DESTAQUES ACADÊMICOS

UNIVATES | CENTRO DE GESTÃO ORGANIZACIONAL

Revista Destaques Acadêmicos, Lajeado, v. 10, n. 1, 2018. ISSN 2176-3070
DOI: http://dx.doi.org/10.22410/issn.2176-3070.v10i1a2018.1775
http://www.univates.br/revistas

IMPLANTAÇÃO DE NOVO PROCESSO DE TRABALHO EM UMA FÁBRICA DE SOFTWARE BASEADO NOS MODELOS ÁGEIS DE DESENVOLVIMENTO

Evandro Franzen[1], Douglas Lutz[2]

Resumo: A qualidade e o resultado financeiro dos projetos de *software* estão fortemente atrelados aos processos de desenvolvimento e ambos são fatores decisivos para o sucesso ou fracasso das organizações. As metodologias ágeis se constituem em técnicas utilizadas pelas empresas de *software* que estão preocupadas com resultados e que buscam melhoria constante dos seus processos de desenvolvimento. Considerando a realidade de uma empresa que atua no mercado de *software* personalizado para clientes e desenvolve um produto próprio, este trabalho teve como principal objetivo a elaboração e validação de um processo baseado nas metodologias ágeis, mantendo pontos positivos do processo atual. A investigação de caráter experimental avaliou de maneira quantitativa e qualitativa o novo processo e os resultados indicam que existe uma percepção de melhoria na qualidade do produto desenvolvido, além da redução dos atrasos nos prazos de entrega em função da nova forma de trabalho.

Palavras-chave: Processo de *software*, Análise e melhoria de processos, Metodologias ágeis, SCRUM.

1 INTRODUÇÃO

O desenvolvimento de *software* é composto por várias fases distintas, que juntas compõem o ciclo de vida de um projeto. Dentre elas pode-se citar: gerenciamento, análise, implementação, testes, homologação e entrega. Estas fases de trabalho envolvem recursos distintos e finitos ao longo do projeto exigindo um acompanhamento próximo para garantir que o andamento siga dentro do esperado, não ocorrendo desvios. Projetos de *software* são conhecidos por seus desvios de custo e prazo, acarretando prejuízos indesejados para o desenvolvedor e também para o cliente.

1 Universidade do Vale do Taquari – Univates.

2 Universidade do Vale do Taquari – Univates.

Para Kerzner (2006), o equilíbrio do gerenciamento de projeto tem como as principais variáveis de risco o custo, o tempo e o escopo. Segundo o autor, o desafio está em planejar e gerenciar as três restrições do processo de desenvolvimento de software, além de uma quarta restrição: o relacionamento com o cliente. Assim, destacam-se a necessidade de um planejamento eficaz e um processo eficiente que consiga realizar as etapas de desenvolvimento de um projeto de forma organizada e rentável.

Os principais modelos de desenvolvimento de *software* conhecidos podem ser categorizados em dois grandes grupos: tradicionais e ágeis. O modelo tradicional é conceitualmente mais antigo e caracteriza-se por ter um processo sequencial, que contempla um esforço maior de análise no início do projeto, seguido por uma fase de desenvolvimento pesada e após, um período de testes e homologação, sendo que o cliente recebe uma entrega de *software* concentrada ao final dos trabalhos. Pesquisas sobre a utilização de metodologias ágeis, combinadas com estratégias baseadas em abordagens tradicionais tem sido tema de diversos trabalhos (FRANZEN, BARDEN, 2103; PEREIRA, 2005).

Esse fluxo de trabalho obriga a empresa a ser bastante assertiva no planejamento inicial, visto que a sequência do projeto dependerá dele. O custo, tempo e escopo devem ser mensurados já na primeira etapa do processo, elevando muito o risco de desvios no planejamento à medida que o projeto evolui para as próximas etapas.

As metodologias ágeis surgiram para oferecer dinamicidade aos projetos, propondo novas técnicas de execução das etapas de trabalho, seguindo fluxos alternativos e flexíveis. Basicamente esse modelo de trabalho propõem que as fases sejam executadas conforme a sua necessidade, ampliando a capacidade de adaptabilidade do processo às características do projeto, ou seja, a análise, o desenvolvimento e os testes podem andar em paralelo, enquanto o cliente recebe pequenas entregas parciais. Esse modelo exige maior participação do cliente no projeto, garantindo um nível de assertividade maior tanto para a empresa quanto para o cliente.

Este trabalho realizou uma análise de quatro metodologias de desenvolvimento de software utilizadas atualmente, duas delas de formato tradicional (modelo cascata e RUP) e duas consideradas ágeis, a XP (eXtreme Programming) e SCRUM. Realizou-se também uma análise da forma de trabalho de uma empresa de desenvolvimento de sistemas, que não tem modelo de trabalho definido, mas tende para um fluxo mais tradicional. Com base no conceito das metodologias ágeis, o estudo identificou gargalos e pontos de melhoria no processo desta empresa, propôs um novo fluxo de trabalho adaptado à realidade da empresa e validou as novas práticas com projetos piloto.

A empresa foco deste estudo possui duas frentes de trabalho, uma como fábrica de software e outra com o seu único produto, um software para gestão da equipe comercial de empresas que possuem venda externa ou

representantes. Por este motivo o perfil dos clientes não é único, visto que são vários clientes que utilizam o mesmo sistema de forma simultânea. Assim, uma das características da empresa é que os projetos não têm a mesma forma, organização e estrutura de trabalho, o que impacta em alguns problemas como alternância de pessoal entre os projetos e diferentes modelos de gerenciamento, acarretando em controles e formas de acompanhamentos variadas para cada processo, além de relatórios e resultados não padronizados.

Alguns processos de trabalho já foram propostos e executados pela empresa ao longo dos anos, sempre objetivando a correção de defeitos ou ineficiências, mas ainda assim existem pontos não satisfatórios, que podem ser melhorados. É possível citar, por exemplo, a estimativa de custo do desenvolvimento de um projeto que não contempla todas as fases da sua execução, gerando assim uma margem considerável de erros, resultando em previsões de prazos que não são cumpridos. Tais problemas vêm sendo corrigidos ao longo do tempo, mas a empresa ainda não conseguiu encontrar um processo padrão que seja aderente a dinamicidade dos projetos que desenvolve.

Considerando os problemas e o cenário exposto, este estudo teve como objetivo implantar um novo processo de desenvolvimento de software, adequado às necessidades da empresa. Com base em projetos recentes e similares, foram executados pilotos do novo processo mapeado, no intuído de mensurar de forma quantitativa o percentual de escopo entregue ao cliente ao final de uma versão em comparação aos dias de atraso no cronograma previsto. Além disso, identificou-se de forma qualitativa e quantitativa a qualidade dos entregáveis apresentados aos clientes.

2 REFERENCIAL TEÓRICO

A Crise de Software da década de 70 deu origem ao processo de desenvolvimento de sistemas. Neste período não havia planejamento, estrutura e organização para seu desenvolvimento, aumentando com isso prazos e gerando custos desnecessários, por não haver processos bem estabelecidos. Surgiu assim a necessidade de planejar e padronizar as metodologias do desenvolvimento de softwares, para atender as necessidades de informação e padronização de processos (UTIDA, 2012).

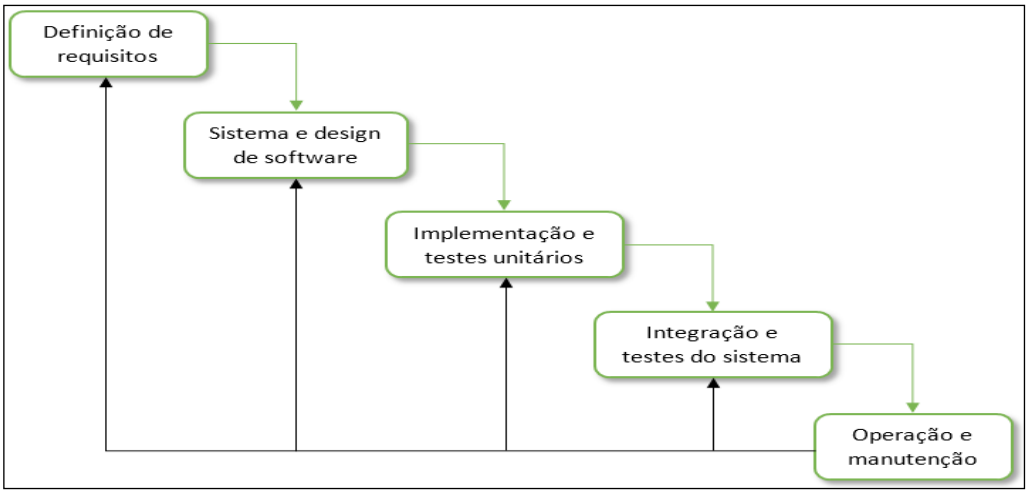
Para Royce (1970) existem duas etapas da construção de software que realmente são necessárias e agregam valor visível ao produto final, que são a análise e o desenvolvimento. Para projetos menores e com um nível de complexidade baixo, utilizar apenas essas duas fases de projeto pode ser interessante, visto que todos os desenvolvimentos precisam destas etapas básicas. Contudo, projetos maiores construídos com base apenas nessas duas fases, estão frequentemente condenados ao fracasso, uma vez que várias outras

etapas são necessárias para a obtenção do sucesso, mesmo que sua importância seja indireta.

2.1 Método Cascata

Segundo Pressman (2011) esse modelo foi o primeiro processo de software a ser publicado e aceito como metodologia de trabalho padronizado e desde então tem sido amplamente utilizado pelas empresas desenvolvedoras, que adotaram a metodologia e suas características de trabalho. Como pode-se observar na Figura 1, neste modelo existe uma sequência de passos a serem seguidos e cada etapa deve ser concluída para que a próxima possa ser iniciada. Sommerville (2011) reforça que, utilizando esta metodologia de trabalho, o software é inicialmente planejado e agendado, para só depois entrar no processo de desenvolvimento.

Figura 1 - Modelo cascata apresentado por Sommerville (2011).



Fonte: Adaptado pelo autor com base em Sommerville, 2011.

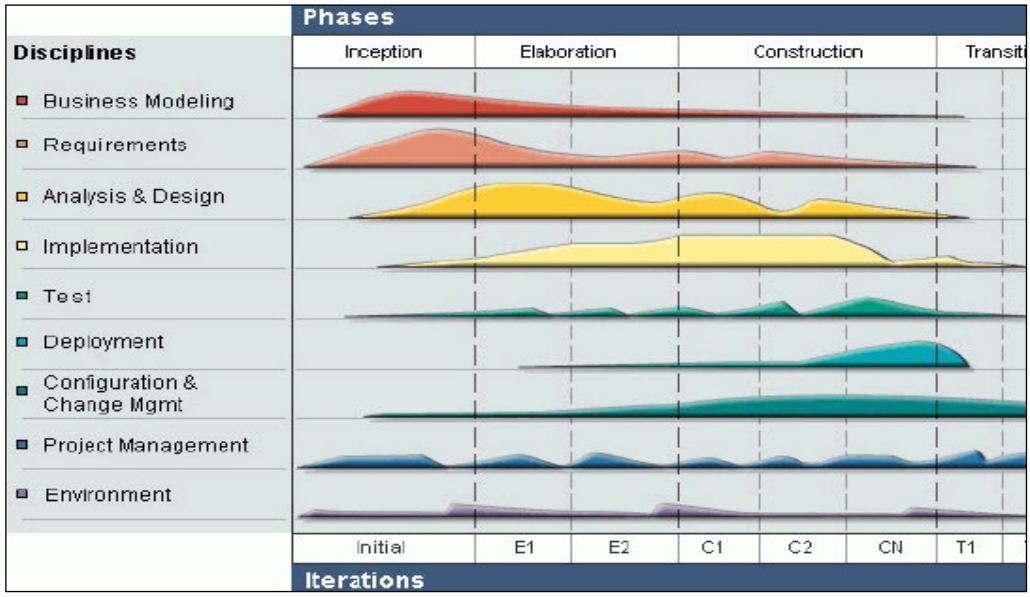
O fato de uma etapa só iniciar após a conclusão de outra, contribui para a detecção dos erros somente na etapa seguinte, por exemplo: problemas de levantamento de requisitos são identificados na fase de design, problemas de projeto na fase de codificação e problemas de integração na fase de operação. Tal constatação reforça o ponto de que, apesar de ser um modelo bastante conceituado e amplamente utilizado mesmo nos dias atuais, o processo de software denominado cascata apresenta pontos de melhoria na estrutura e até mesmo na organização de suas etapas.

2.1 Processo Unificado Racional – RUP

Conforme Barbosa (2011), o Processo Unificado Racional (RUP) nasceu da mistura das melhores práticas de desenvolvimento de software, com o intuito de melhorar e dar respostas aos problemas decorrentes a esta atividade. Este processo faz amplo uso da UML (Unified Modeling Language), que em outros modelos servia somente como ferramenta de apoio, e não como um molde a ser seguido no desenvolvimento.

Existem 4 fases que descrevem o modelo de processo de software (FIGURA 2). Na concepção, ocorre obtenção dos requisitos junto ao cliente, são definidas as regras da negociação e como será o sistema. A segunda fase é a de elaboração, onde ocorre o planejamento e modelagem do software. Na fase de construção do sistema o principal objetivo é o desenvolvimento. É nesta etapa que a otimização de custos se torna ainda mais importante, visto que a qualidade e assertividade do desenvolvimento estão diretamente relacionados ao sucesso do projeto. Por último, a fase de transição é o momento no qual a construção do software é finalizada, ocorrendo a implantação no cliente.

Figura 2 – Fases do processo RUP



Fonte: The IBM Rational Unified Process (2007).

2.2 eXtremeProgramming - XP

As metodologias de desenvolvimento ágil são propostas que surgiram a partir da evolução da metodologia tradicional de desenvolvimento e outros modelos de processo utilizados para a construção de sistemas de software até o

momento. Para Soares (2005), a busca pelas metodologias de desenvolvimento ágil é crescente, mas o número de projetos grandes e críticos que obtiveram sucesso devido ao seu uso ainda é pequeno.

Conforme Soares (2005), a metodologia Extreme Programming (XP), é diferenciada das demais por conter feedback constante, abordagem incremental e a comunicação entre as pessoas é encorajada. Esta metodologia ágil é para equipes que desenvolvem software onde seus requisitos são vagos e podem ser modificados rapidamente. Algumas regras deste método podem causar conflitos e outras não fazem sentido se aplicadas isoladamente, sendo assim, revoluciona o desenvolvimento de software. A XP enfatiza o desenvolvimento rápido do projeto e visa garantir a satisfação do cliente, além de favorecer o cumprimento das estimativas. A forma de comunicação é o fator chave nesta metodologia, onde a comunicação, a simplicidade, feedback e coragem, são os quatro valores aplicados e seguidos da metodologia ágil.

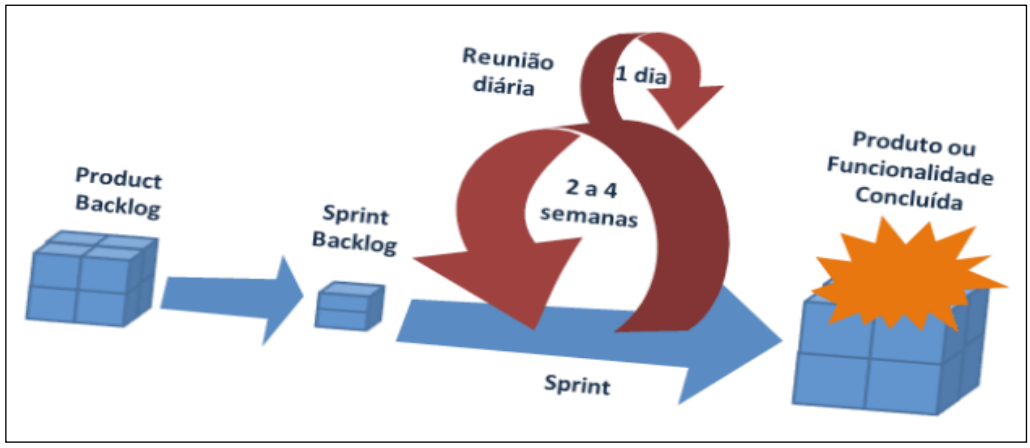
Conforme Wells (2009), gerentes, desenvolvedores e clientes, são todos, parte de uma mesma equipe. Utiliza-se o desenvolvimento orientado por teste (TDD) e a refatoração para ajudar a descobrir o design mais eficaz. Ao longo do projeto os feedbacks são constantes, tanto para o cliente quanto para a equipe. Os desenvolvedores recebem retorno constante trabalhando em pares e testando o código conforme é escrito; os gerentes recebem o reporte do progresso e dos obstáculos na reunião diária. Os clientes recebem feedback sobre o progresso com os resultados dos testes de aceitação e demonstrações a cada iteração.

2.3 SCRUM

Conforme Schwaber e Beedle (2002) a metodologia SCRUM apresenta características similares ao processo XP, onde os requisitos também são pouco conhecidos e existem iterações frequentes, proporcionando assim uma melhor visibilidade do andamento do projeto, tanto para o cliente quanto para a equipe. Para Soares (2006) o ciclo de vida da metodologia Scrum é baseado em três fases principais.

O pré-planejamento (fase 1), é a etapa na qual são descritos todos os requisitos em um documento chamado backlog. O planejamento (fase 2) define as prioridades para o desenvolvimento, as estimativas, e definição das funções de cada membro da equipe e ferramentas que serão utilizadas. Podem haver alterações nos requisitos descritos no backlog, quando observados possíveis riscos e melhorias no desenvolvimento. Na etapa de desenvolvimento (fase 3), o controle é feito continuamente, o que aumenta a flexibilidade de acompanhar todas as etapas de desenvolvimento. O software deve ser desenvolvido em ciclos (*sprints*) onde novas funcionalidades são adicionadas (FIGURA 3).

Figura 3 - Fases do método Scrum



Fonte: Vieira, 2014.

Um clico de desenvolvimento baseado no Scrum inicia pela priorização do Product Backlog, que é feita pelo Product Owner (dono do produto) em conjunto com a sua equipe. Dessa forma as prioridades estarão sempre no topo da pilha de funcionalidades, enquanto itens de menor relevância ficarão no fundo. O Scrum utiliza calendários ou os ciclos que determinam o tempo de cada *sprint*, onde parte do projeto será finalizado e entregue ao cliente para validação. Finalizando uma etapa, segue-se para outra e assim por diante, sempre mantendo a mesma duração entre cada iteração com o cliente. Normalmente o prazo sugerido é de duas a quatro semanas de desenvolvimento para cada entrega.

3 PROCEDIMENTOS METODOLÓGICOS

Quanto à natureza, este trabalho pode ser considerado como exploratório. Conforme Gerhardt e Silveira (2009), este tipo de pesquisa tem como objetivo propor familiaridade com o assunto, tornando-o mais explicativo utilizando levantamentos bibliográficos, entrevistas, análises ou estudos de caso. Para explicar e expor as ideias sobre metodologias de elaboração de software o estudo comparou métodos tradicionais e ágeis, buscando um entendimento sobre as melhores práticas dos conceitos e propor uma metodologia mais eficiente na produção de sistemas na empresa em questão. Utilizou-se uma investigação de caráter experimental para aprofundar o conhecimento sobre construção de softwares, com proposição de uma nova forma de trabalho para a empresa.

A abordagem da pesquisa no desenvolvimento do trabalho pode ser classificada como qualitativa e quantitativa, podendo ser considerada uma pesquisa de método misto.

Conforme Gerhartd e Silveira (2009), a pesquisa qualitativa tem como preocupação a compreensão do cenário e o aprofundamento do assunto, baseando-se em informações, uma vez que este tipo de pesquisa reforça aspectos que não podem ser quantificados, centrando-se em explicações. Já a pesquisa quantitativa tem como objetivo o pensamento lógico e atributos mensuráveis. Com a pesquisa quantitativa é possível ter um maior enfoque na interpretação do estudo, e tem-se mais respostas sobre o contexto estudado, pois os resultados podem ser mensurados e comparados à métricas anteriores (quando disponíveis).

Para esta pesquisa utilizou-se como fonte de dados os projetos de software desenvolvidos pela empresa antes e depois da implantação do novo processo. Para o procedimento de coleta de informações foi utilizado o modo de estudo de caso, que objetiva compreender pontos de vista dos participantes e expectativas do pessoal envolvido, colocando os pontos de vista do investigador. Conforme Yin (2001), um projeto de pesquisa é constituído da lógica que une os dados a serem coletados com as conclusões que serão tiradas deste estudo. Um projeto de pesquisa é um plano de ação onde deve ser definido um conjunto inicial de questões a serem respondidas e por fim um conjunto de conclusões sobre as questões iniciais.

A forma antiga de trabalho da empresa era resultado de várias melhorias feitas ao longo do tempo e reflete uma caminhada de muito esforço, contudo, a dinamicidade do cenário no qual a empresa está inserida não permite comodismo. Para uma organização que quer se manter no mercado oferecendo produtos e serviços de alta qualidade é imprescindível a evolução contínua dos seus processos, eliminando perdas e custos desnecessários, com o objetivo de aumentar a sua competitividade.

4 ANÁLISE DO CENÁRIO ORGANIZACIONAL

A empresa escolhida para este estudo foi a Retta Tecnologia da Informação, uma fábrica de software situada na cidade de Lajeado/RS, que foi fundada em outubro de 2005 e iniciou suas atividades com o objetivo de desenvolver tecnologias informatizadas para o agronegócio. No ano de 2011 a empresa lança um serviço de outsourcing, posicionando-se como Fábrica de Software. Neste mesmo ano é lançado o módulo emissor de NF-e para o software RETTA Comercial, ampliando assim o potencial de clientes para o produto. Ainda em 2011 iniciam as pesquisas e desenvolvimento de aplicações para dispositivos móveis, inicialmente para a plataforma Android.

Em 2012 a empresa lança mais um produto, um aplicativo de força de vendas para que as empresas com equipes externas de vendedores possam automatizar o trabalho da emissão de pedidos. Neste mesmo ano os produtos RETTA Comercial e Gestor RETTA Suínos foram descontinuados para que a

empresa pudesse manter o foco no produto Demander e nos desenvolvimentos de produtos específicos para clientes.

Atualmente a empresa está situada no Parque Tecnológico da UNIVATES, o TECNOVATES e conta com 15 colaboradores, atendendo a clientes de todo Brasil. São 13 anos de experiência no desenvolvimento de sistemas e produtos personalizados, que resultaram em dois segmentos distintos de projetos que buscam satisfazer as necessidades dos clientes: fábrica de software e produto de gestão comercial.

A fábrica de software tem como foco o desenvolvimento de contratações pontuais, que acabam no momento da entrega do software, sob o qual a empresa não mantém nenhum direito. Por se tratarem de produtos de propriedade dos clientes este trabalho não apresenta nenhum exemplo de implementação, limitando-se apenas a dizer que atende diversos segmentos como: telecomunicações, contábil, jurídico, agropecuário, construção, entre outros. Tais projetos chegam ao setor comercial da Retta através de uma necessidade específica do cliente e são atendidos por um determinado fluxo de trabalho, tanto na fase de elaboração da proposta quanto na fase de concepção de produto. O fluxo utilizado atualmente no desenvolvimento de projetos de fábrica é a evolução do processo homologado da empresa, que utiliza algumas técnicas ágeis, mas não possui nenhuma documentação.

Outro segmento de negócio, diz respeito ao produto DEMANDER, que é uma plataforma completa para gestão de vendas em indústrias e distribuidoras, automatizando o trabalho das equipes comerciais internas e externas. O desenvolvimento de todos os ambientes da plataforma de vendas Demander é feito e mantido pela equipe da Retta. Os desenvolvedores são divididos por segmentos (produtos e fábrica de software), mas eventualmente é preciso que alguém de um time ajude o outro. Isso acontece por conta de altas demandas pontuais e falta de organização da empresa. Esta implantação teve foco no processo de desenvolvimento do Demander, deixando o processo de fábrica para outro momento.

A Retta iniciou no ano de 2014, um trabalho de padronização de seus processos, obtendo a certificação nível G do MR-MPS-SW (MPS.BR). Essa certificação é uma melhoria da qualidade do processo de desenvolvimento, garantindo que o desenvolvimento é orientado a projetos e requisitos, fatores que antes não eram observados com tanta importância. Essa certificação levou a empresa a desenhar um novo processo de trabalho na época, utilizado até hoje em alguns dos projetos, mas como o cenário seguiu evoluindo, esta forma de trabalho ficou defasada. O processo, conforme descrito na documentação da interna da empresa (Retta, 2014), tornou-se muito burocrático e pouco eficiente para atender as expectativas dos clientes e da alta direção, principalmente para os projetos de versões do produto, que são liberados frequentemente. Assim, o presente trabalho buscou mapear um novo ciclo de trabalho, com base nos

pontos positivos do fluxo utilizado até então e nas melhores práticas das metodologias de mercado.

O processo de trabalho da empresa foi concebido através da implantação da certificação MPS.BR e é completamente voltado à forma tradicional cascata de desenvolvimento, por conta do seu fluxo fluir constantemente para a frente sem pontos de retorno no processo. Mesmo que em alguns momentos haja interação com o cliente para validação de documentos, não há uma entrega formal a fim de receber um feedback avaliativo sobre o andamento dos trabalhos (iteração). A seguir são descritas de maneira resumida as fases e etapas do processo.

A primeira fase do ciclo de vida é a iniciação, onde a principal atividade é a criação e configuração do projeto na ferramenta de gestão. Na fase de planejamento o Gerente do Projeto (GP) planeja as atividades e o Analista de Sistemas realiza a especificação funcional. Nessa fase o GP obtém o comprometimento da direção e do Patrocinador do Projeto com prazos e custos estimados. A fase de execução pode ser realizada em entrega única ou de forma incremental, com mais de uma fase. O encerramento é a etapa de revisão das atividades do projeto e formalização de encerramento com o cliente. A fase de execução é composta por etapas que iniciam com a especificação técnica, onde o Analista de Sistema divide o escopo do projeto em atividades, que são alocadas para os desenvolvedores. A medida que o analista conclui alguma especificação técnica o desenvolvedor já pode iniciar a criação da funcionalidade, liberando assim a verificação das tarefas implementadas pelo testador, que compara o resultado da funcionalidade com a expectativa do requisito.

Na estabilização os desenvolvedores fazem as correções dos problemas identificados pelo testador. Essa etapa pode ocorrer em paralelo com a conclusão dos testes, mas só deve ser priorizada pelo desenvolvedor após a conclusão do desenvolvimento das outras tarefas do escopo. Uma vez que o software esteja homologado pelo testador ocorre a etapa de entrega, com a disponibilização do sistema em ambiente de produção do cliente e o fim do processo.

4.1 Problemas e melhorias identificadas no processo

Com base nas metodologias pesquisadas, foram identificados pontos que poderiam ser melhorados no processo para torná-lo mais aderente à necessidade atual da empresa e do mercado. Tais melhorias e apontamentos estão descritos a seguir.

Documentação e aprovação de requisitos: a documentação do processo destaca que os requisitos funcionais devem ser descritos a nível de negócio e o cliente deve ser acionado para aprovação. Essa etapa do processo foi deixada de lado por alguns motivos, mas principalmente por causa dos prazos apertados e da falta de retorno do cliente. Dessa forma o escopo é enviado

para o desenvolvimento sem que haja aprovação e comprometimento com a funcionalidade que será implementada.

Processo de testes documentado: todas as versões do Demander passam por revisão de testes antes de serem liberadas, mas como gargalos dessa etapa é possível citar a falta de uma rotina documentada e ausência de colaborador dedicado para a função de teste. Existe um documento com regras mínimas a serem testadas antes de cada versão, mas como não há processo de testes definido, este documento deixou de ser atualizado e é executado eventualmente quando há tempo hábil. Para aumentar a qualidade das entregas a restauração dessa rotina é fundamental.

Fluxograma de trabalho para projetos de produto: este é o ponto de maior prioridade da melhoria, pois o produto recebe novas versões todos os meses e precisa ter um processo com etapas bem definidas e documentadas. Esse fluxo deve ser documentado, exposto e difundido entre os colaboradores para que quando um novo membro se junte a equipe, possa ter uma base de como o processo funciona. A vantagem do segmento de produto em relação ao segmento de fábrica é que a tecnologia não varia, mesmo tendo ambiente mobile e web a linguagem de programação é sempre a mesma em cada ambiente.

Ciclos de entrega com prazo definido: as versões do produto Demander sempre foram planejadas conforme a demanda do escopo, ou seja, o escopo era definido e com base nele a estimativa de entrega era elaborada. Dessa forma os projetos podem ficar muito grandes, aumentando os riscos de desvio e consequentemente atraso nas entregas. Outra característica de prazos longos na Retta é que nas situações em que o prazo não fosse cumprido, um novo prazo não era definido, ou seja, se a versão não for publicada na data prevista, não há nova previsão de publicação. Esse fato implica diretamente na entrega aos clientes e também faz com que funcionalidades que já estejam prontas e poderiam ser liberadas, fiquem prezas no desenvolvimento.

Alternância de pessoal entre os projetos: cada desenvolvedor trabalha em um segmento específico de projeto, mas em algumas situações, este pode ser requisitado em projetos diferentes, de acordo com sua capacidade na linguagem programação, ambiente de desenvolvimento, disponibilidade de agenda ou prazos de entrega apertados. Tal situação atrapalha o planejamento do projeto, tornando-se um risco para desvios de prazo e qualidade, já que a funcionalidade deverá ser implementada por outro profissional que pode não estar familiarizado com a necessidade.

Boa comunicação entre os interessados do projeto: como a equipe de desenvolvimento é pequena, os membros são mantidos juntos no setor de desenvolvimento, por isso a comunicação é constante. Porém, como a interação não está definida no processo, pode estar ocorrendo de uma forma ineficaz, além de não ser compartilhada com o cliente e os demais interessados.

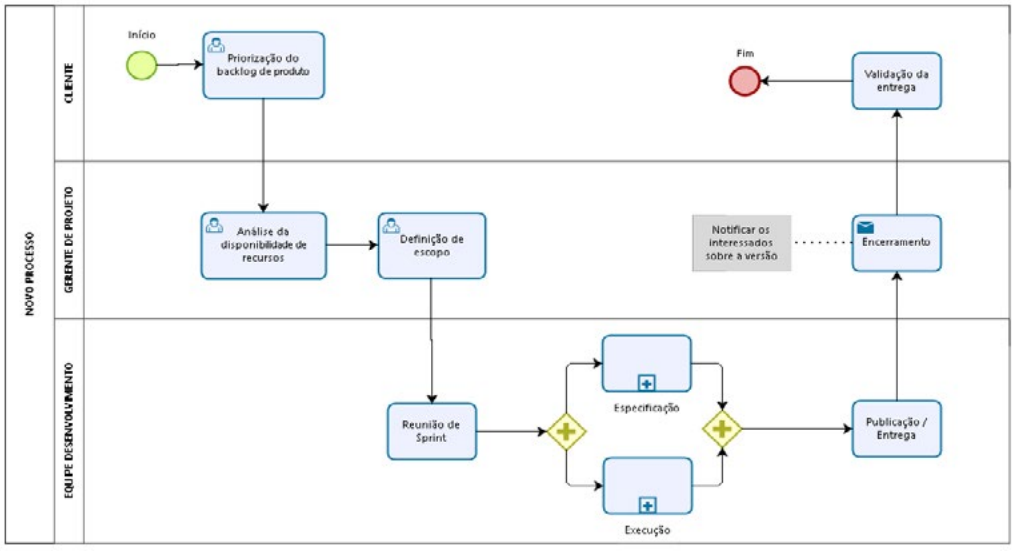
5 RESULTADOS E DISCUSSÃO

Nesta seção serão apresentadas as alterações propostas para as melhorias identificadas, o detalhamento do novo processo proposto e os resultados da aplicação do novo modelo. Será apresentado o novo processo proposto, aplicando conceitos e técnicas das metodologias ágeis para aumentar a eficácia no desenvolvimento. Os resultados obtidos foram subdivididos em qualitativos (questionário de avaliação aplicado aos colaboradores que desenvolveram os pilotos) e quantitativos (comparação dos projetos pilotos com outros similares executados anteriormente).

5.1 Modelo proposto

O modelo utiliza características do framework Scrum, combinadas métodos encontrados no modelo XP, além da manutenção de aspectos considerados adequados no processo atual. Para determinar o novo processo foram feitas reuniões com a equipe, com os gerentes de projeto, com clientes internos e com a direção, obtendo assim a aprovação mutua de todos os envolvidos no ciclo de desenvolvimento da empresa. A Figura 4 apresenta o novo processo, conforme o fluxo proposto por este trabalho, identificando os papéis dos responsáveis por cada etapa.

Figura 4 – Modelo proposto



Fonte: elaborado pelo autor, 2017.

O *backlog* de produto pode ser composto por solicitações de melhorias enviadas pelos usuários do Demander, evoluções estratégicas de mercado propostas pela direção e também inconformidades relatadas à equipe de

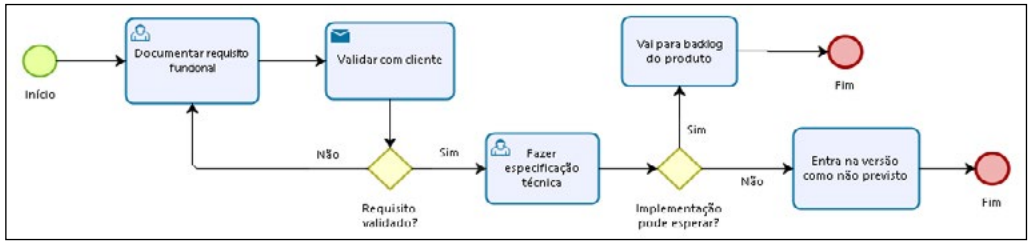
suporte. Por isso, o fluxo inicia sempre com a necessidade de priorização deste *backlog*, feita pelo cliente interno, que conhece cada demanda e a sua respectiva importância.

As próximas tarefas são executadas pelo Gerente de Projeto, que estima dentro do prazo de cada entrega a disponibilidade em horas dos membros da equipe, mensurando o tempo total disponível para desenvolvimento. Para compor o escopo de desenvolvimento de versões é necessário apenas consultar o *backlog* de produto e selecionar tarefas de acordo com a sua prioridade até completar 80% das horas disponíveis para desenvolvimento. Após a definição do escopo o gerente de projetos informa aos *stakeholders* o plano de projeto, contendo o escopo da versão, as estimativas em horas e o percentual de não previstos. Caso seja necessário incluir alguma demanda que exceda o total de não previsto para a versão, o cliente deverá optar em remover alguma outra atividade da lista prevista, ou então o GP remove, iniciando da menos prioritária, tantas tarefas quantas forem necessárias para totalizar a estimativa em horas dessa que está entrando.

A reunião de *sprint* é o marco inicial da execução, onde o responsável apresenta à equipe as definições do projeto, para que assim possam assumir o comprometimento da entrega junto com o cliente, que deve ser incentivado a participar. Neste momento são discutidas dificuldades técnicas do escopo e as estimativas são revistas em conjunto, para identificar possíveis desvios previamente.

As etapas de especificação e execução são compostas por subprocessos que serão representadas no modelo de BPMN como um gateway paralelo, ou seja, ambas devem ser concluídas para que a próxima etapa seja iniciada. Ao término do desenvolvimento do escopo, ou ao término do prazo da *sprint*, a equipe faz a entrega e publicação das funcionalidades desenvolvidas para o ambiente de produção do sistema. Logo após, o gerente formaliza esse momento enviando aos *stackholders* as estatísticas da versão, como: percentual de escopo entregue; problemas enfrentados; quantidade de horas previstas X horas realizadas. O cliente recebe o software e o documento para validar a entrega, encerrando assim o fluxo do processo.

Figura 5 – Subprocesso da etapa de especificação



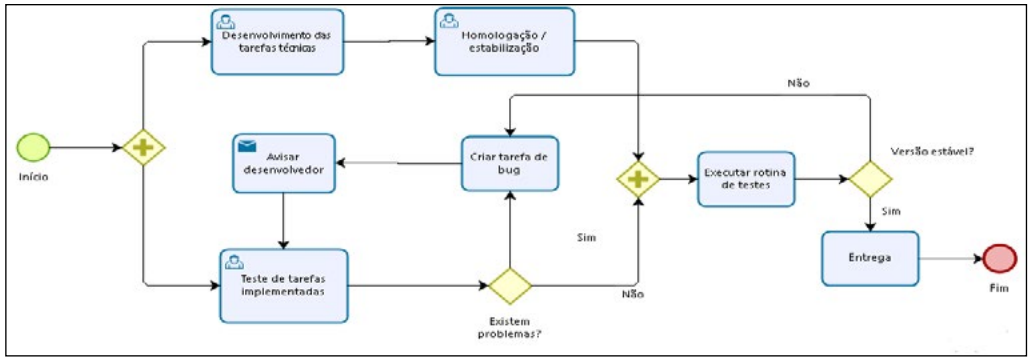
Fonte: elaborado pelo autor, 2017.

A Figura 5 descreve o subprocesso de especificação, etapa realizada basicamente pelo Analista de Software em conjunto com o cliente. O novo processo da empresa tem *sprints* de tempo definidas em 2 semanas, por isso entendeu-se que a especificação já deve estar feita e validada pelo cliente no momento da elaboração do escopo da versão, para não comprometer o prazo.

O processo inicia com uma lista de demandas de análise que foram incluídas pelo gerente conforme priorização do cliente e disponibilidade do analista de software. Inicialmente o analista documenta o requisito. Após o requisito estar em conformidade com a solicitação e necessidade do cliente, o analista criará a tarefa técnica que será utilizada pelo desenvolvedor no momento da criação. Nesta etapa o processo sugere um ponto de questionamento: caso essa demanda seja de caráter emergencial, ela pode ser incluída na versão atual como uma tarefa do tipo não prevista, caso contrário ela será destinada ao *backlog* de produto, ficando disponível para priorização do cliente no próximo projeto. A fase de especificação já fez parte do processo da empresa em algum momento, mas deixou de ser executada.

No subprocesso de execução, demonstrado na Figura 6, ocorre a criação e homologação das tarefas de desenvolvimento. A imagem não mostra, mas nesta etapa ocorrem as reuniões diárias, que são seções rápidas de conversa entre a equipe do projeto para identificar riscos e problemas que possam se apresentar no desenvolvimento, além de garantir que o objetivo de todos está alinhado.

Figura 6 – Subprocesso da etapa de execução



Fonte: elaborado pelo autor, 2017.

O processo prevê que a etapa de testes das tarefas implementadas inicie tão logo alguma das tarefas técnicas tenha sido desenvolvida. Essa proximidade do teste com o tempo de desenvolvimento deve ser encorajada para que o tempo de espera de uma implementação técnica seja mínimo, reduzindo assim a probabilidade de atraso no prazo. Sempre que o testador detectar uma

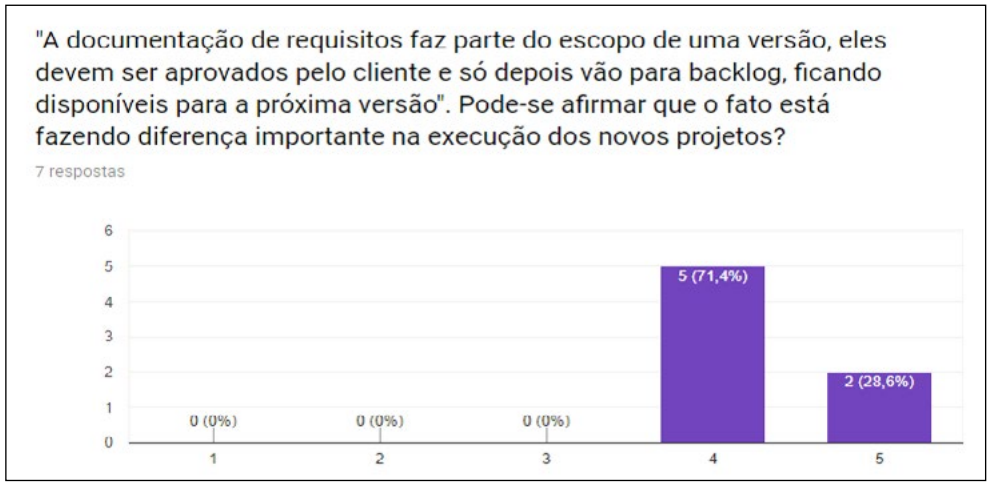
inconformidade de software, ele deve criar uma tarefa para correção e notificar o desenvolvedor responsável.

Novamente identificado com o gateway paralelo, a etapa de execução da rotina de testes só pode iniciar quando o desenvolvimento estiver homologado pelo testador, ou seja, as tarefas de implementação estão de acordo com a expectativa do cliente. Esta rotina consiste em uma lista de funcionalidades mínimas que devem ser testadas antes que a versão possa ser liberada para o ambiente de produção, pois como cada versão é uma evolução de um produto, não se pode permitir que uma nova implementação afete outra funcionalidade já utilizada pelos clientes. Esta é uma etapa que não fazia mais parte do processo e foi um dos focos da melhoria. Se o testador concluir a rotina com êxito o escopo será considerado estável e apto a ser disponibilizado ao cliente, ou então os problemas identificados irão virar tarefas de correção e o processo volta para a etapa de homologação / estabilização.

5.2 Resultados da aplicação do novo modelo

Esta seção apresenta os resultados relacionados aos projetos pilotos que foram desenvolvidos utilizando o novo processo proposto. Serão apresentados os principais resultados e análises relacionados a algumas das melhorias propostas. Os resultados incluem análises qualitativas, quantitativas, e a comparação com projetos similares executados anteriormente, quando aplicável. Foi realizada ainda uma pesquisa com a equipe do projeto das versões Demander, composta por perguntas que seguem uma escala linear, de 1 a 5, onde 1 é “discordo fortemente” e 5 é “concordo fortemente”. Cada pergunta foi relacionada a um ponto de melhoria e suas respostas estão avaliadas nos parágrafos que seguem.

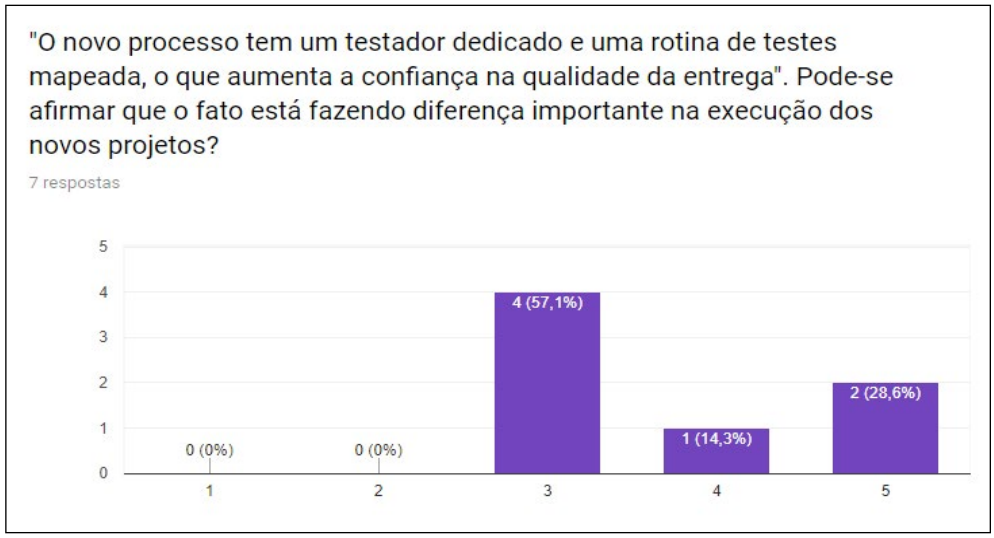
Figura 7 – Questão sobre documentação e aprovação de requisitos



Fonte: elaborado pelo autor, 2017.

De acordo com os entrevistados, a forma como os requisitos estão sendo documentados e validados é bastante satisfatória, pois a resposta média da questão foi 4,29, tendo o menor desvio padrão de todas as questões, com apenas 0,49 pontos. A Figura 7 apresenta na íntegra a pergunta que foi feita aos membros da equipe e mostra as respectivas respostas recebidas, onde nenhum dos entrevistados discordou da nova maneira que o processo conduz a documentação, contra 71,4% que pelo menos concordam.

Figura 8 – Questão sobre rotina de testes de versão



Fonte: elaborado pelo autor, 2017.

Mesmo com um profissional dedicado para a execução dos testes da versão e com uma rotina de testes mínimos mapeada dentro do processo, percebe-se que não há uma segurança quanto ao aumento da qualidade da entrega. Como é possível constatar na Figura 8, a maioria da equipe manifestou opinião neutra (não concorda e nem discorda) sobre a rotina de testes, sendo que na média a resposta foi relativamente boa, com 3,71 pontos e um desvio padrão de 0,95. Entende-se que esse ponto precisa ser discutido com a equipe para que a etapa possa alcançar a eficácia pretendida.

Tabela 1 – Tabela da quantidade de correções lançadas em cada versão de projeto

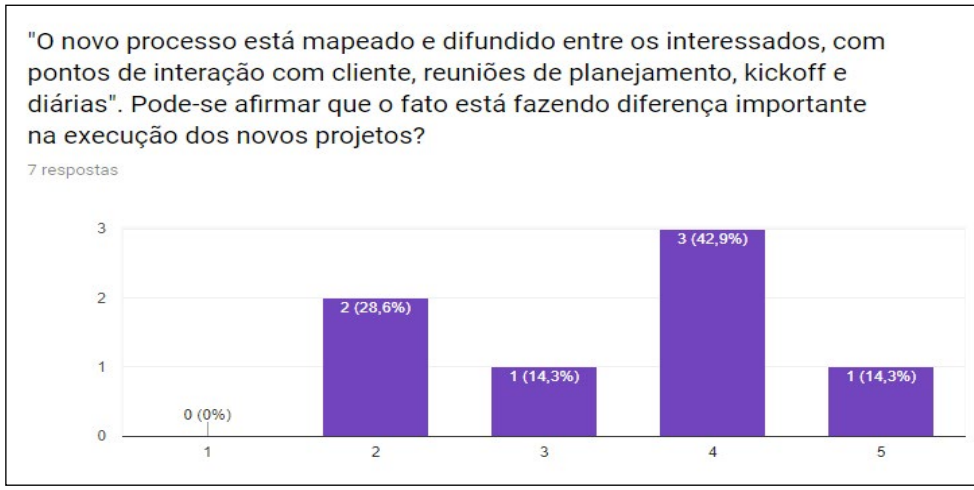
Versão	Lançamento	Correções	Fase
3.5	14/jul	3	Antes da melhoria
3.6	08/ago	3	Antes da melhoria
3.7	14/set	2	Antes da melhoria
3.8	29/set	2	Com processo melhorado
3.9	16/out	1	Com processo melhorado
3.10	30/out	2	Com processo melhorado

Fonte: elaborado pelo autor, 2017.

Um dos indicadores a serem considerados é o número de correções que são liberadas a cada versão, pois elas representam de forma direta os problemas que chegaram até o cliente. A Tabela 1 apresenta as últimas seis versões liberadas para os clientes, das quais três são execuções anteriores a melhoria proposta e as demais são os projetos pilotos deste trabalho.

Pode-se observar claramente que as três versões que foram desenvolvidas antes da implantação do novo processo tiveram mais necessidades de ajustes, 8 no total, do que os projetos piloto, que totalizaram apenas 5 lançamentos de correção.

Figura 9 – Questão sobre o novo fluxo do processo

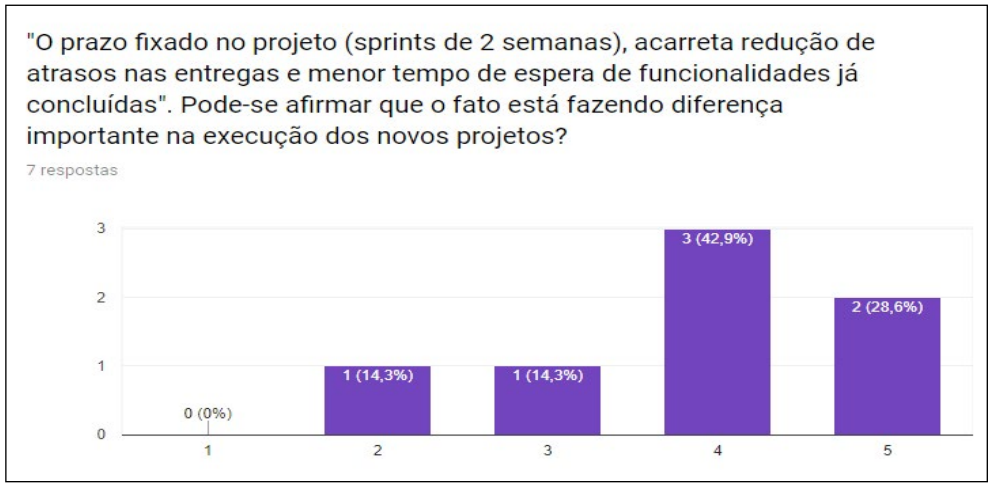


Fonte: elaborado pelo autor, 2017.

O fluxo de trabalho dos projetos de produto destacou-se por ser o ponto de maior discordância entre os entrevistados (FIGURA 9), tendo média de resposta de 3,43, a menor de todas as melhorias e o desvio padrão de 1,13, o maior de todas as questões. Entende-se que esse fato está ligado à forma como o fluxo foi apresentado aos colaboradores, não tendo sido exposto de forma física em lugar de fácil acesso, além de ter sido apresentado apenas uma vez. Por tratar-se de um processo novo, que foge do modelo tradicional executado pela empresa, deveria ter sido apresentado repetidas vezes para os interessados.

Para a maioria dos entrevistados o ciclo de entrega com prazo definido realmente diminui o atraso e o tempo de espera de funcionalidades que já estão validadas, mas estes aguardam o encerramento da versão para serem liberadas para os clientes (FIGURA 10).

Figura 10 – Questão sobre o prazo de entrega fixado



Fonte: elaborado pelo autor, 2017.

A definição de *sprints* de duas semanas de trabalho a frequência cíclica de entregas possibilita um planejamento padronizado de escopo e recursos. A Tabela 2 apresenta uma comparação feita entre os três projetos piloto deste trabalho e as últimas três versões executadas antes da implantação das melhorias. Como é possível observar, o antigo processo garantia uma entrega de 100% do escopo, visto que o prazo era postergado até que toda a execução estivesse concluída, porém esse fato acarretava em desvios consideráveis de planejamento. Mesmo que o novo processo não garanta a entrega de todo o escopo, o mesmo reduz o tempo de espera das funcionalidades que já estão validadas e podem ir para o ambiente de produção, ou seja, as liberações de versão se mantém constantes. Com o novo processo, tanto o cliente quanto a equipe sabem que a demanda será especificada em uma versão de duas semanas e, possivelmente seja desenvolvida na próxima versão de mesmo prazo.

Tabela 2 – Tabela de comparação do atraso em dias em função do escopo entregue

Versão	Início	Previsão	Entrega	Atraso	Escopo	Fase
3.5	19/jun	05/jul	14/jul	9 dias	100%	Antes da melhoria
3.6	17/jul	03/ago	08/ago	5 dias	100%	Antes da melhoria
3.7	09/ago	31/ago	14/set	14 dias	100%	Antes da melhoria
3.8	15/set	29/set	29/set	0 dias	89%	Com processo melhorado
3.9	02/out	13/out	16/out	3 dias	93%	Com processo melhorado
3.10	16/out	30/out	30/out	0 dias	86%	Com processo melhorado

Fonte: elaborado pelo autor, 2017.

6 CONSIDERAÇÕES FINAIS

O objetivo principal deste trabalho foi a implantação de um novo processo de desenvolvimento em uma empresa de software, utilizando técnicas de metodologias ágeis para reduzir ou eliminar problemas de execução. Como resultado da análise foram evidenciados diversos problemas e melhorias necessárias, como a falta de documentação e validação dos requisitos por parte do cliente, uma rotina de testes documentada e constantemente atualizada, um fluxograma de trabalho que favorecesse as técnicas ágeis, ciclos de desenvolvimento com prazo de execução padronizado, um planejamento que permitisse reduzir a alternância de pessoal entre projetos, além da transparência e regularidade na comunicação entre os interessados do projeto.

Dos seis pontos de melhoria destacados, três deles já haviam sido executados em algum processo desenvolvido pela empresa (validação de requisitos, rotina de testes e comunicação entre os participantes), mas com o tempo foram deixados de lado. Por outro lado, três pontos do novo processo (fluxograma ágil, prazos definidos e recursos dedicados) são relativamente novos, mesmo que não sejam assuntos desconhecidos pela empresa, não estão descritos em nenhuma documentação antiga, logo, entende-se que podem ser decisivos para a continuidade da evolução. Contudo, a nova metodologia adaptada para a empresa deverá ser revista anualmente, levantando novas necessidades, simplificando rotinas e ajustando os novos gargalos. Compreende-se que a alternância de recursos entre os projetos é um ponto crítico pois não há garantias de que um dos projetos de fábrica de software, que ainda não utilizam o novo processo, tenha um planejamento tão eficaz ao ponto de não comprometer os demais.

Com base na pesquisa aplicada com os envolvidos, mas também com base nos índices de entrega, conclui-se que a implantação permitiu obter algumas melhorias e atingiu o principal objetivo do trabalho. Alguns pontos do processo devem ser reforçados para obter maior aderência da equipe, que é a protagonista da implantação de qualquer metodologia nova, mas nada que comprometesse o êxito da execução dos projetos piloto. Como continuidade deste trabalho é possível sugerir uma padronização da nova forma de trabalho para todos os projetos da empresa, inclusive os de fábrica de software, além de instituir uma cultura de revisão da metodologia de trabalho objetivando a eliminação de desperdícios, automatização de processos e redução de custos fixos. Dessa forma evita-se que o processo fique defasado novamente.

REFERÊNCIAS

BARBOSA, Welton Luiz de Oliveira. **Processo Unificado e Processo Unificado Racional (UP e RUP)**. 2011. Disponivel em: http://www.webartigos.com/artigos/processo-unificado-e-processo-unificado-racional-up-e-rup/65404/. Acesso em: 12 maio 2017.

FRANZEN, Evandro; BARDEN, Helena. Proposta de adequação do processo de desenvolvimento de software para a melhoria da estimativa de horas. Revista Destaques Acadêmicos, v. 5, n. 1, 2013.

GERHARDT, Tatiana Engel; SILVEIRA, DenoseTolfo. **Métodos de Pesquisa**. Editora UFRGS. Porto Alegre. RS. 2009. Disponível em: http://www.ufrgs.br/cursopgdr/downloadsSerie/derad005.pdf. Acesso em: 15 maio 2017.

KERZNER, H. **Project Management: A Systems Approach to Planning, Scheduling, and Controlling**. 9ª edição. New Jersey: John Wiley & Sons. 2006.

PEREIRA, Eliana Beatriz. **Uma proposta para adaptação de processos de desenvolvimento de software baseados no rational unified process.** Dissertação de mestrado. Porto Alegre. 2005. Disponível em: http://tede2.pucrs.br/tede2/handle/tede/5283. Acesso em: 18 maio 2017.

PMI BRASIL. O que é gerenciamento de projetos? Disponível em: <https://brasil.pmi.org/brazil/AboutUs/WhatIsProjectManagement.aspx>. Acesso em: 30 setembro 2017.

PRESSMAN, R. **Engenharia de Software Uma Abordagem Profissional.** Tradução **Accelerating Velocity and Customer Value with Agile and DevOps**. Disponível em: https://www.ca.com/us/rewrite/articles/agile/accelerating-velocity-and-customer-value-with-agile-and-devops.html. Acesso: em 21 maio 2017.

ROYCE, Winston W. **Managing the Development of Large Software Systems**. Los Angeles, USA. 1970.

SCHWABER, Ken; BEEDLE Mike. **Agile Software Development with Scrum**. Pearson; 1ª edition. 2002.

SOARES, Michel dos Santos. **Comparação entre Metodologias Ágeis e Tradicionais para o Desenvolvimento de Software. Unipac** - Universidade Presidente Antônio Carlos Faculdade de Tecnologia e Ciências de Conselheiro Lafaiete. Gigante, MG. 2005.

SOARES, Michel dos Santos. **Metodologias Ágeis Extreme Programming e Scrum para o Desenvolvimento de Software**. Universidade Presidente Antônio Carlos. Gigante, MG. 2006.

SOMMERVILLE, Ian. **Software Engineering**. 9ª edição. 2011.

VIEIRA, Denisson. **Scrum: A Metodologia Ágil Explicada de forma Definitiva**. 2014. Disponível em: <http://www.mindmaster.com.br/scrum>. Acesso em: 17 setembro 2017.

WELLS, Don. **Introducing Extreme Programming.** 2009. Disponível em: http://www.extremeprogramming.org/introduction.html. Acesso em: 12 abr 2017.

YIN, Robert K. **Estudo de Caso. Planejamento e métodos.** 2ª edição. Editora Bookman. 2001.