UNIVERSIDADE DO VALE DO TAQUARI

CURSO DE SISTEMAS DE INFORMAÇÃO

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

Douglas Lutz

Lajeado, novembro de 2017.

Douglas Lutz

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

Monografia apresentada ao Centro de Ciências Exatas e Tecnológicas da UNIVATES – Universidade do Vale do Taquari, como parte da exigência para a obtenção do título de bacharel em Sistemas de Informação.

Orientador: Prof. Evandro Franzen

Lajeado, junho de 2017.

AGRADECIMENTOS

Em primeiro lugar quero agradecer imensamente a minha família, que é a base de tudo e com quem posso contar sempre. Então, mãe Rosangela, pai Valmor, irmão Gabriel, muitíssimo obrigado.

Agradeço também, de coração, minha namorada Vanessa e filha Alice pela compreensão e apoio durante esta construção.

Um agradecimento especial ao professor orientador, Evandro Franzen e, em nome dele, aos demais mestres que orientam nosso caminho incansavelmente ao longo do curso.

À RETTA Tecnologia da Informação, empresa na qual trabalho e alvo desta implantação, pela oportunidade de desenvolver o projeto.

Amigos e colegas que de alguma forma contribuíram durante o período deste curso.

Cada um de vocês foi responsável por um pouco desta caminhada, muito obrigado!

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. Um grande número de organizações já utiliza essas técnicas em pelo menos algum nível de maturidade. Considerando a realidade de uma empresa que atua no mercado de *software* personalizado para clientes e desenvolve um produto próprio, este trabalho elaborou e validou um processo melhorado, dinâmico e aderente às expectativas dos clientes e da empresa, com base nas metodologias ágeis SCRUM e XP, mantendo pontos positivos do processo atual, que é voltado ao modelo tradicional. Através de uma pesquisa exploratória o autor buscou aprofundar os conhecimentos em metodologias ágeis populares de mercado, comparando-as com o processo tradicional de desenvolvimento de uma empresa para mapear gargalos e sugerir mudanças no novo processo. 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: XtremeProgramming. SCRUM. Metodologias ágeis. Processo de *software*. Melhoria.

ABSTRACT

The quality and financial result of software projects are strongly tied to development processes and both are decisive factors for the success or failure of organizations. The methodologies are built in techniques by software companies that are concerned with results and what constant movements in their development processes. A large number of use already used in digital machines. The objective of the work is to develop an improved work, dynamic and adherent to the expectations of the clients and the company, based on the agile methodologies SCRUM and XP, keeping good points of the current process, which is aimed at the traditional model. Through an exploratory research the author sought to deepen the knowledge in methodologies of the market, comparing it with the traditional process of developing a company to map bottlenecks and suggest changes in the new process. A research of quantitative and qualitative evaluation evaluated or new process and the results indicate that there is a perception of better in the quality of the developed product, besides the reduction of the delays in the deadlines of delivery due to the new form of work.

Keywords: Xtreme Programming. SCRUM. Agile methodologies. Software process. Improvement.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS E ABREVIATURAS

AP – Atributos de Processo

BPMN – Business Process Model and Notation

B2B – Business-to-business

CMMI – Capability Maturity Model Integration ou Modelo Integrado de Maturidade em Capacitação

DSDM – Dynamic Systems Development Method

ERP – Enterprise Resource Planning ou Planejamento de recurso corporativo

MPS.BR – Melhoria de processo do *software* brasileiro

PU – Processo Unificado

RUP – Rational Unified Process

UML – Unified Modeling Language

XP – Extreme Programming

SUMÁRIO

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.

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. Algumas organizações acabam unificando etapas de trabalho ou simplesmente pulando alguma fase por conta de uma estrutura interna enxuta, mas no geral, independentemente da forma como o ciclo de um projeto é composto, as fases e restrições não mudam.

Eventualmente a metodologia de desenvolvimento de *software* adotada pela empresa norteia o fluxo do processo e as técnicas a serem usadas nos projetos, contudo a realidade nem sempre é essa em empresas menores, onde a forma de caracterizar o processo pode ser inversa. A empresa define técnicas de trabalho que mais se adequam a sua realidade e, com base na

forma como está atuando se qualifica como sendo seguidora de uma ou outra metodologia de mercado.

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.

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.

O grupo das metodologias ágeis surgiu justamente 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 e duas consideradas ágeis. 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.

Um dos modelos tradicionais muito utilizado atualmente é o Cascata, onde as etapas devem seguir uma sequência linear de execução, sendo que uma etapa precisa ser concluída

para que a próxima seja iniciada. Apresentado por Royce (1970), o próprio autor já apontava este método como sendo de risco e possivelmente suscetível a falhas.

Outro modelo bastante adotado por empresas é o *Rational Unified Process* (RUP), que segundo Kruchten (2004), é um modelo de processo incremental que fornece uma abordagem de atribuição e responsabilidades de forma disciplinada dentro da organização, focando em prazo e custo de projeto. Kruchten (2004) reforça ainda que esta forma de trabalho é um framework de processo, podendo ser adaptada para as necessidades da organização que está implantando. O RUP é considerado uma metodologia tradicional, mesmo não exigindo que as fases sejam executadas em sequência, pois o modelo tem traços mais pesados de documentação no início, desenvolvimento no meio do projeto e uma fase de homologação pré entrega final ao cliente.

Percebe-se atualmente um movimento migratório de algumas empresas para formas mais ágeis de desenvolvimento, justamente para tentar reduzir o desperdício e os altos custos de projetos. Sabe-se que a forma tradicional de desenvolvimento ainda é bastante utilizada, mas o mercado está forçando uma mudança, já que os próprios clientes estão ficando mais exigentes e participativos. Para poder ter agilidade e flexibilidade no desenvolvimento e entregas de *software* as empresas estão adaptando-se a essas novas formas de trabalho, possibilitando assim uma maior participação do cliente nos projetos e permitindo que as necessidades de mudança sejam incluídas de forma mais rápida, reduzindo o custo de uma nova implementação.

Segundo o estudo *Accelerating Velocityand Customer Value with Agile and DevOps*, divulgado por Rando (2017) que contou com a participação de 1.770 executivos da área, 81% dos participantes acreditam que o desenvolvimento ágil é uma ferramenta crítica para o sucesso da transformação digital e 4 em cada 5 empresas utilizam as práticas ágeis em pelo menos algum nível de maturidade

As metodologias de desenvolvimento ágeis também prometem fazer entregas de *software* com maior qualidade, já que a participação ativa do cliente e o incentivo à colaboração de toda equipe envolvida preza por execuções mais assertivas. Esta coesão entre as partes interessadas é um dos pontos fortes das metodologias novas e possivelmente um dos responsáveis pelo seu sucesso. Existe um processo a ser seguido, mas este processo é flexível

e incentiva alterações de escopo prévias, por entender que o custo de uma alteração no início ou meio de um projeto é menor do que uma alteração após a sua conclusão.

A metodologia ágil denominada eXtreme Programming (XP) é uma das mais utilizadas na atualidade e para Layman (2006) é fortemente dependente da comunicação contínua entre as partes interessadas. Baseia-se nos *feedbacks* constantes para esclarecer a implementação de recursos e responder às mudanças de maneira rápida e eficiente. Ela é ideal para equipes de pequeno e médio porte incumbidas de desenvolver projetos com um escopo vago e poucos requisitos mapeados, onde o cliente inclui novas necessidades com frequência, o que é uma realidade na maioria das empresas. Este modelo visa principalmente a simplicidade e a comunicação para atingir o sucesso dos projetos.

Outro conceito que também foi estudado é o SCRUM, metodologia ágil bastante adotada pelas empresas que querem migrar seus processos tradicionais para o formato ágil. Segundo Sabbagh (2014) o SCRUM é um framework simples utilizado para o desenvolvimento de produtos complexos, com o objetivo de fazer entregas constantes para o cliente e reduzir os riscos de um projeto.

1.1 Justificativa

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. As implementações no produto são sempre solicitações de usuários específicos que serão aplicadas a todos os demais, enquanto os desenvolvimentos de fábrica de *software* são criações únicas para atender às necessidades específicas do cliente em questão.

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 despadronizados.

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.

Para completar a complexidade de alocação de recursos, bem como a elaboração de custos e prazos, a empresa trabalha com várias tecnologias diferentes no setor de desenvolvimento, dentre as quais podemos citar: PHP, Java, C#, Android e iOS, sendo que todos os profissionais são qualificados em pelo menos uma tecnologia, mas pela disponibilidade de agendas as vezes é necessário que alguém com menos experiência execute tarefas em determinado ambiente ou tecnologia que não é do seu domínio. As plataformas também são variadas, pois tanto os projetos de fábrica como o produto podem ser desenvolvidos em ambientes diferentes, dentre os quais destacam-se *mobile*, *web* e *desktop*, onde cada um apresenta características específicas.

A constante evolução da tecnologia, a competitividade e o cenário financeiro vivido pelo setor exige que as empresas busquem novas formas de trabalho que apresentem a agilidade esperada tanto pelo mercado quanto pela alta direção, independentemente da tecnologia utilizada, ambiente em que o projeto será desenvolvido ou a equipe que irá executar. Os clientes também estão cada vez mais exigentes quanto a custo, prazo e qualidade, aumentando a necessidade de agilidade e padronização nos processos da empresa para continuar sendo uma opção competitiva no mercado.

Entende-se que o cenário de qualquer empresa apresenta características específicas, principalmente em um contexto de desenvolvimento de sistemas ou produtos personalizados. Portanto o presente trabalho realizou um estudo das principais formas de desenvolvimento ágeis utilizadas pelo mercado atualmente, mas sem esquecer dos pontos positivos remanescentes das metodologias tradicionais. Foi feita uma comparação dos métodos, visando elaborar um modelo ideal de trabalho para a empresa em questão, considerando características como: equipe, projetos, produto e mercado.

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. Destaca-se a redução dos atrasos nas entregas, a padronização da forma como os mesmos são conduzidos, a entrega de resultados com maior qualidade aos clientes e à alta direção, bem como o resgate de técnicas já utilizadas em algum momento como o teste e a análise funcional.

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.

1.2 Objetivo geral

Elaborar e validar um processo de *software* melhorado, que seja dinâmico e aderente às expectativas dos clientes e da empresa, com base nas metodologias ágeis mais consolidadas do mercado, mantendo pontos positivos do processo atual.

1.3 Objetivos específicos

- Realizar análise das metodologias tradicionais (cascata e RUP) e as principais metodologias ágeis do mercado (XP e SCRUM);
- Apontar problemas e gargalos existentes na forma de trabalho utilizada pela empresa antes deste trabalho;
- Identificar os pontos mais aderentes das metodologias estudadas, objetivando montar um modelo ideal de forma de trabalho;
- Sugerir novo formato de trabalho de acordo com a metodologia estudada;
- Executar projetos piloto para comparação com projetos anteriores;
- Validar o formato de trabalho proposto, coletando dados e validando a melhoria no processo;

1.4 Delimitação do estudo

O escopo deste trabalho contempla um estudo entre o processo de desenvolvimento de uma empresa de *software*, com modelos que estão em alta no mercado, para sugerir melhorias com base nos conceitos estudados. Ao final espera-se ter identificado a metodologia com maior aderência ao processo considerado ideal pela empresa, adaptando os seus conceitos incorporando pontos positivos das demais metodologias, criando assim um modelo próprio de trabalho.

O processo utilizado pela empresa atualmente está certificado pela norma de qualidade de *software* MPS.BR-SW, contudo a reformulação deste modelo de desenvolvimento não seguiu à risca as exigências desta certificação, sendo que não está contemplada nesta proposta a garantia da recertificação, tendo este trabalho apenas o compromisso implantar um processo de trabalho adequado aos interesses da organização no momento. Futuramente esta adaptação às diretrizes do modelo de qualidade pode ser estudada e implantada em um trabalho complementar.

O presente trabalho contempla a definição de novos processos, e na proposição de novas formas de trabalho. Propõe-se a identificação e desenvolvimento de estratégias para obter melhorias nos projetos, visando tanto o aumento do lucro a cada ciclo de projeto, redução de perdas, além do cumprimento das expectativas da equipe e dos clientes.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 Metodologias de desenvolvimento de *software*

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. Neste novo processo de desenvolvimento utiliza-se documentação, ferramentas, prazos e restrições, a ponto de desenvolver um sistema de qualidade. As metodologias podem ser divididas em processos clássicos e processos ágeis, onde em ambos os métodos serão as três principais etapas da construção do *software*: definição, desenvolvimento e manutenção (UTIDA, 2012).

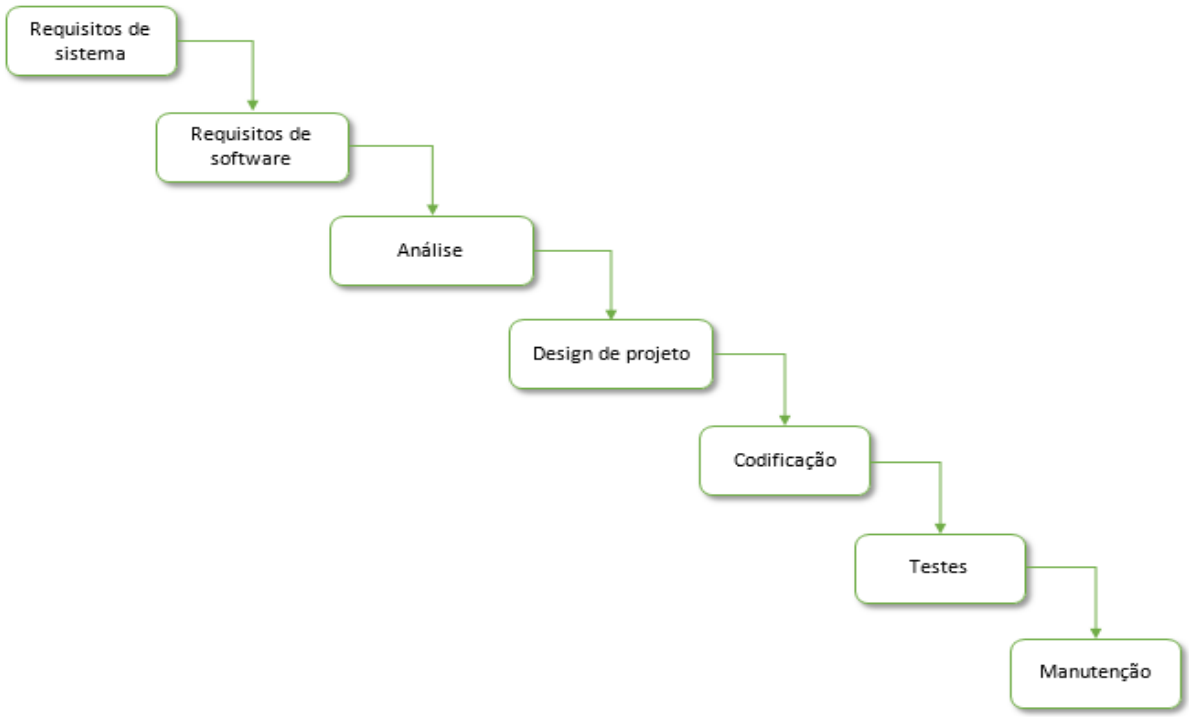
Para Royce (1970) existem duas etapas da construção de *software* que realmente são necessárias e agregam valor visível ao produto final, que são a análise e o desenvolvimento. Ele explica que para projetos menores e com um nível de complexidade baixo, utilizar apenas essas duas fases de projeto pode ser interessante, visto que todos os desenvolvimentos precisam destas etapas básicas. Ainda segundo o autor a maioria dos clientes está disposto a pagar por estas duas fases, por entender que envolvem um trabalho criativo e contribuem para a composição da solução final. Contudo, projetos maiores construídos com base apenas nessas duas fases de desenvolvimento estão frequentemente condenados ao fracasso, uma vez que várias outras etapas são necessárias para a obtenção do sucesso, mesmo que sua importância seja indireta.

2.2 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. Por ter este efeito entre uma fase e outra, ele é conhecido por *Waterfall* cascata. Sommerville (2011) reforça que, utilizando esta metodologia de trabalho, o *software* é inicialmente planejado e agendado, para só depois entrar no processo de desenvolvimento.

> O método tradicional mais conhecido para o desenvolvimento de *software* é o modelo em cascata, ou waterfall. Esse modelo foi inicialmente descrito por Royce em 1970 e se caracteriza por uma sequência de fases de desenvolvimento, em que cada fase somente se inicia quando a anterior se encerra, e a saída de uma fase é a entrada da fase seguinte. Royce, no entanto, criticava o modelo em seu artigo, afirmando que, para o desenvolvimento de *software*, seu uso era arriscado. (Sabbagh, Rafael. 2014. P. 18)

Figura 1 – Modelo apresentado por Royce (1970)



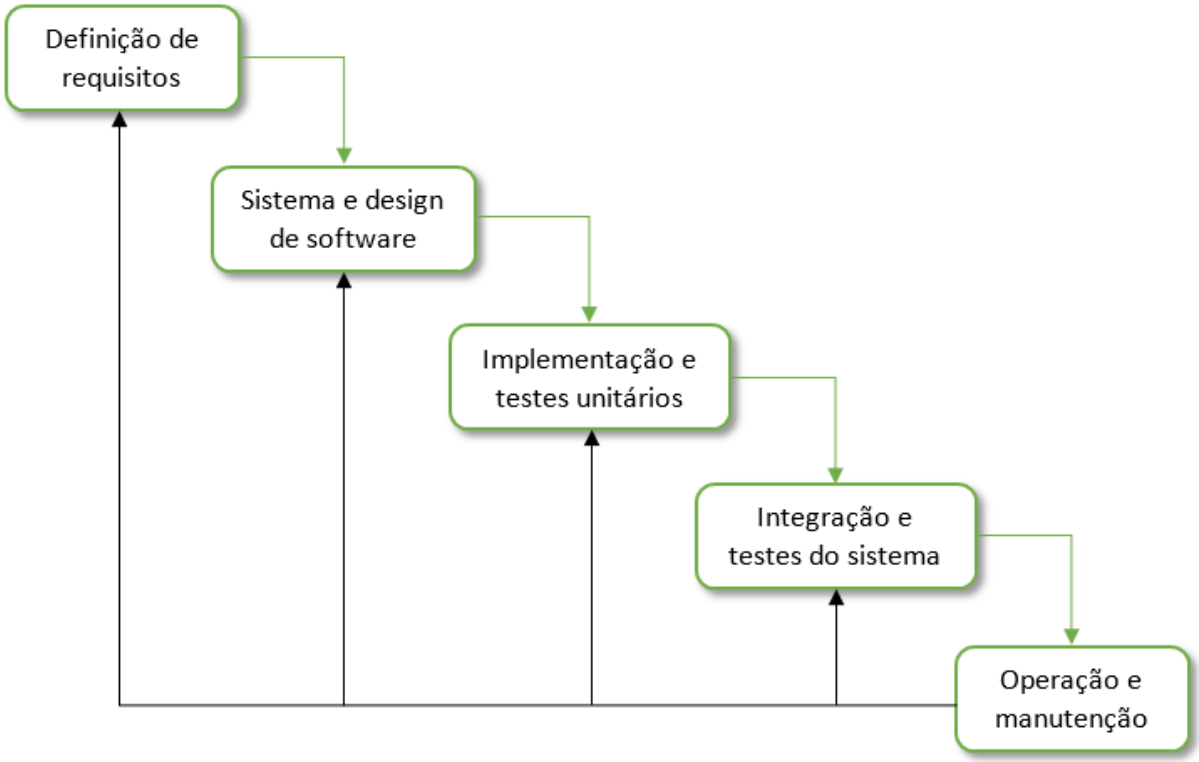
Fonte: Adaptado pelo autor com base em Royce, 1970.

Não muito diferente do modelo desenhado por Royce, Pressman (2011) também afirma que o processo de desenvolvimento de *software* evolui de forma linear seguindo as etapas de:

engenharia do sistema, análise de requisitos, projeto, codificação, testes e manutenção. Cada fase dessas é responsável por artefatos que serão utilizados na fase seguinte, o que torna o processo modularizado e permite que os profissionais sejam especialistas em suas áreas. Ambos entendem que o modelo de etapas seguidas facilita o gerenciamento, mantendo pontos de *check* em cada uma das interações.

Já Sommerville (2011) organiza as etapas do processo de desenvolvimento de uma forma ligeiramente diferente, apresentando-as conforme a Figura 2: Definição de requisitos, sistema e design de *software*, implementação e testes unitários, integração e testes do sistema e operação e manutenção. Apesar de ter fases de processo a menos que o modelo de Royce (1970), as demais etapas documentadas por Sommerville são muito similares, o que nos leva a crer que ambos possuíam o mesmo entendimento da metodologia tradicional de desenvolvimento.

Figura 2 - Modelo apresentado por Sommerville (2011).



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

2.2.1 Etapas do desenvolvimento do método cascata

Cada etapa do processo tradicional de desenvolvimento de *software* (*Waterfall*) só inicia após a conclusão da anterior, sendo que uma etapa alimenta com informações a etapa seguinte. Esta estrutura 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. O que 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.

Brooks (1987) mantém uma posição bastante definida quanto ao o modelo de desenvolvimento de *software* tradicional ao dizer que a ideia de especificar um *software* por completo, para depois implementá-lo do início ao fim e, de forma única, realizar a sua entrega é fundamentalmente errada e responsável por muitos problemas na aquisição dos *softwares*.

No modelo proposto por Pressmann (2011), as etapas inicias do processo são o levantamento de requisitos junto ao cliente (engenharia do sistema) e o detalhamento dos mesmos (análise de requisitos). Já Sommerville (2011) consolida estas duas etapas mencionando somente a definição de requisitos. As demais etapas que se seguem são iguais em ambos os processos, sendo elas: sistema e design de *software*; implementação e testes unitários; integração e testes do sistema; operação e manutenção.

Na etapa de engenharia do sistema, inicialmente o processo tem uma fase de levantamento de requisitos, onde o cliente e os futuros usuários do sistema se reúnem com o analista responsável para discutir o negócio e o processo de trabalho do cliente. Essa análise tem o objetivo de identificar e elencar a lista de necessidades e funcionalidades que o sistema deve ter para atender à necessidade. Nesta etapa os requisitos são identificados e registrados de forma ainda superficial, sem um aprofundamento detalhado no assunto, pois o objetivo é entender o tamanho do escopo e criar a lista de características do sistema para que sejam documentadas na fase seguinte. Após a conclusão do levantamento, os requisitos são revistos e o escopo é delimitado junto ao cliente para que este assuma o comprometimento de não incluir novas necessidades durante o período de execução do projeto (PRESSMANN, 2011).

A etapa de análise de requisitos inicia logo após a conclusão do levantamento de requisitos. Com base na lista criada na reunião de levantamento a equipe do projeto (analistas + responsáveis do cliente) começa a detalhar de forma específica cada uma das funcionalidades, descrevendo a forma de trabalho do cliente em especificações técnicas e de negócio, afim de identificar possíveis problemas de escopo, diminuir o risco de entendimento e gerar a documentação necessária para a próxima etapa de trabalho, que será a fase de projeto. Esta fase é responsável por introduzir clareza e objetividade no escopo de desenvolvimento, pois é determinante para guiar as etapas seguintes (PRESSMANN, 2011).

No sistema e design de *software*, inicia-se o ciclo de desenvolvimento concentrado na divisão e identificação dos requisitos e conceitos criados nas duas (ou uma) primeiras etapas, para que seja criada uma estrutura organizada de sistema que considere tanto os requisitos de hardware, quanto os requisitos de *software* (este último dividido em requisitos funcionais e não funcionais). O design de *software* é responsável por identificar e documentar os pontos críticos do sistema, como os principais relacionamentos dos pontos importantes (SOMMERVILLE, 2011).

Na etapa de implantação e testes unitários, os requisitos são desenvolvidos conforme a estruturação e design elaborados na etapa anterior. As unidades de programas ganham forma e utilidade, sendo cada uma delas criadas para atender à necessidade específica da solicitação. Os testes unitários são a garantia de qualidade da implementação e acompanham cada rotina e unidade do sistema ao longo do seu desenvolvimento integral, aumentando a confiabilidade no *software* que está sendo gerado (PRESSMANN, 2011).

A etapa de integração e testes do sistema baseia-se na assertividade de cada unidade de *software* desenvolvida e segurança na integração das diversas unidades que podem ter sido desenvolvidas de forma simultânea, mas em contextos separados. A partir deste momento as partes começam a ganhar forma como sendo um sistema completo e complexo, tendo as suas rotinas e funcionalidades testadas de modo geral para assegurar que as solicitações do cliente estão sendo atendidas. Após a bateria de testes (unitários e de sistema), o *software* é entregue para o cliente (SOMMERVILLE, 2011).

Por fim, a operação e manutenção é considerada o ciclo de vida mais longo do processo, esta fase contempla a instalação e início do uso do *software* pelo usuário final. A operação

remete à manutenção, uma vez que a correção de pequenos erros não identificados anteriormente é feita nesta etapa. Nesta também surgem as demandas de melhoria das rotinas já implementadas, novas necessidades levantadas pelos usuários e claro, a validação das unidades desenvolvidas (SOMMERVILLE, 2011).

2.3 Processo Unificado Racional – RUP

Conforme Barbosa (2011), o Processo Unificado Racional nasceu da mistura das melhores práticas de desenvolvimento de *software*, querendo 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. Atualmente a UML é utilizada em conjunto com o PU (Processo Unificado) no desenvolvimento de *softwares*. É considerado um processo ágil, tendo como objetivo garantir uma produção de *software* de alta qualidade, que cumpra o que for proposto em relação a tempo e orçamento.

O RUP apresenta um modelo que deve ser seguido com bastante clareza, pois define exatamente quem é responsável por cada parte, como e quando as coisas devem ser realizadas, descreve as metas especificando para que sejam alcançadas, garantindo assim a robustez do modelo. Existem 4 fases que descrevem o modelo de processo de *software*, mostradas na Figura 3, onde as metodologias de desenvolvimento e estruturas de avaliação são comparadas em duas dimensões.

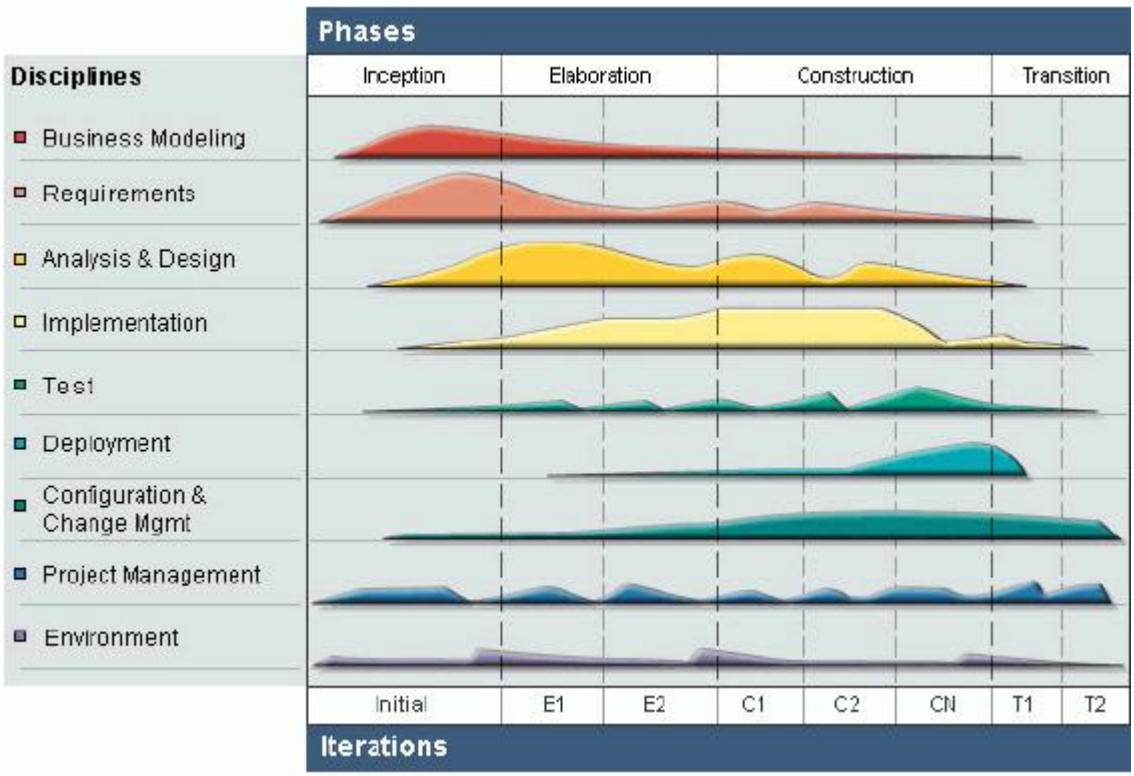
A primeira fase é a de Concepção, onde é feita a obtenção dos requisitos junto ao cliente, são definidas as regras da negociação e como será o sistema. Criam-se os cenários, identificam-se os recursos, define-se um cronograma e os marcos do projeto, de acordo com a necessidade de *software*. Esta etapa concentra boa parte do levantamento de requisitos junto ao cliente, sendo que nas demais etapas esse contato com o cliente diminui bastante.

A segunda fase é a de Elaboração, que não necessariamente inicia após a conclusão da primeira, é onde faz-se o planejamento e modelagem do *software*; documentação do que foi definido na etapa de concepção; ocorre a maior parte do trabalho da análise e design do *software*; o desenvolvimento tem início de forma mais objetiva.

Na fase de construção do sistema o principal objetivo é concluir o desenvolvimento. É nesta etapa que a otimização de custos torna-se ainda mais importante, visto que a qualidade e assertividade do desenvolvimento estão diretamente relacionados ao sucesso do projeto. Nesta etapa ainda resta um pouco de análise a ser feita e os testes da aplicação iniciam de forma tímida.

Por último, a fase de Transição é onde a construção do *software* é finalizada e ocorre a implantação no cliente. Essa implantação já pode ter sido iniciada na fase anterior, de forma parcial, mas é na transição que o *software* é levado até o usuário para realizar testes. Podem ser criados manuais para informações e ao fim fecha-se o ciclo do modelo onde foi criada uma versão do *software*.

Figura 3 – Fases do processo RUP



Fonte: The IBM Rational Unified Process for System z, 2007.

Na imagem acima pode-se observar que existem disciplinas que decompõe as fases citadas anteriormente. Tais disciplinas lembram um pouco a organização do modelo cascata, pois uma iteração tem o objetivo de gerar artefatos para a próxima, mas com a diferença de que

a primeira não precisa estar concluída para que a próxima inicie. Para Borges (2007), uma disciplina gera um conjunto de funcionalidades um pouco mais próximo do requisito inicial.

Ele descreve a iteração de *Business Modelling* (ou Modelo de Negócio) como sendo as atividades de descrição do negócio do cliente, onde o processo é descrito e, em conjunto com o cliente, ocorre a definição dos pontos mais importantes para o sistema. Assim, a medida que o negócio é modelado os artefatos estão prontos para que sejam documentados, organizados e descritos de forma mais completa, dando início à disciplina de *Requirements* (ou Requisitos).

Analysis and Design (ou Análise e Desenho) comporta as atividades de construção da arquitetura e criação do sistema, permitindo que a implementação tenha início. Essa iteração se concentra fortemente na etapa de elaboração.

Iniciando na etapa de elaboração, mas destacando-se na etapa de construção, *Implementation* (ou Implementação) representa as atividades de desenvolvimento do sistema e criação dos testes unitários. Pouco desenvolvimento deve ficar para a etapa de transição, mantendo apenas alguns ajustes e correções para a próxima fase.

O período de *Test* (ou Teste) não tem fase de início e fim definida, pois ela pode ocorrer em vários momentos do projeto, sempre acompanhando o desenvolvimento do sistema. Por outro lado o Deployment (ou Distribuição), que consiste em documentar o *software* produzido (manual do sistema) e gerar o pacote de instalação têm seu início no final da etapa de construção e é a disciplina de maior peso na fase de transição.

Definidas por Borges (2007) como disciplinas de apoio ao projeto, *Configuration and Change Management* (ou Gestão de Configurações e da Mudança) e *Project Management* (ou Gestão de Projeto) são trabalhos que não abandonam o projeto em nenhuma das fases, pois são atividades de planejamento, monitoramento e gestão, que proporcionam os meios necessários para o desenvolvimento, prestando suporte aos interessados do projeto conduzindo as versões do sistema e as alterações solicitadas pelo cliente.

O *Environment* (ou Ambiente) é um estágio que seleciona as ferramentas de suporte ao desenvolvimento e executa as devidas adaptações do processo que se fazem necessárias para atender as expectativas do projeto e da organização.

Martins (2007) resume o RUP como um processo de engenharia destinado a orientar organizações de *software* em seus esforços para criar sistemas sólidos, onde os ciclos de desenvolvimento são compostos por quatro fases (início, elaboração, construção e transição) que por sua vez são divididas em nove disciplinas. Juntas, etapas e disciplinas compõem um ciclo de processo, que pode ser uma parte do todo, visto que a vida útil de um projeto é composta por um número finito de ciclos de desenvolvimento.

2.4 Métodos ágeis

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. À medida que mais empresas se basearem neste processo e compartilharem as suas experiências, mais embasamentos existirão para que possam ser mensuradas as vantagens, desvantagens, riscos e formas de introduzir esta nova forma de trabalho. Soares complementa dizendo que os resultados parciais se mostram bastante promissores quanto a prazos, custos, qualidade e confiança.

Para Libardi e Barbosa (2010) o desenvolvimento de *software* apresenta variáveis diversas em cada projeto, sendo elas na forma, na equipe ou no tempo em que está sendo executado. Assim, devemos entender este processo como algo imprevisível e complicado, características que não respondem muito bem ao modelo tradicional de desenvolvimento, devido ao alto custo de revisão de cada etapa e o esforço necessário para incluir uma alteração.

2.4.1 Manifesto ágil

Em fevereiro de 2001, dezessete pessoas reuniram-se no The Lodgeat Snowbird, nas montanhas Wasatch de Utah, para conversar, esquiar, relaxar e tentar encontrar um terreno comum no mundo do desenvolvimento de *software*. Deste encontro emergiu o *Manifesto Agile*

'Software Development', elaborado em conjunto pelos representantes das metodologias Extreme Programming, SCRUM, DSDM, Desenvolvimento de Software Adaptável, Crystal, Feature-Driven Development, Pragmatic Programming e outros simpatizantes da necessidade de uma alternativa aos processos de desenvolvimento de *software* pesados e orientados a documentação que foram convocados. Um breve resumo desta história está descrito e assinado por Jim Highsmith na página oficial do manifesto (*http://agilemanifesto.org*). Jim foi um dos entusiastas que esteve presente no evento e contribuiu para a construção deste manifesto, cujo principal objetivo foi elaborar melhores meio de desenvolvimento de *software*.

Este manifesto foi um marco na história da evolução do desenvolvimento de *software* pois apresentou de forma oficial uma nova proposta de trabalho que já vinha sendo testada por várias empresas e possuía muitas definições diferentes. Este manifesto foi responsável por reunir os conceitos comuns destas metodologias e transcrever a partir destes os valores e princípios da comunidade de desenvolvimento ágil. Os diferentes métodos continuam existindo, cada um com suas características, porém o manifesto fez despertar a atenção de todos para o movimento e uniu de certa forma as diferentes culturas em torno do termo ágil (Neiva e Matos, 2010).

2.4.2 Valores e princípios do manifesto ágil

Conforme o site oficial da organização do manifesto Ágil, destaca-se que o manifesto foi descobrindo melhores maneiras e formas para desenvolver *softwares*, buscando otimizar o processo de desenvolvimento das empresas envolvidas.Com isso, o manifesto descreveu alguns valores, sendo eles:

- Indivíduos e interações ao invés de processos e ferramentas;
- *Software* operante ao invés de documentações completas;
- Colaboração do cliente ao invés de negociações contratuais;
- Responder às mudanças ao invés de seguir um planejamento.

Após dias de debate, o grupo de desenvolvedores não chegou a uma solução que poderia ser definida em somente um processo, pois o desenvolvimento de *software* é composto por variáveis em todas as suas etapas. Contudo, o grupo chegou ao consenso sobre valores e

princípios que entendiam como sendo fundamentais, assim nasceu o Manifesto Ágil (FILHO, 2008).

Conforme Neto (2002), o movimento ágil é descritopor12 princípios. Porém nem todo modelo que utiliza processo ágil aplicará todos eles, alguns modelos optam por não atender um ou mais princípios. Sendo eles:

- Nossa prioridade mais alta é satisfazer o cliente através de entregas contínuas e antecipadas de *software* válido;
- Mudanças nos requisitos são bem-vindas, mesmo as que chegam tarde no desenvolvimento. Processos ágeis asseguram a mudança como uma vantagem competitiva do cliente;
- Entregar *software* produtivo frequentemente, de algumas semanas a alguns meses, de preferência os tempos mais curtos;
- Pessoal de negócio e desenvolvedores trabalham juntos diariamente durante o projeto;
- Criar projetos em torno de indivíduos motivados, proporcionar o ambiente e suporte que eles necessitam e confiar que eles farão o serviço;
- O método mais eficiente e efetivo para transmitir informações entre e para a equipe de desenvolvimento é a conversa cara a cara;
- *Software* produtivo é a medida primária do progresso;
- Processos ágeis promovem um desenvolvimento sustentável. Os patrocinadores, desenvolvedores e usuários devem ser capazes de manter um ritmo constante indefinidamente;
- Atenção contínua à excelência técnica e boa solução melhoram a agilidade;
- Simplicidade – a arte de maximizar a quantidade de trabalho não feita – é essencial;
- As melhores arquiteturas, requisitos e projetos emergem de equipes auto organizadas;
- Em intervalos regulares, a equipe reflete sobre como se tornar mais efetiva e então sintoniza e ajusta seu comportamento de forma apropriada.

2.5 ExtremeProgramming - XP

Conforme Soares (2005), a metodologia Extreme Programming (XP), é diferenciada 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 algumas 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.

A XP baseia-se em 12 práticas para seu desenvolvimento:

- Planejamento: organizar o que é necessário ser feito e o que não é necessário, podendo ser adiado no projeto.
- Entregas frequentes: entregar versões a cada mês (ou período pré-determinado), aumentando o *feedback* rápido para com o cliente.
- Metáfora: descrições de funcionalidades sem a utilização de termos técnicos, com o intuito de guiar o desenvolvimento do *software*.
- Projeto simples: o programa desenvolvido precisa ser o mais simples possível, podendo serem adicionados requisitos futuros.
- Testes: a validação deve ser feita durante o processo de desenvolvimento.
- Programação em pares: o desenvolvimento deve ser feito em duplas, onde em um único computador, o colaborador que terá o controle de teclado e mouse implementa o código, enquanto o outro observa, buscando identificar erros e propor melhorias para o código implementado, podendo a dupla alternar as funções continuamente.
- Refatoração: presente em toda fase de desenvolvimento, esta rotina visa ajustar alguma rotina já desenvolvida, reescrevendo seu código parcialmente para melhorar o entendimento ou a qualidade do *software*.
- Propriedade coletiva: todos da equipe tem liberdade e pode adicionar ou remover valor de um código, mesmo não sendo o próprio desenvolvedor, pois no método XP todos colaboradores são responsáveis pelo produto de *software* que será entregue. A vantagem deste ponto é que caso algum membro da equipe saia do projeto antes do seu término, todos têm conhecimento do desenvolvimento.

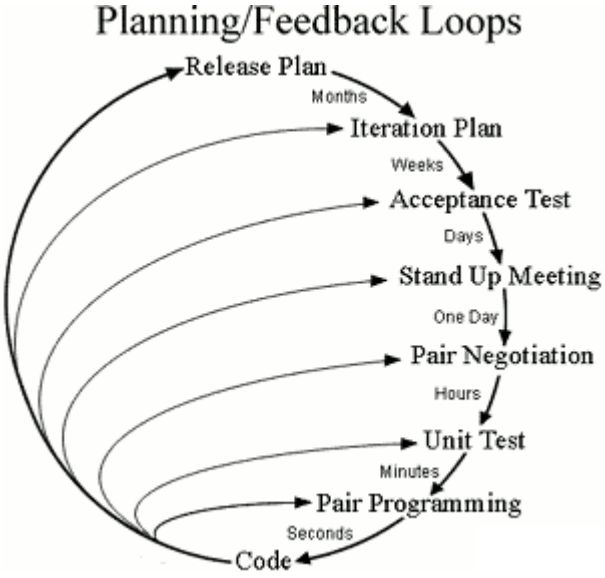
- Integração contínua: o sistema deve ser construído parcialmente várias vezes ao dia. Integrar apenas um conjunto de modificações de cada vez é uma prática que funciona bem e facilita as possíveis correções quando um teste falhar.
- 40 horas de trabalho semanal: para o método XP não deve ser necessário trabalhar mais do que 40 horas semanais, caso isso ocorra, possivelmente há algum problema no projeto. Neste caso deve se fazer um novo planejamento para não sobrecarregar os membros do time.
- Cliente presente: o cliente deve estar presente durante todo o desenvolvimento do projeto, recebendo entregas parciais e retornando *feedbacks* constantes, minimizando assim o número de implementações erradas.
- Código padrão: padronização na arquitetura do código, para que este possa ser compartilhado entre todos os programadores.

Conforme Wells (2009), gerentes, desenvolvedores e clientes, são todos parte de uma mesma equipe. Sempre visando fazer o melhor, o escopo e plano do projeto podem ser criados de forma simples. O XP trabalha em três planos, onde o primeiro visa objetivos para o futuro e agrupa implantações maiores, o segundo visa sempre a próxima interação e o terceiro que visa a interação atual. Estes planos não são fixos e podem ser recriados sempre que necessários.

O XP utiliza desenvolvimento orientado por teste (TDD) e 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.

A Figura 4 ilustra o processo de desenvolvimento com base nas etapas do XP, onde o planejamento da iteração é feito para um curto período de tempo e, através das técnicas da metodologia, as funcionalidades previstas para esta entrega são desenvolvidas. De forma cíclica esse processo é repetido e, a cada iteração o cliente recebe uma parte funcional do sistema.

Figura 4 – Modelo de trabalho Extreming Programming.



Fonte: Morse, 2017.

2.6 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:

Pré-planejamento: em primeira etapa, são descritos todos os requisitos em um documento chamado *backlog*. No planejamento, são descritas as prioridades para o desenvolvimento, estimativas, 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.

Desenvolvimento: nas metodologias tradicionais o comum seria considerar as variáveis somente no início do projeto, para o método Scrum este 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. Cada ciclo deve ser planejado para durar entre uma ou duas semanas, e devem ser desenvolvidos de forma tradicional, fazendo-se primeiramente a análise, depois o projeto, implementação e por últimos os testes.

Pós-planejamento: após o desenvolvimento são feitas reuniões para analisar o projeto e demonstrar o *software* elaborado. Nesta etapa são feitos testes finais, treinamentos e documentação de entrega ao cliente.

Conforme Vieira (2014), o Scrum não é um método padronizado onde segue-se etapas, o modelo é indicado para gerenciar e organizar trabalhos complexos tais como desenvolvimento de *softwares*. Como pode ser observado na Figura 5, o processo baseia-se em alguns passos que possuem características flexíveis.

Figura 5 - 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, conforme exemplo da Figura 6.

Figura 6 – Exemplo de *Product Backlog*

Fonte: Vieira, 2014.

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 *sprint*, 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 2 a 4 semanas de desenvolvimento para cada entrega.

> A *Sprint Backlog* é o ponto de partida de cada *sprint*. Consiste em um conjunto de funcionalidades selecionadas do *Product Backlog* para serem implementadas durante a *sprint*. Os itens são selecionados pelo *Scrum Team* em conjunto com o *Scrum Master* e o Dono do Produto na Reunião de Planejamento da *sprint*, de acordo com as prioridades dos itens. Quando todos os itens da *Sprint Backlog* estiverem prontos, um novo incremento do sistema é entregue (Fagundes, Priscila. 2005. P.35)

Sprint Planning é a reunião que acontece após o termino de cada iteração (*sprint*) com o objetivo de compor uma nova *Sprint Backlog* a partir do *Product Backlog*.

Chamada de reunião diária (ou *Daily Scrum*), deve ocorrer sempre no mesmo horário, com todos os membros envolvidos, podendo ser de 15 minutos ou menos, para pontuar e registrar para cada membro da equipe os três itens considerados importantes por Schwaber e Beedle (2002): o que fez até agora; o que fará até a próxima iteração; quais são os impedimentos;

No método Scrum, o *Definition of Done*, é considerada a entrega do produto, ou uma funcionalidade concluída. Para Sabbagh (2007) ao concluir uma s*print* a equipe deverá ter

gerado um novo incremento entregável do produto a partir do *Sprint Backlog*, para validação do cliente. Dessa forma o cliente tem visibilidade do andamento do projeto.

Nas metodologias tradicionais o processo é definido e as saídas de uma fase são necessariamente a entrada da próxima. Assim, Ken Schwaber apud Bassi Filho (2008), destaca que a complexidade da concepção de *software* torna essa abordagem inadequada ao problema em questão, visto que pode-se obter resultados melhores ao tratar o desenvolvimento de sistemas como um processo empírico.

2.7 Comparação dos métodos clássicos e ágeis

Para Soares (2005) as metodologias ágeis não apresentam grandes mudanças nas etapas quando comparadas com as metodologias clássicas, o que as diferencia é a forma como são aplicadas e os seus valores. Nas metodologias ágeis o enfoque é nas pessoas e na comunicação, enquanto um modelo mais tradicional foca na execução das fases.

Soares (2006) comenta que projetos elaborados com metodologias ágeis obtiveram melhores resultados comparados a metodologias tradicionais, em questão de prazos, custos e qualidade do *software*.

Taroco e Werner (2001) descreveram em seu artigo uma análise comparativa entre métodos tradicionais e ágeis. Destacam que na metodologia tradicional o seu planejamento é realizado logo no início do projeto e não contempla alterações durante seu desenvolvimento, concluindo que seu planejamento deve ser prévio e detalhado. Já nas metodologias ágeis o planejamento ocorre de forma interativa entre desenvolvedores e clientes, além de ser desenvolvido em ciclos, onde as etapas são priorizadas conforme a necessidade do cliente. Para os autores não existe o melhor modelo para ser utilizado em um desenvolvimento, pois este deve ser escolhido conforme as necessidades e possibilidades de aplicação da empresa.

Entende-se que as metodologias tradicionais de desenvolvimento de *software* são mais burocráticas e concentram a execução das atividades em etapas bem definidas. No modelo Cascata os atributos de processo gerados em uma etapa são necessariamente a entrada para a etapa seguinte. No modelo RUP as atividades de uma etapa podem iniciar durante a execução de outra, mas ainda assim quando se observa a Figura 5 (Fases do processo RUP) fica clara a

divisão e concentração de esforços entre as fases do projeto. Por outro lado, as metodologias ágeis incentivam o desapego à burocracia, tornando o processo mais dinâmico.

Uma das principais características que distingue os modelos de desenvolvimento é a forma e o momento em que a documentação é gerada, sendo que na metodologia tradicional o *software* precisa estar todo (ou quase todo) especificado para que o desenvolvimento possa ter início, enquanto que os processos ágeis focam em desenvolvimento e entregas ao invés de ter documentações pesadas.

Outra característica que diferencia bastante as metodologias é a comunicação com o cliente. O processo tradicional necessita de um contato inicial com o cliente e, após isso, ocorre uma entrega apelidada de *Big Bang*, pois todo o *software* é entregue em uma única vez. Já na metodologia ágil existem as iterações, que são entregas pequenas e funcionais, ou seja, o cliente tem a entrega de uma ou mais funcionalidades já concluídas Assim ele pode realizar a validação parcial do *software* e fazer os apontamentos necessários muito antes do que faria no processo tradicional. Dessa forma o custo de alteração, quando existente, é muito menor do que seria no evento de entrega final do projeto.

Prazos também podem ser revistos com maior facilidade na metodologia ágil pois as entregas parciais permitem que o cliente e a equipe de desenvolvimento percebam o impacto de possíveis alterações de curso. Os clientes priorizam os requisitos do escopo, elencando a prioridade do desenvolvimento. Com base nessa priorização são definidas as iterações, onde serão disponibilizadas uma lista de funcionalidades para a avaliação do cliente. (Fagundes, Priscila Bastos et. Al. 2008)

Enquanto a metodologia ágil prioriza o desenvolvimento antes da criação de documentação extensa, a metodologia tradicional foca no micro gerenciamento de todas as etapas, como pode-se observar no texto de Sabbagh:

> Os métodos tradicionais são fortemente prescritivos e, de forma geral, se caracterizam pelo foco em planos detalhados definidos no princípio do projeto, como custo, escopo e um cronograma detalhado, em microgerenciamento, no poder centralizado, em processos cada vez mais complicados e em extensa documentação. Mudanças são fortemente indesejadas (Sabbagh. 2014. P.19).

Assim, entende-se que os conceitos da metodologia ágil são bastante interessantes e aplicáveis aos projetos de desenvolvimento de *software*, contudo a metodologia utilizada pode variar de acordo com as características do projeto e do cenário específico da empresa.

2.8 Gestão de projetos de *software*

Este capítulo apresenta de forma sucinta os conceitos de gerenciamento de projetos pois eles serão utilizados na elaboração de proposta do novo processo de trabalho da empresa. De acordo com o PMI Brasil, gerentes de projeto são pessoas orientados para um objetivo, com habilidades e competências que os permite trabalhar bem sob pressão, sentindo se confortáveis com mudanças em projetos. Para isso precisam ser agentes de mudança, estabelecendo objetivos, métricas e compartilhando este sentimento de propósito para os demais membros da equipe. Organização, flexibilidade, dinamicidade e comunicação são algumas das características fundamentais ao gerenciamento, permitindo que o gerente conquiste a confiança do time e dos patrocinadores do projeto.

Conforme Russo e Ruiz (2005), para que haja uma boa gestão, habilidades e competências são necessárias para o líder de um projeto. O responsável pelo projeto deve focar nos seguintes aspectos:

- Gerenciamento da atenção, é a competência do líder em fixar o foco, a visão e a direção da equipe nos objetivos, sensos, estratégias e metas do projeto;
- O Gerenciamento do significado é a responsabilidade do líder em comunicar o que irá significar o resultado do projeto para a estratégia da empresa;
- Gerenciamento da confiança é a habilidade do líder em ser um exemplo para a sua equipe. Essa habilidade é importante para que os membros da equipe sintam-se seguros para seguir o líder mesmo que estejam ocorrendo diferenças de percurso;
- Gerenciamento de si mesmo, é o gerenciamento crítico. Se o líder estiver motivado, possuir conhecimento e habilidades sobre o projeto gerido, aumentará a chance de empregá-lo de uma maneira efetiva.
- Competência pessoal, essa capacidade que determina como os líderes lidam consigo mesmo, tendo autoconsciência e possibilidade de compreender as próprias emoções, bem como as possibilidades e os seus limites, valores e motivações.

- Competência social é a capacidades de gerenciar seus relacionamentos levando em consideração os sentimentos dos liderados para tomar decisões inteligentes, capazes de converter possíveis divergências de opiniões em ações efetivas.
- Administração dos relacionamentos é a forma como o líder lida com as emoções alheias, utilizando as ferramentas mais visíveis, como persuasão, gerenciamento de conflitos e colaboração para alcançar os objetivos do projeto.

Projeto é um conjunto de atividades temporárias, que podem ser executadas em grupo ou de forma individual e que objetivam a produção de algum produto, serviço ou atingimento de resultados. Cada projeto é único pois deve apresentar escopo e recursos definidos para sua execução, assim como prazos de início e fim definidos (PMI Brasil).

2.9 Modelagem de processo

Um "Modelo" é uma representação, podendo ser matemática, gráfica, física ou narrativa. Uma modelagem possui diversas aplicações, incluindo organizações, previsões, medição, explanação, verificação e controle. "Processo" pode ser considerado um processo de negócio e detalhado em vários níveis, podendo ser contextualizado para mostrar uma visão operacional detalhada que pode ser avaliada e caracterizada no seu desempenho. Um modelo de processo pode ser explicado de forma gráfica ou com diagramas, com informações sobre objetos, relação de objetos, como estes objetos podem ser representados, como se comportam ou desempenham (BARROS e NETO, 2013).

2.9.1 BPMN

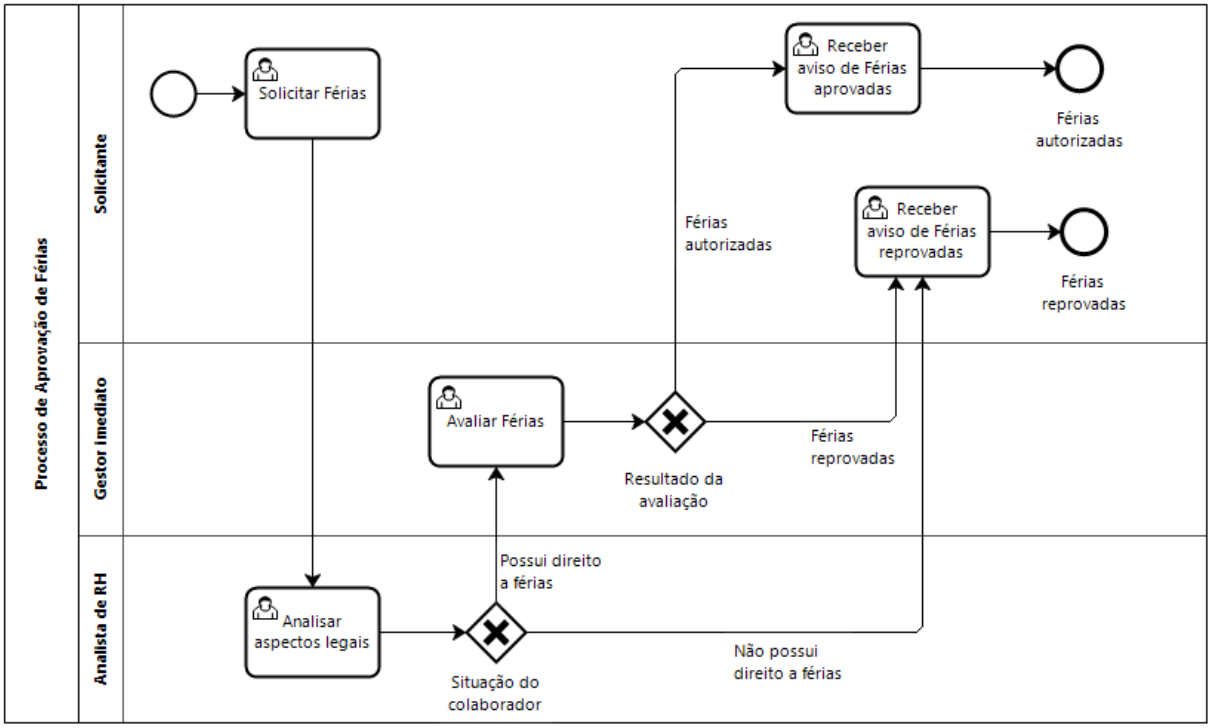
O BPMN (Business Process Modeling Notation) é uma notação gráfica, que utiliza um conjunto de figuras para transmitir as informações e ajuda a melhorar a gestão de processo de negócio. Assim este processo utiliza a lógica das atividades, as mensagens entre os diferentes participantes e toda a informação necessária para que um processo seja analisado, simulado e executado. As figuras e imagens utilizadas para os diagramas, são utilizados de forma clara e

padronizada, facilitando assim a comunicação e um entendimento geral (BARROS e NETO, 2013).

Essa notação gráfica foi desenvolvida pela organização Business Process Management Initiative (BPMI) em 2004 e segue sendo utilizada amplamente por empresas para desenhar o workflow dos seus processos. Segundo White (2004) o BPMN é uma rede de objetos gráficos cujas atividades e controles de fluxo definem a ordem que cada tarefa é executada. Através de objetos de fluxo, conexão e artefatos o processo é descrito de forma objetiva e de fácil compreensão.

A Figura 7 exemplifica um processo hipotético de solicitação de férias de um funcionário, apresentando as etapas e responsáveis por cada tarefa. Além de destacar o fluxo das atividades o BPMN facilita a compreensão dos pontos de decisão, os caminhos possíveis e o final de cada fluxo.

Figura 7 - Exemplo de fluxo BPMN



Fonte: Sganderla, 2013.

Neste estudo o BPMN foi utilizado pelo pesquisador para exemplificar as formas de trabalho que eram utilizadas pela empresa antes da implantação do novo modelo. A

metodologia também serviu como base para a elaboração do processo, sendo utilizada para desenhar o novo fluxo de trabalho e apresentar aos colaboradores a forma como os projetos seriam conduzidos pós implantação.

3 TRABALHOS RELACIONADOS

Nesta seção serão apresentados trabalhos relacionados ao presente estudo, trazendo exemplos de implantação de metodologias de trabalho, estudo comparativo entre metodologias tradicionais e ágeis, bem como aprimoramento de etapas de processos levando a um novo modelo de trabalho, que seja mais aderente e permita obter métricas mais precisas de execução.

Barden (2013) estudou uma proposta de adequação do processo de desenvolvimento de *software* para a melhoria da estimativa de horas. Em seu trabalho a autora teve como objetivo analisar e propor melhorias no processo de desenvolvimento de *software* na empresa Interact Solutions Ltda. O trabalho proposto visou melhorar a estimativa de horas utilizadas para um projeto e mensurar as atividades dos envolvidos.

A empresa foco do estudo trabalha com o desenvolvimento de *software* e visa um produto de qualidade atendendo aos padrões estabelecidos, utilizando metodologias reconhecidas como RUP, Open UP, modelos CMMI e PMI. Além disso a empresa divide em fases o projeto a ser executado: definição do projeto, análise dos requisitos, construção do projeto, transição do projeto do ambiente de desenvolvimento para o ambiente oficial de produção, onde o *software* é disponibilizado para o cliente.

Através de documentações e entrevistas com a empresa e os colaboradores, a autora propôs melhorias para cada fase de desenvolvimento. Algumas das melhorias propostas são: adoção de práticas da metodologia Scrum; formação de times de trabalho; ajuste no conceito de Use Case Points e pontos de função; utilização de pontos de função como métrica para monitoramento da produtividade da equipe.

Em suas conclusões, a autora indica que foram atingidos os principais objetivos propostos. Iniciando uma base histórica de dados com o projeto do estudo, a autora espera evoluir esse histórico para refinar gradativamente a assertividade do processo da empresa. Para que a melhoria proposta continue evoluindo será necessário que todos os envolvidos auxiliem na disseminação do conhecimento da base histórica, realizando possíveis ajustes identificados nos primeiros projetos que executem este modelo. Somente após alguns projetos e ajustes na metodologia poder-se-á afirmar sobre a assertividade da proposta.

Pereira (2005) elaborou em seu trabalho uma proposta para adaptação de processos de desenvolvimento de *software* baseados no *Rational Unified Process*. O autor teve como objetivo desenvolver um modelo e assertivas estruturais para dar apoio a processos de adaptação.

Este trabalho foi desenvolvido em várias etapas, onde primeiro realizou-se um estudo sobre o RUP, depois o autor identificou os elementos de um processo padrão para desenvolvimento de *software*. Na terceira etapa especificou um meta-modelo para desenvolver um protótipo e na quarta etapa utilizou o RUP como processo padrão a ser adaptado. Algumas de suas disciplinas foram cadastradas no protótipo e simulações foram realizadas a fim de avaliar estruturas assertivas para possíveis inclusões e exclusões de atividades.

Como resultados Pereira considerou que o uso do meta-modelo aplicado com parte do processo RUP foi considerado compatível, embora algumas alterações foram necessárias para completar o processo e sua adaptação. O autor também acredita ser um resultado satisfatório as assertivas estruturais para inclusão e exclusão de atividades. Esta operação é considerada de grande impacto em processos de adaptação, tornando assim possível garantir conformidades no processo resultante em relação ao processo padrão. Isto acontece pois após a exclusão de algumas atividades o autor verificou que as dependências e dificuldades neste ponto de processo excluído também foram eliminados.

Awad (2005) fez uma comparação entre metodologias ágeis e tradicionais no desenvolvimento de *software*, onde seu objetivo foi detalhar as metodologias, revisar métodos, avaliar lacunas existentes nos processos e questionar profissionais da área para analisar suas visões.

Para as metodologias tradicionais, que são baseadas em sequência de etapas, o autor estudou os modelos cascata, processo unificado e método espiral. Os três são baseados em etapas definidas e suas metodologias são caracterizadas por demandarem muito tempo para ficarem prontas, pois seguem de forma burocrática todas as etapas.

Já para um modelo ágil observou que o principal objetivo é contribuir para um desenvolvimento ágil, entre estas metodologias o autor cita os métodos XP, Scrum, metodologias dinâmicas, desenvolvimentos com recursos (FDD) e Adaptive *Software* Development (ASD). Estas metodologias seguem a partir do manifesto ágil alguns ou todos os seus valores e princípios. Conforme o autor o que a de novo na metodologia ágil não é a forma prática que utilizam, mas sim o reconhecimento das pessoas envolvidas como o principal objetivo deste processo.

Awad (2005) conclui a comparação entre os métodos tradicionais e ágeis afirmando que ambos têm seus pontos fracos e fortes. Os principais fatores que afetam a decisão da metodologia a ser utilizada variam de acordo com o projeto, tamanho e risco do *software* a ser elaborado. A principal limitação do método ágil é o tamanho do projeto, sendo ele muito extenso, são necessários mais requisitos, pessoas e mais coordenação envolvida. Já o método tradicional, suporta projetos maiores fornecendo planos, documentações e processos para melhor comunicação e coordenação em grandes grupos.

Com base nos artigos relacionados percebe-se a crescente preocupação das empresas em adotar um processo mais aderente às necessidades reduzindo assim custos e desperdícios e também aproximando o cliente do processo produtivo, com o objetivo de diminuir as alterações posteriores à entrega final. Realizando iterações constantes entre equipe e cliente o projeto tem um ganho considerável de assertividade em relação ao escopo. Ambos os artigos citados visam a melhoria do processo produtivo e, consequentemente, maior eficácia no momento da entrega final, tanto em custos, quanto em escopo e prazo.

A pesquisa realizada por Barden (2013) pode ser comparada a este trabalho em dois pontos, a adoção de práticas da metodologia Scrum e formação de times de trabalho. Além disso a autora realizou um ajuste no conceito de *Use Case Points* já utilizado pela empresa e aplicou o uso de pontos de função como métrica para monitoramento da produtividade da

equipe, pontos que não serão adotados nesta melhoria. Pereira (2005) estudou o conceito de desenvolvimento RUP, propondo a implantação de um novo modelo de trabalho baseado nas principais técnicas da metodologia, enquanto este trabalho baseou-se neste modelo para fazer a comparação com os métodos ágeis, que serviram como base para a elaboração do novo processo.

O trabalho de Awad (2005) foi o mais compatível com a atual pesquisa, pois ele também fez uma comparação entre os métodos tradicionais e ágeis de desenvolvimento com objetivo de detalhar e avaliar os seus conceitos. Além de outras metodologias tradicionais e ágeis, o autor comparou as mesmas abordadas por esse trabalho: Cascata (tradicional), RUP (tradicional), XP (ágil) e Scrum (ágil), porém o estudo parou na comparação dos métodos, não resultando em nenhum novo processo implantado.

Assim, este estudo baseia-se na evolução das metodologias de trabalho e da necessidade de sucesso dos projetos, o que é impactante para empresas e clientes, uma vez que o projeto poderá ser entregue conforme as suas especificações, dentro do prazo acordado e continuar sendo rentável para organização que o desenvolveu. Entende-se que com as metodologias tradicionais de desenvolvimento o cliente estava desassistido ao longo de todo o desenvolvimento e este fato pode ser apontado como um dos principais pontos de fracasso. O presente trabalho identificou pontos de melhorias no processo de desenvolvimento utilizado pela empresa e, com base na pesquisa propôs uma nova forma de trabalho, mais aderente à realidade do cenário atual, considerando a maior participação do cliente como sendo um ponto fundamental para sucesso das entregas de *software*.

4 METODOLOGIADA PESQUISA

Este capítulo apresenta os pressupostos metodológicos utilizados no trabalho e na implantação da nova forma de trabalho na empresa Retta Tecnologia da Informação. Segundo Silva e Menezes (2005), a metodologia da pesquisa descreve como será desenvolvimento do trabalho em questão, definindo métricas que serão utilizadas para concluir se a aplicação teve êxito. É o momento de alinhar como pesquisa, implantação e prática.

A natureza deste trabalho pode ser considerada como pesquisa exploratória. 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 ágeis e tradicionais, buscando um entendimento sobre as melhores práticas dos conceitos e propor uma metodologia mais eficiente na produção de sistemas na empresa Retta Tecnologia da Informação.

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. Este tipo de pesquisa normalmente utiliza formas relativas à pesquisa experimental, como por exemplo um objeto de estudo, no qual são identificadas variáveis que participam do processo, visando, ao final do estudo, a pratica na própria realidade.

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

> Com o desenvolvimento e a legitimidade percebida tanto da pesquisa qualitativa quanto da pesquisa quantitativa nas ciências sociais e humanas, a pesquisa de métodos mistos, empregando a combinação de abordagens quantitativas e qualitativas, ganhou popularidade. Essa popularidade deve-se ao fato de que a metodologia da pesquisa continua a evoluir e a se desenvolver, e os métodos mistos são outro passo adiante, utilizando os pontos fortes das pesquisas qualitativa e quantitativa (Cresswel, 2010. p. 238).

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. Nesta pesquisa o pesquisador desenvolve conceitos, ideias e entendimentos, a partir de padrões ao invés de comprovar teorias e modelos pré-concebidos. 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).

Creswell (2010) mostra a pesquisa qualitativa como uma abordagem diferente de investigação acadêmica, com métodos de coleta e análise de interpretação de dados, com procedimentos qualitativos que baseiam-se em elementos de texto e imagem. Por outro lado descreve que os métodos quantitativos são de extrema importância, pois é a parte mais concreta e especifica de uma proposta. Para ele, um projeto de levantamento apresenta descrição quantitativa ou numérica sobre as tendências e opiniões do estudo. A partir de alguns resultados já obtidos, o pesquisador faz afirmações sobre o assunto em questão.

Para esta pesquisa utilizou-se como fonte de dados os projetos de *software* desenvolvidos pela empresa antes e depois da implantação do novo processo. Segundo Gerhadt e Silveira (2009), esse modelo caracteriza-se como pesquisa de campo, por fazer investigações ao caso, além da pesquisa bibliográfica, realiza-se coleta de dados e recursos diferenciados para estudar e analisar o assunto.

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.

Este estudo de caso objetivou implantar uma metodologia de desenvolvimento mais eficiente na empresa Retta Tecnologia da Informação. A forma antiga de trabalho 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.

A pesquisa iniciou com uma análise comparativa entre as metodologias tradicionais e ágeis mais comuns do mercado, gerando assim o embasamento teórico necessário para que os principais gargalos do processo de desenvolvimento fossem identificados. O processo utilizado até a implantação tinha pontos positivos que foram mantidos e outros que não estavam mais sendo executados, mas foram considerados importantes para a nova forma de trabalho. Considerando os processos e as técnicas ágeis, mais a expectativa da empresa com o novo processo, o estudo desenhou e implantou um novo modelo de trabalho.

Esse novo modelo foi executado em projetos pilotos e resultou em uma pesquisa qualitativa com analise dos benefícios da implantação, ouvindo clientes e colaboradores da equipe. A forma de coleta de dados desta etapa foi com base em um questionário elaborado pelo autor. Por último foi feita a comparação quantitativa dos resultados, mensurando indicadores de prazos e esforços obtidos e comparando estes com as execuções de projetos passados.

5 ANÁLISE DO CENÁRIO ORGANIZACIONAL

Este capítulo apresentará o cenário identificado no momento em que o estudo iniciou, apresentando de forma sucinta a história da empresa, segmentos que atende e os projetos que executa. Além da composição da equipe, os próximos tópicos descrevem o produto foco deste trabalho e o processo utilizado para o seu desenvolvimento. O processo alternativo utilizado em determinados projetos de fábrica de *software* com tamanho grande ou escopo incerto também será explicado.

5.1 A empresa

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. Ao longo dos três anos seguintes a empresa trabalhou na criação de um sistema para gestão de granjas suinícolas, lançando-o como seu primeiro produto no ano de 2008. No ano de 2010 a empresa ingressa como incubada na INOVATES (Centro de Inovação Tecnológica), incubadora empresarial da Univates, iniciando o desenvolvimento de um produto de gestão de microempresas, chamado RETTA Comercial.

No ano de 2011 a empresa alcança um novo patamar de mercado, lançando o serviço de *outsourcing* e 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 e os trabalhos de *outsourcing* encerrados para que a empresa pudesse manter o foco no produto Demander e nos desenvolvimentos contratados pela Fábrica de *Software*.

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.

5.2 Fábrica de *software*

A fábrica de *software* tem como foco o desenvolvimento de sistemas específicos para clientes, ou seja, contratações pontuais que acabam no momento da entrega do *software*, sob o qual a empresa não mantém nenhum direito. Neste segmento de fábrica de *software* a Retta possui inúmeras experiências de desenvolvimento, atendendo diferentes segmentos de mercado, nas mais variadas tecnologias e linguagens de programação, como por exemplo: PHP, Java, C#, Android, iOS, HTML, CSS e Java Script; além de realizar integrações com bancos de dados Oracle, SQL Server, MySQL, Firebird e Postgres. A empresa também realiza integrações com hardwares terceiros, como impressores térmicas, coletores de dados, leitores RFID e outros.

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.

5.3 O produto DEMANDER

O produto da empresa é uma plataforma completa para gestão de vendas em indústrias e distribuidoras, automatizando o trabalho das equipes comerciais internas e externas. O Demander é um *software* integrado que possui plataformas *mobile* e *web*, atingindo usuários de perfis bastante distintos. Na Figura 8 pode-se ver algumas das funcionalidades do aplicativo para o vendedor externo, que tem o objetivo de facilitar e agilizar o seu dia de trabalho, fornecendo informações e estatísticas importantes para o evento da venda.

Figura 8 - Tela principal do aplicativo

Fonte: versão de demonstração disponível na conta da Retta na Play Store, 2017.

O sistema funciona de maneira *off-line*, enviando e recebendo dados somente quando houver conexão com a internet. Na Figura 9 observa-se uma tela onde o usuário pode realizar um novo pedido ou então consultar o relatório periódico das vendas, fazendo filtros por período,

por status ou tipo de pedido e por cliente. Com o aplicativo o vendedor pode concentrar seus esforços na sua atividade principal, que é a venda para o cliente, o resto do trabalho fica a cargo do sistema, pois ele garante que todas as regras de negócio e políticas da empresa sejam respeitadas.

Figura 9 – Listagem de pedidos emitidos pelo sistema

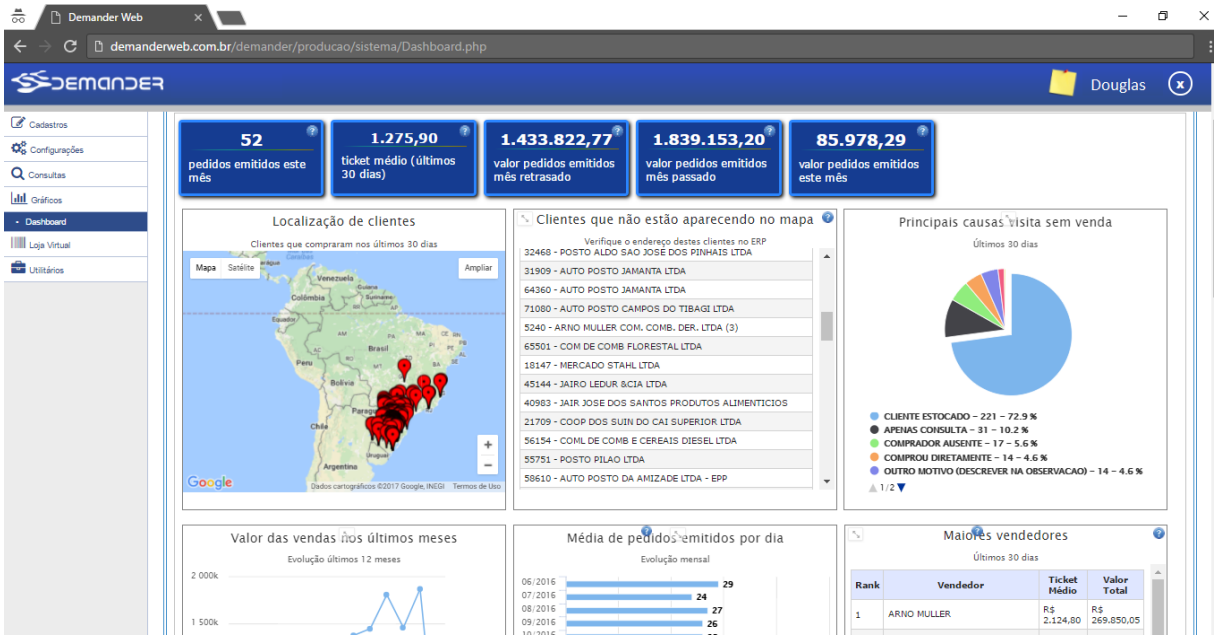
Fonte: versão de demonstração disponível para download na Play Store, 2017.

Uma versão de demonstração do produto pode ser encontrada na Play Store (loja de aplicativos Android da Google) pesquisando pelo nome ‘Demander’, ou diretamente no link https://play.google.com/store/apps/details?id=br.com.retta.demander&hl=fr. A versão do aplicativo para dispositivos iOS está em desenvolvimento no momento e tem previsão de lançamento para março de 2018, podendo ser encontrada na APP Store (loja de aplicativos iOS da Apple) após esse período.

A plataforma de vendas Demander também possui um painel web (chamado de DemanderWeb) que possibilita ao gestor fazer o controle dos dados e informações enviadas pela equipe. O site também é utilizado por empresas que não possuem um sistema interno

(ERP), permitindo que os cadastros base sejam inseridos de forma rápida e fácil. O Demander pode ser integrado ao sistema de gestão das empresas, dessa forma o cadastro de novos clientes ou pedidos emitidos pelos vendedores entram de forma automática no ERP. A Figura 10 apresenta a tela principal do sistema, uma *dashboard* com alguns gráficos, relatórios de acompanhamento e indicadores de resultados. Uma versão de demonstração pode ser solicitada através do link http://demanderweb.com.br.

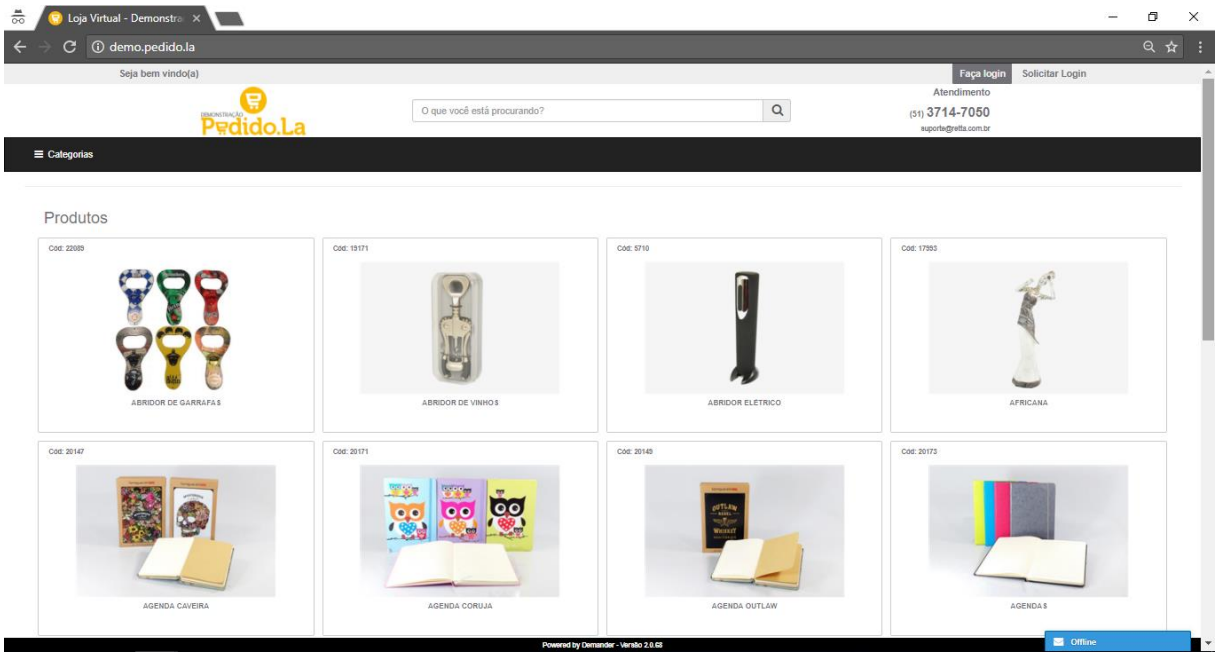
Figura 10 – Demanderweb: site para gestão das informações



Fonte: ambiente de demonstração do sistema web (http://demanderweb.com.br)

O sub produto Pedido.La é uma plataforma B2B completamente integrada ao processo do Demander, utilizada pelas empresas para disponibilizar os seus produtos de forma on-line, sem que seja necessária a presença física de um vendedor. Dessa forma os clientes das empresas que usam o Demander podem acessar o canal direto de compra, realizando pedidos de forma simplificada a qualquer hora e de qualquer lugar. A Figura 11 traz o exemplo de loja B2B fictícia, que pode ser acessada em http://demo.pedido.la/. Dessa forma é possível validar o processo da compra, onde diferente de um e-commerce tradicional, o cliente precisa estar previamente cadastrado e receber permissão para efetuar pedidos pelo sistema.

Figura 11 – Plataforma de vendas B2B – Pedido.La.



Fonte: loja demo na plataforma B2B (http://demo.pedido.la/)

O desenvolvimento de todos os ambientes da plataforma de vendas Demander é feito e mantido pela equipe da Retta. Como já foi descrito neste trabalho, 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.

5.4 Processo anterior ao desenvolvimento do presente trabalho

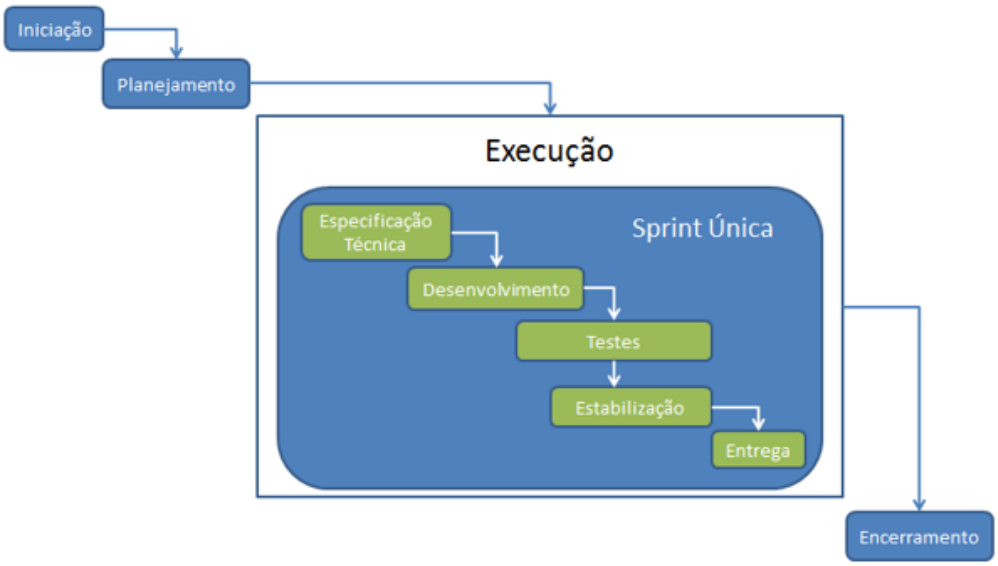
A Retta iniciou no ano de 2014, um trabalho de padronização de seus processos, obtendo a certificação nível G do MR-MPS-SW (MPS.BR). Essa certificação é uma melhoria da qualidade do processo de desenvolvimento, garantindo que o desenvolvimento é orientado a projetos e requisitos, fatores que antes não 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.

A Retta utiliza o Modelo de Referência MPS para *Software* (MR-MPS-SW), que segue os requisitos para modelo de referência descritos na ISO/IEC 15504-2 declarando o propósito e os resultados esperados de sua execução. A capacidade do processo é a caracterização da habilidade para alcançar os objetivos, atuais e futuros; estando relacionada com o atendimento aos atributos de processo associados aos processos de cada nível de maturidade (SOFTEX, 2016).

O processo de trabalho da empresa segue dois modelos elaborados e concebidos através da implantação da certificação MPS.BR. Um deles 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). Na Figura 12, retirada da documentação gerada pelo processo de implantação do MPS.BR, pode-se observar que o ciclo de desenvolvimento respeita a sequência sugerida no modelo tradicional, onde cada fase deve ser concluída para virar um AP (Atributo de processo) para que a próxima fase inicie.

Figura 12- Fluxograma de trabalho atual – modelo cascata.



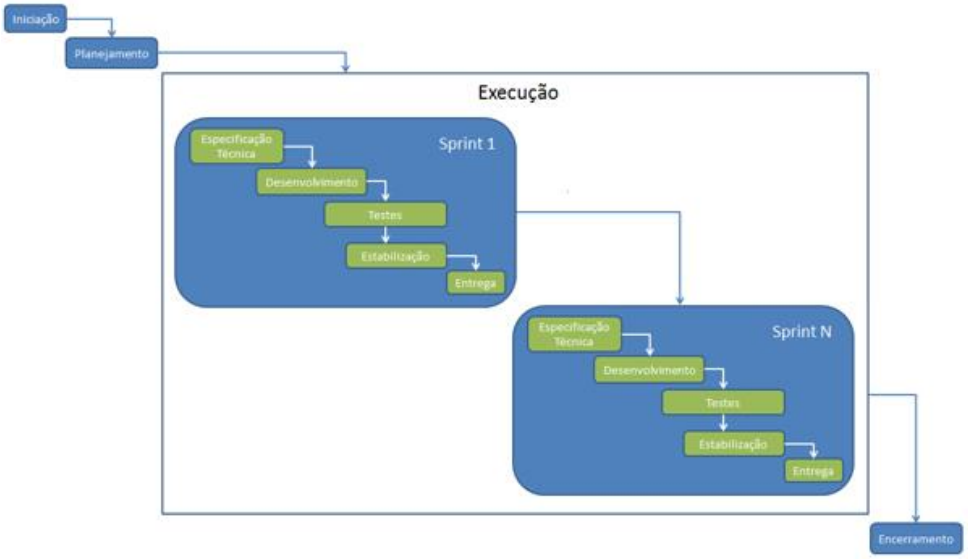
Fonte: adaptado da documentação de processos Retta, 2014.

Assim como observado por Royce (1970) os documentos de implantação da certificação MPS.BR (Retta2014) também destacam que o ciclo de vida cascata não é aconselhado nos seguintes cenários:

- Sistemas muito grandes (mais de 200 horas de desenvolvimento);
- Sistemas com requisitos vagos ou não mapeados;
- Sistemas com mudanças frequentes em requisitos ou tecnologias;

Para os casos de sistemas grandes, requisitos vagos ou mudanças frequentes, é adotado um fluxo de trabalho incremental semelhante ao modelo cascata, com a diferença que a fase de execução permite mais de um ciclo de *sprint* (Figura 13). Este modelo tem características da metodologia ágil, mas ainda é enraizado na metodologia tradicional. As fases são as mesmas independente do ciclo de vida, o que pode mudar é a frequência e sequência delas.

Figura 13 – Fluxograma de trabalho atual – modelo incremental



Fonte: documentação de processos Retta (2014).

Com o tempo alguns pontos mais burocráticos do processo foram sendo deixados de lado, mas basicamente a forma de trabalho utilizada até a implantação segue conforme documentado. A seguir são descritas de maneira resumida as fases e etapas dos processos utilizados pela Retta:

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 *sprint* única ou de forma incremental, com mais de uma *sprint*. 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.

5.4 Problemas e melhorias identificadas no processo

Para a certificação do MPS.BR a empresa definiu e homologou o processo de trabalho utilizado até hoje. Esse processo é fortemente baseado nas metodologias tradicionais de desenvolvimento de *software*, podendo ser comparado principalmente aos modelos Cascata e RUP, pela forma como as etapas estão documentadas e são executadas. Com base nas metodologias abordadas pelo presente trabalho, o pesquisador identificou os 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.

6 RESULTADOS E DISCUSSÃO

Neste capítulo serão apresentadas as alterações propostas para as melhorias identificadas, o detalhamento do novo processo proposto e os resultados da aplicação do novo modelo. A primeira parte descreve os pontos de melhorias identificados no processo de desenvolvimento utilizado pela empresa até o momento deste trabalho. A segunda apresenta o novo processo proposto, aplicando conceitos e técnicas das metodologias ágeis para aumentar a eficácia no desenvolvimento. Por último, 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).

6.1 Alterações propostas para as melhorias

Essa seção apresenta as melhorias propostas pelo autor para minimizar os problemas que foram identificados no processo utilizado pela empresa. Após definir um novo processo, foram executados projetos pilotos que permitiram avaliar os resultados da implantação da nova metodologia de trabalho. A seguir será apresentada uma revisão dos problemas apontados no capítulo anterior e as ações e atividades desenvolvidas para implementação das melhorias propostas.

Documentação e aprovação de requisitos: para que esta etapa tenha prazo e responsável, definiu-se que a especificação de um requisito deve ser uma demanda de versão, ou seja, a

documentação e validação de cada funcionalidade, será feita dentro de uma *sprint* de desenvolvimento. A forma de validação de um requisito funcional será em formato de arquivo de texto que é enviado por e-mail ao cliente e, após os ajustes e validação da implementação esse requisito é documentado na ferramenta de gestão da empresa e as tarefas técnicas são especificadas dentro do *backlog* de produto, para que sejam incluídas em versões futuras do projeto.

Processo de testes documentado: mesmo que a etapa de testes já seja executada de alguma forma, foi definido que haverá um responsável dedicado para esta tarefa. Foi criado um documento com as funcionalidades mínimas que devem ser testadas em cada versão, além do escopo que foi implementado, pois como é um produto de atualização constante, as implementações podem impactar em funcionalidades já liberadas anteriormente. Esse profissional passou a ser responsável pela evolução da rotina mínima, documentando novas implementações que devem ser consideradas como uma funcionalidade mínima nas próximas versões.

Fluxograma de trabalho para projetos de produto: um novo fluxo de trabalho foi proposto com base nas metodologias ágeis estudadas, contemplando fases já utilizadas pela empresa e outras levantadas nesta melhoria. O fluxograma foi dividido em três macroprocessos, apresentados para os interessados e está acessível na documentação interna da empresa. O fluxo principal contempla desde a composição do escopo a partir do *backlog* de produto até a entrega para o cliente, o processo de desenvolvimento define a sequência e as etapas da fase de execução e o processo de validação dos requisitos, explica a sequência da especificação, aprovação e documentação.

Ciclos de entrega com prazo definido: um dos pontos mais problemáticos do antigo processo eram os atrasos constantes, que acarretavam em insatisfação dos clientes e funcionalidades estocadas no desenvolvimento. Considerando esta realidade, o novo processo de desenvolvimento é programado sobre *sprints* de prazo fixo, sendo que o cliente e a equipe sabem o dia em que a entrega será realizada. Para evitar atraso nas entregas, o escopo de desenvolvimento é priorizado e, caso alguma tarefa não possa ser concluída dentro do tempo previsto, a funcionalidade de menor prioridade deixa o escopo e volta para o *backlog* do produto.

Alternância de pessoal entre os projetos: montou-se uma equipe fixa para os projetos de produto, que conta com desenvolvedores, gerente, analista e testador, o que diminui o risco de destes profissionais serem requisitados em outro projeto. Além disso, como o escopo de desenvolvimento é reduzido (prazo de duas semanas) é mais fácil programar a agenda dos envolvidos, possibilitando que um remanejo inevitável possa ser adiado para a próxima *sprint*, onde o tempo de afastamento pode ser considerado na estimativa.

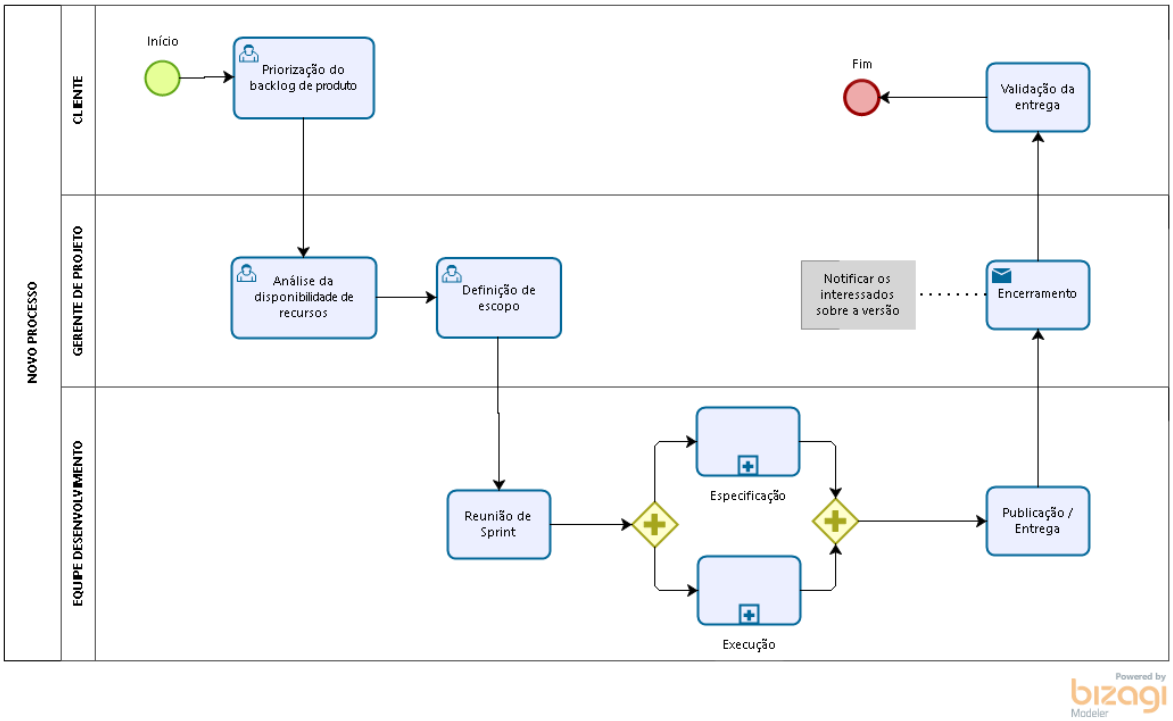
Boa comunicação entre os interessados do projeto: a comunicação entre os integrantes da equipe de desenvolvimento é fundamental e deve ser incentivada, por isso o novo processo prevê etapas semelhantes ao que é proposto na metodologia Scrum, nas reuniões de s*print* e diárias. Também é importante aproximar o cliente do projeto, dividindo com ele as decisões de prioridade e forma de atuação, isso permite que os desvios sejam percebidos com maior antecedência. A melhoria de validação do escopo e o envio de relatórios periódicos, aumentou essa interação, possibilitando mais transparência do andamento da versão.

6.2 Novo modelo proposto

Após analisar metodologias ágeis de desenvolvimento e compará-las com o ciclo utilizado até o momento desta intervenção, foi elaborado um modelo que pode contribuir para minimizar ou resolver os problemas citados. Foram utilizadas 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.

Para descrever o fluxo de desenvolvimento proposto para os novos projetos de produto, foi utilizada uma ferramenta para desenho de diagramas BPMN, denominada Bizagi. A Figura 14 apresenta o novo processo, conforme o fluxo proposto por este trabalho, identificando os papéis dos responsáveis por cada etapa.

Figura 14 – Novo processo de trabalho



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 (GP), que estima dentro do prazo de cada entrega a disponibilidade em horas dos membros da equipe, mensurando o tempo total disponível para desenvolvimento. Por exemplo: se para uma versão de projeto o GP tiver 2 desenvolvedores dedicados ele irá considerar 8 (horas de trabalho por dia) X 10 (número de dias úteis da *sprint*) X 2 (número de desenvolvedores) = 160 horas disponíveis para desenvolvimento. São consideradas apenas horas dos desenvolvedores, sendo que as horas de análise, gerenciamento e testes não entram na estimativa inicial.

Seguindo o exemplo de estimativa, supondo que a disponibilidade para uma versão seja de 160 horas o gerente irá reservar 20% deste tempo para não previstos, ou seja, uma margem

na estimativa que pode ser consumida por riscos, problemas ou então tarefas de desenvolvimento não consideradas na priorização e que recebem urgência em algum momento. Este percentual ainda é alto e deve ser reduzido com o tempo, mas nos projetos pilotos executados ele foi importante para a assertividade das previsões. Assim, 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* (interessados no projeto) o plano de projeto, contendo o escopo da versão, as estimativas em horas e o percentual de não previstos. A partir deste momento sabe-se que qualquer tarefa de desenvolvimento que for incorporada ao projeto deverá ser contabilizada como não previsto. 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.

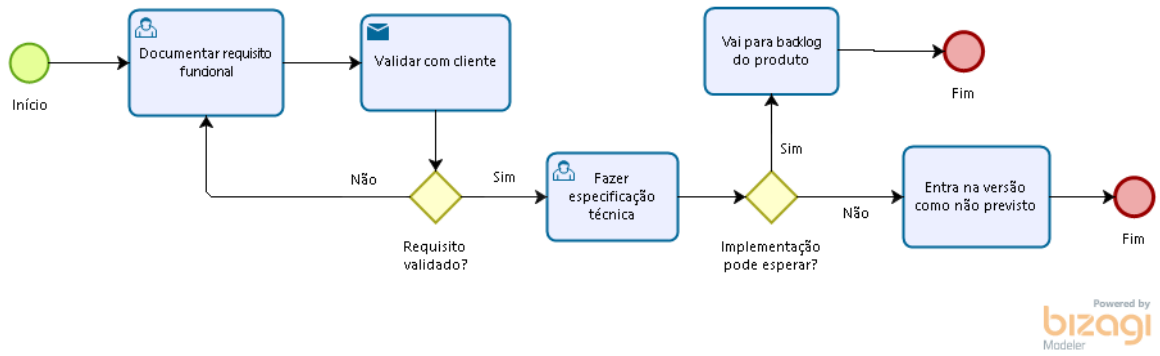
A reunião de *sprint*, é o marco inicial da execução, onde o responsável apresenta à equipe as definições do projeto, para que assim possam assumir o comprometimento da entrega junto com o cliente, que deve ser incentivado a participar. Neste momento são discutidas dificuldades técnicas do escopo e as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

As etapas de especificação e execução são compostas por sub processos que serão detalhados no decorrer do trabalho, mas são 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.

A priorização do *backlog* de produto e a reunião de *sprint*, são dois pontos novos para os processos da Retta, enquanto a análise da disponibilidade de recursos e a formalização do encerramento, são pontos já executados outrora, mas que deixaram de fazer parte do processo por motivos específicos. As demais etapas são corriqueiras para os colaboradores, tendo sofrido apenas ajustes na forma como são encaradas.

A Figura 15 descreve o sub processo 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. Como o trabalho de análise deve estar dentro das atividades de um projeto para que possam ser mensuradas e gerenciadas, o novo processo prevê que cada *sprint* documentará de forma funcional e técnica as atividades que serão executadas em versões futuras.

Figura 15 – Sub processo da etapa de especificação



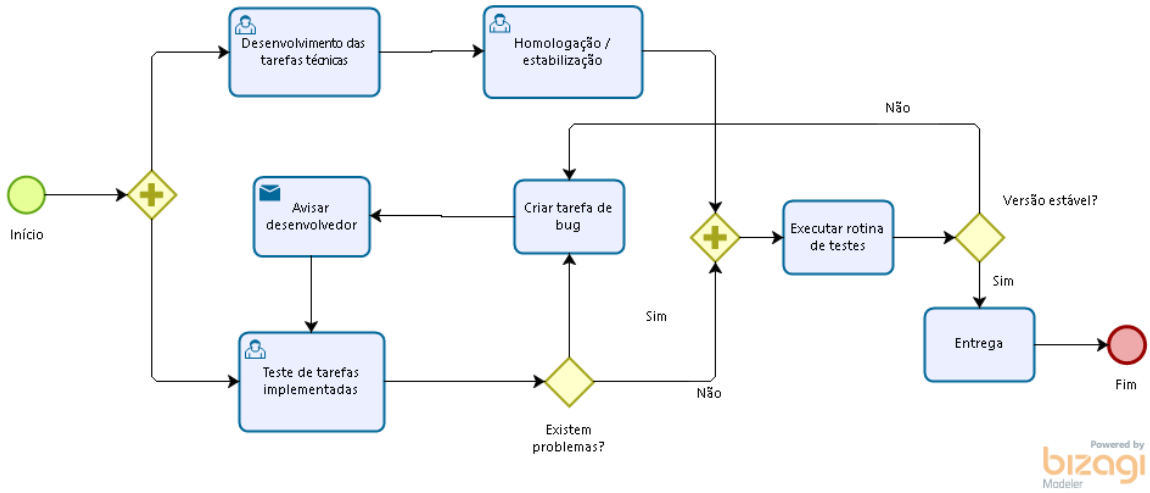
Fonte: elaborado pelo autor, 2017.

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*. Por ser uma etapa paralela, o tempo de especificação não é contabilizado na estimativa de horas de desenvolvimento, mas o intuito de incluir esse ponto no processo é envolver o cliente com a priorização e validação destas análises. Inicialmente, o analista documenta o requisito de forma funcional, ou seja, em linguagem de negócio, para buscar a concordância do cliente sobre o entendimento.

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 Retta em algum momento, mas deixou de ser executada. Após essa implementação a rotina volta a compor o fluxo de desenvolvimento com uma nova forma de execução, sendo considerada como ideal para o novo modelo proposto.

No sub processo de execução, demonstrado na Figura 16, 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 16 – Sub processo 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. Assim, logo que o desenvolvimento das tarefas técnicas for concluído, a equipe já pode começar a trabalhar nas tarefas de *bug*.

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.

6.3 Resultados da aplicação do novo modelo

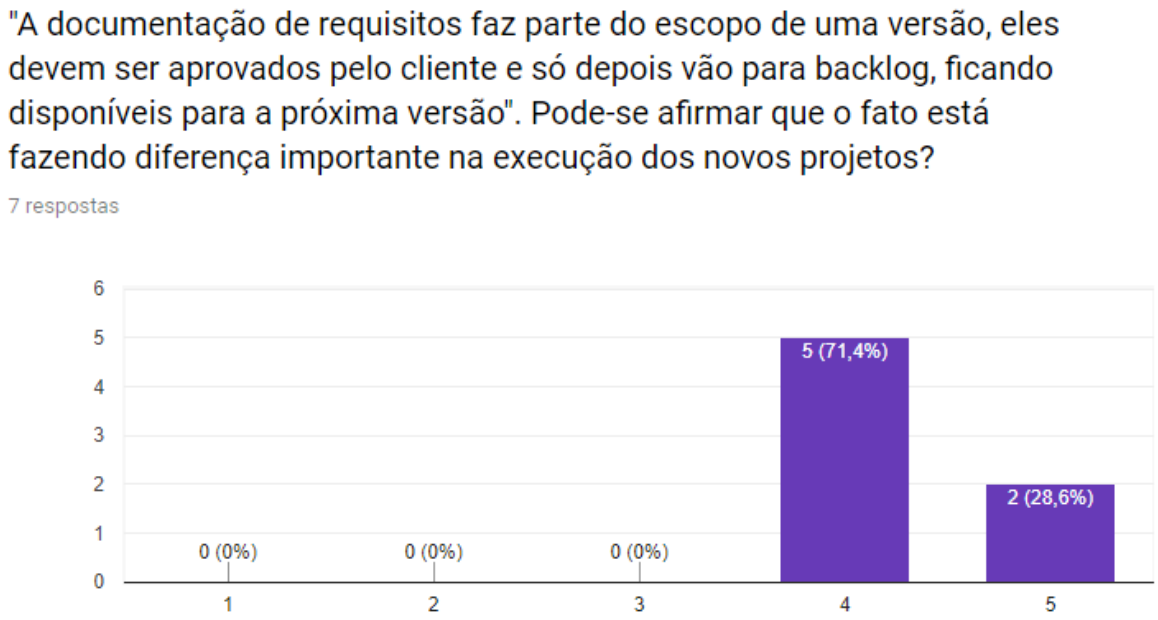
Esta seção apresenta os resultados relacionados aos projetos pilotos que foram desenvolvidos utilizando o novo processo proposto. Serão apresentados os resultados e análises relacionados a cada uma das melhorias propostas. Os resultados incluem análises qualitativas, quantitativas, e a comparação com projetos similares executados anteriormente, quando aplicável. Considerando a abordagem qualitativa, foi realizada 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 atrelada a um ponto de melhoria e suas respostas estão avaliadas nas subseções que seguem.

6.3.1 Documentação e aprovação de requisitos

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 17 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.

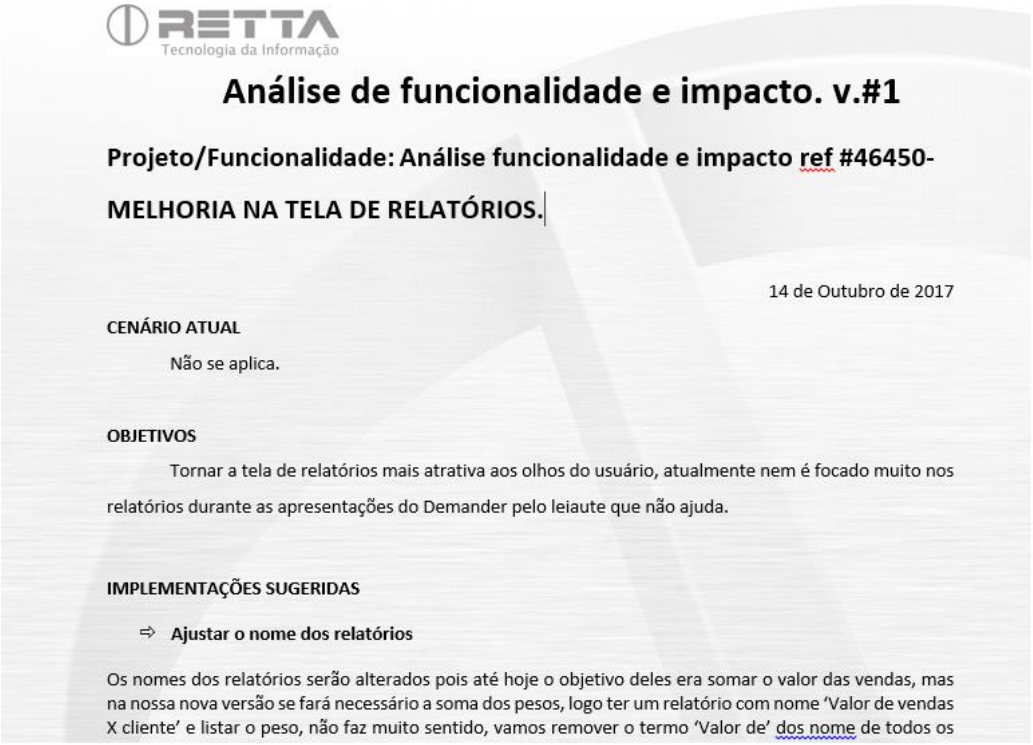
Figura 17 – Questão sobre documentação e aprovação de requisitos



Fonte: elaborado pelo autor, 2017.

Todas as solicitações de novas funcionalidades demandadas por clientes ou pela direção da empresa são documentadas em forma de requisito funcional, a nível de negócio e devem ser validados pelo solicitante. O documento (FIGURA 18) é acompanhado por um número interno, onde será registrado no sistema de gerenciamento e pelo número da versão, podendo evoluir e sofrer alterações até que a ideia esteja entendida por ambos os lados. Essa comunicação é feita exclusivamente por e-mail, por ser uma ferramenta padrão utilizada pela Retta e por manter registro formal das conversas.

Figura 18 – Exemplo de documento de requisito enviado ao cliente para validação



RETTA
Tecnologia da Informação

Análise de funcionalidade e impacto. v.#1

Projeto/Funcionalidade: Análise funcionalidade e impacto ref #46450-

MELHORIA NA TELA DE RELATÓRIOS.

14 de Outubro de 2017

CENÁRIO ATUAL

Não se aplica.

OBJETIVOS

Tornar a tela de relatórios mais atrativa aos olhos do usuário, atualmente nem é focado muito nos relatórios durante as apresentações do Demander pelo leiaute que não ajuda.

IMPLEMENTAÇÕES SUGERIDAS

⇨ **Ajustar o nome dos relatórios**

Os nomes dos relatórios serão alterados pois até hoje o objetivo deles era somar o valor das vendas, mas na nossa nova versão se fará necessário a soma dos pesos, logo ter um relatório com nome 'Valor de vendas X cliente' e listar o peso, não faz muito sentido, vamos remover o termo 'Valor de' dos nome de todos os

Fonte: ferramenta de gerenciamento de projetos, tarefa #46450. Retta, 2017.

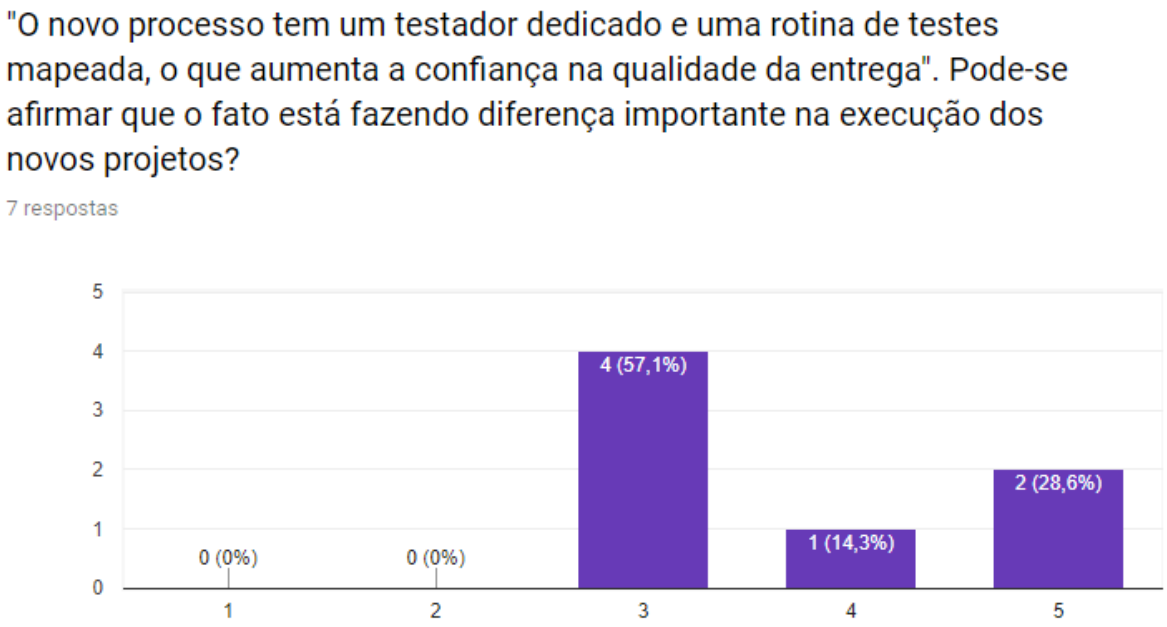
Esta melhoria não teve comparação com outros projetos anteriores pois a documentação dos requisitos foi suprimida das versões do Demander a algum tempo, tendo sido feita de maneira informal e sem nenhum registro de interação com o solicitante. Uma característica importante é que, como a especificação não era previamente criada, não há registro de requisitos que são enviados ao *backlog*, ou seja, toda análise feita para uma versão era necessariamente executada dentro da mesma.

6.3.2 Processo de testes documentado

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 19, a maioria da equipe manifestou opinião neutra (não concorda e nem discorda) sobre a pergunta, 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.

Figura 19 – Questão sobre rotina de testes de versão



Fonte: elaborado pelo autor, 2017.

Inicialmente a rotina de testes mínimos foi criada em um documento de texto (FIGURA 20), que está sendo desenvolvido pelo testador responsável, em conjunto com demais membros da equipe. Como esta implantação executou apenas três projetos pilotos não foi possível completar esta rotina, mas o objetivo é concluir a documentação para posteriormente registrar todos os testes dentro da ferramenta de gestão. Até que seja devidamente registrada a rotina ficará disponível em um ambiente de documentos compartilhados da empresa.

Figura 20 – Documento da rotina de testes mínimos a serem executados em uma versão

Rotina de testes mínimos de versão

O processo de teste deve ser feito em ambiente de homologação, salvo exceções muito pontuais.

Os testes serão divididos em critérios, para ser possível ser executado em partes e mesmo por pessoas diferentes.

1. **Testar telas**

Este teste tem por objetivo encontrar erros em telas, temos algumas características em telas que são baseadas em configurações, então iremos elencar aqui as configurações que precisam ser ativadas ou desativadas para efetuar os testes mínimos para este critério de teste ser aprovado.

a) Logar com a versão demo (email: suporte.demander@retta.com.br) com a versão anterior, cadastrar cliente e fazer pedido, logo em seguida atualizar para a nova versão. não pode dar problema de migração.

b) Fazer este procedimento de atualizar versão para as últimas 3 versões. Como não é possível sincronizar com esta empresa, o objetivo é que alguém que esteja testando não de problema ao atualizar o sistema.

Fonte: documentação interna de projetos. Retta, 2017.

A nova proposta para os testes também não pode ser comparada a projetos anteriores pois, mesmo que fosse executada de alguma maneira, os devidos registros não eram feitos. Logo, a empresa sabe que testes são feitos em cada versão, mas não é possível mensurar qual o percentual de cobertura sobre o escopo e nem a relação custo/benefício deste esforço. A finalidade desta nova rotina será expor à direção e demais *stackholders* a importância de ter um profissional dedicado e responsável pela qualidade.

Contudo, é possível considerar como um indicador, 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.

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

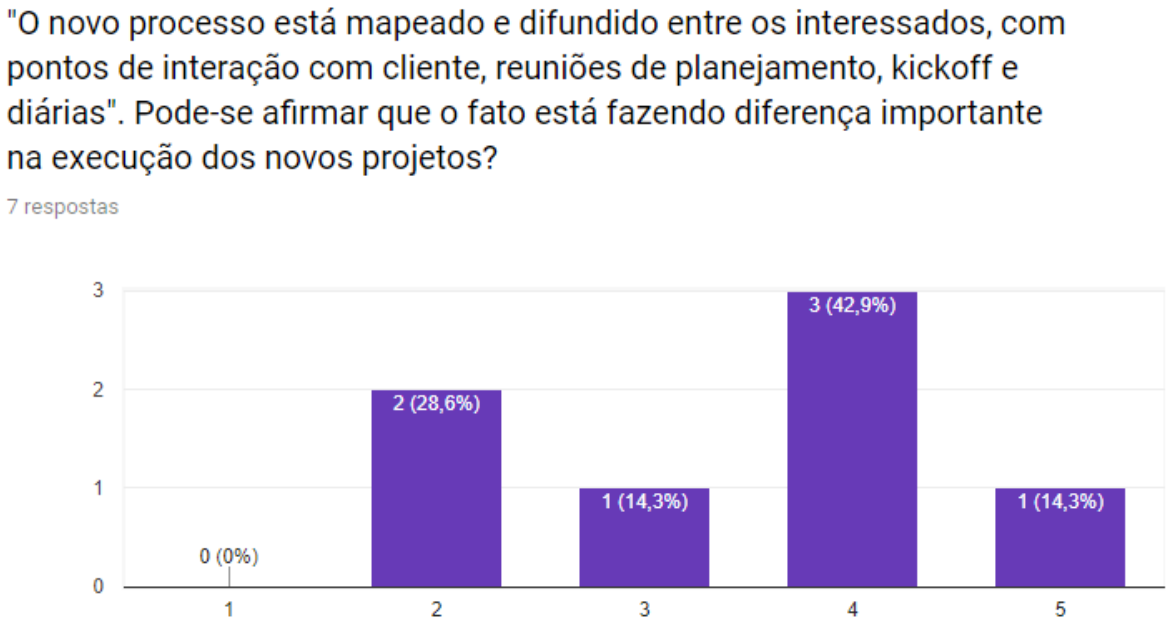
Fonte: elaborado pelo autor, 2017.

Pode-se observar claramente que as três versões que foram desenvolvidas antes da implantação deste trabalho, tiveram mais necessidades de ajustes, 8 no total, do que os projetos piloto, que totalizaram apenas 5 lançamentos de correção. Não é possível afirmar que esse avanço se deve unicamente a melhoria da rotina de testes, mas é visível que houve uma evolução na qualidade das entregas.

6.3.3 Fluxograma de trabalho para projetos de produto

O fluxo de trabalho dos projetos de produto destacou-se por ser o ponto de maior discordância entre os entrevistados (FIGURA 21), 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.

Figura 21 – Questão sobre o novo fluxo do processo



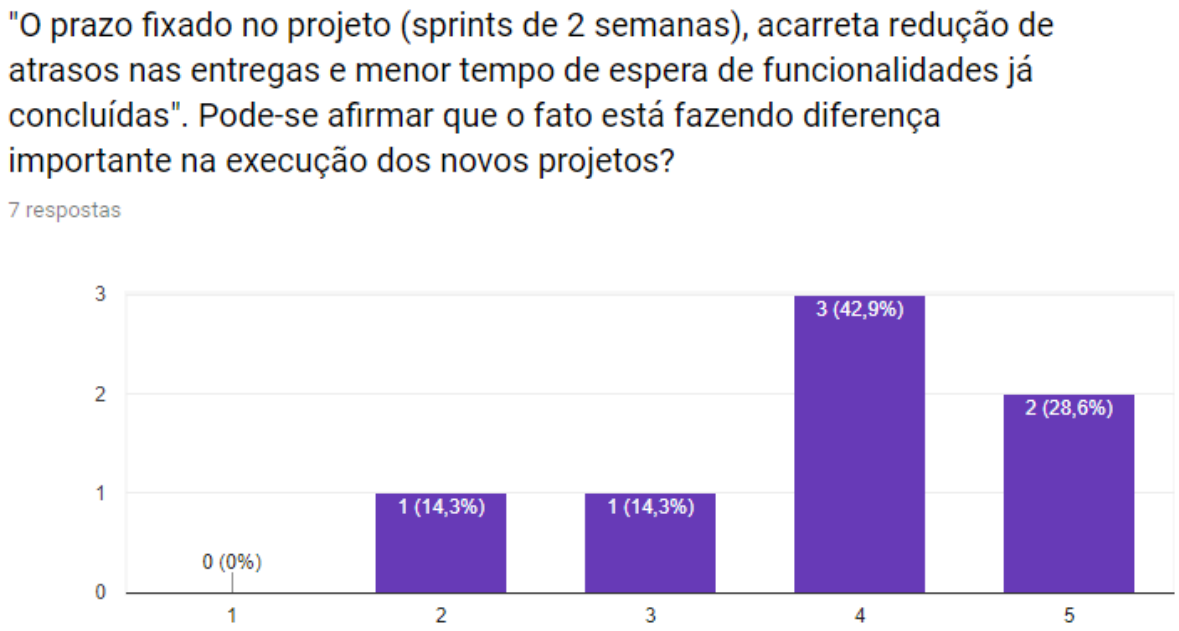
Fonte: elaborado pelo autor, 2017.

Apesar dos objetivos de prazo e qualidade terem sido atingidos com o novo fluxo, esta melhoria é a que mais depende da avaliação qualitativa, pois não é possível mensurar de maneira comparativa e tangível. Com base na avaliação dos entrevistados, neste ponto a implantação deixou a desejar, mas será criado nas próximas semanas um plano de ação para difundir entre os colaboradores o fluxo e a documentação gerada sobre ele.

6.3.4 Ciclos de entrega com prazo definido

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, conforme Figura 22. A média das respostas foi de 3,86 pontos, com um desvio padrão de 1,07, o que mostra que o entendimento não é unânime, por isso será necessário discutir um pouco mais esse assunto com a equipe e clientes para ver se é possível melhorar mais o processo.

Figura 22 – Questão sobre o prazo de entrega fixado



Fonte: elaborado pelo autor, 2017.

Com a definição de *sprints* de duas semanas de trabalho, a frequência cíclica de entregas possibilita um planejamento padronizado de escopo e recursos. Esse prazo de versão é experimental e pode ser ajustado com o tempo, mas o que se pode constatar é que, com a relação de custo fixo das etapas de priorização de escopo, execução das rotinas de teste, reuniões e publicação de versão, esse é o tempo mínimo que o processo suporta. É possível que a empresa faça testes com *sprints* de três semanas para poder avaliar qual das duas formas apresentou melhores resultados, mas um prazo maior, por exemplo quatro semanas, é muito longo e implica em riscos 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. Para o cliente que fez uma solicitação relativamente simples é difícil explicar um tempo de espera de tão longo, pois como o planejamento era falho o processo não garantia que uma demanda seria incluída no próximo projeto, logo, não era possível estimar uma data para a entrega e, quando feito, corria-se o risco de atrasar, sem nova data prevista.

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.

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. O que pode ocorrer é que implementações de menor prioridade retornem ao *backlog* do produto. Por mais que o atraso em dias não pareça um número intimidante, se comparado ao número de dias previstos para o desenvolvimento, percebe-se que o atraso foi de no mínimo 30%, na versão 3.6. As outras versões tiveram um atraso superior a 50% em relação ao previsto.

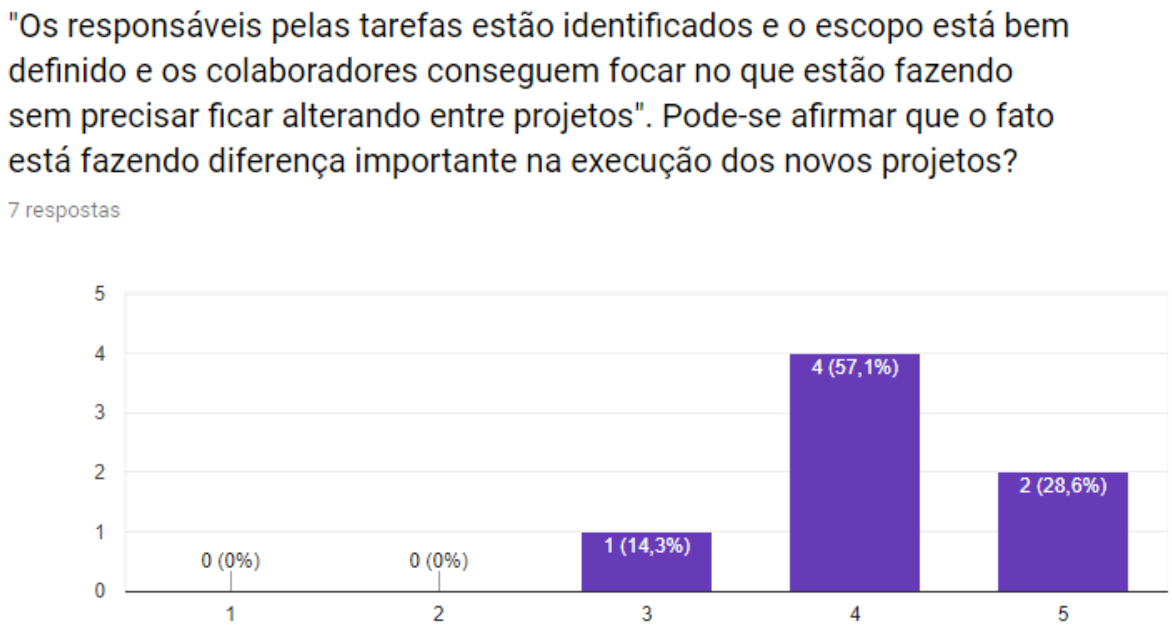
Atualmente, com o novo processo, tanto o cliente quanto a equipe sabem que a demanda será especificada em uma versão de duas semanas e, possivelmente seja desenvolvida na próxima versão de mesmo prazo. Contudo, caso a demanda não entre na lista de prioridades, um novo prazo já pode ser estimado, pois outro projeto será iniciado e concluído em mais duas semanas. Por mais que o novo processo não garanta 100% de entrega ao final do período, as demandas que são removidas de uma versão entram automaticamente na próxima, pois a sua prioridade não sofre alteração. Assim, é mais fácil explicar aos clientes, tanto internos quanto externos, que a demanda será movida para a próxima publicação, com data já definida.

6.3.5 Alternância de pessoal entre os projetos

A avaliação desta questão também é qualitativa pois não há registro formal de projetos anteriores, tanto da saída quanto da entrada de novos recursos. Portanto a avaliação feita com a equipe interna (FIGURA 23) torna-se mais importante para a definição de sucesso desta

mudança. Entende-se que com a fixação de uma equipe específica para o segmento de produto, o risco de alternância de pessoal já diminui consideravelmente, mas ainda assim não garante êxito.

Figura 23 – Questão sobre a alternância de pessoal



Fonte: elaborado pelo autor, 2017.

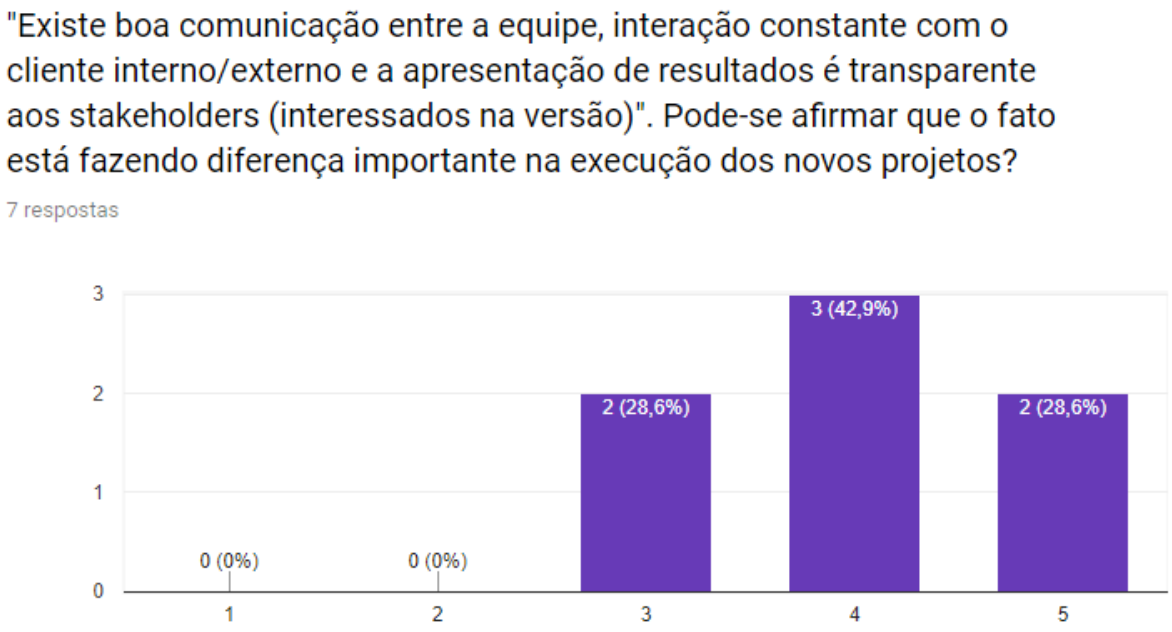
De acordo com os membros da equipe que responderam este questionamento, a afirmação de que definir responsáveis e escopo garante o foco dos envolvidos é verdadeira. Na média, a resposta foi de 4,14 pontos, a segunda mais alta da pesquisa, tendo também o segundo menor desvio padrão, com 0,69 pontos. Compreende-se que com o tempo o processo irá criar um efeito cascata para a situação das alternâncias, pois sabe-se que ela ocorre principalmente por conta de atrasos em projetos ou capacidade técnica da equipe, fatores que tendem a ser mitigados pelo novo processo.

6.3.6 Boa comunicação entre os interessados do projeto

Mesmo no antigo processo de desenvolvimento a comunicação entre os integrantes da equipe de desenvolvimento ocorria, porém não existia um evento marcado e nem registros de

interações. Além disso, o escopo e o andamento do projeto não eram transparentes para os interessados que não participavam das conversas informais. A Figura 24 demonstra que a equipe concorda, na sua maioria, que o novo processo provê maior visibilidade da situação para os demais envolvidos, sendo a média das respostas de 4 pontos, com um desvio padrão de 0,82. Percebe-se pela pesquisa que a comunicação ainda pode ser afinada, sendo este assunto passível de um novo plano de ação.

Figura 24 – Questão sobre a comunicação dos *stackholders*



Fonte: elaborado pelo autor, 2017.

As etapas de priorização de *backlog* e aprovação de requisitos trouxeram maior proximidade do cliente ao projeto, fator considerado por todos como um ponto de sucesso da nova implantação. Envolvendo o cliente é possível identificar possíveis erros ou desvios com bastante antecedência, o que diminui especialmente o custo de possíveis problemas e aumenta a qualidade das entregas. Mesmo recebendo o relatório periódico e a formalização do encerramento, a participação do cliente também é incentivada em outras atividades do projeto, principalmente na definição das demandas não previstas.

Por outro lado, a reunião de *kickoff* foi ponto crucial para a melhoria do entendimento do escopo por parte da equipe. Em outras versões a equipe poderia ser surpreendida pela

complexidade do escopo no momento do desenvolvimento, não tendo um evento de apresentação mapeado no processo. Agora, com uma reunião de alinhamento inicial e as atualizações nas reuniões diárias, os envolvidos com o desenvolvimento estão mais engajados com o objetivo final da entrega.

No passado, as reuniões diárias e envio de relatórios periódicos já fez parte do processo da Retta, mas com o passar do tempo a parte burocrática e gerencial foi abandonada em função de vários fatores relacionados. Acredita-se que os eventos de priorização do *backlog* de produto e *kickoff* das versões, que aproximam cliente e equipe, são fatores primordiais para que o problema não torne a acontecer.

7 CONSIDERAÇÕES FINAIS

Com base na pesquisa, constatou-se uma tendência migratória das empresas de *software* para alguma das metodologias tidas como ágeis, mesmo que iniciando de forma tímida, os processos de desenvolvimento começam a apresentar características dinâmicas. Outro fato relevante percebido é que essa mudança inicia geralmente pela base da pirâmide hierárquica da empresa, ou seja, são os colaboradores que fomentam pequenas transformações que eventualmente se tornam implantações de novos processos.

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. Para embasar o estudo foram analisados os métodos tradicionais Cascata e RUP que, tem tendência sequencial de evolução e as metodologias SCRUM e XP, conhecidos por sua forma de trabalho mais dinâmica. Posteriormente, analisou-se o processo de desenvolvimento da empresa, buscando identificar pontos positivos e negativos. Os pontos positivos foram destacados e devem ser mantidos no processo, enquanto os negativos foram considerados como potenciais melhorias.

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 *stackholders*), mas com o tempo foram deixados de lado. O processo utilizado pela empresa Retta até o momento desta implantação foi criado no final de 2014, ou seja, são praticamente três anos sem ter sua documentação atualizada. Evidentemente as fases foram modificadas em função das necessidades de entrega, mas nenhuma alteração foi feita no manual do processo.

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 Retta, 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.

Principalmente com base na pesquisa qualitativa aplicada com os envolvidos, mas também com base nos índices de entrega, conclui-se que a implantação foi um sucesso e atingiu o seu objetivo de maneira completa. Alguns pontos do processo deveriam ter sido 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.

Para continuidade desta pesquisa o autor sugere dois pontos relevantes para a empresa Retta ou outra empresa de desenvolvimento que se encontre na mesma situação: padronizar a nova forma de trabalho para todos os projetos da empresa, inclusive os de fábrica de *software* e 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 BIBLIOGRÁFICAS

ARTIGOS GP3. **O Triângulo das Restrições de Gerenciamento de Projetos**. Disponível em: <https://www.gp3.com.br/artigo/182-o-triangulo-das-restricoes-de-gerenciamento-de-projetos/>. Acesso em: 05 novembro 2017

AWALD, M. A. A **Comparison between Agile and Traditional**. Software Development Methodologies. 2005. Disponível em: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.464.6090&rep=rep1&type=pdf. Acesso em: 14 abril 2017.

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

BARDEN, Helena. **Proposta de adequação do processo de desenvolvimento de software para a melhoria da estimativa de horas.** Disponível em: http://www.univates.br/revistas/index.php/destaques/article/view/263. Acesso em: 01 maio 2017.

BARROS, Rodrigo Janot; NETO, Lauro Pinto. **Manual de Gestão por Processos Secretaria Jurídica e de Documentação. Escritório de Processos Organizacionais do MPF.** Brasília. DF. 2013. Disponível em: http://www.mpf.mp.br/conheca-o-mpf/gestao-estrategica-e-modernizacao-do-mpf/escritorio-de-processos/publicacoes/livros/manualdegestaoporprocessos.pdf. Acesso em: 12 abril 2017.

BASSI FILHO, Dirton Luiz. **Experiências com desenvolvimento ágil**. Universidade de São Paulo. 2008.

BORGES, Pedro Miguel Pereira. **Configuração do RUP com Vista à Simplificação dos Elencos Processuais em PMEs de Desenvolvimento de Software**. Universidade do Minho. Novembro de 2007.

BROOKS, Frederick. **No silverbulet. Essence and accident in software engineering**. 1987. Disponível em: http://worrydream.com/refs/Brooks-NoSilverBullet.pdf. Acesso em: 10 maio 2017.

CRESWELL, John W. **Projeto de pesquisa. Métodos qualitativo, quantitativo e misto**. 3ª edição. Editora Bookman. 2010.

FAGUNDES, Priscila. **Framework para comparação e análise de métodos ágeis.** Universidade Federal de Santa Catarina. 2005. Disponível em: https://repositorio.ufsc.br/handle/123456789/101860. Acesso em: 01 junho 2017.

FILHO, Dairton Luiz Bassi. **Experiências com desenvolvimento ágil**. São Paulo, março de 2008. Disponível em: https://www.ime.usp.br/~dairton/files/Dissertacao-DairtonBassi.pdf. Acesso em: 16 maio 2017

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

GRIESI, Ariovaldo; FECCHIO, Mario Moro. **Engenharia de Software Uma Abordagem Profissional**. 7º ed. - São Paulo, SP: MGH Editora Ltda, 2011.

HIGHSMITH. Jim. **History: The Agile Manifesto**. 2001. Disponível em: http://agilemanifesto.org/history.html. Acesso em: 16 maio 2017.

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

KRUCHTN, Philippe. **The Rational Unified Process na introduction**. 3ª edição. Pearson Education, Inc. 2004.

KRUCHTN, Philippe. **The Rational Unified Process Made Easy: A Practitioner's Guide to the RUP**. Pearson Education, Inc.4ª edição. 2004.

LAYMANA, Lucas. WILLIAMS, Laurie. DAMIAN, Daniela. **Essential communication practices for Extreme Programming in a global software development team**. Hynek Bures c. 2006. Information and Software Technology.

LEIDEMER, Romulo Henrique**. Implantação de SCRUM em uma empresa de desenvolvimento de software.** Univates. 2013.

LIBARDI, Paula, BARBOSA, Vladmir. **Metodologias tradicionais e metodologias ágeis: análise comparativa entre rational unified process e extreme programming.** Faculdade de Tecnologia do EST. São Paulo – SP. 2012

LUIZ, Ronaldo Rezende Vilela. **Obtendo Qualidade de Software com o RUP.** Uberaba, MG. 2011. Disponível em: http://javafree.uol.com.br/artigo/871455/Obtendo-Qualidade-de-Software-com-o-RUP.html. Acesso em: 21 maio 2016.

MARTINS, J. C. C. **Gerenciando projetos de desenvolvimento de software com PMI, RUP e UML**. 4ª edição. Rio de Janeiro: Brasport.2007.

MORSE, Andrew Powell. **Extreme Programming: what is it and how do you use it?** Airbrake. 2017. Disponível em: <https://airbrake.io/blog/sdlc/extreme-programming>. Acesso em: 14 setembro 2017.

NEIVA, Danuza Ferreira, MATOS, Ecivaldo de Souza. **Uma Estratégia para Substituição de Sistemas Legados. Departamento de Ciência da Computação.** Universidade Federal da Bahia, Salvador, BA. 2010. Disponível em: http://ecivaldo.com/arquivos/10.3-anais-de-eventos-completo/10.3.24.pdf. Acesso em: 16 maio 2017.

PEREIRA, Eliana Beatriz. **Uma proposta para adaptação de processos de desenvolvimento de software baseados no rational unified process.** 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 Agileand Dev Ops**. 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.

RANDO, Gianluigi. **Agile e DevOps per la Trasformazione Digitale**. 2017. Disponível em: <http://www.gianluigirando.com/?p=693>. Acesso em: 28 junho 2017.

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

RUSSO, Rosalia; RUIZ, Jose. **Liderança e influência nas fases da gestão de projetos.** Revista Produção, v. 15, n. 3, p. 362-375, Set./Dez. 2005. Disponível em: http://www.scielo.br/pdf/%0D/prod/v15n3/v15n3a06.pdf. Acesso em: 22 maio 2017.

SABBAGH, Rafael. **SCRUM Gestão ágil para projetos de sucesso**. **Editora:** Casa do Código. 2014.

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

SGANDERLA, Kelly. **Um guia para iniciar estudos em BPMN (VI): Swimlanes e Artefatos**. 2013. Disponível em: <http://blog.iprocess.com.br/2013/01/um-guia-para-iniciar-estudos-em-bpmn-vi-swimlanes-e-artefatos/>. Acesso em: 04 outubro 2017.

SILVA, E. L.; MENEZES, E. M. **Metodologia da Pesquisa e Elaboração de Dissertação.** Florianópolis. 2005. Disponível em: <https://projetos.inf.ufsc.br/arquivos/Metodologia_de_pesquisa_e_elaboracao_de_teses_e_dissertacoes_4ed.pdf >. Acesso em: 19 maio 2017.

SOARES, Michel dos Santos. **Comparação entre Metodologias Ágeis e Tradicionais para o Desenvolvimento de Software.** 2004. Disponível em: <http://www.dcc.ufla.br/infocomp/artigos/v3.2/art02.pdf > acesso em: 29outubro 2017

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.

TAROCO, Bruna Avanci; WERNER, Claudete. **Análise comparativa entre as metodologias de desenvolvimento tradicionais e ágeis.** Universidade Paranaense (Unipar) Paranavaí, PR. 2001.

VIEIRA, Denisson. **O guia passo-a-passo para implantar scrum em seu próprio projeto**. 2014.Disponivel em: http://www.mindmaster.com.br/wp-content/uploads/2015/09/Guia-Passo-a-Passo-Como-Implantar-Scrum.pdf. Acesso em: 27 maio 2017.

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.