INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DA PARAÍBA
UNIDADE ACADÊMICA DE INFORMAÇÃO E COMUNICAÇÃO
COORDENAÇÃO DO CURSO SUPERIOR DE TECNOLOGIA EM SISTEMAS PARA INTERNET

RELATÓRIO DE ESTÁGIO

Processo de desenvolvimento para agências digitais de pequeno porte

Paulo Virgílio Gomes Linhares Neto

João Pessoa – PB
Novembro/2015

Instituto Federal de Educação, Ciência e Tecnologia da Paraíba
Unidade Acadêmica de Informação e Comunicação
Coordenação do CST em Sistemas para Internet

Processo de Desenvolvimento de Software para agências digitais de pequeno porte

Paulo Virgílio Gomes Linhares Neto

Relatório de Estágio Supervisionado apresentado à disciplina Estágio Supervisionado da Coordenação do Curso Superior de Tecnologia em Sistemas para Internet do Instituto Federal de Educação, Ciência e Tecnologia da Paraíba como requisito parcial para obtenção do grau de Tecnólogo em Sistemas para Internet.

Orientador: Nadja da Nóbrega Rodrigues
Supervisor: João Paulo Barbosa Neto
Coordenador do Curso: Valéria Cavalcanti
Empresa: Brazil Digital Experience LTDA - ME
Período: 09/03/2015 a 16/07/2015

João Pessoa

APROVAÇÃO

João Paulo Barbosa Neto

Supervisor da Empresa

Valéria Cavalcanti

Coordenador do CST de Sistemas para Internet

Nadja da Nóbrega Rodrigues

Professor Orientador

Paulo Virgílio Gomes

Linhares Neto

Estagiário

Para Vera Lúcia Silva Pessoa, responsável por guiar meus passos em direção ao conhecimento, me estimulando sempre a nunca desistir de aprender.

AGRADECIMENTOS

Aos professores do Instituto Federal de Educação, Ciência e Tecnologia (IFPB), por estarem sempre dispostos a multiplicar o conhecimento sem medir esforços, à equipe da BRZ Digital por confiar no meu potencial na busca contínua da excelência e à minha orientadora, Nadja Rodrigues, por compartilhar esse momento, ajudando sempre que necessário.

RESUMO

Retrato do cenário de desenvolvimento de software da BRZ Digital, uma agência digital, que está inserida no mercado publicitário em João Pessoa – PB. A trajetória de definição, aplicação e refatoração de um processo de desenvolvimento de software baseado nas melhores práticas difundidas na indústria, aliadas a necessidades específicas do mercado local.

Palavras-chave: Engenharia de software, processo de desenvolvimento de software, metodologia ágil, agência digital.

LISTA DE FIGURAS

Figura 1: Diagrama de etapas da primeira versão do PDS

Figura 2: Diagrama de etapas da segunda versão

Figura 3: Diagramas de casos de uso do sistema de lista de presentes

Figura 4: Painel de gerenciamento de tarefas na última release

LISTA DE SIGLAS E ABREVIATURAS

BABOK	Business Analysis Body of Knowledge
PDS	Processo de desenvolvimento de Software

SUMÁRIO

LISTA DE FIGURAS 8

LISTA DE SIGLAS E ABREVIATURAS 9

1. Introdução 11

1.1 Objetivo 11

1.2 A Empresa 11

1.3 Descrição Geral das Atividades 13

1.4 Organização do Relatório 13

2. Embasamento Teórico 14

2.1 Processo de desenvolvimento de software 14

2.2 Disciplinas da engenharia de software 14

2.3 Modelos de processo de software 17

2.3.1 Modelo linear sequencial (cascata) 17

2.3.2 Modelo incremental 17

2.4 Metodologias ágeis 18

2.4.1 Scrum 19

2.4.2 *Extreme Programming* (XP) 21

3. Atividades Realizadas 24

3.1 Etapa centralizada (cascata) 24

3.2 Etapa descentralizada (iterativa e incremental) 37

3.3 Execução de projetos utilizando o PDS 45

4. Considerações Finais 48

Referências 50

Apêndice A – Documento de *User Stories*: Casa Tudo Premium 51

1. Introdução

O presente relatório descreve o cenário referente ao estudo de caso realizado em 2015 para elaboração e implantação do Processo de Desenvolvimento de Software (PDS) da Brazil Digital Experience Ltda - ME, localizada em João Pessoa – PB. A definição do processo a ser utilizado baseou-se no estudo de diversas metodologias e conceitos utilizados na Engenharia de Software, especialmente aqueles relacionados à sistematização das suas etapas, e através de uma análise comparativa nos seus diversos aspectos, buscou-se pensar em um processo que atendesse aos principais requisitos para o trabalho em uma agência digital de pequeno porte: Agilidade de execução dos projetos, simplicidade de entendimento, adaptabilidade em ambientes incertos, baixo número de membros na equipe e documentação clara e objetiva.

Além do PDS, serão apresentados exemplos de projetos utilizando as técnicas descritas pelo PDS, bem como os resultados alcançados através de sua utilização.

1.1 Objetivo

Elaborar o processo de desenvolvimento de software da BRZ Digital a fim de aumentar a qualidade dos produtos desenvolvidos pela empresa.

1.2 A Empresa

A BRZ Digital é uma agência de internet, fundada em novembro de 2014 com o objetivo de prover soluções nas áreas de marketing e publicidade, no âmbito digital. Desde sua criação, a empresa vem tentando melhorar a qualidade dos seus produtos através da organização dos seus processos internos.

O estagiário é co-fundador/sócio da BRZ Digital, o que implica que ele possuiu uma maior liberdade na tomada de decisão da empresa, facilitando o desenvolvimento do processo descrito neste relatório.

Durante o estágio, a equipe de desenvolvimento possuiu quatro membros, inicialmente organizados da seguinte maneira: um designer de interfaces, dois

desenvolvedores web e um gerente de projetos, que também possuía atribuições de analista de requisitos.

As tarefas da equipe foram distribuídas entre os papéis da seguinte maneira:

Gerente de Projetos: responsável por planejar, estimar e controlar variáveis como tempo, custo e qualidade dos produtos desenvolvidos pela empresa, teve também como obrigação direcionar a equipe de desenvolvimento, priorizando e canalizando os seus recursos para atingir os objetivos da melhor forma possível.

Analista de Requisitos (Especificador): responsável por fazer a conexão entre as necessidades do cliente e a equipe de desenvolvimento, entender as especificidades de cada cliente e sugerir as melhores soluções para que a equipe de desenvolvimento pudesse produzir os produtos de software.

Designer de Interface: responsável por montar a parte visual dos projetos em formato que pudesse ser utilizado para o desenvolvimento front-end, e que refletisse os requisitos e especificidades de cada projeto, seguindo as diretrizes e padrões adotados pela empresa.

Desenvolvedor Web: responsável por transformar os projetos de interface em código front-end de alto nível, bem como elaborar soluções para os problemas propostos pela análise de requisitos específica de cada projeto, transformando-as em código back-end.

A BRZ Digital trabalha em grande parte das vezes em parceria com agências de publicidade tradicionais, funcionando como um núcleo de desenvolvimento externo através do modelo de terceirização. Por esse motivo, muitas vezes a sua equipe não tem contato direto com o usuário de seus produtos, que por sua vez, são parte de campanhas publicitárias e possuem características diferenciadas de produtos de software comum, como por exemplo, os prazos fixos (e geralmente curtos), além de alta volatilidade na especificação dos requisitos. Essas características demandam que a equipe seja flexível e ágil. Também por esse motivo, é dificil para a equipe controlar as variáveis referentes à gerência dos projetos (tempo, escopo, recursos, entre outras).

Os produtos desenvolvidos pela BRZ Digital em parceria com as agências, em sua maioria fogem do padrão de desenvolvimento de sistemas comum. Como citado

anteriormente, os produtos geralmente são parte de campanhas publicitárias, como *hotsites*, ferramentas de interação com redes sociais, entre outros.

Para atingir esse propósito, foi necessário o desenvolvimento de um PDS que balanceasse todas essas características e, além disso, tivesse uma implantação de simples entendimento, pois nem todos os membros da equipe possuem conhecimento sobre a engenharia de software.

1.3 Descrição Geral das Atividades

As atividades desenvolvidas durante o período de estágio foram as seguintes:

- Análise das metodologias de desenvolvimento de software mais difundidas na indústria de software.
- Criação do PDS da empresa.
- Desenvolvimento de sites e sistemas utilizando o PDS proposto.

1.4 Organização do Relatório

Este relatório está divido em três partes principais: a primeira parte consiste de uma breve introdução, a segunda parte descreve as atividades desenvolvidas no estágio, inclusive o embasamento teórico, resultado da revisão bibliográfica sobre as metodologias, e a terceira e última parte apresenta as considerações finais.

2. Embasamento Teórico

2.1 Processo de desenvolvimento de software

Processo de Desenvolvimento de Software (PDS) pode ser definido como um conjunto de atividades necessárias para a criação de um produto de software. Pressman (2001) define um processo de software como a moldura para as tarefas que são necessárias para construir software de alta qualidade. Os PDS servem para definir quem, quando, como e o quê deve ser feito para alcançar um determinado objetivo. Eles também são importantes para a organização de equipes, pois um processo bem definido implica numa equipe que sabe bem qual é o seu papel, quais são as ferramentas necessárias para executar suas tarefas, a ordem que as mesmas devem ser executadas e suas respectivas dependências. Para apoiar a definição dos processos de software, a engenharia de software divide as técnicas utilizadas em disciplinas como modelagem de negócio, análise de requisitos, análise e projeto, implementação (codificação), testes, implantação, gerência de projetos, entre outras.

2.2 Disciplinas da engenharia de software

As disciplinas da engenharia de software têm como objetivo agrupar os métodos, técnicas e ferramentas utilizadas na construção de produtos de software em áreas de atuação afins. Alguns exemplos de disciplinas seguem abaixo:

Modelagem de negócio:

A modelagem de negócio consiste em mapear e documentar os processos administrativos de uma empresa com o objetivo de melhorar o entendimento da equipe desenvolvedora sobre a proposta de valor oferecida pela mesma. Para alcançar esse objetivo existem vários métodos e ferramentas, entre eles a entrevista de membros atuantes nos setores onde o produto de software será implantado para que o produto possa refletir a realidade da empresa.

A definição do *BABOK* (*Business Analysis Body of Knowledge*) segue abaixo:

> "Análise de Negócios é o conjunto de atividades e técnicas utilizadas para servir como ligação entre as partes interessadas, no intuito de compreender a estrutura,

> políticas e operações de uma organização e para recomendar soluções que permitam que a organização alcance suas metas."
> (BABOK, 2011, p. 5)

Análise de requisitos:

Consiste no ato de elencar as necessidades ou problemas que o produto de software deve sanar, criando um roteiro inicial para o desenvolvimento da solução (produto). Sommerville (2011) divide os requisitos em duas categorias: requisitos de usuário e requisitos de sistema, que seguem respectivamente as seguintes definições:

> "Requisitos de usuário são declarações, em uma linguagem natural com diagramas, de quais serviços o sistema deverá fornecer a seus usuários e as restrições com as quais este deve operar. [...] Requisitos de sistema são descrições mais detalhadas das funções, serviços e restrições operacionais do sistema de software. O documento de requisitos do sistema (às vezes, chamado especificação funcional) deve definir exatamente o que deve ser implementado. Pode ser parte do contrato entre o comprador do sistema e os desenvolvedores de software."
> (SOMMERVILLE, 2011, p. 58)

Projeto de software:

> "O projeto de arquitetura está preocupado com a compreensão de como um sistema deve ser organizado e com a estrutura geral desse sistema. No modelo do processo de desenvolvimento de software [...]. É o elo crítico entre o projeto e a engenharia de requisitos, pois identifica os principais componentes estruturais de um sistema e os relacionamentos entre eles. O resultado do processo de projeto de arquitetura é um modelo de arquitetura que descreve como o sistema está organizado em um conjunto de componentes de comunicação. Em processos ágeis, geralmente se aceita que um estágio inicial do processo de desenvolvimento se preocupe com o estabelecimento de uma arquitetura global do sistema. O desenvolvimento incremental de arquiteturas geralmente não é bem-sucedido. Embora a refatoração de componentes em resposta às mudanças costume ser relativamente fácil, a refatoração de uma arquitetura é geralmente cara."
> (SOMMERVILLE, 2011 p. 103)

Com base na definição de Sommerville, podemos entender que o projeto de arquitetura de software tem como objetivo guiar o desenvolvimento da solução, fazendo a conexão entre os requisitos de negócio e a implementação técnica, servindo como base para a escolha de tecnologias, paradigmas de programação,modelos de comunicação entre sub-sistemas, entre outros.

Implementação (codificação):

Segundo Pressman (2001), essa é a etapa que se traduz o projeto em linguagem legível para a máquina, ou seja, transforma-se a arquitetura em um produto utilizável. O autor ainda diz que se o projeto for feito de maneira detalhada, a etapa de codificação pode ser executada de forma mecânica. Também é nesta etapa que são utilizados os padrões de projeto de software.

Testes:

Segundo Pressman (2001), a disciplina de testes de software é um elemento crucial na garantia de qualidade seja da especificação (requisitos), do projeto ou do código gerado. Esta disciplina tem como objetivo principal corrigir possíveis erros nas diversas etapas do processo de desenvolvimento de um produto de software através de uma série de técnicas e garantir que o produto gerado irá melhorar a proposta de valor do cliente, resolvendo problemas reais do seu dia-a-dia.

Implantação:

Define as etapas necessárias para a implantação do produto no ambiente de produção do cliente, bem como o treinamento da equipe que irá manuseá-lo e o suporte para que tudo ocorra da forma mais suave possível.

Gerenciamento de Projetos:

“[...]O Gerenciamento de Projetos, portanto, é a aplicação de conhecimentos, habilidades e técnicas para a execução de projetos de forma efetiva e eficaz.[...]” (PMI, 2015).

A disciplina de gerência de projetos tem como objetivo planejar, executar e controlar variáveis como escopo, tempo, recursos (físicos e humanos), otimizando-as para entregar melhores resultados.

2.3 Modelos de processo de software

Modelos de processo de software são padrões de processos estabelecidos pela indústria para determinados tipos de projetos. Os modelos são constituídos de conjuntos específicos de tarefas agrupados de uma forma também específica para alcançar os objetivos dos projetos que se encaixem neles. Alguns exemplos de modelos seguem abaixo:

2.3.1 Modelo linear sequencial (cascata)

"Às vezes chamado de ciclo de vida clássico ou modelo cascata, o modelo linear sequencial sugere uma aproximação sequencial, sistemática ao desenvolvimento de software [...]" (PRESSMAN, 2001, p. 29)

O modelo linear sequencial divide o projeto em cinco etapas: análise, projeto, codificação, testes e suporte. Essas etapas são executadas uma após a outra, geralmente uma única vez durante todo o projeto. Os artefatos gerados durante cada etapa são tomados como verdade absoluta durante toda a criação do produto, o que pode vir a ser um problema, principalmente para projetos grandes e de longo prazo, onde os requisitos para implementação da solução podem mudar, ou até mesmo um mal-entendido na elaboração dos mesmos só será percebido no final do projeto, quando ele for implementado e/ou testado. Além disso, muitas vezes o cliente não entende bem quais são as suas reais necessidades ou quais soluções ele pode obter através de um produto de software, trazendo também mudanças no curso do projeto.

Por outro lado, uma das vantagens do modelo cascata é o alto nível de definição das etapas e o que se espera de cada uma. Uma vez que a equipe entende quais são as etapas e o que fazer em cada uma delas, a sua comunicação melhora e com isso, melhoram também os seus resultados.

2.3.2 Modelo incremental

> "O desenvolvimento incremental é baseado na ideia de desenvolver uma implementação inicial, expô-la aos comentários dos usuários e continuar por meio da criação de várias versões até que um sistema adequado seja desenvolvido. Atividades de especificação, desenvolvimento e validação são intercaladas, e não separadas, com rápido *feedback* entre todas as atividades."
>
> (SOMMERVILLE, 2011 p. 21)

O modelo incremental inicialmente modela a arquitetura básica de toda a solução, depois divide os requisitos em pedaços (incrementos), que são partes da solução que serão entregues gradativamente para que o cliente possa analisar, utilizar e dar *feedback* válido para a equipe responsável pela elaboração da solução, fazendo com que possíveis distinções entre a solução proposta e as suas necessidades reais sejam eliminadas o mais cedo possível. As metodologias que se encaixam no modelo incremental possuem uma alta incidência de refatoração, entretanto, em menor escala, visto que os incrementos que são entregues são apenas fatias da solução, o que torna a refatoração mais simples e ágil.

Esse modelo geralmente atua com bastante proximidade ao cliente, pois além da necessidade de *feedback* constante, a equipe define o que estará e qual a ordem de implementação de cada incremento baseado nas prioridades estabelecidas por ele.

O modelo incremental também serve de base para a criação de várias metodologias ágeis.

2.4 Metodologias ágeis

As metodologias ágeis são as metologias que seguem o manifesto ágil.Segundo Beck et al (2001), valorizam:

- "Indivíduos e interações mais que processos e ferramentas"
- "Software em funcionamento mais que documentação abrangente"
- "Colaboração com o cliente mais que negociação de contratos"
- "Responder a mudanças mais que seguir um plano"

As metodologias ágeis também seguem princípios que vão desde a abertura para as mudanças de requisitos até a reflexão contínua sobre a atuação da equipe para melhoria interna da qualidade. Alguns exemplos de metodologias seguem abaixo:

2.4.1 Scrum

"Um framework dentro do qual pessoas podem tratar e resolver problemas complexos e adaptativos, enquanto produtiva e criativamente entregam produtos com o mais alto valor possível." (SCHWABER; SUTHERLAND, 2013, p. 3).

O scrum é uma metodologia ágil que se baseia em três pilares: transparência, inspeção e adaptação. Transparência para que todos os membros da equipe tenham a mesma visão do produto e do processo, além da definição de pronto, que deve ser única entre os que realizam o trabalho e os que o aprovam. Inspeção para que sejam detectadas falhas no produto, feitas preferencialmente pelos especialistas no assunto, mas que não sejam tão frequentes a ponto de atrapalhar as atividades executivas. Adaptação para quando o resultado da inspeção apontar falhas no produto que possam diferenciar as necessidades reais do cliente da solução proposta, sejam feitas as mudanças necessárias da maneira mais ágil possível, minimizando o desperdício de tempo no desenvolvimento de partes do produto que não serão utilizadas.

O time scrum é composto por três tipos de profissionais: *Product owner, scrum master* e desenvolvedores. Existem apenas um *product owner* e um *scrum master* por equipe, enquanto recomenda-se que existam de três a nove desenvolvedores por equipe. O *product owner* é o representante do cliente dentro do time scrum e tem como responsabilidades garantir a clareza dos itens do *product backlog* para que a equipe de desenvolvimento possa desenvolvê-los corretamente, ou ainda priorizá-los de acordo com as necessidades do cliente ou a complexidade e outros fatores de risco, por exemplo. O *scrum master* tem como objetivo fazer com que o scrum seja entendido e aplicado pela equipe, maximizando a eficiência e melhorando a comunicação da mesma. É importante notar que nem o *scrum master* nem o *product owner* atuam como gerentes de projeto, pois a equipe de desenvolvedores deve ser auto-gerenciável e encontrar a sua própria maneira de entregar os incrementos propostos no espaço de tempo proposto. O time de desenvolvimento engloba todos os profissionais responsáveis pela criação do produto, para isso, deve possuir uma pluralidade de profissionais e habilidades que serão utilizadas para garantir que todos os requisitos serão atendidos como esperado

no *product backlog*. Como citado anteriormente, a equipe de desenvolvimento é totalmente auto-gerenciável, ou seja, ninguém, nem mesmo o *scrum master* deve dizê-la qual a melhor maneira de transformar o *backlog* em um produto usável para o cliente.

O scrum possui uma série de eventos que foram definidos para definir determinados acontecimentos durante o projeto, esses eventos têm como objetivo diminuir o número de reuniões não previstas na metodologia, além de criar uma rotina de execução de tarefas que deve ser entendida e aplicada por toda a equipe. Alguns dos eventos seguem abaixo:

Sprint:

O elemento principal do scrum, que pode ser comparado a um incremento do modelo incremental, é um pedaço do *product backlog* que será implementado num dado período de tempo não maior do que um mês, mas que será definido e inalterado durante sua execução. O incremento gerado como resultado de uma sprint deve ser completamente funcional, mesmo que não contenha o projeto completo. Os requisitos que serão priorizados em uma *sprint* devem ser escolhidos pelo *product owner* e aprovados pela equipe desenvolvedores, confirmando que o tempo escolhido para execução da mesma é suficiente.

Reunião Diária:

A reunião diária é uma reunião entre todos os membros da equipe, que não deve ultrapassar 15 minutos e tem como objetivos inspecionar o trabalho executado desde a última reunião diária e planejar o trabalho a ser executado nas próximas 24 horas. Além de inspecionar o trabalho, a reunião diária também serve para sanar possíveis impedimentos que aconteçam da forma mais rápida possível, por isso é necessário que todos os membros exponham suas dúvidas nela.

Para complementar e documentar os eventos presentes na metodologia, o scrum também declara alguns artefatos:

Backlog do produto (*Product backlog*):

O backlog do produto é o conjunto de todos os requisitos necessários para a criação do produto. Ele é mantido e priorizado pelo *product owner*. É importante ressaltar que o backlog do produto é aberto a mudanças e se mantém vivo enquanto o produto existir, podendo ser repriorizado a cada adição de requisito.

Backlog da *sprint* (*Sprint backlog*):

O backlog da *sprint* é uma delimitação do backlog do produto destinado a ser implementado no espaço de uma *sprint*. *Sprints* podem ter tempos variados dentro de um projeto, desde que não quebrem as regras estabelecidas pela metodologia. Essa definição do tempo da *sprint* é feita pela equipe de desenvolvimento de acordo com as definição de pronto para um determinado incremento. Ao contrário do backlog do produto, o backlog da *sprint* não pode sofrer alterações durante a sua execução. Nesse caso, novos requisitos ou possíveis ajustes nos requisitos de uma *sprint* serão adicionados ao backlog do produto e repriorizados em *sprints* futuras.

Incremento:

É o conjunto de todos os requisitos implementados de uma *sprint* somados aos requisitos implementados em *sprints* anteriores. O incremento deve ser um produto utilizável, ou seja, deve atender sempre o estado de pronto, independente da decisão do *product owner* de liberá-lo para o cliente ou não.

2.4.2 *Extreme Programming* (XP)

A metodologia XP também segue o manifesto ágil, por isso tem muitas características parecidas com o scrum, como por exemplo as reuniões curtas diárias, ou as entregas de incrementos em pequenas releases. Diferentemente do scrum, o XP

é uma metodologia totalmente voltada para o desenvolvimento de software, por isso tem algumas regras específicas para esse ambiente, como por exemplo:

- Utilizar *user stories* ao invés de casos de uso, onde as *user stories* são escritas pelo cliente, não por especificadores. Além de criar as estórias, os usuários também ajudam na criação dos testes de aceitação para as respectivas estórias.
- Incentivar a equipe a trocar de papéis com frequência para que todos saibam o necessário para que nenhum membro seja indispensável e/ou insubstituível.
- O desenvolvimento é guiado pelos testes, ou seja, os testes são desenvolvidos antes mesmo do código do produto, que por sua vez é escrito para atender os requisitos descritos nos testes.
- Todo código deve ser testado e só será liberado para uso após passar em todos os testes.
- Toda a programação é feita em pares.
- O cliente deve estar disponível a todo momento para esclarecer dúvidas da equipe sobre os requisitos (*user stories*).

Para cumprir com as regras citadas acima, o XP prega os seguintes valores:

- Simplicidade: fazer apenas o que é pedido, nem mais, nem menos. Caminhar a passos curtos em direção ao objetivo, e atenuar as falhas quando elas acontecerem.
- Comunicação: todos da equipe se comunicam face a face diariamente e trabalham em conjunto, desde os requisitos até a codificação.
- *Feedback*: entregar software funcional ao final de cada iteração, ouvir atentamente o que o cliente/usuário tem a dizer e adaptar o processo de acordo com as necessidades.
- Respeito: todos respeitam os papéis uns dos outros, os desenvolvedores entendem que o cliente possui um *expertise* maior sobre o seu negócio, enquanto o cliente entende que os desenvolvedores têm maior capacidade para desenvolver a solução da melhor forma possível.

- Coragem: deixar sempre claras as estimativas e o progresso do projeto.

O quadro abaixo lista as características abordadas no desenvolvimento do PDS de forma relacionada aos modelos e práticas que foram utilizados como fonte de inspiração para a sua definição:

Modelo / Prática	Modelo cascata	Modelo iterativo e incremental	Metodologias ágeis	Scrum	*Extreme programming*
Definição em 4 etapas (APCT)	X				
Dividir o projeto em *releases*		X	X	X	X
Abraçar mudanças			X	X	X
Reuniões diárias				X	*X*
Product backlog e sprint backlog				X	X
Padrão *user story*				X	X

3. Atividades Realizadas

A realização do estágio se deu em dois grandes ciclos: centralizado e descentralizado. O primeiro foi dividido nas seguintes etapas: análise do ambiente da empresa, levantamento bibliográfico, criação do PDS e aplicação e feedback da equipe. Já o segundo consistiu em colher o feedback dado pela equipe, revisitar a literatura para adaptar as práticas do PDS à realidade da empresa, redefinir o PDS e aplicá-lo nos projetos. Além de alterar o PDS durante os dois ciclos, os papéis da equipe se alteraram durante a realização do estágio. A descrição completa dos papéis da equipe será feita dentro da descrição de cada ciclo, neste capítulo.

3.1 Etapa centralizada (cascata)

Para efeitos de nomenclatura, chamaremos esta etapa de centralizada ou cascata, pois os relacionamentos na equipe se deram de forma hierárquica, sendo que a equipe de criação e desenvolvimento respondia à figura de um gerente de projetos, que por sua vez, era responsável pela organização de todos os projetos da empresa através das atividades descritas pela disciplina de gerência de projetos.

O ciclo centralizado se iniciou com a análise do ambiente da empresa, que foi desde a análise individual dos perfis dos membros da equipe e suas peculiaridades até a análise dos clientes e parceiros com os quais a empresa se relacionava, passando também pelos tipos de projetos que a empresa executava, a frequência com que cada tipo ocorria e quais eram os problemas específicos em cada um deles.

É importante ressaltar que a definição do modelo sequencial (cascata) feita por Pressman (2001, p. 28) foi amplamente utilizada na criação da primeira etapa do PDS. O autor divide o ciclo de vida de um projeto em quatro etapas: análise, projeto, codificação e testes, etapas estas que são executadas uma única vez cada, nessa mesma sequência. Apesar de não utilizar totalmente o modelo sequêncial na definição do PDS, os conceitos APCT (Análise, Projeto, Codificação e Testes) foram adaptados para que todas as atividades executadas no processo se encaixassem em uma dessas etapas. O autor ainda divide o fluxo do projeto em duas macro etapas, que também foram adaptadas para utlização no PDS desenvolvido: planejamento e execução, onde o

planejamento engloba a análise e o projeto, enquanto a execução é composta por codificação e testes. Na primeira versão do PDS, a etapa de planejamento é executada uma única vez, enquanto a etapa de execução era executada de forma iterativa e incremental, sempre revisando a documentação gerada na etapa de planejamento no início de cada *release* e fazendo as adaptações necessárias para dar continuidade ao projeto. Esse modelo é uma mistura entre o cascata e o modelo iterativo e incremental, mas que no dia-a-dia acabou tornando-se mais próximo do modelo cascata, pois mesmo a macro etapa de execução muitas vezes era executada uma só vez durante alguns projetos.

A escolha desse modelo se deu porque os membros da empresa trabalhavam de forma totalmente sequencial antes da implementação do PDS, tornando difícil a quebra total desse paradigma, que viria se mostrar ineficiente mais tarde.

A empresa conta com uma única equipe para a execução dos vários projetos possui, equipe essa que é formada por quatro membros, que se dividem para preencher os papéis descritos no PDS. Os projetos são executados alternadamente e simultaneamente pela equipe, que alterna entre as diversas etapas descritas durante o seu dia-a-dia.

A análise do ambiente da empresa resultou em algumas características:

- A equipe é pequena e multidisciplinar, ou seja, seus membros possuem vários papéis (muitas vezes simultâneos) durante a execução dos projetos. Essa característica pode ser vista sob dois aspectos: por um lado, a comunicação interna é muito boa e todos os membros são acessíveis, tornando a resolução de pequenos problemas ou dúvidas mais simples, mas por outro lado, muitas vezes a equipe está sobrecarregada com tarefas de vários projetos e/ou disciplinas, deixando-a com pouco ou nenhum tempo extra para imprevistos ou alterações no escopo dos projetos sem que haja nenhum prejuízo.
- Nem todos os membros da equipe possuem conhecimento suficiente de engenharia de software para entender a parte técnica da definição de um PDS, fazendo com que ele precise ser de simples entendimento e pouco

disruptivo sobre como a equipe está acostumada a trabalhar em suas experiências anteriores.

- A maioria dos projetos é feitaem parceria com outras agências de publicidade (terceirizados), onde a equipe da BRZ funciona como um núcleo de desenvolvimento para a agência de publicidade (ou seja, a agência de publicidade, por sua vez, terceiriza o projeto de desenvolvimento para a BRZ). Esse cenário implica que a BRZ trabalha com restrições, ou seja, muitas vezes não possui flexibilidade para estipular custo, prazo e/ou escopo dos projetos, que são impostos pela agência de publicidade, que muitas vezes não tem nenhum conhecimento técnico para estipulá-las, fazendo com que a equipe da BRZ muitas vezes trabalhe em condições desfavoráveis nesse sentido. Além da necessidade de adaptação quase instantânea, a equipe precisa trabalhar para preencher todos os espaços de tempo possíveis, maximizando a eficiência e a agilidade na entrega dos projetos, pois como citado anteriormente, a maioria dos projetos era feita em conjunto com agências de publicidade, o que implica que eles interagiam diretamente com campanhas publicitárias e atrasos, por menores que sejam, podem resultar no fracasso total de um projeto, a ponto dele ser descartado. É importante frisar que neste tipo de projeto, a variável de gerência que menos (ou nunca) pode sofrer alterações é o tempo, enquanto as outras podem ser negociadas, apesar da baixa flexibilidade.

Com as informações sobre o ambiente onde a empresa estava inserida em mãos, a próxima etapa foi o levantamento bibliográfico, com o objetivo procurar soluções técnicas para os problemas encontrados. O levantamento bibliográfico feito nesta etapa serviu como base para o referencial teórico deste trabalho, além de embasar a tomada de decisão na elaboração do PDS proposto.

Este primeiro ciclo define seis papéis para a execução dos projetos, que foram divididos entre os membros da equipe, sendo eles gerente de projetos, atendimento, especificador, desenvolvedor, testador e gerente de configuração. As descrições dos papéis seguem abaixo:

Gerente de projetos:

Responsável por planejar, organizar, direcionar e controlar os recursos disponíveis dentre os projetos, além de controlar variáveis como tempo, custos, escopo e qualidade, procurando garantir sempre o melhor resultado tanto para o cliente quanto para a empresa. O gerente também é responsável por delegar e priorizar a execução das tarefas dentro da equipe, assim como remover quaisquer impedimentos que venham acontecer durante a execução das mesmas.

Atendimento:

O profissional de atendimento é o representante direto do cliente dentro da empresa, seu papel é colher as informações do negócio do cliente. É o atendimento que sugere para o cliente quais produtos ele deve desenvolver baseado no seu conhecimento tanto da empresa quanto do negócio do cliente, sendo também responsável pela elaboração inicial dos requisitos dos projetos. Vale salientar que em alguns projetos esse profissional pertence a outra empresa e o único contato que a equipe de especificação e desenvolvimento tem com ele é através da documentação gerada pelo mesmo.

Especificador:

Responsável por entender os problemas dos clientes e propor soluções para eles. Diferente do profissional de atendimento, o especificador tem um perfil mais técnico, sendo a ponte entre a parte externa (atendimento) e a parte interna (desenvolvimento) da empresa através da elaboração de documentação que é compreensível por ambas as partes em diferentes situações.

Desenvolvedor:

O papel do desenvolvedor é transformar num produto de software todas as especificações feitas pelo especificador, garantindo sempre a qualidade e seguindo os padrões de projeto propostos pela empresa.

Testador:

Tem como missão garantir que o produto criado atende aos requisitos definidos pelo especificador, além dos padrões de qualidade e performance, elaborando e

executando roteiros de teste em suas várias vertentes (funcional, unitário, de usabilidade).

Gerente de configuração:

Organiza os ambientes de desenvolvimento, testes, homologação e produção para que o produto passe por todas essas etapas sem que os outros membros da equipe precisem se preocupar com as configurações necessárias para tal.

Na primeira etapa, o estagiário fez o papel de gerente de projetos, juntamente com o papel de analista de requisitos (especificador).

A figura 1 representa graficamente todas os passos pelos quais os projetos podem passar na primeira versão do PDS:

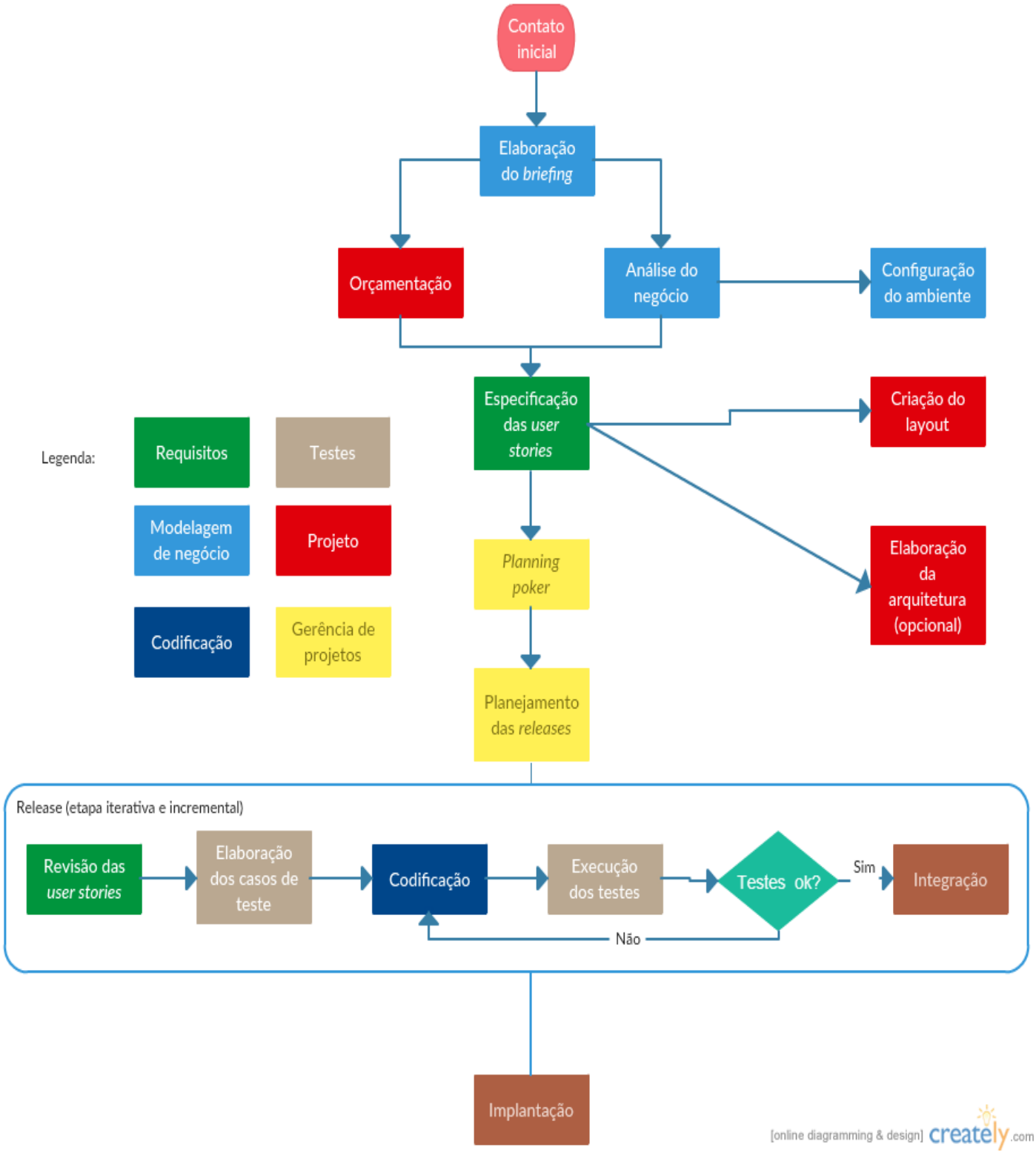

Figura 1 - Diagrama de etapas da primeira versão do PDS – Fonte: Próprio autor

A figura 1 demonstra graficamente a sequência que um projeto deve seguir no PDS, sendo a parte de fora do retângulo a macro etapa de planejamento, (exceto a etapa de implantação) e a parte de dentro do retângulo a macro etapa de execução, que como foi citado, ocorre de forma iterativa até a finalização (implantação) do projeto.

Na macro etapa de planejamento, a ideia é que o problema seja bem compreendido pela equipe, bem como o ambiente onde a solução será implementada para iniciar a modelagem da solução (projeto). Para alcançar esse objetivo, são feitas as seguintes atividades:

- Elaboração do *briefing*
- Orçamentação do projeto
- Análise do negócio
- Especificação das *user stories*
- Criação do *layout*
- Elaboração da arquitetura da solução (opcional)
- Configuração do ambiente
- *Planning poker*
- Planejamento das *releases*

Elaboração do Briefing:

O *briefing* é o pontapé inicial de qualquer projeto, é quando o profissional de atendimento vai até o cliente/usuário para entender as suas necessidades, fazendo a análise do modelo de negócio e dando início à especificação do produto de software a ser construído. O *briefing* também pode ser construído de forma remota, através de um questionário preparado pelo profissional de atendimento e respondido pelo cliente.

- Ordem de execução: Primeira etapa/Início do projeto.
- Documentação recebida: Reunião, documentos utilizados, dados necessários, etc.
- Documentos gerados: Documento de *briefing*.
- Papéis participantes: Atendimento.

Orçamentação:

A orçamentação é a etapa onde o especificador e o gerente de projeto listam os recursos necessários para a construção de um produto, gerando um custo para o desenvolvimento do projeto. Esse custo é apresentado para o profissional de atendimento, que analisa se o custo do projeto é compatível com o capital que o cliente disponibilizou para o projeto. Pode haver uma negociação interna sobre o valor obtido, ou sobre os requisitos a serem implementados.

- Ordem de execução: Após o briefing.
- Documentação recebida: Documento de briefing.
- Documentos gerados: Orçamento.
- Papéis participantes: Atendimento, especificador e gerente de projetos.

Análise do negócio:

A etapa de análise de negócio é quando o especificador procura entender o modelo de negócio do cliente. Ao fim desta etapa, o especificador deve ter plena consciência do problema do cliente, bem como uma proposta de solução, que será passada no documento de requisitos. Essa etapa funciona como um detalhamento técnico do *briefing*, que por ter sua origem na área de publicidade, não apresenta características técnicas de engenharia de software. É importante notar que essa etapa pode se repetir algumas vezes até que os requisitos do produto reflitam as necessidades apontadas pelo cliente para a execução do projeto, sendo finalizada apenas quando o cliente aprova o documento de requisito através de assinatura.

- Ordem de execução: Após a aprovação do orçamento, quando é fechado o contrato.
- Documentação recebida: Reunião, documento de briefing, documentos utilizados, dados necessários, etc.
- Documentos gerados: Documento de requisitos.
- Papéis participantes: Especificador.

Espeficicação das user stories:

Descrição específica de cada requisito, contendo as entidades (dados) a ser tratados, quais os caminhos, possibilidades de visualização, seguindo o padrão *user story*, definido pela metodologia XP.

- Ordem de execução: Após a análise do negócio.
- Documentação recebida: Documento de requisitos na versão final (aprovada).
- Documentos gerados: Especificação dos requisitos (documento de user stories).
- Papéis participantes: Especificador.

Planning poker:

A etapa de planning poker consiste na estimativa de esforço para cada user story dentro do projeto. Cada participante (estimador), recebe um conjunto de cartas que representa o número de horas que serão necessárias para implementar um determinado requisito. O especificador descreve o requisito para todos, tirando quaisquer dúvidas que possam surgir. Quando todos os estimadores tiverem pleno entendimento, cada um escolhe uma carta e todos mostram ao mesmo tempo. Se os valores forem correspondentes, a estimativa é adotada para a implementação, se existirem valores discrepantes, os extremos explicam o motivo para as suas escolhas e uma nova rodada acontece até que exista uma estimativa concreta. O processo se repete até que todos os requisitos estejam devidamente estimados.

- Ordem de execução: Após a especificação dos requisitos.
- Documentação recebida: Documento de user stories.
- Documentos gerados: Estimativa de trabalho para os requisitos.
- Papéis participantes: Especificador e desenvolvedores (estimadores).

Planejamento das *releases*:

Com as especificações dos requisitos e as estimativas de tempo provenientes do planning poker, o gerente de projeto divide as releases levando em consideração a complexidade dos requisitos, as suas respectivas importâncias para o cliente e o prazo de entrega acertado. Quando os requisitos são divididos nas releases, o gerente organiza todas

as tarefas necessárias para a implementação dos requisitos da release atual e as delega para os seus responsáveis.

- Ordem de execução: Após o planning poker.
- Documentação recebida: Documento de user stories.
- Documentos gerados: Tabela de acompanhamento das tarefas, tarefas organizadas na ferramenta de gerenciamento de tarefas.
- Papéis participantes: Gerente de projetos.

Após a macro etapa de planejamento, inicia-se a macro etapa de execução, que como foi citado anteriormente, ocorre de forma iterativa. O objetivo dessa macro etapa é validar a especificação tanto do ponto de vista técnico junto aos desenvolvedores quanto do ponto de vista de negócio, que é quando o cliente analisa o resultado das entregas anteriores e direciona o desenvolvimento das próximas entregas. Após essa validação, a equipe começa o desenvolvimento do produto, afim de integrá-lo com incrementos pré-existentes e disponibilizá-lo para uso do cliente.

A macro etapa de planejamento é composta das seguintes etapas:

- Revisão das *user stories*
- Elaboração dos casos de teste
- Codificação
- Execução dos casos de teste
- Integração

Revisão das *user stories*:

Antes de implementar os requisitos, os desenvolvedores analisam o documento de *user stories* para identificar possíveis erros de especificação, sejam elas na ordem de negócio ou de codificação, por exemplo: Requisitos que não são possíveis de implementar da forma como foram especificados (ou inviáveis), erros na estimativa de tempo ou incompletude na explicação. Se o requisito estiver completo e for viável, parte-se para a implementação, se não, os desenvolvedores se reúnem com o especificador e chegam a um denominador comum, seja apenas a resolução de uma dúvida ou a alteração completa da *user story*.

- Ordem de execução: Início da *release*.
- Documentação recebida: Documento de *user stories*.
- Documentos gerados: Documento de *user stories* revisado, relatório de lições aprendidas.
- Papéis participantes: Desenvolvedores e especificador.

Elaboração dos casos de teste:

Os testadores pegam o documento de *user stories* revisado e com ele criam os casos de teste, baseados nas expectativas do cliente em relação a cada requisito. Os testes podem ser de unidade, de interface ou funcionais.

- Ordem de execução: Após a revisão das user stories.
- Input recebido: Documento de user stories revisado.
- Documentos gerados: Documento de casos de teste.
- Papéis participantes: Testadores.

Codificação

Os desenvolvedores codificam a solução com base nas user stories, para que elas passem nos testes escritos, atendendo assim todas as expectativas em relação ao produto final. Além disso, é nessa etapa que a interface do projeto é implementada de acordo com o layout.

- Ordem de execução: Após a elaboração dos casos de teste.
- Input recebido: Documento de user stories revisado, documento de casos de teste, layout de interface.
- Documentos gerados: Repositório de código, imagens, ícones, fontes, etc.
- Papéis participantes: Desenvolvedores.

Execução dos testes:

Execução dos roteiros de testes baseados na especificação dos requisitos, podendo fazer com que o requisito volte para a etapa de implementação se houver inconsistência entre a especificação e a implementação (*bugs*).

- Ordem de execução: Após a finalização da implementação de cada requisito
- Input recebido: Código-fonte e especificação do requisito.
- Documentos gerados: Relatórios de teste.
- Papéis participantes: Testador.

Integração:

A integração consiste em adicionar os requisitos desenvolvidos ao projeto, para que o cliente possa testar e aprovar os requisitos, garantindo que os requisitos foram especificados de acordo com as reais necessidades dele. A integração sempre deve seguir a seguinte ordem:

1. Executar os testes de integração, testar se os requisitos implementados na nova versão não influenciaram no funcionamento dos requisitos que já estão em funcionamento.
2. Criar uma tag no repositório com o número de versão incrementado no primeiro decimal (ex: v0.1 -> v0.2), apenas versões completas incrementam em número (ex: v0.8 -> v1.0). Se a integração for de um ajuste de versão, utilizar a segunda casa decimal (ex: v0.2 -> 0.2.1)
3. Enviar o código e o banco de dados para o servidor de homologação, procurando manter os dados de teste quando possível.
4. Notificar o cliente por e-mail de que existe uma nova versão do seu software no servidor e indicar quais foram as alterações feitas.

- Ordem de execução: Após a finzalização dos testes de uma release.
- Input recebido: Repositório com o código do projeto.
- Documentos gerados: E-mail de release para o cliente, versão do software funcional para homologação.
- Papéis participantes: Gerente de configuração e gerente de projetos.

Após a definição do PDS, foi elaborada uma apresentação para que os membros da equipe pudessem conhecê-lo e começassem a utilizá-lo. Nessa apresentação foram definidos os papéis de cada membro e as tarefas que cada um devia realizar durante os projetos. Nas semanas seguintes, o PDS começou a rodar dentro da empresa, e a primeira mudança latente a ser notada foi que a definição de papéis e documentos para cada etapa trouxe maturidade para a equipe. Pouco tempo depois do início da utilização do PDS, os membros já executavam as etapas com naturalidade e sabiam exatamente onde procurar a documentação, o que fazer ou quem procurar após a finalização de cada etapa do projeto. Com isso, projetos que estavam "presos" ou atrasados há muito tempo foram entregues aos seus respectivos clientes, pois a equipe conseguia entender melhor quais eram os problemas de cada um e resolvê-los um a um.

Quando esses projetos foram finalizados e novos projetos foram captados, dessa vez, o PDS foi utilizado desde o início. Nesta etapa, os projetos que eram tratados diretamente com cliente final ainda obtiveram sucesso, pois a comunicação entre o cliente e a equipe era feita de forma clara e objetiva. O cliente sabia, de forma superficial, como funcionava o processo, em que etapa o projeto estava, quando ele seria necessário e quais as suas obrigações para contribuir com o sucesso do projeto. Por outro lado, os projetos que vinham de outras agências de publicidade tornaram-se maioria na pauta de desenvolvimento da empresa. Esses projetos, por sua vez, não permitiam contato direto com o cliente, fazendo com que a comunicação fosse menos eficaz e o processo nem sempre funcionasse da forma correta, principalmente na etapa de análise, que era feita externamente por pessoas sem conhecimento técnico algum de engenharia de software. A deficiência na análise dos requisitos acarretava em grandes falhas em todas as etapas do projeto, tendo em vista que ele era baseado no modelo cascata e a análise é a base desse modelo.

As constantes falhas na análise geraram dois fatores primordiais para o insucesso na implantação do PDS na equipe: alta incidência de refatoração em larga escala nos projetos e, por conseguinte, prejuízo de tempo e dinheiro na execução dos projetos, tendo em vista que essas duas variáveis muitas vezes eram controladas por outra empresa, nos casos de projetos terceirizados.

Somados esses acontecimentos ao constante aumento da demanda de desenvolvimento por causa de novos projetos, a equipe começou a ficar sobrecarregada e o cenário de execução dos projetos, que no início favorecia a utilização do PDS, começou a prejudicá-la, pois os projetos começaram a ficar ingerenciáveis, visto que a quantidade era maior do que a equipe conseguia executar, os planejamentos não se concretizavam e o gerente de projetos começou a se tornar ineficiente na equipe, já que não conseguia cumprir o seu papel devido aos fatos já citados.

Nesse momento a equipe se viu em uma situação parecida com aquela que havia na empresa antes da implantação do PDS: vários projetos estavam presos na pauta de desenvolvimento, a comunicação com os clientes e parceiros era confusa, os membros da equipe não sabiam com clareza quais eram suas tarefas e a sequência exata para executá-las, pois mesmo que isso estivesse na definição do PDS, o excesso de projetos simultâneos acabou por prejudicar o fluxo de trabalho da equipe. Foi nesse momento que a equipe constatou que o PDS era muito burocrático para os tipos de projetos que viriam se tornar maioria com o passar do tempo. Para sanar esse problema, a equipe decidiu revisitar o processo e refatorá-lo, dando origem a etapa descentralizada.

3.2 Etapa descentralizada (iterativa e incremental)

Essa etapa foi referenciada como descentralizada pois nela, a figura do gerente de projetos foi desmembrada em dois papéis (*scrum master* e *product owner*) e a equipe passou a adotar um modelo de autogerenciamento, onde todas as decisões dos projetos eram tomadas em conjunto, somando-se os conhecimentos de cada membro afim de chegar nas melhores soluções. Além disso, o PDS revisado nesta etapa possui mais características do modelo iterativo e incremental, remanescente das metodologias ágeis descritas anteriormente.

Como citado anteriomente, o que deu início a essa etapa, foi uma mudança nos tipos de projetos desenvolvidos pela empresa, bem como no relacionamento com clientes e parceiros. Com essas mudanças, foi necessário que o PDS adptasse para atender às novas demandas. Nesse momento, houve uma revisão na literatura afim de buscar novamente as soluções para os novos problemas. Após essa revisão, a equipe se reuniu e tomou duas grandes decisões: mudar o modelo que servia como base para o PDS, além de mudar a organização da equipe, passando de um modelo centralizado na figura do gerente de projetos para um modelo autogerenciável e descentralizado.

Nesse momento, a empresa estava em uma situação que necessitava de mais desenvolvedores para poder finalizar os projetos que estavam em sua pauta de desenvolvimento, mas por outro lado, não possuia recursos suficientes para aumentar a equipe. Juntando isso à falta de experiência e ineficiência da centralização das atividades de gerência em uma só pessoa, foi decidido que o cargo de gerente de projetos seria diluído e o gerente de projetos iria integrar a equipe de desenvolvimento. Entretanto, o PDS havia sido criado baseado na centralização dessas atividades na figura do gerente, que era primordial para o sucesso dos projetos que o seguiam. Foi então que se decidiu que era necessário mudar o PDS baseado na experiência adquirida durante a utilização do mesmo.

As principais justiticativas detectadas pela equipe para a remodelagem do PDS foram as seguintes:

- Havia alto nível de incerteza na análise dos requisitos.
- Era necessário encontrar quaisquer problemas de análise, projeto e codificação o mais rápido possível, pois os clientes muitas vezes demoravam a responder as dúvidas da equipe, fazendo com que os projetos atrasassem.
- Havia a necessidade de intercalar recursos rapidamente entre projetos sem compremeter a sua execução.
- Era preciso simplificar as atividades e a geração dos artefatos, visto que não existia mais um profissional dedicado a cada etapa.
- Era importante aumentar a frequência nas entregas do produto para satisfazer os clientes

Juntando essas necessidades com o resultado da pesquisa bibliográfica, foi constatado que seria melhor seguir o modelo iterativo e incremental, mais precisamente as metodologias XP e Scrum, por causa das seguintes características:

- A etapa de análise e projeto é bem mais leve nas metodologias ágeis, dispensando documentação desnecessária e focando no desenvolvimento da solução, mas sem deixar de pensar na arquitetura da solução por completo.
- Metodologias ágeis incentivam a refatoração como garantia de qualidade.
- As duas metodologias dividem o projeto em *user stories*, cada uma à sua maneira, viabilizando a implementação de uma unidade por vez e facilitando a divisão entre os membros da equipe.
- Como os incrementos são entregues ao cliente com mais frequência, erros na definição do projeto são identificados mais rapidamente e consequentemente consertados também com maior rapidez.

Baseando-se nessas metodologias, a equipe decidiu focar a documentação do PDS na parte de implementação, deixando a parte de análise mais leve, diminuindo o tempo que é passado nessa etapa e chegando mais rapidamente na parte de codificação, onde a maioria dos problemas eram encontrados, agilizando a busca pelas suas soluções.

A figura 2 ilustra o fluxo do PDS após a refatoração:

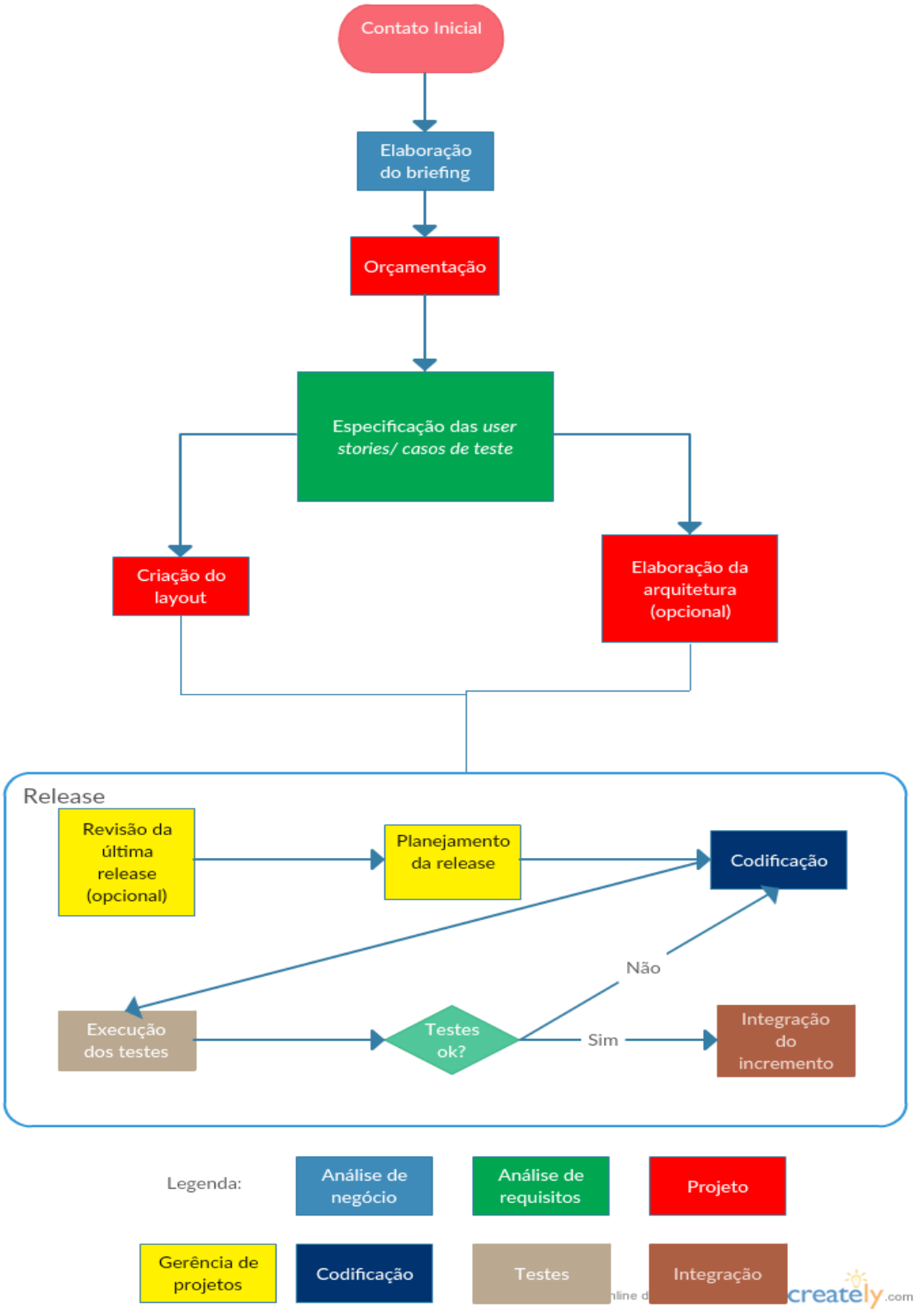

Figura 2: Diagrama de etapas da segunda versão – Fonte: Próprio autor

É possível notar pela figura que a etapa de planejamento não foi extinta, mas sim diminuida. A documentação gerada é bem mais leve, pois é feita apenas para dar uma base para a iniciação do projeto, sendo amadurecida durante o curso das *releases*. Por isso, a primeira *release* funciona como um protótipo com alto índice de aproveitamento, pois é baseado em informações colhidas com um bom nível de precisão.

É importante notar também que a equipe participa de todos as atividades de gerência, analisando a situação de cada projeto e tomando decisões a cada *release*.

Segue abaixo a descrição de cada atividade listada no diagrama:

Elaboração do *briefing*:

A metodologia de elaboração do briefing se manteve a mesma, pois se mostrou efetiva. O profissional de atendimento (agora chamado de *product owner*, segundo a metodologia scrum) faz a análise do negócio do cliente, identificando quais são os seus problemas e os passa para a equipe, seja através de um documento, ou de uma reunião de iniciação do projeto, que consiste em juntar toda a equipe (desenvolvedores, designers e *product owner*) para identificar quais são os problemas do cliente e iniciar o projeto da solução. Dessa reunião pode sair tanto um *briefing* quanto um documento de *user stories* preliminar, e mais importante, o orçamento do projeto.

- Ordem de execução: Primeira etapa/Início do projeto.
- Documentação recebida: Reunião, documentos utilizados, dados necessários, etc.
- Documentos gerados: Documento de *briefing*.
- Papéis participantes: *Product owner* e *scrum master* (quando necessário).

Orçamentação:

Elaborado na reunião inicial em conjunto com toda a equipe, o orçamento tende a ser mais preciso do que anteriormente, quando os desenvolvedores não participavam da sua concepção. Dependendo do projeto, o orçamento é feito em conjunto com o documento

incial de *user stories*, que provê um maior detalhamento dos requisitos de sistema, por isso uma maior precisão.

- Ordem de execução: Após a elaboração do *briefing*.
- Documentação recebida: Documento de *briefing*.
- Documentos gerados: Orçamento.
- Papéis participantes: *Product owner*, *scrum master* e desenvolvedores.

Especificação das *user stories* / casos de teste:

O template do documento de *user stories* se manteve inalterado, entretanto, nessa versão do PDS ela é executada de forma mais rápida, pois a primeira *release* servirá como protótipo para a criação do produto e os documentos criados nessa etapa serão refatorados a cada *release*. A outra diferença é que a primeira especificação dos casos de teste é feita também nessa etapa, enquanto na primeira versão se esperava até o início da primeira *release* para gerá-los. Assim como as *user stories*, os casos de testes são refatorados e incrementados a cada *release*. Participam dessa equipe todos os membros da equipe, onde o *product owner* cria as estórias enquanto os desenvolvedores analisam a viabilidade e dificuldade (estimativa de tempo) de implementação de cada uma, juntamente com os testes.

- Ordem de execução: Após a aprovação do orçamento.
- Documentação recebida: Documento de *briefing*.
- Documentos gerados: Documento de *user stories* e casos de teste.
- Papéis participantes: *Product owner*, *scrum master* e desenvolvedores.

Criação do layout:

A partir da documentação gerada até o momento, e também pelo fato de ter participado da reunião inicial do projeto, o designer já possui os dados necessários para criar o projeto de interface do produto (layout).

- Ordem de execução: Após a definição das *user stories*.

- Documentação recebida: Documento de *user stories*.
- Documentos gerados: Layout (especificação visual da interface).
- Papéis participantes: Designer de interfaces (desenvolvedor).

Elaboração da arquitetura:

Tendo em vista que todos os desenvolvedores participaram da reunião inicial do projeto e elaboraram o documento de *user stories* juntos, muitas vezes a definição da arquitetura da solução fica junto do documento de *user stories*. Entretanto, quando o projeto possui um nível maior de complexidade, e por conseguinte, necessita de um maior nível de detalhamento da arquitetura, a equipe cria documentos separados que definem os componentes da solução, sejam eles diagramas de casos de uso, modelos de bancos de dados, diagramas de classes, entre outros. Desta forma, esta documentação se torna opcional, mas a etapa de elaboração da arquitetura é extremamente importante para o sucesso dos projetos.

- Ordem de execução: Após a definição das *user stories*.
- Documentação recebida: Documento de *user stories* e casos de teste.
- Documentos gerados: Arquitetura do sistema (opcional).
- Papéis participantes: Desenvolvedores.

Após essa curta etapa de planejamento, a equipe começa a desenvolver o projeto da solução. Nessa etapa, a equipe se reune diariamente por aproximadamente 10 minutos (*daily scrum*) para discutir o andamento dos projetos, possíveis impedimentos na execução nas tarefas, possíveis alterações nos requisitos, além de se reunir uma vez ao início de cada nova *release* (*sprint*), reunião esta que possui uma duração um pouco maior (em média uma hora) e tem como objetivo analisar os resultados da *sprint* anterior, se todos os requisitos planejados foram implementados, se surgiram novos requisitos durante a execução da mesma, se será necessário repriorizar ou adaptar os requisitos e finalmente decidem o que será implementado na *release*. Se necessário, os documentos de *user stories* e/ou casos de teste são atualizados para refletir a realidade do projeto, que parte para a etapa de

codificação. Ao final da codificação de cada *user story*, são executados os testes e se, e somente se o código gerado passar em todos os testes (unitários, funcionais e de performance) a *user story* é incrementada à solução existente e liberada para o cliente no final da *sprint*. O projeto termina quando não existem mais requisitos para formar uma *sprint*.

Com a diminuição na carga do processo, a equipe também sofreu alterações nos papéis, que foram reduzidos para apenas três, que seguem a metodologia *scrum*:

Product owner:

O *product owner* é o equivalente ao profissional de atendimento descrito na primeira etapa. Ele também recebeu algumas atribuições de gerência de projetos, sendo a voz do cliente dentro da equipe, também é o único responsável por alterar o *backlog* do produto, priorizando os requisitos dentro das *sprints*, aprovando o desenvolvimento das *user stories* antes que elas sejam enviadas ao cliente e planejando as entregas de acordo com os prazos.

Desenvolvedor:

Para a metodologia scrum, todo membro da equipe que participa na construção do produto é considerado um desenvolvedor, então, seguindo esse modelo, o papel de desenvolvedor integra as atividades de especificação, codificação, testes, elaboração da arquitetura e desenvolvimento do projeto de interface (layout). É importante notar que essas atividades não são executadas apenas por uma pessoa, nem ao mesmo tempo, mas sim divididas entre os membros da equipe de desenvolvimento de acordo com as habilidades de cada membro, fazendo com que cada um trabalhe onde possui mais experiência, gerando uma maior produtividade na equipe.

Scrum master:

O *scrum master* recebeu algumas das atribuições do gerente de projetos, mas é importante salientar que os dois não possuem as mesmas atribuições, sendo o *scrum master* responsável por garantir que a metodologia está sendo bem executada e que nenhum dos membros está tendo nenhum impedimento na execução de suas tarefas, além de controlar a execução dos projetos pela equipe.

Após a implantação da segunda versão do PDS, foi possível notar uma melhora considerável na comunicação da equipe, fazendo com que a identificação e resolução de problemas e dúvidas fosse mais ágil, resultando na otimização do uso de tempo e dos recursos humanos na execução dos projetos. Além disso, o processo de encontrar problemas nos projetos ficou mais ágil, fazendo com que a comunicação com as outras empresas com as quais a BRZ Digital trabalhava em parceria fosse mais clara e precisa. Além das melhoras na comunicação, a equipe obteve uma melhora no ambiente de trabalho, pois como todos os membros sabiam o que os outros estavam fazendo, nenhum deles se sentia sobrecarregado ou injustiçado. Faz-se importante observar que as melhorias foram apenas observadas, mas não constatadas, não tendo sido usadas métricas para garanti-las – o que deverá ser feito em trabalhos futuros.

Na segunda etapa, o estagiário fez o papel de *scrum master*, juntamente com o papel de desenvolvedor front-end e especificador (ambos categorizados como desenvolvedor, segundo o processo), sendo responsável por garantir que o processo estava funcionando da forma correta e com que a equipe entendesse o seu funcionamento.

3.3 Execução de projetos utilizando o PDS

Um dos projetos implementados utilizando a segunda versão do PDS foi o sistema de listas de presentes, do cliente Casa Tudo Premium. Esse sistema tem como objetivo fazer com que os clientes do Casa Tudo possam montar suas próprias listas de presentes para eventos e que os seus respectivos convidados possam acessá-las e comprar os presentes. Esse projeto foi desenvolvido em parceria com a agência digital Ponto D, que foi responsável por desenvolver a parte visual (layout) do sistema, enquanto a BRZ Digital foi responsável por codificar as telas (front-end) e desenvolver a parte lógica do sistema (back-end).

A figura 3 ilustra os casos de uso do sistema.

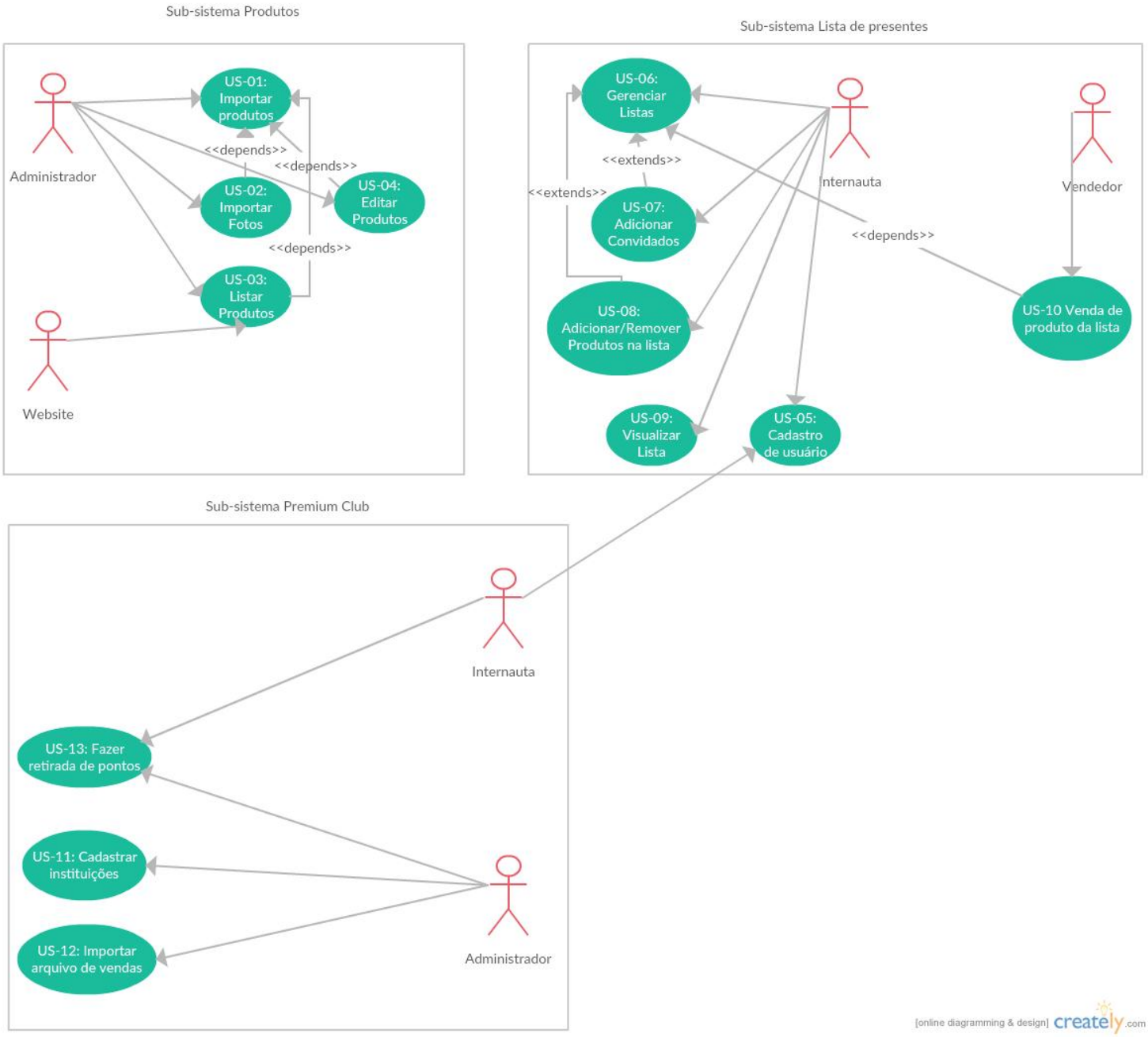

Figura 3: Diagramas de casos de uso do sistema de lista de presentes - Fonte: Documentação BRZ Digital

A gerência desse se deu em conjunto com a gerente de projetos da Ponto D, que acompanhou o desenvolvimento através da ferramenta utilizada para o gerenciamento de tarefas, o Trello[1]. A divisão das tarefas foi relacionada diretamente às *user stories* definidas para o projeto, tendo em vista que o documento central do PDS é o documento de *user*

[1] http://www.trello.com

stories. Após a definição das *user stories* (vide apêndice X), a equipe definiu a prioridade de implementação junto ao cliente final (Casa Tudo Premium) e organizou as tarefas no Trello, como pode ser visto na figura 4:

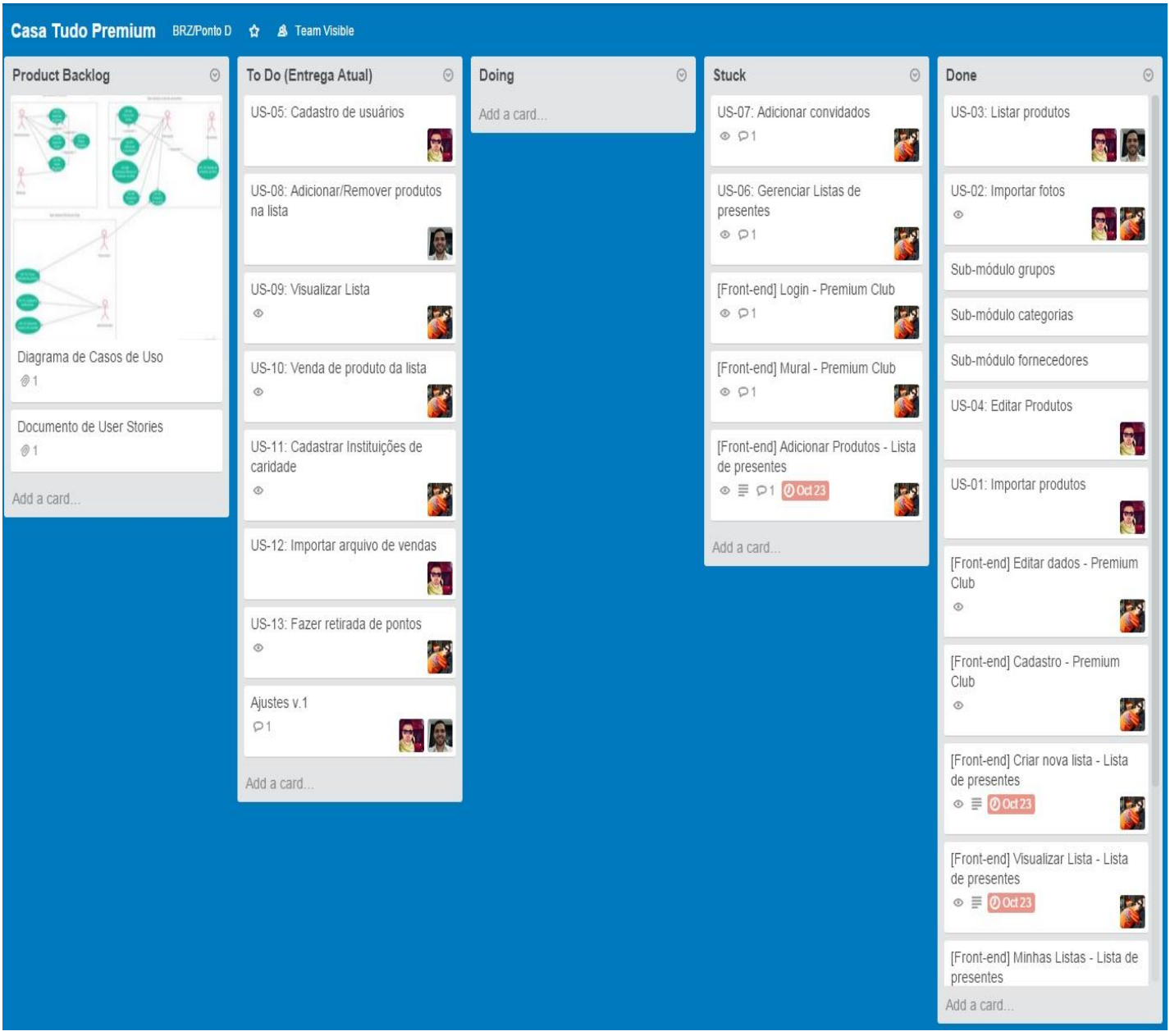

Figura 4: Painel de gerenciamento de tarefas na última release - Fonte: Documentação BRZ Digital

4. Considerações Finais

A segunda versão do PDS ainda está em fase de implantação, mas pode-se notar a melhora na produtividade da equipe, pois a escolha do modelo iterativo e incremental aliado às metodologias ágeis atende o tipo de projeto executado na empresa. Todo o trabalho realizado beneficiou tanto o estagiário, que pôde aplicar os conhecimentos adquiridos na academia dentro de um ambiente de trabalho real, quanto a empresa, que saiu de um nível de maturidade onde não existia nenhuma documentação sobre o seu PDS para outro nível onde todos os membros de sua equipe entendem através de uma documentação formalizada como proceder em cada etapa da execução dos seus respectivos projetos, melhorando a produtividade da equipe e a qualidade dos produtos desenvolvidos.

A experiência adquirida nesse processo foi bastante satisfatória, pois o estagiário pôde presenciar a realidade de uma empresa de desenvolvimento de software e além disso, também mudar essa realidade através dos conhecimentos de engenharia de software adquiridos na academia.

A grande dificuldade na execução do processo residiu em fazer o papel de gerente de projetos, pois o estagiário não possuia experiência alguma antes de desempenhá-lo, o que pode ter prejudicado a análise dos resultados, pois a equipe pode ter subestimado o valor do gerente de projeto, mas a causa para tal falha foi a falta de experiência na função, e não a existência da mesma. Além disso, foi interessante aplicar conhecimento de engenharia de software em um ambiente diferenciado das empresas de desenvolvimento de software propriamente ditas, pois como foi citado durante o relatório, o escopo desse trabalho é de agências de marketing digital.

Os objetivos traçados para esse trabalho foram satisfatoriamente alcançados, pois além de desenvolver um PDS com base nas melhores práticas da indústria, o estagiário pôde aplicá-lo, entendendo quais eram os defeitos na sua concepção e refatorar o processo, deixando a equipe mais motivada e satisfeita com o modo de trabalho.

Para os trabalhos futuros está prevista a próxima etapa desse processo, que é criar uma cultura de análise e refatoração constante do PDS para que a equipe possa sempre melhorar, entendendo quais são seus pontos fortes e fracos, potencializando os fortes e

minimizando os fracos. Essa cultura enriquece não só os profissionais envolvidos na equipe como gera conhecimento organizacional para a empresa, que é um bem intangível de valor inestimável.

Referências

BECK, K. et al. ***Manifesto para Desenvolvimento Ágil de Software*****.** Disponível em <http://www.agilemanifesto.org/iso/ptbr/>. Acesso: em 23 nov. 2015.

BUSINESS ANALYSIS BODY OF KNOWLEDGE. ***Um guia para o corpo de conhecimento de análise de negócios. Versão 2.0*****.** 2011.

PRESSMAN, R.S. ***Software Engeneering. A practioner's approach. 5.ed.*** 2001.

SCHWABER, K; SUTHERLAND, J. ***Guia do Scrum*****.** 2013.

SOMMERVILLE, I. ***Engenharia de Sofware. 9.ed.*** 2011.

Apêndice A – Documento de *User Stories*: Casa Tudo Premium

Título	US-01: Importação de produtos
Descrição	O administrador do sistema irá importar um arquivo em formato CSV (*comma separated values*), com os seguintes campos: • Fornecedor • Grupo • Codigo • Descrição (Nome) • Preço
Entidades	Produto: • Código (int) • Fornecedor (varchar2(255)) • categoria_id (int) (referência CategoriaProduto) • grupo_id (int) (referência GrupoProduto) • Galeria (GaleriaProduto) • Nome (varchar2(255)) • Nome web varchar2(255)) • Descrição web (text) • Preço (float) • Ativo? (boolean) • Site? (boolean) CategoriaProduto: • id (int) • Nome (varchar2(255)) GrupoProduto: • id(int) • Nome (varchar2(255))
Testes de aceitação	• Manter a unicidade do atributo código na tabela produtos • Deve ser possível importar uma quantidade alta de linhas no arquivo CSV
Observações	• A descrição no arquivo corresponde campo “Nome” do produto. • O campo código é unico (unique), então se o arquivo contiver um código que já exista no sistema, o sistema deve atualizar aquele produto ao invés de adicionar um novo.

Título	US-02: Importar fotos
Descrição	O administrador do sistema irá importar um conjunto de fotos, seguindo os seguintes padrões de nomenclatura • <codigo>-c.<extensão> • <codigo>-<ordem>.<extensão> Onde <código> é o código do produto onde a imagem deve ser relacionada, <ordem> é a ordem que ele deve ser adicionado na galeria e <extensão> é a extensão do arquivo, se o arquivo vier com "-c" antes da extensão, significa que essa foto é a foto de capa da galeria.
Entidades	GaleriaProduto: • id (int) • capa (Image) • cod_produto (int) (referência de Produto) • imagens (Image 1:N) ImagemProduto • id (int) • url (varchar2(255) • galeria_id (int) (referência GaleriaProduto)
Testes de aceitação	• A importação das imagens deve refletir o padrão de nomenclatura. Se as imagens não refletirem o padrão, o sistema deve retornar um erro.

Título	US-03: Listar produtos
Descrição	O sistema deve permitir ao usuário visualizar todos os produtos, incluindo os que possuem o estado "ativo" como falso, além de possuir uma interface de programação de aplicações (API) para acesso por parte do website, esta interface, por sua vez, deve apenas listar os produtos classificados como ativos para essa interface, utilizando o campo "site" como parâmetro.
Testes de aceitação	• A API deve ser acessível de domínios fora do sistema.
Observações	A API de listagem de produtos, deve fornecer também as imagens nas respectivas galerias para exibição no website.

Título	US-04: Editar produtos
Descrição	O administrador do sistema poderá editar qualquer produto no sistema manualmente, inclusive alterar os status “ativo” e “site” para exibição, bem como visualizar a galeria correspondente e editar as fotos.
Testes de aceitação	• O administrador deve poder excluir ou adicicionar novas fotos na galeria a qualquer momento.
Observações	• Não será possível editar o código do produto, pois o mesmo se comporta como uma chave primária no sistema

Título	US-05: Cadastro de usuários
Descrição	O usuário do sistema poderá criar o seu próprio cadastro, informando os dados necessários
Entidades	Usuário: • Tipo de conta (varchar2(255)) • CPF/CNPJ (int/varchar2(255)) • Data de Nascimento/Fundação (date) • Nome completo (varchar2(255)) • E-mail (varchar2(255)) • Senha (varchar2(255)) • Número do CREA (varchar2(255))
Testes de aceitação	• O CPF/CNPJ deve ser único no sistema • O e-mail deve ser único no sistema

Título	US-06: Gerenciar listas de presentes
Descrição	O usuário do sistema poderá criar suas listas de presentes e convidar os seus amigos através do envio de um e-mail. As listas possuem categorias para facilitar na busca. O usuário poderá a qualquer momento adicionar ou remover produtos, além de acompanhar o status de compra e entrega dos presentes.
Entidades	ListaDePresentes: • Nome (varchar2(255)) • Categoria (varchar2(255)) • Nome do responsável 1 (varchar2(255))

	• Nome do responsável 2(varchar2(255)) • Data do evento (date) • Endereço de entrega (varchar2(255))
Testes de aceitação	
Observações	• As listas devem ser públicas para que os convidados possam ver através de um link que será enviado por e-mail.

Título	US-07: Adicionar Convidados
Descrição	O usuário poderá convidar uma ou mais pessoas através do seu endereço de e-mail. Ao fazer o convite, o convidado receberá uma mensagem de e-mail informando que foi convidado e o link para a lista de presentes.
Entidades	Convidado: • Nome (varchar2(255)) • E-mail (varchar2(255)) • lista_id (int) (referência ListaDePresentes)
Testes de aceitação	• O e-mail enviado não deve cair na caixa de spam

Título	US-08: Adicionar/Remover Produtos na lista
Descrição	O usuário do sistema poderá adicionar ou remover produtos de sua lista a qualquer momento, exceto se algum convidado já tiver comprado esse item, podendo somente diminuir a quantidade pedida na lista. essa atualização deve refletir na visualização que os convidados têm da lista, para que não haja inconsistência entre o pedido e as compras.
Entidades	ItemLista: • produto_id (int) (referência Produto) • quantidade_total (int) • quantidade_disponivel (int)
Observações	• A adição/remoção de um item na lista será feita via AJAX.

Título	US-09: Visualizar lista

Descrição	O usuário do sistema poderá visualizar o status da sua lista, ver quais produtos já foram comprados e as respectivas quantidades, além de quem comprou
Observações	• É necessário que o vendedor possa imprimir a lista para que os convidados que visitem a loja tenham acesso

Título	US-10: Venda de produto da lista
Descrição	O vendedor irá pesquisar qual a lista que o convidado deseja comprar, podendo utilizar como parâmetros os nomes dos responsáveis, a data do evento e a categoria escolher. Escolhida a lista, o convidado poderá escolher quais produtos irá comprar e as respectivas quantidades, bem como o modo de entrega.
Entidades	Venda: • lista_id (int) (referência ListaProdutos) • Data da Venda (date) ItemVenda: • produto_id (int) (referência Produto) • quantidade_total (int) • venda_id (int) (referência Venda) • Forma de entrega (varchar2(255))
Observações	• Cada vez que uma venda é feita, a quantidade dos produtos contidos na mesma é deduzida do respectivo item na lista.

Título	US-11: Cadastrar instituições
Descrição	O administrador do sistema poderá cadastrar instituições de caridade para que elas sejam beneficiadas com o premium club.
Entidades	InstituicaoDeCaridade: • CNPJ (varchar2(255)) • Nome (varchar2(255))

Título	US-12: Importação do arquivo de vendas (premium club)
Descrição	O administrador do sistema importará um arquivo CSV (*comma separated values*) onde as linhas representam vendas, a partir dos valores dessas vendas, serão creditados os pontos no premium club do consumidor, da sua

	instituição e do especificador relativo a venda.
Entidades	VendaPremium: • CPF Consumidor (varchar2(255)) • id_instituicao (int) (referência InstituicaoDeCaridade) • CPF Especificador (varchar2(255)) • Total da venda (float) • Data da venda (date)
Observações	• A cada R$ 100,00 em compras (valores "quebrados" não serão contabilizados), o cliente acumulará 5 pontos, caso seja por indicação, o especificador mais 5 pontos e 1 ponto a instituição que deseja apoiar, selecionada no momento do cadastro. • Os pontos são válidos por 2 anos após a execução da venda

Título	US-13: Retirada de pontos
Descrição	O consumidor poderá fazer a retirada dos pontos, cada ponto equivale a R$1,00 em produtos casa tudo premium. Cada troca de pontos deverá ser registrada em log.
Entidades	TrocaPremium: • id_usuario (int) (referência Usuário) • data-hora (timestamp) • id_admin (int) (referência Usuário) • pontos retirados (int)
Observações	