com Sommerville (2011), a Engenharia de Software está relacionada com todos os aspectos da produção de software, desde os estágios de especificação do sistema até sua manutenção. Sendo assim, é importante saber filtrar as técnicas mais apropriadas e eficazes para os diferentes contextos, ou seja, não existe uma única forma ou uma maneira ideal para a ES, embora noções de processo e de

1 Refere-se a todos os envolvidos, direta ou indiretamente, no projeto de um software.

2 GPES-IFPB: Espelho do grupo no CNPq <http://dgp.cnpq.br/dgp/espelhogrupo/0686408453733553>.

organização de sistemas sejam a base de todas as suas técnicas.

Entende-se por processo um conjunto de atividades, ações e tarefas que, uma vez executadas, geram um produto. Essas atividades requerem esforços para atingir um objetivo e podem ser utilizadas em quaisquer projetos, variando de acordo com algumas características dos projetos em questão, como, por exemplo, o seu tamanho. Na produção de softwares, uma ação envolve um conjunto de tarefas que resulta em um artefato de software, enquanto a tarefa é voltada para um objetivo pequeno, que resulta em produzir um resultado tangível (SOMMERVILLE, 2011).

No contexto da ES, um processo não deve ser visto como algo rígido a ser seguido para construir/desenvolver softwares, ao contrário, pode representar uma abordagem adaptável, que possibilita à equipe identificar o conjunto apropriado de ações e tarefas, de acordo com as características do projeto, da equipe ou do cliente. Segundo Pressman e Maxim (2016), a intenção é, sempre, entregar software dentro do prazo estabelecido e que atenda às necessidades daqueles que investiram em sua criação.

Pode-se então definir um processo de software como uma metodologia que orienta atividades, ações e tarefas necessárias para desenvolver um produto de alta qualidade, uma vez que estabelece o alicerce para um processo de engenharia de software constituído de atividades estruturais e de apoio, aplicáveis a qualquer projeto dessa natureza , ao longo de todas as suas etapas. Segundo Pressman e Maxim (2016), os processos de software definem um conjunto de atividades, ações e tarefas necessárias, que pode ser adaptado a uma realidade, a um contexto durante o ciclo de desenvolvimento, desde a especificação de requisitos até a sua manutenção.

De acordo com Sommerville (2011), apesar de existirem diversos tipos de processos de softwares, algumas atividades lhes são comuns: especificação, projeto e implementação, validação e evolução do software. Entre alguns tipos de processos, metodologias ou conjuntos de práticas de desenvolvimento de software, podem ser citados: cascata, iterativo e incremental, prototipação, *Rational Unified Process* (RUP) e o *eXtreme Programming* (XP).

Entre esses, o RUP é um modelo constituído de quatro fases: concepção, elaboração, construção e transição; estreitamente relacionadas ao negócio (SOMMERVILLE, 2011). O XP, por sua vez, é uma metodologia ágil, que envolve um conjunto de regras e práticas constantes de quatro atividades metodológicas: planejamento, projeto, codificação e testes (PRESSMAN; MAXIM, 2016).

2.1 Processos ágeis de desenvolvimento

Os processos ágeis surgem como uma alternativa aos modelos tradicionais e prezam pela satisfação do cliente, propondo entregas incrementais constantes, equipe motivada, métodos informais e simplicidade no desenvolvimento geral. Para Paula Filho (2013), processos ágeis ou métodos ágeis constituem um grupo de metodologias diferentes entre si, porém possuem princípios comuns, baseados mais no trabalho cooperativo do que no formalismo e na documentação escrita.

O que diferencia os métodos ágeis dos métodos mais tradicionais é o enfoque maior nas pessoas e o seu conjunto de valores, práticas e princípios, ou seja, centram-se menos em documentações, em processos e prezam pela adaptabilidade em vez de serem predeterminantes.

De acordo com Pressman e Maxim (2016), os processos ágeis surgem para sanar a pergunta: Como criar um processo capaz de administrar a imprevisibilidade? A resposta consiste na adaptabilidade do processo para alterar rapidamente o projeto e as condições técnicas.

Segundo Sbrocco e Macedo (2012), o uso de metodologias adaptativas, que pregam o desprendimento à previsibilidade, não significa voltar ao caos incontrolável; tais metodologias proporcionam um controle da imprevisibilidade, sendo este seu ponto mais relevante, já que, estando em um contexto não previsível, não se pode usar uma metodologia previsível. Um processo ágil, portanto, deve ser essencialmente adaptável.

Como base para um processo ágil, necessita-se de uma equipe empenhada, motivada, auto-organizada, composta por integrantes qualificados e que interajam de maneira contínua.

Além dessas características, também se espera da equipe: busca pela simplicidade; priorização da satisfação do cliente e sua maior colaboração no projeto; software em funcionamento mais que documentação abrangente; entregas contínuas e frequentes de software, em intervalos curtos... enfim, mudanças produtivas sempre são bem-vindas.

De acordo com Fowler (2005), os métodos ágeis podem ser considerados um meio termo entre um

processo pesado ou trabalhoso (no sentido de requerer mais esforço para desenvolvimento de todos os subprodutos do PDS) e a ausência de processo. Como exemplo desse meio termo, pode-se citar a geração de documentos, que continua indispensável, focando-se, porém, apenas na criação de documentos que serão úteis. A ideia é fornecer apenas o necessário.

Segundo Teles (2004, *apud* MAINART; SANTOS, 2010), os processos ágeis partilham a premissa de que o cliente, à medida que é capaz de manipular o sistema que está sendo produzido, aprende sobre suas necessidades e, com isso, ele mesmo pode reavaliar suas necessidades e prioridades do sistema. Essas informações são incorporadas em seu feedback sobre o sistema. Este aprendizado permite que o cliente direcione o desenvolvimento para que a equipe produza sempre aquilo que tem maior valor para o seu negócio.

Como exemplo de metodologias ágeis, podemos citar o XP e o *Scrum*. O *Scrum* é um framework que segue os princípios do manifesto ágil e se baseia em seis características: flexibilidade dos resultados, flexibilidade dos prazos, times pequenos, revisões frequentes, colaboração e orientação a objetos (SBROCCO; MACEDO, 2012).

Segundo Sommerville (2011), os processos e metodologias ágeis "são mais adequados ao desenvolvimento de aplicativos nos quais os requisitos de sistema mudam rapidamente durante o processo de desenvolvimento".

2.2 Utilização de metodologias ágeis no meio acadêmico e trabalhos relacionados

Cada vez mais se percebe a utilização de métodos ágeis em diversos contextos (SOMMERVILLE, 2011). Seus princípios e características são voltados para projetos de portes variados, além de prezar por agilidade e adaptabilidade. Assim sendo, os processos ágeis passaram a ter visibilidade no meio acadêmico, justamente por se mostrarem menos "pesados" do que os processos tradicionais e por atenderem mais às expectativas do público acadêmico e ao seu contexto: equipe reduzida e com pouca experiência, curto tempo para seu desenvolvimento (?), projetos menores.

Seja para disciplinas que trabalham diretamente com desenvolvimento, seja em projetos de pesquisa ou extensão que gerem softwares, tem-se utilizado processos ágeis customizados, visando atender da melhor maneira esses contextos e facilitar o desenvolvimento e o trabalho em grupo. É o caso do Grupo de Trabalho Ágil (GTA) do Centro Universitário Luterano de Santarém (CEULS), da Universidade Luterana do Brasil (ULBRA), que desenvolve uma pesquisa com objetivo de criar um processo ágil para pequenos sistemas desenvolvidos dentro da academia. Como consequência, o grupo elaborou o PDS *P@PSI*[3] (Processo Ágil para Pequenos Sistemas), descrito como sendo gerenciado pelo *Scrum*, adotando práticas XP e com fluxos organizados do Processo Unificado. Esse PDS constitui-se a partir de três etapas: Planejamento, Desenvolvimento e Encerramento.

De acordo com Geller *et al*. (2008), os alunos do curso de Curso de Bacharelado em Sistemas de Informação do CEULS/ULBRA aplicaram o PDS e obtiveram os seguintes resultados: o PDS obteve bom nível de aceitação; as práticas foram percebidas como fáceis de serem utilizadas; a visão de andamento de tarefas foi citada como uma ajuda para seguir o cronograma; a agilidade no fluxo do projeto foi atribuída ao PDS; e a representação gráfica foi posta como facilitadora do entendimento do processo.

Outro modelo que se pode citar é o XP1[4], uma metodologia de desenvolvimento ágil criada na UFCG (SAUVÉ *et al.*, 2007), utilizada por alunos de graduação em Ciência da Computação. O XP1 é baseado nas práticas de *Extreme Programming* (XP), com algumas mudanças e simplificações, mas que englobam tarefas imprescindíveis a qualquer processo de desenvolvimento de software. A metodologia segue um processo iterativo e incremental e foi elaborada considerando um ambiente universitário em que alunos não trabalham no projeto 40 horas por semana e nem sempre terão muitas horas compartilhadas para realizar o trabalho. Por ser voltado para alunos, inclusive aqueles que fazem uso de um PDS pela primeira vez, o modelo tem foco na simplicidade.

Observando que as referências a processos de software estão sendo cada vez mais adaptadas a cenários específicos, tanto para empresas como para a academia, com o intuito de proporcionar um maior suporte na construção de produtos de software do GPES, proporcionando mais qualidade ao processo e aos produtos, os integrantes do projeto RPPDS definiram um processo que busca agregar valor ao

3 P@PSI: <http://www.periodicos.ulbra.br/index.php/urissane/article/view/1059>.

4 XP1: <http://www.dsc.ufcg.edu.br/~jacques/projetos/common/xp1/xp1.html>.

trabalho dos discentes e serve como validador das boas práticas pesquisadas pelos integrantes do projeto.

Compreender o PDS P@PSI foi importante para que a equipe pensasse que características fossem incorporadas ao processo do GPES, como, por exemplo, ciclo organizado, inexperiência na equipe e *sprint*s iterativas. Diferentemente do P@PSI, que foi idealizado partindo do contexto dos alunos iniciantes em desenvolvimento de sistemas (que se veem com mais frequência), o PDS do GPES foi formulado partindo do contexto do Grupo de Pesquisa, ou seja, alunos que também trabalham *home office*, se comunicam por aplicativos de mensagens e que necessitam de documentação para proporcionar um entendimento mais completo do projeto por qualquer pessoa da equipe (ou novo integrante que dela venha participar).

No processo XP1, podem-se observar papéis, responsabilidades e atividades bem definidas, delimitação de quando cada uma das atividades deve ser realizada, incluindo exemplos que podem ser utilizados como templates. Por se tratar de um PDS com foco em meio acadêmico, essas características do XP1 puderam ser utilizadas pelo PDS do GPES, pois ambos – XPI e PDS – compartilham o fato de que os alunos nem sempre terão disponibilidade de atuar o total de horas estipulado para o projeto e nem sempre terão horas compartilhadas. Assim, enquanto no XP1 preza-se pela simplicidade (pois foi construído para ser utilizado por alunos que nunca utilizaram um PDS) como, por exemplo, ausência de um modelo conceitual (por ter o cliente sempre à mão), o PDS do GPES abarca todas as disciplinas com o intuito de concentrar, de forma organizada e acessível, a maior quantidade possível de informações sobre o projeto para que qualquer integrante (ou alguém que venha a integrar o time) possa entendê-lo.

3 Método de pesquisa

De acordo com Prodanov e Freitas (2013), considerando a metodologia de pesquisa científica, este projeto é de Natureza Aplicada, pois procura gerar conhecimentos, para aplicação prática, direcionados à solução de problemas específicos. Considerando o Objetivo do Estudo, pode-se classificar este projeto de pesquisa como Exploratório, pois ele possibilita maior familiaridade com o problema, tornando-o explícito. A abordagem de pesquisa adotada é quali-quantitativa e os principais procedimentos técnicos utilizados são: pesquisa bibliográfica, levantamento e estudo de caso.

Do ponto de vista de execução, esta pesquisa foi desenvolvida nas seguintes etapas:

- Definição do PDS: foi realizada através do levantamento das boas práticas em PDS, colhidas, analisadas e tratadas pelo projeto RPPDS. Sua elaboração também considerou o cenário do grupo: quantidade de membros, tempo disponível dos integrantes, afinidade com ferramentas de desenvolvimento, *home office,* porte do sistema e a comunicação da equipe.
- Apresentação do PDS para a equipe: envolveu todos os integrantes do GPES, sendo realizada durante uma reunião, e seu documento foi disponibilizado dias antes de ser aplicado ao projeto. Conjuntamente ao documento, foram disponibilizados todos os templates dos artefatos (quando necessários) que deveriam ser gerados, para análise da equipe. Assim sendo, o PDS foi validado por todo o grupo.
- Aplicação em três *releases* (versões do produto) do ABPRES: este projeto vem sendo executado de forma iterativa e incremental, e assim o PDS foi instanciado em algumas iterações. Cada iteração durou duas semanas.
- Coleta de informações das aplicações do PDS nas *release*s: por meio de comunicação contínua nas iterações com a equipe (antes, durante e depois), foram sendo coletados dados avaliativos sobre a utilização do PDS. A última iteração utilizou um formulário com questões quantitativas e qualitativas para avaliação do PDS, pois se constatou que este instrumento seria mais eficaz para se obterem os resultados.
- Refinamento do PDS: a partir dos resultados da etapa anterior, o PDS foi sendo analisado e modificado de acordo com seus resultados de aplicação no projeto, necessidades e sugestões dadas pela equipe.
- Comparação dos resultados das *release*s: nesta última etapa foi analisado o desenvolvimento do projeto antes do PDS e após sua utilização, como forma de identificar as suas devidas contribuições.

4 O contexto do projeto do ambiente de boas práticas

O projeto ABPRES é desenvolvido por uma equipe composta por oito discentes pesquisadores, sete deles da área de Sistemas para Internet e um da área de Design Gráfico. A supervisão é feita por professores orientadores do Curso Superior de Tecnologia em

Sistemas para Internet (CSTSI)[5], que representam os clientes do projeto, nesse primeiro momento.

A equipe trabalha *home office* com horários flexíveis, ou seja, cada pessoa executa um (ou mais) determinado(s) papel(eis) em seus horários disponíveis. Os integrantes se comunicam por aplicativo de mensagens (*Whatsapp*) e possuem dois tipos de reuniões: as reuniões presenciais, que ocorrem quinzenalmente, e reuniões virtuais, sempre que forem necessárias, sendo, na maioria das vezes, realizadas uma a cada semana.

Desenvolvido de forma iterativa e incremental, o projeto teve suas duas primeiras iterações (com duração de duas semanas cada uma) realizadas fazendo uso do Framework de Gerenciamento Ágil de Projetos (FGAP), definido pelo RPGPS (MORAIS *et al.*, 2017b) e fundamentado no *Scrum*, e de algumas boas práticas de PDS, especialmente orientadas pelo XP, porém de forma não sistematizada. Em virtude de a equipe ter passado por uma capacitação técnica em GP (ágil), a velocidade de desenvolvimento do framework de gerência foi maior que a do PDS. Isto fez com que cada integrante do time assumisse a postura de escolher as atividades a serem executadas por si só (de acordo com o que considerasse conveniente), o que deveria gerar artefatos ou subprodutos do projeto, trabalhando de forma *adhoc*. Ou seja, a equipe utilizou algumas boas práticas de PDS, tais como *Users Stories* e prototipação.

A ausência de um PDS consistente e claro para o time fez com que houvesse muita confusão quanto às atividades, responsabilidades e artefatos a serem produzidos, assim como quanto à sequência ou dependência das atividades.

Após essa experiência, a equipe percebeu que ficaram muitas lacunas no PDS, o que resultou em baixa produtividade. Entre as lacunas, podemos destacar: problemas no entendimento da construção das *Users Stories* e na sua divisão em tarefas, além do fato de a equipe estar despreparada para utilizar a tecnologia sugerida. Atividades relacionadas a outras disciplinas da ES, tais como Análise e Projeto, Desenvolvimento e Testes não chegaram a ser realizadas.

4.1 O Processo de desenvolvimento de software

A definição do PDS para o GPES foi elaborada de acordo com o cenário do projeto, revisando as práticas já utilizadas pela equipe, adicionando novas práticas provenientes do RPPDS, relacionando-as entre si, a fim de preencher as lacunas encontradas e definindo os artefatos e etapas que deveriam ser cumpridos pela equipe. Desse modo, foram definidas as práticas para todas as áreas da ES, a partir de uma visão sistêmica do trabalho previsto para a construção do ABPRES.

A definição do PDS considerou que todos os seus elementos precisavam ser bem delineados e que precisava focar-se em disciplinas como Análise e Projeto, Desenvolvimento e Testes, vistos como cruciais para o desenvolvimento do projeto. Essas considerações foram feitas a partir do pressuposto de que o ABPRES, ao ser disponibilizado na web, vai requerer usabilidade, facilidade de uso, intuição, interface amigável e pouca complexidade, e assim disponibilizar conteúdos de ES de maneira fácil e atrativa.

A definição do PDS foi supervisionada pelos orientadores e seu uso foi validado por toda a equipe e sua construção. Os papéis propostos pelo processo foram os de: analista, desenvolvedor, testador, designer gráfico e gerente (este sendo orientado pelo framework de GP). Além dos papéis, o PDS propôs responsabilidades que deveriam ser atribuídas a todos os integrantes do projeto.

Quanto às responsabilidades referentes aos papéis e o modo como exercê-las, tem-se as seguintes designações:

- **Time:** Identificação dos requisitos funcionais e não-funcionais – geração de documento que define uma parte do sistema ou função e características associadas a essa parte ou função. O artefato deverá ser construído seguindo um template pré-estabelecido, com a ferramenta selecionada (*Google Docs*[6]*).*
- **Homologação** – etapa realizada em reunião com todo o time e clientes, a fim de ser realizado o aceite ou não aceite do sistema e de suas partes.

- **Analista:**
  - Elaboração de *Users Stories* (US) – geração de documento que descreve as necessidades ou expectativas do usuário. O artefato deverá

5 Página curso de CSTSI <https://estudante.ifpb.edu.br/cursos/39>.

6 Página do Google Docs<https://www.google.com/docs/about/> .

ser construído seguindo um template pré-estabelecido, com a ferramenta selecionada (*Google Docs*).

- Prototipação – geração de documentos com a visão inicial das telas do sistema. O artefato deverá ser construído seguindo um template pré-estabelecido, com a ferramenta selecionada (*Balsamiq Mockups*[7]).
- Elaboração de modelo do sistema – geração de documento que consiste na descrição da comunicação entre as camadas do sistema e sua funcionalidade. O artefato deverá ser construído seguindo um template pré-estabelecido, com a ferramenta selecionada (*Astah*[8]).
- Elaboração de diagrama de sequência – geração de documento que objetiva mostrar como são feitas as trocas de mensagens entre instâncias de classes, componentes, subsistemas ou atores. Esse documento será gerado quando houver demanda de entendimento por parte de equipe (sobretudo a de desenvolvimento) sobre a comunicação entre as camadas da aplicação que não estejam evidentes nos cenários elencados, regras ou protótipos de telas. O artefato deverá ser construído seguindo um template pré-estabelecido, com a ferramenta selecionada (*Astah*).
- Elaboração das mensagens e regras de negócio – geração de documentos que descrevem, respectivamente, todas as mensagens lançadas pelo sistema e definem as premissas/restrições do sistema. O artefato deverá ser construído seguindo um template pré-estabelecido, com a ferramenta selecionada (*Google Docs*).
- Construção do Design de arquitetura do software – geração de documento que demonstra a estruturação e o funcionamento dos componentes do software. O artefato deverá ser construído seguindo um template pré-estabelecido, com a ferramenta selecionada (*Google Docs*).

- **Desenvolvedor:**
  - Implementação de interface visual do sistema – construção da versão definitiva do software. Esta ação demanda códigos que deverão ser construídos na ferramenta selecionada (*Visual Studio IDE*[9]).
  - Codificação das tarefas utilizando testes unitários – construção de código, utilizando-se o *Test-Driven Development* (TDD), ou seja, realização do desenvolvimento guiado por testes. Os testes são escritos para cada funcionalidade, antes de codificá-las, o que obriga o programador a ter um planejamento de interface, classe e método. Esta ação demanda códigos que deverão ser construídos na ferramenta selecionada (*Visual Studio IDE*).
  - Organização e manutenção do repositório de código do sistema – promove a sistematização do código gerado e constante organização a cada iteração, ou seja, todos os artefatos serão organizados em suas respectivas áreas no repositório do projeto, de acordo com sua estrutura. Esta ação não demanda produção de documento.
  - Correção de bugs/erros – consertos das falhas encontradas – provenientes da execução dos testes ou da utilização do cliente. Esta ação demanda códigos que deverão ser construídos na ferramenta selecionada (*Visual Studio IDE*).
  - Controle de versão do produto – gerenciamento das diferentes versões do sistema geradas nas iterações, ou seja, o versionamento, de acordo com o que foi produzido por *sprint* (no contexto do *Scrum*, *sprint* é definido como um tempo determinado dentro do qual um conjunto de atividades deve ser executado) – que deverá ser feito ao final de cada *sprint*. Esta responsabilidade foi atribuída ao desenvolvedor, pois se acredita que, por seu constante contato com o repositório e código do sistema, seja mais fácil para esse agente manter tal responsabilidade. Esta ação não demanda produção de documento e deverá ser

7 Página do Balsamiq Mockups<https://balsamiq.com/products/mockups/>.

8 Página do Astah: <http://astah.net/>.

9 Página do Visual Studio: <https://www.visualstudio.com/pt-br/>.

realizada no repositório escolhido pelo grupo, o GitHub[10].

- Implantação do sistema na máquina do GPES – disponibilização do software no ambiente de produção do grupo, ao final de cada iteração. Esta ação não demanda produção de documento.

- **Testador:**
  - Elaboração de casos de teste com os dados – documento que define as entradas, saídas e resultados esperados, para todos os passos do teste, incluindo os dados que serão utilizados nos testes. Esta ação demanda um artefato que deverá ser feito seguindo o template pré-estabelecido na ferramenta selecionada (*Google Docs*).
  - Elaboração de *scripts* para os testes automatizados – são construídos códigos que automatizam o teste na própria ferramenta de testes. Esta ação produz códigos que deverão ser construídos com a ferramenta selecionada (*Selenium IDE*[11]).
  - Execução de testes funcionais automatizados – realização dos testes codificados. Esta ação não demanda produção de documentos e deverá ser executada na ferramenta selecionada (*Selenium IDE*).
  - Reporte de erros e/ou *bugs* do sistema – informação dos erros encontrados com todos os dados necessários para a sua compreensão, os quais poderão surgir na execução dos testes. Esta ação produz artefato que deverá seguir um template pré-estabelecido, construído na ferramenta selecionada (*Google Docs*).

- **Designer Gráfico**
  - Elaborar interface visual do sistema –geração de documentos com a visão do sistema, que proporciona a interação do software com o usuário. Esta ação demanda artefato (interface visual final do sistema) que deverá ser elaborado na ferramenta selecionada (*Adobe Illustrator CC*[12] e *Adobe Photoshop CC*[13]).

- **Gerente:**
  - As atividades e responsabilidades deste papel são definidas pelo trabalho realizado por um integrante do GPES-IFPB, como foi citado anteriormente, utilizando o Framework de Gerenciamento Ágil de Projetos – FGAP do GPES, detalhado no trabalho de Morais *et al.* (2017b).

A sugestão das ferramentas no PDS foi feita pelos alunos, baseada nas experiências vividas por eles nas disciplinas do CSTSI, como, por exemplo, o *Balsamiq Mockups*, *Astah*, *Selenium IDE* e *Visual Studio IDE*. O *Google Docs* é uma ferramenta que vem sendo utilizada pelo projeto de pesquisa desde seu início, além de amplamente utilizada em ambiente acadêmico, por ser colaborativa. Há também o uso de ferramentas como o *Adobe Illustrator CC* e o *Adobe Photoshop CC*, que foram sugeridas por um dos membros, já que possuía conhecimento sobre elas, e assim conduziu uma capacitação dos demais integrantes da equipe.

Os artefatos que devem ser gerados podem ser visto nos fluxogramas apresentados nas Figuras 1, 2 e 3. Conforme a Figura 1, antes do ciclo iterativo, é realizada a listagem de requisitos, elaborado o design de arquitetura de software e a visão inicial do modelo do sistema. O ciclo inicia-se com a revisão dos requisitos funcionais e não funcionais, o que deve ser realizado por toda a equipe. Em seguida é revisado o projeto da arquitetura de software, e dá-se início à elaboração das *Users Stories*, documentos de regras de negócio e documentos de mensagens. Estes documentos são gerados separadamente, mas são referenciados pelas respectivas *Users Stories*. Após a conclusão das *Users Stories*, é iniciada a elaboração dos protótipos, a elaboração/revisão do modelo de dados e a construção dos diagramas de sequência (caso seja necessário). Com auxílio dos protótipos elaborados, é possível construir os casos de testes, que incluem os dados que serão utilizados nos testes e consideram a interface visual do sistema. Essa interface é construída por um designer gráfico (embora

10 Página do GitHub <https://github.com/>.

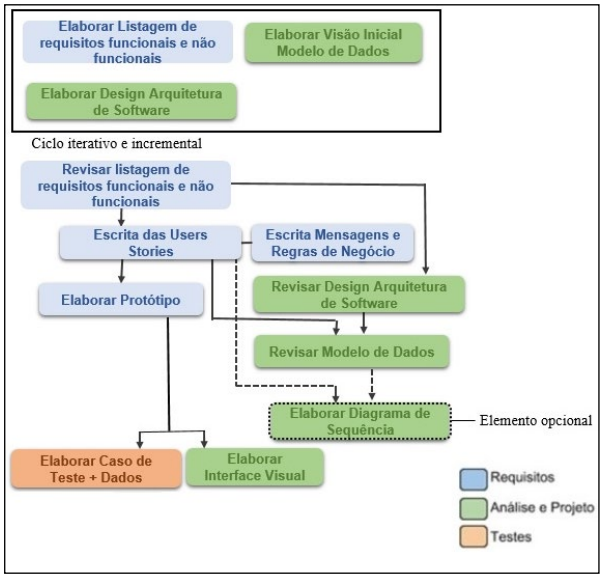
11 Página do Selenium IDE <http://www.seleniumhq.org/projects/ide/>.

12 Página do Adobe Illustrator CC: <http://www.adobe.com/br/products/illustrator.html>.

13 Página do Adobe Photoshop CC <http://www.adobe.com/br/products/photoshop.html>.

os protótipos sejam de alto nível, ou seja, aptos para serem utilizados na fase de desenvolvimento).

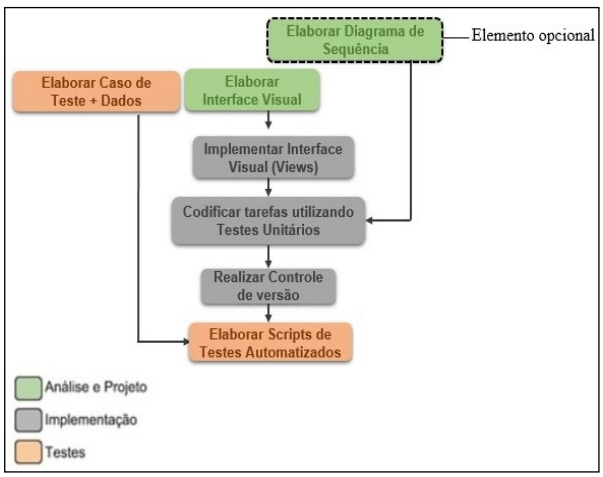
Figura 1 – Fluxograma do PDS – Parte 1



Fonte: Elaboração própria

Seguindo o fluxo, conforme pode ser visto na Figura 2, com a interface finalizada, é dado início à codificação, por meio da implementação das *views*, utilizando-se os testes unitários (baseados em TDD) e os diagramas de sequência. Após a conclusão dessas tarefas, é gerada uma versão do sistema que será adicionada no repositório do projeto. O controle de versão é responsabilidade do desenvolvedor.

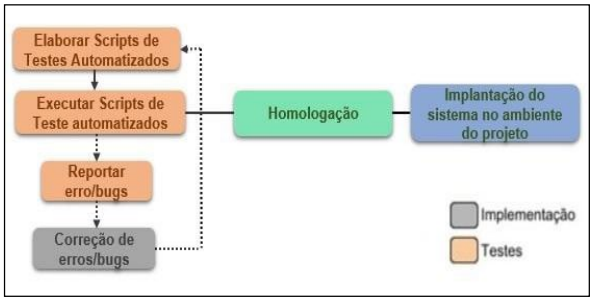
Figura 2 – Continuação do Fluxograma do PDS – Parte 2



Fonte: Elaboração própria

Após a disponibilização do projeto no repositório, conforme apresentado na Figura 3, o testador poderá iniciar a construção dos *scripts* dos testes automatizados, seguindo os casos de testes. A execução dos testes será feita à medida que forem sendo finalizados os *scripts*. A partir dos resultados dos testes, dois caminhos podem ser seguidos: (1) ou será reportado algum erro/*bug*, caso o teste não passe, o que fará com que a tarefa retorne ao desenvolvedor para que este a corrija; (2) ou o teste será bem sucedido e a tarefa será considerada finalizada. Após todos os testes serem concluídos com sucesso e/ou se chegar ao final da *sprint*, será feita a homologação, etapa em que o sistema será apresentado para avaliação pelos clientes. Caso aprovado, o sistema será implantado, o que consiste em configurá-lo em um computador do projeto presente no IFPB. É importante ressaltar que todos os artefatos vão sendo realizados à medida que suas dependências são finalizadas, e essa realização se dá ao longo de iterações, de forma incremental.

Figura 3 – Continuação do Fluxograma do PDS – Parte 3.



Fonte: Elaboração própria

Por se tratar de um processo adaptativo (sendo esta sua primeira versão), com a chegada de novas práticas obtidas do projeto RPPDS, o PDS poderá ter várias possibilidades técnicas (com a inclusão ou exclusão de tarefas, ferramentas, artefatos, em versões futuras), o que potencializará as suas características aos contextos dos projetos.

5 Resultados

Conforme foi relatado, a primeira e segunda iteração foram executadas sem um PDS definido. A primeira iteração que utilizou o PDS proposto foi a terceira, com duração de duas semanas. Esta iteração se iniciou em uma reunião com toda a equipe discente, momento em que o time elaborou os requisitos funcionais e não funcionais e procedeu

à sua divisão em papéis. Devido a problemas de disponibilização de tempo no time, um dos integrantes do grupo propôs assumir os papéis de desenvolvedor e analista, pois queria demonstrar à equipe que a linguagem e framework sugeridos por ele (PHP: *Hypertext Preprocessor*[14] e Slim *Framework*[15]) seriam mais adequados ao projeto do que a ferramenta de gerenciamento de conteúdo escolhida (*DSpace*[16]).

Com esta decisão, uma grande parte do PDS foi instanciada, o que proporcionou um desenvolvimento substancial ao projeto. Nesta *sprint*, foram elaborados os seguintes artefatos: *User Stories*, modelo de dados, diagrama de sequência, design de arquitetura, código utilizando TDD, organização e controle de versão. Sendo assim, foram desenvolvidas as áreas de Requisitos, Análise e Projeto, Desenvolvimento, Homologação.

Nessa iteração, houve a implementação de dois requisitos, que constituíram a primeira versão executável do sistema entregue em uma *release*. As demais áreas e artefatos não puderam ser elaborados devido à falta de tempo da equipe, pois um integrante acabou assumindo dois papéis (desenvolvedor e analista), o que comprometeu sua produtividade. Como foi dito anteriormente, esse mesmo integrante tomou essa atitude e, apesar de não ter conseguido instanciar todas as áreas do PDS, comprovou que a linguagem e o framework por ele sugeridos tiveram um melhor rendimento no projeto, reduzindo tempo e complexidade no desenvolvimento.

Quanto à utilização do PDS, podemos elencar os seguintes resultados:

- Documentação do PDS disponível e acessível a todos;
- Artefatos elaborados na *sprint*, que proporcionaram um melhor entendimento sobre o sistema e também servirão de base para futuras *sprints* (e futuros integrantes do projeto);
- Comunicação e integração mais reforçadas da equipe;
- Dependências mais visíveis e bem definidas dos elementos.

Ao final da iteração, foram coletadas algumas impressões sobre o sistema, entre as quais podemos destacar:

- Com a documentação do PDS, houve um melhor esclarecimento acerca das tarefas e atividades a serem feitas;
- Melhor entendimento sobre o sistema, pois, ao documentá-lo, muitas dúvidas foram sanadas devido ao debate entre a equipe sobre o projeto;
- Entendimento de quando e como os artefatos devem ser gerados;
- Melhor comunicação e integração da equipe, devido ao fato de se ter uma visão melhor das dependências dos elementos.

Dessa forma, os participantes sugeriram deixá-lo como estava, a fim de ver como seria sua aplicação com toda a equipe.

Na quarta *release*, com duração de duas semanas, todos os integrantes do grupo foram designados a um papel. Nesse período, também houve o acréscimo de discentes ao projeto, o que facilitou a distribuição dos papéis e melhorou o desenvolvimento do trabalho. Ficou decidido que os integrantes que tinham mais conhecimento acerca de determinada área ficariam responsáveis de orientar e criar materiais que servissem de apoio aos novatos e futuros membros do projeto, a fim de cumprirem os papéis designados e obterem mais conhecimento sobre a área.

Esta iteração teve três requisitos previstos. Todos os três foram implementados e todos os artefatos previstos pelo PDS foram desenvolvidos, exceto o TDD (ou seja, o desenvolvimento foi executado sem ser guiado por testes), devido à falta de tempo e de conhecimento sobre a prática com a linguagem utilizada, pois não houve tempo de estudar e aplicar dentro desta *sprint*. A falta de tempo afetou o desenvolvimento dos diagramas de sequência (foi construído o diagrama de apenas um requisito), a elaboração e execução dos testes (um requisito foi feito parcialmente) e a implantação do sistema.

Na aplicação do PDS, alguns resultados puderam ser percebidos:

- Entrega de cerca de 80% das atividades previstas;
- Familiaridade da equipe com o processo;

14 Página do PHP<http://php.net/>.

15 Página do Slim Framework<https://www.slimframework.com/>.

16 Página do DSpace: <http://www.dspace.org/>.

- Compartilhamento de conhecimentos em algumas áreas;
- Artefatos e materiais produzidos serão de grande utilidade para o projeto;
- Padronização das atividades e artefatos gerados.

Analisando esses resultados pudemos observar alguns aspectos:

- Com a equipe já familiarizada com o processo, o desenvolvimento nas áreas foi mais produtivo, pois todos já conheciam o fluxo e os artefatos a serem produzidos;
- O compartilhamento de conhecimento em áreas como testes e desenvolvimento, pois os que possuíam mais conhecimento acerca de uma área puderam repassar esse conhecimento para outros, como, por exemplo, um novo integrante que foi designado para o teste recebeu apoio de um integrante com mais experiência na área, uma vez que este repassou todas as informações necessárias para que o recém-chegado pudesse exercer o seu papel. Além de o aluno menos experiente sair com mais conhecimento teórico e prático sobre testes, ele conseguiu desenvolver quase todas as atividades relativas a seu papel, não as cumprindo totalmente devido ao tempo;
- Todos os artefatos e demais materiais produzidos pela equipe e relativos a cada área foram e são de grande utilidade para o entendimento do projeto, tanto para quem já faz parte dele, quanto para qualquer pessoa que ingresse na equipe;
- A padronização das atividades e dos artefatos se dá devido ao fato de, em sua maioria, obedecerem a templates e serem elaborados seguindo suas especificações.

Quanto a sugestões dadas ao PDS, após a iteração, o time elencou a utilização dos seguintes pontos:

- Utilizar versionamento em documentos gerados, para facilitar o entendimento de mudanças nesses documentos provenientes das iterações e ou modificações;
- Fazer revisões sempre que fossem necessárias, ou seja, caso aconteçam mudanças em funcionalidades, requisitos, entre outros aspectos, para que a documentação esteja sempre atualizada com o desenvolvimento do sistema.

A quinta *sprint* foi instanciada com a mesma duração de tempo das demais e com a pretensão de corrigir e finalizar as etapas inacabadas da iteração anterior. Sendo assim, nela entraram os artefatos não concluídos na anterior e mais um requisito novo. Houve a entrada de mais um membro na equipe. O PDS continuou o mesmo, acrescentando apenas a sugestão de versionamento de documentação gerada que auxiliaria a equipe a manter controle das modificações realizadas. A sugestão de revisões também se manteve, porém não foi realizada porque não havia ocorrido mudança nos artefatos previamente construídos. A equipe não demonstrou dúvidas ou problemas ao utilizar o PDS, mesmo tendo um novo membro.

As atividades não concluídas na *sprint* anterior foram executadas e concluídas nesta *sprint* (construção de diagramas de sequência, e execução das atividades relacionadas a testes para um requisito). Nesta iteração também houve o acréscimo de artefatos não realizados na terceira *sprint*, tais como: construção dos casos de testes e *scripts*, realização de testes e construção de diagramas de sequência das *User Stories* elaboradas. Todos esses artefatos foram construídos pela equipe. Apesar do atraso da especificação, o desenvolvedor pode implementar o requisito a tempo de ser finalizado na iteração.

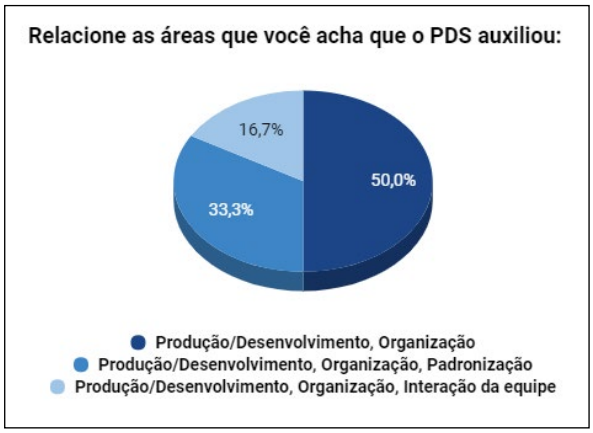
Das tarefas previstas para esta *sprint*, algumas não puderam ser executadas: construção de diagramas de sequência para dois requisitos (sendo um requisito acrescentado nessa *sprint* e dois da *sprint* anterior), geração de um protótipo de uma *User Story* (dessa *sprint*), elaboração do caso de teste, construção do *script* e a execução do teste da mesma *User Story*. Todos os demais artefatos foram concluídos. De acordo com a equipe, as tarefas não puderam ser concluídas por falta de tempo.

Ao final da *sprint*, dos oito membros da equipe, sete responderam ao formulário, a partir dos quais obtivemos os seguintes dados:

- Quando perguntados se o PDS tem todos os elementos necessários para execução do projeto, 100% deles disse que "sim". Sendo assim, os elementos definidos no PDS permanecerão para próximas iterações do projeto;

- Quando perguntados se o uso do PDS melhorou o desempenho das *sprint*s 100% deles disse que "sim". Com isso, a equipe escolheu continuar utilizando-se do PDS nas futuras iterações;
- Sobre as áreas que a utilização do PDS mais auxiliou, entre as opções "produção/desenvolvimento", "padronização", "organização", "interação com a equipe", "nenhum", 50% apontou para "produção/desenvolvimento" e "organização", conforme pode ser visto na Figura 4. A partir dessa resposta, será analisado mais a fundo o porquê destas áreas serem as mais auxiliadas, de maneira que todas as outras também possam ser melhoradas. Essa resposta também demonstra que o PDS proporcionou uma ajuda maior em áreas com um grande déficit nas *sprint*s anteriores, como desenvolvimento/produção e organização das atividades e elementos a serem utilizados.

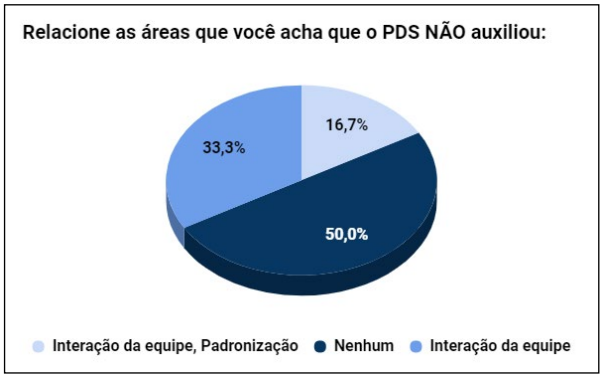
Figura 4 – Áreas que o PDS auxiliou



Fonte: Elaboração própria

- Quanto às áreas que o PDS não auxiliou, com as mesmas opções da pergunta anterior, 50% apontou para "nenhum", enquanto 33,3% indicou "interação da equipe" e "padronização", conforme pode ser visto na Figura 5. Com a obtenção dessa resposta, pode-se concluir que o PDS tem demonstrado ser um grande auxiliador no desenvolvimento do projeto, pois a porcentagem para a opção "nenhuma" foi bem relevante. Com relação às áreas relatadas, serão levantados com a equipe quais fatores precisam ser melhorados no PDS, para auxiliar mais as áreas de interação com a equipe e sua padronização.

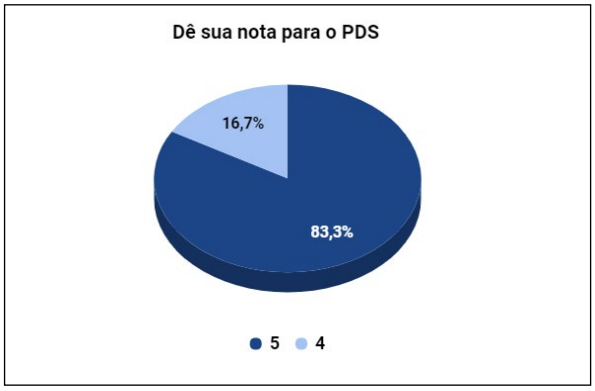
Figura 5 – Áreas que o PDS não auxiliou



Fonte: Elaboração própria

- Quando perguntados sobre sugestões para o PDS, todos responderam não ter nenhuma naquele momento;
- Quando perguntados sobre a nota que dariam para o PDS, em uma escala de 1 (muito ruim) a 5 (muito bom), 80% indicaram 5 e 20% avaliaram o PDS com nota 4, conforme apresentado na Figura 6. Com isso, demonstra-se que o PDS conseguiu auxiliar muito e se relaciona a um alto nível de contentamento da equipe.

Figura 6 – Avaliação do PDS



Fonte: Elaboração própria

- Sobre as considerações finais referentes à utilização do PDS, os relatos apontaram: "A existência do processo no projeto é muito importante porque garante que a execução do projeto sai de acordo com o esperado", escreveu um membro da equipe; outro integrante respondeu que "Demorou um

pouco para a equipe pegar o ritmo, mas ajudou bastante para o desenvolvimento e o entendimento do andamento do projeto"; e por fim, outro integrante relatou "Gostei, ajudou muito no desenvolvimento". Sendo assim, o PDS mostrou aumentar o desempenho da equipe e desenvolvimento do projeto e que ainda precisa ser instanciado mais vezes para que possam ter maturidade com tais atividades.

Por fim, a equipe será consultada mais profundamente sobre o que poderia ser acrescentado ao PDS para que fossem melhoradas as áreas citadas na quarta questão. Com isso, o PDS será modificado (caso sejam sugeridas mudanças) para ser instanciado na próxima *sprint*. No geral, a equipe não apresentou nenhuma sugestão de mudança e demonstrou satisfação, como foi constatado no formulário avaliativo mostrado na Figura 6. A avaliação do PDS continuará a ser realizada a cada interação para melhoria contínua.

6 Conclusão

Diante da diversidade de tantos cenários e projetos de software, sugere-se a avaliação da necessidade de adaptar os processos e metodologias já existentes, a fim de que eles possam atender da melhor maneira às especificidades dos requisitos propostos por cada cenário e projeto. Sendo assim, o PDS proposto pelo GPES teve como finalidade auxiliar os integrantes deste grupo na construção do ABPRES e também validar as boas práticas utilizadas no projeto RPPDS, aplicadas ao PDS.

Com a instanciação das práticas do PDS, a equipe foi desenvolvendo maturidade e afinidade com o processo, no decorrer das interações, o que trouxe dinamicidade ao desenvolvimento do projeto.

Observando que as referências a processos de software estão sendo cada vez mais adaptadas a cenários específicos, tanto para empresas como para a academia, e com o intuito de auxiliar, da melhor maneira, a construção dos produtos de software do GPES, proporcionando mais qualidade ao processo e aos produtos em si, os integrantes do projeto RPPDS viabilizaram um processo que ainda busca agregar valores aos discentes e que serve como validador das boas práticas pesquisadas pelos integrantes do projeto.

Inicialmente nosso cliente foi a própria equipe do GPES, pois a construção do PDS objetivou orientar os trabalhos dos alunos, que foram acompanhados pelos professores orientadores. A partir do feedback da equipe sobre as etapas realizadas e produtos gerados, acredita-se que a aplicação do PDS resultou em melhorias no desenvolvimento do projeto, entendimento do sistema, interação entre a equipe, além de ter sido bem recebido pelo grupo, sendo sugeridas poucas mudanças. Os professores orientadores concluíram que as demandas por um processo de desenvolvimento em um contexto acadêmico seguiram os mesmos padrões daqueles encontrados no meio empresarial, onde a adaptação dos processos aos contextos de uso é fundamental para a evolução dos sistemas.

O PDS deverá ser aplicado no referido projeto e certamente em outros que possam surgir no GPES, com o intuito de consolidar as suas práticas entre os membros do GPES. A ideia é que ele seja otimizado a cada iteração, de acordo com o feedback da equipe, sempre de forma adaptável aos contextos e às mudanças que poderão surgir.

Como melhorias futuras, atividades poderão ser inseridas ao PDS, como *o Acceptance Test Driven Development* – Desenvolvimento Orientado a Testes de Aceitação (ATDD), ou o *Behaviour Driven Development* – Desenvolvimento Orientado ao Comportamento (BDD), além de serem identificados indicadores para melhorar o acompanhamento do projeto e do PDS, a fim de proporcionarem maior suporte e qualidade à disciplina e ao processo em si; outras atividades também poderão ser modificadas (opcional) ou removidas, de acordo com a necessidade e avaliação da equipe sobre a aplicação do PDS.

Por meio de um trabalho cíclico e contínuo de busca de novas práticas e retroalimentação do PDS, como trabalhos futuros esta pesquisa deve fomentar o aprendizado e a contextualização dessas práticas em PDS. Indo mais além, esperamos que esse feedback – de uso dessas práticas – seja repassado ao público externo através do ABPRES.

REFERÊNCIAS

FOWLER, Martin. The New Methodology. **MartinFowler.com**, 13 dez. 2005. Disponível em: <https://martinfowler.com/articles/newMethodology.html>. Acesso em: 27 set. 2017.

GELLER, Marla; ARAÚJO, Carlos A. P.; ELIAS, João; BASTOS, Mythian; ROMA, Tayna; MARTOS, Pedro; KNEBEL, Clóvis. P@PSI (Processo Ágil para Pequenos Sistemas): Processo de Desenvolvimento de Software Adaptado para o Ensino nos Cursos de

Graduação. **Urissanê**, v. 1, n. 1, 2008. Disponível em: <http://www.periodicos.ulbra.br/index.php/urissane/article/view/1059/822>. Acesso em: 13 set. 2017.

MAINART, Domingos de A.; SANTOS, Ciro M. Desenvolvimento de Software: Processos ágeis ou tradicionais? Uma visão crítica. In: ENCONTRO ANUAL DE COMPUTAÇÃO, 8., 2010, Catalão-GO. **Anais...** Catalão-GO: Universidade Federal de Goiás, 2010. Disponível em: <http://www.enacomp.com.br/2010/cd/artigos/completos/enacomp2010_4.pdf>. Acesso em: 27 set. 2017.

MEDEIROS, J. D. R. V; ALVES, D. C. P; VASCONCELOS, A. M. L.; SCHUENEMANN, C. T. L. L. S.; WANDERLEY, E. Engenharia de requisitos em projetos ágeis: uma revisão sistemática da literatura. **Revista Principia – Divulgação Científica e Tecnológica do IFPB**, n. 28, p. 11-24, dez. 2015.

MORAIS, A. D. S.; ALMEIDA, S. L. F.; SILVA, S. C.; RANGEL, H. J. L.; LIRA, H. B.; RODRIGUES, N. N. **Boas Práticas Reconfiguráveis em Engenharia de Software**. In: EXPOTEC, 2016, João Pessoa. (Trabalho apresentado na forma de pôster).

MORAIS, A. D. S.; RANGEL, H. J. L.; LIRA, H. B.; RODRIGUES, N. N.; MEDEIROS, F. P. A. Repositório de Práticas em Gerenciamento de Projetos de Software. In: SEMANA DE CIÊNCIA E TECNOLOGIA DO IFPB, 11, 2016, João Pessoa. **Anais...** João Pessoa: IFPB, 2017a. p. 364-366. Disponível em: <http://editora.ifpb.edu.br/index.php/ifpb/catalog/view/72/40/114-1>. Acesso em: 12 set. 2017.

MORAIS, A. D. S.; LIMA, C. D. Q.; RANGEL, H. J. L.; LIRA, H. B.; RODRIGUES, N. N.; ALMEIDA, S. L. F.; BARBOSA, T. C. Proposta de Framework de Gerenciamento Ágil de Projetos do Grupo de Pesquisa em Engenharia de Software do IFPB. In: CONGRESSO DA SOCIEDADE BRASILEIRA DE COMPUTAÇÃO, 37.; Encontro Nacional de Computação dos Institutos Federais, 4., 2017, São Paulo. **Anais...** São Paulo: Sociedade Brasileira de Computação, 2017b. p. 212-215. Disponível em: <http://csbc2017.mackenzie.br/public/files/4-encompif/2.pdf>. Acesso em: 12 set. 2017.

PAULA FILHO, W. P. **Engenharia de Software**: fundamentos, métodos e padrões. 3. ed. Rio de Janeiro: LTC, 2013.

PRESSMAN, Roger; MAXIM, Bruce. **Engenharia de Software**: uma abordagem profissional. 8. ed. Porto Alegre: AMGH, 2016.

PRODANOV, C. C.; FREITAS, E. C. **Metodologia do trabalho científico:** métodos e técnicas da pesquisa e do trabalho acadêmico. 2. ed. Novo Hamburgo: Feevale, 2013. Disponível em: <http://www.feevale.br/Comum/midias/8807f05a-14d0-4d5b-b1ad-1538f3aef538/E-book%20Metodologia%20do%20Trabalho%20Cientifico.pdf>. Acesso em: fev. 2017.

SAUVÉ, J. P.; NETO, E. R; FEITOSA, A. L.; SILVA, A. F.; NETO, F. R. de A.; CABRAL, F. W. L. **XP1**: Um Processo de Desenvolvimento. 21 mar. 2007. Disponível em: <http://www.dsc.ufcg.edu.br/~jacques/projetos/common/xp1/xp1.html>. Acesso em: 14 set. 2017.

SBROCCO, José Henrique Teixeira de Carvalho; MACEDO, Paulo Cesar de. **Metodologias ágeis**: engenharia de software sob medida. 1. ed. São Paulo: Érica, 2012.

SILVA, S. C.; ALMEIDA, S. L. F.; RANGEL, H. J. L.; LIRA, H. B.; RODRIGUES, N. N. Repositório de Práticas em Processos de Desenvolvimento de Software. In: SEMANA DE CIÊNCIA E TECNOLOGIA IFPB, 11., 2016, João Pessoa. **Anais...** João Pessoa: IFPB, 2017. p. 367-370. Disponível em: <http://editora.ifpb.edu.br/index.php/ifpb/catalog/view/72/40/114-1>. Acesso em: 12 set. 2017.

SOMMERVILLE, Ian. **Engenharia de Software**. 8. ed. São Paulo: Pearson Addison-Wesley, 2007.

_____. **Engenharia de Software.** 9. ed. São Paulo: Pearson Prentice Hall, 2011.