Universidade Federal de Minas Gerais
Instituto de Ciências Exatas
Departamento de Ciências da Computação

ALAN CASSIANO DE CARVALHO

MUDANÇA DE PARADIGMA EM INSTITUIÇÕES QUE POSSUEM PROCESSOS DE DESENVOLVIMENTO DE SOFTWARE CONSAGRADO PARA PROCESSOS ÁGEIS

Belo Horizonte
2016

Universidade Federal de Minas Gerais
Instituto de Ciências Exatas
Departamento de Ciências da Computação
Especialização em Informática: Ênfase: Engenharia de Software

MUDANÇA DE PARADIGMA EM INSTITUIÇÕES QUE POSSUEM PROCESSOS DE DESENVOLVIMENTO DE SOFTWARE CONSAGRADO PARA PROCESSOS ÁGEIS

por
Alan Cassiano de Carvalho

Monografia de final de Curso
CEI-ES 066 - T20 - 2016 - 1

Clarindo Isaias Pereira da Silva e Padua
Orientador

Belo Horizonte
2016

Alan Cassiano de Carvalho

Mudança de Paradigma em Instituições que possuem processos de Desenvolvimento de Software Consagrado para Processos Ágeis

Monografia apresentada ao Curso de Especialização em Informática do Departamento de Ciências Exatas da Universidade Federal de Minas Gerais, como requisito parcial para a obtenção do grau de Especialista em Informática.

Área de concentração: Engenharia de Software

Orientador: Clarindo Isaias Pereira da Silva e Padua

Belo Horizonte

2016

Ficha catalográfica elaborada pela Biblioteca do ICEx - UFMG

Carvalho, Alan Cassiano de.

C331m Mudança de paradigma em instituições que possuem processos de desenvolvimento de software consagrado para processos ágeis / Alan Cassiano de Carvalho. Belo Horizonte, 2016.
x, 46 f.: il.; 29 cm.

Monografia (especialização) - Universidade Federal de Minas Gerais – Departamento de Ciência da Computação.

Orientador: Clarindo Isaías Pereira da Silva e Pádua.

1. Computação 2. Engenharia de software. 3. Software - Desenvolvimento. I. Orientador. II. Título.

CDU 519.6*32(043)

UNIVERSIDADE FEDERAL DE MINAS GERAIS

INSTITUTO DE CIÊNCIAS EXATAS
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO
ESPECIALIZAÇÃO EM INFORMÁTICA: ÁREA DE CONCENTRAÇÃO ENGENHARIA DE SOFTWARE

MUDANÇA DE PARADIGMA EM INSTITUIÇÕES QUE POSSUEM PROCESSOS DE DESENVOLVIMENTO DE SOFTWARE CONSAGRADO PARA PROCESSOS ÁGEIS

ALAN CASSIANO DE CARVALHO

Monografia apresentada aos Senhores:

Prof. Clarindo Isaías Pereira da Silva e Pádua
Orientador
DCC - ICEx - UFMG

Prof. Marco Túlio de Oliveira Valente
DCC - ICEx - UFMG

Belo Horizonte, 15 de abril de 2016

Resumo

As metodologias ágeis estão ganhando mercado ao longo dos anos, logo, este trabalho tem como objetivo acompanhar uma empresa privada que tem o intuito de modificar seus processos de desenvolvimento de software e aplicar modelos ágeis em sua fábrica de software. Atualmente, muitas empresas adotaram essa estratégia e muitas obtiveram sucesso. Na empresa onde foi realizada o trabalho, o desejo de atualização de seus processos para fazer frente a concorrência tem ganhado força, com apoio da gerência e dos colaboradores, o processo de desenvolvimento de software melhora de maneira que, cliente, empresa e colaboradores saiam satisfeitos. Este trabalho demonstrou também como foi o comportamento da fábrica de software perante as mudanças e evidenciou que métodos ágeis buscam inovação, comprometimento, adequação e qualidade na entrega de seus produtos.

Palavras-chave: Desenvolvimento, Software, Agilidade, Scrum, XP, Processos, Engenharia de Software.

Abstract

Agile methodologies are gaining market over the years, so this study aims to follow a private company that aims to modify their software development processes and apply agile models in its software factory. Currently, many companies have adopted this strategy and many have succeeded. In the company where the work was carried out, the desire to upgrade their processes to meet the competition has gained strength with the support of management and employees, the software development process improvement so that customer, company and employees leave satisfied . This work has also shown how was the behavior of the software factory before the changes and showed that agile methods seek innovation, commitment, fitness and quality in the delivery of its products.

Keywords: Development, Software, Agility, Scrum, XP, Process, Software Engineering.

Lista de ilustrações

Lista de tabelas

Sumário

1 Introdução

O presente trabalho trata das ações que uma empresa privada da cidade de Belo Horizonte realiza em prol da melhoria contínua dos processos de desenvolvimento de sua fábrica de software. Tais ações são referentes a atualização e implantação de métodos ágeis de desenvolvimento de software. Tendo em vista que uma fábrica de software segue a regra disciplinada na engenharia de software, onde Sommerville (2008) afirma que é uma engenharia capaz de buscar ocupação para todas as fases do desenvolvimento e aspectos produtivos do software, ou seja, as fases de especificação até a manutenção de software são preenchidos pelos modelos.

Baseando na Engenharia de Software, o crescimento, maturidade, adequação e atualização dos processos da empresa, ao qual deseja possuir vantagem competitiva frente as demais concorrentes, voltam-se para o confrontamento e discussão dos modelos de desenvolvimento tradicionais e ágeis. É nesse sentido que Schwaber e Beedle (2002) defendem que as metodologias surgiram com diversos propósitos, cada uma seguindo um raciocínio diferente mas com o objetivo de sistematizar e aprimorar o desenvolvimento de software.

Com a necessidade de renovar seus conceitos, a empresa começou a buscar alternativas, da mesma forma que Sbrocco e Macedo (2012) discutem que as organizações buscam modelos para se aplicar engenharia de software de maneira que seja menos burocrática e que sejam de qualidade, denominadas metodologias ágeis.

Autores como Loftus e Ratcliffe (2005), Svensson e Höst (2005), Ilieva, Ivanov e Stefanova (2004) e Freire, Kon e Torteli (2005), demonstram uma percepção que, além de cada peculiaridade de seus estudos, todos obtiveram bons resultados com a implantação desses novos modelos. Perceberam-se diversos ganhos para a empresa, colaboradores e clientes.

Dessa forma, considerando essa nova forma de pensar e agir de uma fábrica de software é que a busca pela integração contínua de seus processos podem caminhar em direção ao sucesso, com maior qualidade na entrega de seus produtos, podendo assim, possuir maior potencial competitivo no mercado.

1.1 Objetivos

1.1.1 Geral

O objetivo do trabalho será demonstrar as dificuldades e comportamentos que uma empresa possui quando ela pretende migrar seus processos de desenvolvimento de software tradicional para processos ágeis de desenvolvimento.

1.1.2 Específicos

Os objetivos específicos podem ser listados da seguinte forma:

- Observar o ambiente ao qual está sendo implantada a metodologia ágil.
- Verificar quais práticas do desenvolvimento ágil estão sendo utilizadas.
- Identificar práticas do modelo tradicional que não foram abandonadas.
- Verificar se o time está colaborando e propondo opções para melhorar o processo ágil.
- Propor mudanças no processo de desenvolvimento ágil utilizado atualmente.

1.2 Metodologia

Jung (2004) afirma que, uma metodologia de pesquisa é um conjunto de métodos, técnicas e procedimentos que objetiva facilitar a execução da pesquisa para se chegar a um resultado de um produto, processo ou conhecimento.

Esta seção demonstra como será classificado o trabalho, assim como dos componentes e ferramentas que serão utilizadas para o desenvolvimento do projeto.

1.2.1 Tipo de Pesquisa

De acordo com o apresentado por Jung (2004) sobre o tipo de pesquisa, esse trabalho pode ser classificado da seguinte forma:

- Quanto à natureza, é classificado como pesquisa tecnológica ou aplicada, visto que, aplica os conhecimentos básicos da criação de um produto e é dirigido à solução de problemas específicos de uma instituição.
- Quanto aos objetivos, é classificado como exploratório, visto que tem como finalidade observar e analisar vários fatores de desenvolvimento e ter maior familiaridade com o problema para torná-lo explícito.

- Quanto à abordagem do problema, é classificada como qualitativo, visto que, é confrontado um ambiente real e o sujeito, ao qual os processos e a maneira como funcionam são os pontos focais dessa abordagem.

- Em relação aos procedimentos técnicos, é classificado como estudo de caso, pois, tem como objetivo estudar um ou mais objetos, de maneira que permita o maior entendimento dos problemas. Estudo de caso é uma ferramenta importante para os pesquisadores e tem como finalidade entender como e porque funcionam determinados problemas.

1.2.2 Procedimentos Metodológicos

A pesquisa foi realizada considerando o ano de 2014 e 2015 em uma instituição privada situada em Belo Horizonte, Minas Gerais.

Será realizada uma revisão da literatura com o intuito de identificar estudos relacionados ao assunto abordado, bem como, entender e contextualizar um ambiente que será investigado.

Após, projetos que utilizaram abordagens diferentes, tradicionais e ágeis, foram observados durante esse período de tempo. Será realizada a pesquisa exploratória com o intuito de aprimorar os resultados.

1.2.3 Estrutura do Trabalho

No capítulo 2, são citados os principais conceitos referentes a Engenharia de Software, fases do desenvolvimento de software, metodologias de desenvolvimento tradicionais e ágeis.

No capítulo 3, é apresentada a descrição geral do problema, o projeto piloto utilizado, como foi a implantação do novo modelo e o comportamento da equipe e do projeto.

No capítulo 4, são apresentadas as conclusões, as contribuições e os possíveis trabalhos futuros.

2 Fundamentação Teórica

Tendo em vista o mercado de desenvolvimento de software, no qual a competitividade entre empresas é cada vez maior, a busca por melhoria de processos ao qual atenda seus clientes de maneira rápida e satisfatória se torna frequente.

Este capítulo apresentará os modelos de desenvolvimento de software tradicional e ágil.

2.1 Engenharia de Software e as fases do desenvolvimento

Para abordar os conceitos de metodologias de desenvolvimento de software, é necessário entender o motivo de elas existirem, ou seja, entender o que a engenharia de software propõe com relação aos métodos e práticas de sucesso.

É nesse sentido que a definição de Engenharia de Software segundo Maia (2007) é de ser um processo para criação de software de maneira organizada, utilizando de ferramentas e técnicas predefinidas para se chegar ao objetivo, que é a criação do software.

De acordo com Sommerville (2008), a engenharia de software é uma engenharia que busca ocupar todas as fases e aspectos produtivos do software, ou seja, busca preencher os estágios iniciais de especificação até o crescimento e manutenção do sistema utilizado.

Seabra (2013b) demonstra que em um processo de engenharia de software, são gerados artefatos importantes que demonstram as entradas e saídas para cada atividade do desenvolvimento de software. Ainda de acordo com o autor, utilizando os artefatos nos momentos certos e de acordo com suas entradas e saídas, é possível entender melhor o contexto do produto.

É nesse contexto que, de acordo com Pressman (2015), a Engenharia de Software se preocupa com métodos, técnicas e ferramentas para o processo de criação de sistemas. Com relação as técnicas, está incluída as maneiras para se comunicar, analisar requisitos, modelar projetos, construir, testar e fornecer suporte.

Sommerville (2008) afirma que não existe um processo ideal para desenvolvimento de software, logo, cada instituição procura criar o seu próprio processo. Esses processos são criados para definir tarefas que deverão ser seguidas por pessoas da instituição e as características que o sistema a ser desenvolvido possui. Também de acordo com o autor supracitado, um sistema crítico exige processos mais robustos e bem definidos, já para sistemas que sofrem com mudanças contínuas de requisitos, requerem um processo mais flexível e ágil. No entanto, independente do processo, algumas atividades em comum são

encontradas em todos os processos, são elas:

- Especificação de Software: Definir as funcionalidades que um software irá possuir e como será o seu uso.
- Projeto de Software e implementação: Criação de um projeto de software ao qual o desenvolvimento irá se apoiar.
- Validação de Software: É verificar se o software faz o que o cliente necessita.
- Evolução do Software: Para atender novas funcionalidades do cliente, o sistema deve evoluir de acordo com as mudanças necessárias.

Concomitante com as tarefas e atividades de criação de software definidas pela Engenharia de Software, é possível traçar os passos para criação de software. Esses passos são denominados modelos pré-definidos de sucesso que seguem uma sequência lógica de atividades/tarefas.

2.2 Metodologias de Desenvolvimento

A criação de software pode ser classificada como uma cadeia de atividades a serem realizadas por equipes multidisciplinares, ou seja, divide-se as atividades em etapas com tarefas bem divididas. Esses processos tem como objetivo acompanhar o projeto desde o levantamento inicial até o aceite do produto final pelo cliente. Nesse intuito é que a Engenharia de Software cria métodos, técnicas e diversas ferramentas para apoiar estes processos e torná-los eficientes e mais baratos (SEABRA, 2013b).

De acordo com VASCONCELOS et al. (2006), o desenvolvimento de software tem objetivo de atender as necessidades de um determinado grupo de clientes e usuários que necessitam facilitar suas atividades. Com isso, a especificação e detalhamento dos requisitos de Software são fundamentais para o sucesso de entrega do sistema. Visualizando esse contexto, percebe-se que a utilização de analistas de requisitos está cada vez maior e desempenha um papel importante para o projeto.

Para se alcançar a qualidade de um produto final, as empresas tem se adaptado aos clientes, respondendo a mudanças e resolvendo seus problemas, e é nesse cenário que Bassi (2009) diz que é necessário um conjunto de tarefas e atividades organizadas para entregar um software funcional. De acordo com o autor, esse conjunto de tarefas, atividades e processo é conhecido como Metodologia de Desenvolvimento de Software.

O grande desafio enfrentado pelas empresas é entregar produto de qualidade dentro do prazo, em perfeito funcionamento e com baixo custo de manutenção. No entanto, muitas dessas empresas sofrem com o inverso do que é planejado. É nesse cenário que Schwaber e

Beedle (2002) dizem que diversas metodologias surgiram, cada um com seu propósito mas todas com o intuito de sistematizar e aprimorar o desenvolvimento de software. Ainda de acordo com o autores dessa publicação, diversos outros autores classificam essas metologias em duas modalidades:

- Tradicionais: que são, em sua grande maioria, metodologias criadas antes dos modelos ágeis que possuem o foco em documentação de cada etapa do desenvolvimento;
- Ágeis: são consideradas como um novo paradigma para desenvolvimento de software onde busca focar menos nos formalismos e privilegiando a conversa e respostas rápidas a respeito de mudanças e definições de requisitos do projeto.

Soares (2004) também afirma que metodologias são um conjunto de métodos que possuem os procedimentos necessários para atingir um objetivo. Uma técnica é uma maneira que se dispõe para resolver uma parte do problema que está alocado em um universo de interesse. Alguns exemplos são: Análise Estruturada, Análise Essencial, Projeto Estruturado e Análise Orientada a Objetos.

Possuindo a definição de metodologias de desenvolvimento de software, é possível descer um nível e descrever as metodologias mais conhecidas e utilizadas pelo mercado.

2.3 Metodologia de Desenvolvimento Tradicional

Para Pressman (2015), os modelos tradicionais (ou prescritivos) são aqueles que prezam pela estrutura e ordem no desenvolvimento de software. Todas as atividades e tarefas que um projeto necessita ocorrem de maneira sequencial e são definidas para o seu progresso. Eles são considerados pelo autor como prescritivo pelo fato de possuírem, prescritivamente, um conjunto de atividades de processo que interagem entre as diversas fases existentes.

Nos próximos capítulos será apresentado alguns modelos que seguem o modelo tradicional.

2.3.1 Modelo em Cascata

Sommerville (2008) afirma que esta é a primeira metodologia de desenvolvimento de software publicada e que deriva do processo geral de engenharia de sistemas de Royce.

Calcado (2007) definiu que o Modelo em Cascata se baseia na execução sequencial das atividades. O autor ainda afirma que este é um modelo linear ao qual cada passo deve ser terminado para que o próximo possa começar.

De acordo com a Figura 1, VASCONCELOS et al. (2006) descrevem as etapas do modelo da seguinte forma:

- Definição dos Requisitos: Especificar o que o sistema deve fazer, os serviços que deve realizar, bem como as restrições impostas pelo cliente referente ao sistema.
- Projeto do Sistema: Com os requisitos definidos, eles são utilizados para a definição dos componentes de hardware e software a fim de apoiar a equipe de desenvolvimento. Esta é a fase de fundamentação da arquitetura geral do sistema.
- Implementação e testes unitários: Utilizando a linguagem de programação definida no projeto de software, os módulos funcionais do sistema são desenvolvidos.
- Integração e teste do sistema: Com o intuito de garantir que o desenvolvimento atende ao pedido pelos clientes, os módulos funcionais são integrados e testados como um sistema completo. Testes devem ser realizados a medida que novas funções são adicionadas ao sistema.
- Operação e Manutenção: O sistema, após terminado é disponibilizado para uso. Posteriormente, é alterado quando surgirem falhas ou quando o cliente necessitar de novas funcionalidades.

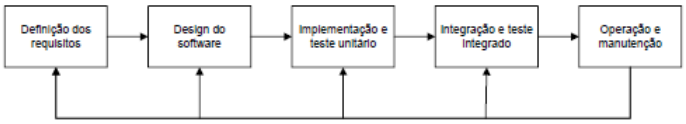

Figura 1: Modelo em Cascata

Segundo Pressman (2015), esse modelo possui alguns problemas que fazem com que a sua adoção seja descartada. O autor afirma que raramente o fluxo proposto pelo modelo é seguido, principalmente quando existem alterações no projeto. Como resultado dessas alterações, a equipe fica mais confusa e o produto perde a qualidade.

Pressman (2015), também afirma que outro problema complicado é o cliente expor todas as funcionalidades que o sistema deve possuir. Muitas vezes o cliente não lembra de tudo, algumas regras e incertezas são naturais. O autor também afirma que falhas desastrosas podem ocorrer e o cliente precisa ter paciência, pois, o cronograma será afetado.

2.3.2 Modelo Incremental

Sommerville (2008) afirma que o modelo incremental é uma evolução da metodologia em cascata, porém, o cliente identifica as funcionalidades do sistema priorizando os requisitos que são mais importantes e que serão entregues primeiro. Feito isso, é combinado

um conjunto de entregas a cada incremento, disponibilizando assim, parte do sistema com as novas funcionalidades. Dessa forma, as fases do projeto podem ser desenvolvidas de maneira incremental, evoluindo em direção ao produto final.

De acordo com VASCONCELOS et al. (2006), cada funcionalidade do sistema é disponibilizada mais rapidamente, seguindo os incrementos desejados pelo cliente. Dessa forma os incrementos finais tem seu entendimento facilitado, visto que, para cada incremento, o foco estará em uma funcionalidade diferente. Esta também é uma forma que diminui falhas no projeto e aumenta a quantidade de testes realizados no sistema.

2.3.3 RUP: Rational Unified Process

De acordo com Seabra (2013a), o Rational Unified Process (RUP) é um processo de propriedade da IBM e que está disponível para venda. Ele se origina do Processo Unificado, que tem como objetivo obter as melhores práticas para desenvolvimento de software.

Sommerville (2008) demonstra que o RUP possui 03 perspectivas, são elas:

- Dinâmica: Esta perspectiva demonstra as fases do processo ao longo do tempo de execução do projeto;
- Estática: Esta perspectiva mostra como e quando as atividades do processo são executadas pelos colaboradores do projeto;
- Prática: Esta perspectiva procura demonstrar as melhores práticas a serem seguidas durante a execução do processo.

De acordo com Seabra (2013a), a perspectiva Prática busca demonstrar aos praticantes de RUP as melhores práticas para desenvolvimento de sistemas. O autor ainda diz que as melhores práticas tendem ao desenvolvimento orientado ao negócio e não possuem dependências. Seguem as melhores práticas:

- Adaptação do processo;
- Equilíbrio as prioridades dos envolvidos;
- Colaboração entre as equipes;
- Demonstração de valor do produto iterativamente;
- Nível de abstração elevado;
- Foco contínuo na qualidade.

2.4 Metodologia de Desenvolvimento Ágil

Metodologias para desenvolvimento de software são roteiros de sucesso a serem seguidos que contribuem para agilizar os processos internos de desenvolvimento.

Nesse contexto de métodos ágeis é que Beck (2015) afirma que elas possuem uma abordagem mais simples e que possui um sentido objetivo se comparado aos modelos tradicionais, ou seja, estes métodos possuem foco em menos documentação e mais comunicação, com respostas rápidas as mudanças de requisitos e fazendo com que os clientes se tornem peças importantes e ativas no contexto do projeto. Dessa forma, a minimização de risco no desenvolvimento ocorre de maneira natural.

Ainda de acordo com Beck (2015), o uso de métodos ágeis foi impulsionada em fevereiro de 2001 quando 17 especialistas em desenvolvimento de software apresentaram e discutiram os seguintes métodos: Extreme Programing (XP), Scrum, Dynamic Systems Development Method (DSDM), Crystal, entre outros. Com esse contexto, foi estabelecido o conhecido "Manifesto Ágil", que apresenta ideias e valores conforme a seguir:

- Valorizar os indivíduos e interações mais que os processos e ferramentas;
- Software em funcionamento mais que documentação abrangente;
- Colaboração com o cliente mais que negociação de contratos;
- Respostas às mudanças mais que seguir um plano.

Ademais, em virtude dessa reunião que surgiu o manifesto ágil "Manifesto Ágil", os componentes não afirmam que as características do modelo tradicional sejam dispensáveis, mas que, a importância maior está nos pontos citados anteriormente. A proposta é executar tarefas prioritárias em conformidade com o desejo do cliente, aplicando os artefatos do projeto de maneira simples e mais objetiva possível, porém, sem deixar de analisar as características importantes que esses artefatos trazem do modelo tradicional. (BECK, 2015)

Nessa reunião também foi definido os doze princípios a serem considerados durante o desenvolvimento de software, são eles:

1. A principal prioridade é satisfazer o cliente por meio de entrega adiantada e contínua de software de valor
2. Aceitar mudanças nos requisitos, mesmo que seja no final do desenvolvimento. Processos ágeis valorizam mudanças para que o cliente possa ganhar vantagem competitiva.

3. Entrega com frequência de software funcionando, em escalas de semanas e até meses, preferencialmente em períodos de tempo menores.

4. Pessoas relacionadas aos negócios e os desenvolvedores devem trabalhar em conjunto e diariamente ao longo da execução do projeto.

5. Construir os projetos em volta de pessoas motivadas. Dando a eles o ambiente e apoio que necessitam e confie que eles vão fazer o serviço;

6. O método mais eficiente e eficaz para transmitir informações para uma equipe é a conversa cara a cara.

7. Software funcionando é a principal medida de progresso.

8. Processos ágeis promovem o desenvolvimento sustentável. Os patrocinadores, desenvolvedores e usuários devem ser capazes de manter um ritmo constante.

9. Atenção contínua referente a excelência técnica e bom design aumenta a agilidade.

10. Simplicidade: a arte de maximizar a quantidade de trabalho que não precisou ser realizado.

11. As melhores arquiteturas, requisitos e projetos emergem de equipes auto organizáveis.

12. Em intervalos regulares, o time reflete sobre como ficar mais efetivo, então, se ajustam e otimizam seu comportamento de acordo com a reflexão.

Abrahamsson et al. (2003) afirmam que o desenvolvimento ágil tem como objetivo aumentar o poder de reação frente a mudanças e reponder a elas.

Highsmith (2002) afirma que um método ágil deve se adequar suficientemente para permitir as alterações necessárias e atender os princípios e práticas conforme demanda um projeto e sua equipe. Também, de acordo o autor supracitado, um processo deve ser colaborativo e disciplinado para incentivar o trabalho de uma equipe, ou seja, promovendo entregas rápidas de software de qualidade e que alinha as necessidades do cliente com os objetivos da empresa.

No decorrer da história, diversos métodos ágeis foram propostos, porém, atualmente os mais utilizados são Scrum, que propõe uma alternativa para gerenciamento e controle do projeto, Extreme Programing (XP) e Teste Driven Development (TDD), que buscam boas práticas para desenvolvimento de sistemas. Abaixo, alguns métodos serão descritos.

2.4.1 Scrum

De acordo com Schwaber (2004), Scrum foi criado seguindo o modelo de gestão da indústria automobilística e vem se destacando nas demais indústrias devido a

suas características de gerenciamento e controle que podem ser utilizadas em contextos diferentes.

Para compreender melhor as práticas do Scrum e conscientizar sua metodologia, foi criada a organização sem fins lucrativos chamada Scrum Alliance. Ela fornece os recursos necessários para pessoas e organizações além de fornecer melhorias contínuas para melhores práticas do Scrum. (ALLIANCE, 2015)

De acordo com Alliance (2015), o Scrum possui ciclos iterativos denominados Sprints. Nesses ciclos, a equipe de desenvolvimento se reúne com o cliente objetivando priorizar tarefas que deverão ser entregues naquele ciclo e os responsáveis por elas. Durante a Sprint, a equipe deve realizar reuniões diárias com o intuito de discutir o que foi feito, o que não foi realizado, os impedimentos e dificuldades encontradas e o que será feito a partir do momento da reunião. Ao final de cada Sprint, a equipe entrega um incremento do produto executável, ou seja, funcionando.

De acordo com Koscianski e Soares (2007), o Scrum se sustenta em uma abordagem que considera a experiência e observação do contexto da equipe. Nesse intuito o Scrum é baseado em um ciclo de vida em três fases: planejamento, desenvolvimento e pós-planejamento. Elas serão apresentadas a seguir:

- Planejamento: Esta é a fase no qual o Product Owner e o cliente descrevem os requisitos e os ordenam de acordo com uma ordem de prioridade, gerando o Sprint Backlog. Feito isso, é realizado o planejamento pelo qual é definido a equipe, ferramentas e os possíveis riscos do projeto. Para a finalização dessa fase é entregue uma proposta de uma arquitetura de desenvolvimento.

- Desenvolvimento: Esta fase é dividida em ciclos que são chamados de Sprints, nela inclui a reunião diária, Daily Meeting, e semanal, Sprint Review. Nessa fase, são implementadas as novas funcionalidades do sistema e são adicionadas a cada Sprint terminada. Os riscos são observados e controlados durante a fase de desenvolvimento e ao final é feito a entrega de um incremento de software funcionando.

- Pós-planejamento: Nesta fase é realizada a integração do software, além dos testes finais, homologação e documentação do software para o usuário. É realizado também uma análise do progresso do projeto e uma demonstração do produto para o cliente.

Schwaber (2004) demonstra que no modelo de desenvolvimento Scrum possui papéis, formalidades e artefatos que serão apresentados nas Tabelas 1, 2 e 3 a seguir:

Tabela 1: Papéis desempenhados no Scrum.

Papel	*Descrição*
Product Owner	É a pessoa que representa todos os interessados no projeto e também é responsável pelo gerenciamento do Product Backlog, maximizando o que é importante no projeto.
Team	É a equipe responsável pelo desenvolvimento do projeto. No time incluem os desenvolvedores, analistas de negócio e os testadores. O time deve ser auto organizável e auto gerenciável, possuindo autonomia suficiente para tomar decisões estratégicas do projeto.
Scrum Master	Sua principal missão é garantir que as práticas do Scrum estejam sendo seguidas, bem como, age como um facilitador resolvendo conflitos entre equipe e o Product Owner.
Cliente	É o solicitante do projeto. Participa ativamente do projeto definindo funcionalidades e prioridades entre as diversas tarefas do sistema. É considerado membro da equipe.

Tabela 2: Formalidades do Scrum.

Formalidade	*Descrição*
Sprint Planning	É a reunião realizada antes de se começar o projeto onde o Product Owner apresenta a lista de tarefas com suas respectivas prioridades do Product Backlog para a equipe.
Daily Meeting	É uma reunião realizada diariamente para alinhar as atividades entre os membros da equipe e relatando o progresso do trabalho, bem como, informando qualquer impedimento ao Scrum Master.
Sprint Review	É uma reunião realizada no fim de cada Sprint para verificar o trabalho realizado na Sprint anterior.
Sprint Retrospective	Tem o intuito de fazer uma retrospectiva da Release terminada, apresentando o que funcionou e o que não funcionou e o que poderia ser mudado. Esta reunião tem duração fixa de 3 horas e é a oportunidade de o time alterar o que não funcionou para a próxima Sprint ser mais produtiva.

Tabela 3: Artefatos do Scrum.

Artefatos	*Descrição*
Product Backlog	Lista de requisitos de acordo com suas prioridades e seu tempo estimado.
Sprint Backlog	É a lista de tarefas definidas pela equipe para ser entregue na respectiva Sprint.
Burndown Chatt	Estima o restante do trabalho ao longo do tempo de uma Sprint.

2.4.2 Extreme Programming (XP)

O Extreme Programming (XP) foi criado por Kent Beck em 1997 e é uma das primeiras abordagens ágeis criadas. Devido a isso ela é considerada o ponto de partida para diversas outras abordagens criadas.

De acordo com Beck(2004), o Extreme Programming tem como foco principal o trabalho em equipe, pelo qual ela deve ser capaz de se auto organizar para resolver um problema. O XP preza pelos cinco valores essenciais que serão descritos a seguir:

- Comunicação: tem o objetivo de manter contato direto entre programadores e cliente, entendendo que todos os participantes do projeto fazem parte da equipe, trabalhando em conjunto para se chegar a melhor solução.
- Simplicidade: Busca criar um produto simples e limpo, com o intuito principal de criar apenas o necessário e que foi solicitado pelo cliente.
- Feedback: Os testes são realizados durante toda a criação do produto de software, obtendo-se assim o feedback para os desenvolvedores. Pequenas entregas permitem maior entendimento a mudanças de requisitos.
- Respeito: Todos os envolvidos do projeto dão o respeito e o merecem, contribuindo de forma individual ou em grupo, ou seja, os desenvolvedores respeitam a experiência dos clientes e vice-versa.
- Coragem: As práticas do XP proporcionam coragem aos desenvolvedores quando há a necessidade de mudanças. Também, as dúvidas, medos e experiências existentes pelos desenvolvedores devem ser expostas.

De acordo com Wells (2009), o processo XP é composto por seis fases: exploração, planejamento de entregas, iterações para lançamento, produção, manutenção e morte do projeto.

1. **Exploração:** Esta fase compreende no levantamento de informações/requisitos e a escrita das histórias de usuário. Paralelamente, a equipe de desenvolvimento define as ferramentas, tecnologias e as práticas que deverão ser utilizadas no projeto. Esta fase pode durar semanas ou até meses variando com a maturidades da equipe.
2. **Planejamento de entregas:** Nesta fase são definidas as prioridades de cada história de usuário e as estimativas de cada uma delas. São realizados acordos para definir quantos e quais histórias serão entregues em um determinado período de tempo. Esta fase possui duração de alguns dias e a entrega não deve ocorrer em até três meses.

3. **Iterações:** Esta fase inclui algumas iterações do trabalho, ou seja, é estabelecido uma arquitetura do sistema para desenvolvimento e que pode ser alterada durante toda a fase de desenvolvimento. O trabalho é deduzido em tarefas em que cada programador deverá atuar, porém, deverá utilizar práticas de programação em pares.

4. **Produção:** É a fase em que o sistema será disponibilizado para o cliente, porém, requer testes adicionais e revisão de desempenho antes de entregar. Ao mesmo tempo, novas funcionalidades são discutidas e planejadas para serem adicionadas a versão atual.

5. **Manutenção:** Esta fase compreende no desenvolvimento de novas iterações enquanto o sistema já está em uso. A equipe modifica o sistema, com a ajuda do cliente, de acordo com suas necessidades. Dessa forma, a velocidade do desenvolvimento pode ser reduzida após o início da produção do sistema.

6. **Morte do projeto:** Esta fase ocorre quando o cliente não possui mais funcionalidades a serem incluídas no sistema. Neste caso, existe a necessidade de atender a expectativas do cliente como desempenho, usabilidade e confiabilidade. Nesta fase, a documentação de usuário são finalizadas e não ocorrem mais modificações na arquitetura. A morte do projeto também pode ocorrer quando o software cai em desuso ou o custo para manutenção se torna elevado.

2.5 Evolução e implantação de novos processos

De acordo com Poppendieck e Poppendieck (2011), baseando-se nos princípios do Toyotismo, que é a produção "enxuta", as metologias ágeis vem ganhando mercado com sua forma menos burocráticas de aplicar a Engenharia de Software.

Sbrocco e Macedo (2012) afirmam que as organizações estão, a cada dia, buscando processos leves e com qualidade, e é nesse intuito que se percebe a aplicação de diversos modelos de qualidade apoiados em métodos ágeis.

Loftus e Ratcliffe (2005) apresentaram os resultados de um estudo realizado com estudantes de pós-graduação cujo objetivo, entre diversos propostos, se o XP ajuda ou melhora o aprendizado de novas tecnologias. A conclusão é que, a maior parte dos estudantes relatou que foi produtivo e que o aprendizado foi mais rápido em um curto espaço de tempo, além do que, facilita a introdução de novas tecnologias. De acordo com os autores, é recomendável apresentar os métodos tradicionais de projetos em grupos antes de inseri-los no modelo XP.

Já no estudo desenvolvido por Svensson e Höst (2005), que descreveram o resultado da adoção do modelo XP (Programação Extrema) em uma empresa considerada no estudo de grande porte, com 1500 pessoas empregadas, sendo 250 em desenvolvimento de software.

A avaliação aconteceu utilizando-se um projeto piloto que teve duração de 8 meses. Para esse estudo, foram escolhidas três pessoas com diferentes posições perante o projeto para emitir diferentes pontos de vista sobre a implantação de modelos ágeis. Devido a mudança de paradigma, a organização pôde perceber um ganho positivo sobre a colaboração entre as pessoas envolvidas no projeto. Os autores indicam que se deve subestimar o esforço necessário para implantar e adaptar XP em uma organização.

Outro estudo foi realizado por Ilieva, Ivanov e Stefanova (2004), ao qual é descrito um caso uso comparando projetos que utilizaram o modelo tradicional de desenvolvimento e outro utilizando o XP. Os projetos eram parecidos em tamanho, esforço e tecnologia. Para composição da equipe, foi utilizado 4 pessoas e o projeto foi organizado para fazer uma entrega com 3 iterações. O resultado foi que houve um aumento de produtividade de 41,23%, redução do esforço por parte dos colaboradores de 11,45% e redução de problemas (bugs) em 13,33%.

Em Freire, Kon e Torteli (2005) é descrito como foi a inserção do modelo XP em uma Start-Up brasileira. São discutidas as modificações realizadas para se praticar o modelo XP e como os aspectos culturais e econômicos brasileiros podem afetar o uso do método. Para o estudo, as práticas foram inseridas de uma única vez e quatro projetos foram criados ao longo de 12 entregas, considerando iterações a cada duas semanas. Os autores relataram que existe uma dificuldade para se implantar um novo pensamento para desenvolvimento ágil em equipes / times heterogêneos, mas que pode ser resolvido com paciência e com a persistência brasileira.

3 Desenvolvimento

3.1 Descrição Geral

Atualmente a empresa utilizada para a implantação de modelos ágeis em seus processos conta com um quadro de aproximadamente 3000 colaboradores e, em específico, cerca de 35 pessoas em sua fábrica de software. No dia-a-dia de trabalho das pessoas na fábrica de software, surgem demandas de novos projetos, criação de software, desenvolvimento, análise, especificação de requisitos, manutenção e gestão de projetos, entre outras atividades comuns para a criação de um produto de software e definidos pela Engenharia de Software.

Para cada atividade citadas anteriormente, existe documentado e descrito todas as entradas para as atividades, ou seja, o que é necessário para começar tal atividade e o que deverá ser entregue ao final de cada atividade. Isso se faz necessário para manter maior organização dentro da fábrica de software para ate as exigências das normas CMMI, ao qual a empresa situa-se no Nível 3 e, MPS-BR, ao qual conquistou o nível C de maturidade de seus processos.

Estas normas possuem exigências aos quais diversos clientes impõem ao contratar um serviço de criação de software. Isso, também, para a disputa de licitações que envolvem desenvolvimento de software, os editais obrigam que as empresas concorrentes possuam níveis de maturidade adequados aos níveis previstos nas normas CMMI e MPS-BR.

Apesar deste cenário a favor da empresa e como ela vive em prol da inovação e tecnologia em todos os seus setores, ela se viu em um cenário de mudanças em relação aos seus processos de desenvolvimento. Mesmo contando com um processo maduro o suficiente para realizar suas entregas com qualidade e que agregue valor ao seu cliente, ela se viu capaz de melhorar a forma de atendimento de seus clientes, visando atender melhor as expectativas de seus clientes, de forma ágil e seguindo paradigmas que estão ganhando mercado a cada dia que passa.

É nesse intuito que a empresa resolveu investir na utilização de práticas ágeis em seus processos de desenvolvimento, focando em suas entregas ao cliente, institucionalizando diálogos entre os integrantes do time, encorajando cada integrante do time a tomar decisões estratégicas para o projeto e o principal, utilizando práticas do SCRUM e do XP(Extreme Programming) para atender de forma satisfatória seus clientes.

3.2 Projeto

Baseando-se em que a empresa decidiu mudar seus processos de desenvolvimento de software para modelos ágeis de desenvolvimento, um projeto foi escolhido para utilizar as novas práticas de desenvolvimento.

3.2.1 Características do Projeto

Esse projeto escolhido foi obtido por meio de licitação, ao qual a empresa abordada obteve sucesso e aprovação para o desenvolvimento da solução. Ele teve início em janeiro de 2015, com prazo de término para janeiro de 2016 divididos em três entregas, ou seja, foram criadas três iterações de entrega para o cliente durante o ano válido para o edital.

O edital prevê que a empresa seja certificada nas normas CMMI nível 3 e MPS-BR nível C e possuir o conhecimento do método de desenvolvimento RUP (Rational Unified Process), ao qual a contratante utiliza em seus processos.

A empresa contratante possui um processo de visitas à seus clientes para verificar se eles possuem o perfil para receber algumas bonificações. Essas visitas são realizadas pelas empreiteiras terceirizadas e todos os insumos coletados nas visitas são anotados manualmente em papéis. Posteriormente essas informações coletadas são analisadas e de acordo com um conjunto de regras pré-definidas, os bônus são liberados para os beneficiários.

Esse projeto consiste basicamente em criar um aplicativo móvel que será utilizado para realizar visitas nas casas dos possíveis beneficiários. Ele irá coletar informações dos residentes, fotografar a casa, gravar as informações e, posteriormente, estas informações e fotos serão utilizadas para definir quais pessoas serão beneficiadas pela empresa.

Logo, o projeto consiste em duas aplicações previstas, ou seja, uma aplicação móvel e uma aplicação Web.

3.2.1.1 Aplicação Web

Capaz de gerenciar as visitas de acordo com as regras pré-definidas, ou seja, para que haja visitas previstas em um determinado município, uma série regras devem ser informadas e aprovadas para que haja o benefício para um município.

De acordo com o edital, essa aplicação web deverá seguir e possuir algumas tecnologias. Basicamente, o sistema deverá utilizar a linguagem C-Sharp e .Net, seguir a arquitetura MVC (Model, View e Controller), Entity Framework para modelagem e controle de transações com o banco de dados e para gerar relatórios será utilizado o Report Services como ferramenta padrão.

3.2.1.2 Aplicação Móvel

Esta aplicação será capaz de efetivar a visita nas casas de seus beneficiários. Nesta aplicação é que o usuário poderá anotar informações das visitas, fotografar os bens passíveis de sofrerem o benefício, salvar as informações e enviá-las para a aplicação Web.

De acordo com o previsto no edital, essa aplicação deverá funcionar em aplicativos que possuem o sistema operacional Android, irá utilizar a linguagem de programação Java, utilização da plataforma de desenvolvimento PhoneGap, utilização do apache Cordova, que é capaz de simular e acessar funções de aparelhos móveis como câmera, geolocalização e permite aos desenvolvedores criarem aplicações utilizando HTML5, CSS3 e Javascript.

3.2.2 Equipe do projeto

Para a realização e comprometimento com o projeto, a empresa necessitou apresentar a equipe e seus papéis que irão atuar no projeto, conforme previsto no edital. Essa equipe é composta por 10 colaboradores que estão distribuídos de acordo com seus papéis da seguinte forma:

- Um Gerente de Projetos: responsável por gerenciar a equipe, fazer a comunicação entre as empresas e negociar prazos.

- Um Arquiteto de Soluções: responsável por ser um facilitador no entendimento e conhecimento das tecnologias utilizadas no projeto. Também é responsável por definir a arquitetura base dos sistemas, arquitetura de banco de dados e os pontos de comunicação entre os sistemas.

- Dois Analistas de Requisitos: responsáveis por realizar as entrevistas para definir escopo, requisitos, casos de uso e regras que deverão conter nos sistemas.

- Quatro Analistas Desenvolvedores: são responsáveis pelo desenvolvimento e criação dos sistemas baseando-se nos casos de uso. São ainda divididos em duas equipes pelo qual, três fazem parte da equipe que desenvolve a aplicação Web e um desenvolvedor da aplicação móvel.

- Dois Analistas de Testes: responsáveis pela garantia da qualidade do software que será entregue para o cliente.

Dessa forma, a equipe que deverá atender as exigências declaradas no edital ficou definida.

3.3 Implantação e Incentivo

Definida a equipe que atendeu o projeto, a empresa se viu em um momento de criar e repassar conhecimento. A empresa passou a fornecer treinamentos, incentivos a leitura de livros específicos para metologias ágeis e demonstrando como seria em uma situação hipotética o uso de processos ágeis em um projeto de desenvolvimento na fábrica de software.

Devido o pensamento da empresa estar voltado para treinamento de seus colaboradores, houve um incentivo para que fosse realizado treinamentos externo em processos ágeis e Scrum. Essa estratégia teve o intuito de fornecer conhecimento suficiente para se certificarem e aplicar a metologia na fábrica de software.

O treinamento também teve o objetivo de especializar pessoas em modelos de desenvolvimento ágeis e que pudessem repassar conhecimento sobre esses modelos, planejando e modelando a forma de trabalho dentro da fábrica de software, pregando a importância de se seguir o modelo ágil conforme treinamento interno fornecido.

Aos poucos, com repasse do treinamento, livros e textos disponibilizados, a equipe foi conhecendo e entendendo como uma fábrica de software se comporta em um ambiente de trabalho ágil e foi a partir desse momento que a empresa tomou a decisão de se utilizar métodos ágeis em um projeto da fábrica.

3.3.1 Processos e Práticas Ágeis

Definida a equipe do projeto e com todos os integrantes possuindo conhecimento suficiente para utilizar a nova metodologia de desenvolvimento, foi necessário adequar o modelo de desenvolvimento ágil as suas práticas tradicionais de sucesso. Essa adequação foi motivada pelo fato da empresa e seus colaboradores não abandonarem totalmente suas práticas de sucesso tradicional e pelo fato desse projeto em específico possuir regras em seu edital que cita práticas do modelo RUP e que devem ser atendidas para a entrega do produto.

Visto o cenário de mudanças, os processos e práticas ágeis ficaram definidos da seguinte forma:

3.3.1.1 Fase 1: Pré - Projeto

É uma etapa que contempla as atividades que são executadas antes do projeto ser iniciado. Nessa fase, são realizadas as tarefas de gerenciamento de portfólio, entendimento inicial dos requisitos, estimativas iniciais para desenvolvimento do projeto, análise inicial do projeto e a alocação dos recursos humanos que irão atuar no projeto.

Nesta fase não são utilizados as práticas dos métodos ágeis.

3.3.1.2 Fase 2: Especificação e Planejamento do Projeto

Nesta fase contempla o planejamento e a especificação técnica do projeto. É realizado o planejamento das atividades de cada recurso, seguido de um levantamento de requisitos contendo os requisitos de cliente, requisitos funcionais, requisitos não funcionais e casos de uso.

É realizado o refinamento das estimativas levando em consideração os requisitos apontados. Também são definidas quantas iterações levará para o sistema ser concluído. Essas iterações podem ser consideradas as Sprints do projeto.

Nesta etapa utiliza-se o Release Planning, vindo do Scrum, que define as iterações necessárias para atender o projeto e quais casos de uso serão utilizados em cada iteração, ou seja, são utilizados os casos de uso para se apoiar o desenvolvimento contrariando o princípio de se utilizar as Estórias de Usuário, mas o resultado se torna o mesmo se utilizado de forma correta.

As estimativas são baseadas em um método criado e utilizado pela própria empresa.

3.3.1.3 Fase 3: Análise, Projeto, Construção, Homologação e Implantação (sprint)

Como o próprio nome indica, nessa fase são realizadas atividades relacionadas à análise, projeto, construção, homologação e implantação de alguns casos de uso.

Para esta etapa, são utilizadas as práticas do Scrum Sprint Planning no qual os casos de usos são escolhidos para a Sprint da iteração, bem como, os casos de usos são detalhados e analisados.

Desta forma, cada caso de uso é trabalho por toda equipe do projeto, desde a análise, passando pelo desenvolvimento, testes, até a homologação interna, sempre valorizando o caso de uso mais importante para o cliente.

São realizadas as reuniões diárias, Daily Sprint, para monitoramento do projeto com a participação da equipe do projeto e do Gerente de Projetos.

As reuniões de Sprint Retrospective acontecem também ao final de cada Release entregue com o intuito de recolher lições aprendidas e melhorar o processo para a próxima Sprint. Esta reunião também tem o objetivo de transparecer e evidenciar aspectos acontecidos durante o projeto que não foram discutidos durante a execução do projeto devido a uma série de motivos. Dessa forma cria-se uma equipe homogênea e multidisciplinar com o intuito de favorecer tanto os clientes quanto a empresa.

3.3.1.4 Fase 4: Treinamento

Nesta fase é feito o planejamento e execução dos treinamentos necessários para que o cliente possa usar e operar o produto entregue. Esta não é uma fase obrigatória, pois o

cliente pode ou não solicitar treinamento. Práticas ágeis nessa fase não são utilizadas.

3.3.1.5 Fase 5: Produção

Esta é uma fase voltada para que o cliente possa tirar dúvidas e corrigir problemas encontrados por eles. Dessa forma, com os problemas encontrados, novos Sprints podem ser planejados para o projeto.

Assim, os processos a serem seguidos nesse projeto ficaram definidos e para novos projetos que possuem mesmas características essas práticas podem ser utilizadas.

3.3.2 Ferramentas de Apoio

Para auxiliar o novo processo implantado na fábrica de software da empresa, foram necessários apoios de algumas ferramentas, cujo conhecimento de Engenharia de Software seja disseminado de maneira controlável.

As ferramentas utilizadas pela empresa nesse projeto foram as seguintes:

- **Microsoft Project:** É a ferramenta utilizada para definir, controlar e adequar o cronograma de um projeto. Todas as atividades foram elaboradas em sequência, com a dependência entre elas, situando os papéis responsáveis por cada atividade a serem executadas. Um template já utilizado pela empresa também foi usado para o planejamento das atividades desse projeto, porém, realizando algumas adaptações para prover as atividades derivadas das adaptações sofridas no processo de desenvolvimento.

- TFS (Team Foundation Server) - Ferramenta utilizada pelo cliente, ao qual, de acordo com o edital, é necessário fazer o uso para criação de casos de uso, documentos de arquitetura, arquitetura de banco de dados, releases entregues do sistema, entre outros. Essa ferramenta possui padrões/templates que são usados para criação dos documentos que foram citados anteriormente. Sendo assim, as elaborações dos documentos ficaram mais fáceis e rápidas, pois, faz-se necessário preencher apenas as seções que podem ser alteradas e adaptadas.

- Subversion (SVN) e TotoiseSVN - Foi utilizada para o projeto, como controle de versões, a ferramenta SVN, pelo qual possível fazer a gerência de configuração de todo o código fonte. A teoria relacionada a Branchs e Tags foram utilizadas para definir o processo de Gerência de Configuração, o que gera linhas de base para controle dos itens de configuração. A ferramenta Tortoise é utilizada para fazer uma interface amigável de acesso aos repositórios do SVN.

- Enterprise Architect - Ferramenta utilizada para análise e projetos de caso de uso antes de ser submetido ao TFS. É usada para criar modelos de arquiteturas, comunicação entre componentes e a manutenção da rastreabilidade desses componentes.

- NetProject - Esta é uma ferramenta utilizada para controlar e planejar as atividades do projeto de acordo com os colaboradores alocados no projeto. Permite emitir uma visão do andamento do projeto. Também é utilizada para lançamento do tempo gasto em cada atividade utilizado pelos colaboradores.

3.4 Projeto: Comportamento da Equipe

Após ter escolhido o projeto piloto, definir a equipe desse projeto, definir o processo a ser utilizado e as ferramentas de apoio, é possível colocar em prática toda a filosofia de desenvolvimento ágil. Sendo assim, pode-se realizar observações para captar aspectos que demonstram se o time esta caminhando de encontro ao sucesso e maturidade do processo, se os colaboradores estão cumprindo com os papéis definidos pra eles, se estão cumprindo com o processo definido e se estão gostando de praticar esse novo paradigma.

Lembrando que, o projeto foi dividido em 3 iterações, logo, muitas situações foram mudando no decorrer de cada iteração fornecendo assim fatores observados de crescimento, maturidade e aceitação do processo.

3.4.1 Primeira Iteração

A primeira iteração foi o início das práticas pregadas pelos modelos Scrum e XP dentro da empresa. Como este foi o projeto piloto, algumas atividades foram traçadas para serem seguidas durante o projeto.

3.4.1.1 Cronograma e Atividades Iniciais

Algumas metas foram definidas para que a equipe pudesse se adaptar as novas tecnologias que o projeto exigia. Esse foi um ponto crucial de aprendizado entre os colaboradores do time de desenvolvimento. Nesse momento foram realizadas pesquisas e estudos sobre melhores práticas para criação do sistema. Uma Prova de Conceito foi desenvolvida para provar que as tecnologias utilizadas seriam de grande valor e, o principal, o envolvimento de todos os desenvolvedores trabalhando juntos, de forma homogênea, com o intuito do aprendizado rápido e benéfico.

Nessa primeira iteração, ficou definido o cronograma com as atividades de cada colaborador, atividades essas que são relacionadas aos casos de uso elencados na Fase 2 do desenvolvimento do projeto.

3.4.1.2 Daily

Houve a definição do horário para realização das Dailys, que eram todos os dias as 11:30 da manhã. Inicialmente a equipe abraçou a ideia sobre a realização das Dailys, porém, o horário muitas vezes não era respeitado conforme prega a metodologia. Inúmeras vezes a reunião começava atrasada porque os colaboradores não possuíam a prática dessa realização.

Ainda durante essa iteração, com a dedicação das pessoas responsáveis pela implantação do novo paradigma, as Dailys passaram a ser realizadas as 11:00 da manhã, por ser o melhor horário comum a todos e os próprios integrantes do time começaram a cobrar a realização das reuniões no horário marcado.

Apesar de uma Daily possuir a característica de duração de no máximo 15 minutos, dependendo do andamento do projeto ela estava se delongando por 30 a 40 minutos. O simples motivo era o fato do time ainda não ter abstraído o foco da Daily, que era: “o que fiz?”, “o que vou fazer?” e “Possuo algum impedimento que não me deixa continuar?”. Os integrantes do time começavam a discutir uma solução para um possível impedimento de um integrante do time.

Além disso, muitas vezes a Daily possuía características voltadas para cobrança de prazo e esforço para se terminar uma determinada tarefa. Efeitos esses, às vezes causados pela participação do gerente de projetos na daily. A participação do gerente não é proibida, mas este deve seguir os princípios da daily para que ela seja produtiva e que consiga aproximar os integrantes do time. Qualquer questão fora do escopo da daily deve ser tratado em um momento à parte.

3.4.1.3 Quadro Kanban

Passaram-se a utilizar o quadro kanban como forma de demonstrar as tarefas que estão alocadas e sendo executadas por cada componente do time. O quadro possui as três colunas padrão: "A Fazer", "Fazendo"e "Feito".

Como ainda era muito no início, as primeiras tarefas foram colocadas no quadro como sendo o próprio caso de uso. Uma tarefa "A Fazer"possuía o título de um caso de uso, logo, o kanban ficava um período longo sem atualizações.

3.4.1.4 Reunião de Retrospectiva

Ao final da iteração, após a entrega do pacote correspondente, foi realizada a Sprint Retrospective, com o intuito de verificar o que funcionou, o que não funcionou e apresentar propostas para alterar o que não funcionou de forma a tornar a próxima Sprint mais produtiva.

Pela primeira vez na empresa, uma reunião desse tipo foi realizada, com o intuito de melhorar práticas que não funcionaram, trazendo retorno nos resultados da empresa.

Esta reunião teve um prazo além do esperado, porém, nela foi possível elencar vários pontos que não estavam evidentes anteriormente. Algumas práticas que não funcionaram surtiram algumas propostas de melhorias conforme a seguir:

- Integrantes do time devem assumir o comprometimento da realização da Daily no horário especificado.
- Geração de pacotes de publicação de uma única maneira e centralizado.
- Integrantes buscarem soluções para seus problemas primeiramente entre o time, de forma conjunta e madura.
- Equipe de Levantamento de Requisitos escreverem os casos de uso com o apoio do desenvolvedor.
- Repasse do caso de uso aprovado para o desenvolvedor responsável.

Desse modo ficou estipulada as novas metas de melhoria para a próximo iteração/s-print conforme pode-se ver na Figura 2 e Figura 3.

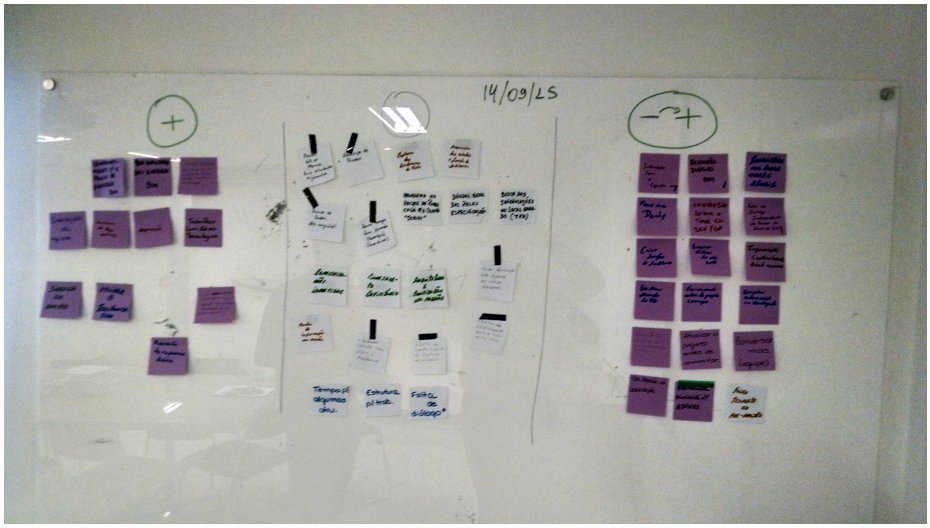

Figura 2: Quadro gerado com a reunião de retrospectiva

3.4.2 Segunda Iteração

A segunda iteração iniciou-se antes mesmo da reunião de retrospectiva da fase anterior ter ocorrido, mas mesmo assim, as melhorias abordadas nessa reunião foram adotadas com sucesso durante a segunda iteração.

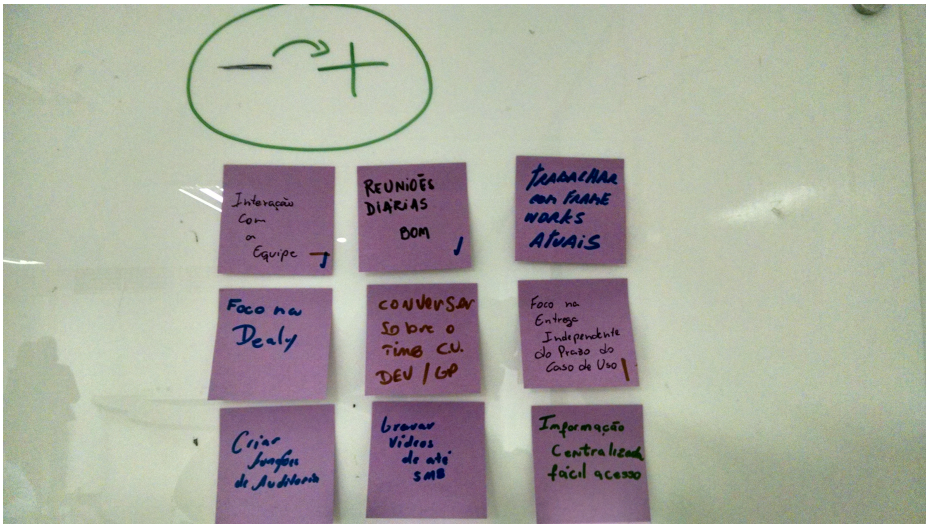

Figura 3: Coluna de melhorias

3.4.2.1 Daily

As reuniões diárias passaram a seguir o horário como uma rotina diária de uma pessoa. Quando algum integrante do time faltava ou simplesmente não se lembrava da Daily, ele era cobrado pelos outros integrantes.

Durante o andamento da segunda iteração, muitas vezes durante a Daily, um integrante era lembrando sobre qual o foco que a Daily deve conter, ou seja, as três perguntas básicas: "O que fiz?", "O que vou fazer?", "Possuo algum impedimento?". Isso acontecia porque alguns integrantes citavam soluções criadas para atender um de seus impedimentos ou até, outros integrantes interrompia as pessoas para emitir seu posicionamento perante a alguma discussão.

Nesse contexto, é que se gerou uma discussão maior sobre o que é o foco da reunião diária o qual o Scrum defende. Os componentes do time que possuíam um pensamento mais tradicional, que sempre trabalharam com modelos tradicionais, juntamente com o Gerente de Projetos, sempre defendiam que a reunião deveria conter mais informações sobre o andamento do projeto e as soluções para alguns impedimentos, infringindo algumas regras:

- Daily deve durar no máximo 15 minutos: reuniões longas são boas formas para começar mal o dia, acaba com a energia das pessoas.
- Informar impedimentos: impedimentos pequenos podem ser resolvidos rapidamente na própria reunião diária, porém, caso contrário, a solução deve partir do Scrum Master em outro momento que não seja durante a reunião.

Após conversa e apresentação sobre o princípio ágil ao qual a Daily é inserida, os

integrantes começaram a concordar melhor sobre o "foco" da reunião. Mesmo as vezes contrariados, estes resolveram aderir ao método com o intuito de testar e verificar se a reunião estava sendo produtiva ou não.

3.4.2.2 Quadro Kanban

O quadro estava sendo usado, porém sem muitas atualizações. Muitas vezes o quadro era esquecido e a equipe não estava atualizando o quadro de maneira correta.

Um fato que acarretou o desleixo em relação ao quadro foi a falta de divisão das tarefas. Muitas vezes um componente da equipe possuía como tarefa um caso de uso inteiro, logo, existia de fato uma demora para atualização do quadro.

Com esses problemas, o quadro acabou sendo deixado de lado e passou a não ser utilizado durante a segunda iteração.

3.4.2.3 Reunião de Retrospectiva

Esta segunda reunião de retrospectiva foi marcada por ser a mais produtiva. Houve muitos debates, problemas levantados, discussões sobre melhores práticas e soluções adotadas para quase todas.

Nesta reunião foram observados os seguintes pontos:

- O Scrum Master deve estar mais presente e mais ativo com a equipe do projeto
- Dúvidas devem ser perguntadas e discutidas pessoalmente, evitando utilizar e-mails para tal.
- Um modelo do processo será descrito e fechado com o intuito de ser seguido e formalizado na empresa.
- As tarefas serão subdivididas para gerar entregas menores
- Passar a utilizar o kanban online

Os pontos observados ficaram registrados na Figura 4 e Figura 5.

3.4.3 Terceira iteração

Nesta fase estava prevista a última iteração e última entrega ao cliente. Alguns pontos que puderam ser observados demonstrando que o pensamento "agilista" esta tomando força no setor.

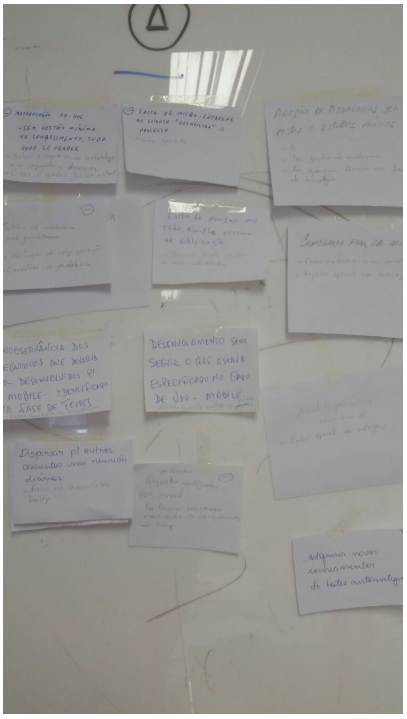

Figura 4: Quadro geral com as melhorias propostas

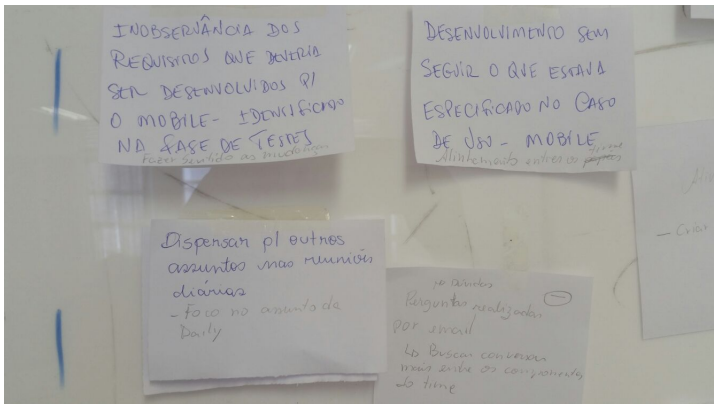

Figura 5: Coluna de melhorias

3.4.3.1 Daily

Não apresentou problemas ou discussões na terceira iteração. Todos integrantes do time passaram a seguir a Daily como ela deve ser.

3.4.3.2 Quadro kanban

Também não foi nessa iteração que o quadro passou a ser utilizado.

3.4.3.3 Jenkins

Nessa iteração a equipe deu um passo importante para que a metodologia ágil implantada na fábrica de software seja eficiente.

É nesse sentido que foi implantada para esse projeto a ferramenta Jenkins, que tem o intuito de montar a aplicação que está sendo desenvolvida em um ambiente de testes. Esta ação pode ser programada para ser realizada em períodos pré-determinados de tempo ou por uma regra pré-determinadas. Essa ferramenta ajuda na detecção rápida de erros que foram inseridos no SVN pelos desenvolvedores.

Esta ferramenta provém da integração contínua entre o SVN e a construção do projeto que o Jenkins faz automaticamente. Dessa forma, essa prática está diretamente ligada a desenvolvimento ágil e principalmente a metodologia XP. Essa integração visa garantir que qualquer alteração realizada no código fonte seja integrada e validada o mais rápido possível dentro da aplicação.

Além destas característica, a ferramenta possui um retorno imediato e preciso quando existem falhas, cabendo ao integrante do time de desenvolvimento solucionar o problema o quanto antes. Ela também executa, quando configurada para fazer, os testes unitários criados pelos desenvolvedores e essa prática esta sendo usada neste projeto.

3.4.3.4 Selenium - Testes automatizados

Outra ferramenta aliada ao grande avanço do modelo ágil nesse projeto é o Selenium. O principal objetivo dessa ferramenta é automatizar ações do navegador, tais como, inserção de informações em um cadastro, alteração, testes de campos obrigatórios, entre outros.

Esta ferramenta provê a seguinte situação: a equipe de teste e qualidade do sistema se vê em um momento de regressão em todo o sistema, ou seja, testar tudo que o sistema já faz. Para projetos que estão no início, é fácil realizar a regressão, porém quando existem várias sprints e o sistema cresce muito, isso se torna inviável. Com esta ferramenta, os testes automatizados vão sendo criados a medida que as sprints vão se concluindo e o teste automatizado ajuda a não deixar que uma modificação nova impacte negativamente outros pontos do sistema.

3.4.3.5 Reunião de Retrospectiva

Para a terceira iteração não foi realizada a reunião porque a equipe decidiu esperar conclusão do cliente e correção de possíveis falhas encontradas no sistema.

4 Conclusão

A proposta inicial estabelecida para implantar um modelo ágil em uma fábrica de software de uma empresa privada foi realizada no decorrer do desenvolvimento desse trabalho. Essa proposta se mostrou eficaz, pois, a partir dela obteve-se:

- Informações suficientes, a partir de reuniões, para traçar a solução em cada etapa
- Formação de pessoas com o intuito de fomentar o pensamento ágil no ambiente de trabalho
- Incentivo a implantação e utilização de novas ferramentas que fazem o modelo ágil ser eficaz
- Interação entre as pessoas e um ambiente melhor para se trabalhar

Desse modo, os objetivos, geral e específicos do trabalho foram atingidos. Considerando que hoje, esse processo foi ponderado como sucesso pelos gestores, pois, surtiu efeito com relação a entregas dentro do prazo e comprometimento da equipe. Sendo assim, novos projetos com mesmas características irão adotar a metodologia ágil, talvez com algumas mudanças que serão discutidas antes de iniciá-los.

Mudanças estas que podem estar relacionada com os seguintes pontos de melhorias que podiam acontecer:

- Equipe de desenvolvimento melhorar os testes de unidade para usá-los continuamente com o Jenkins
- Disseminar e melhorar o desenvolvimento de testes automatizados executados pelo Selenium
- Diminuir o tamanho do time, focando em pessoas auto-gerenciáveis que são capazes de exercer diversos tipos de papéis durante o projeto.
- Criar o kanban de acordo com a realidade da empresa, com as fases que existem no projeto
- Criar punições para falta ou atraso na reunião diária, de forma a ser uma prática que os componentes gostem e se torne competitivo
- Dividir melhor as Sprint em entregas menores e utilizar Dashboards, que são quadros demonstrando a situação de cada caso de uso em desenvolvimento.

- Deixar apenas ao Scrum Master fornecer informações sobre o andamento do projeto, de preferência semanalmente
- Aprender e melhorar os testes de unidade que serão utilizados pelo Jenkins, fornecendo melhor resposta a erros que o Jenkins provê.

Portanto, o trabalho foi essencial para demonstrar como se transforma um ambiente de uma fábrica de software quando se muda seu processo de desenvolvimento, principalmente quando o assunto é metodologia ágil. Esse trabalho pode demonstrar situações vivenciadas por uma equipe, o crescimento e maturidade obtida por tal.

Referências

ABRAHAMSSON, P. et al. New directions on agile methods: A comparative analysis. In: *Proceedings of the 25th International Conference on Software Engineering*. Washington, DC, USA: IEEE Computer Society, 2003. (ICSE '03), p. 244–254. ISBN 0-7695-1877-X. Disponível em: <http://dl.acm.org/citation.cfm?id=776816.776846>. Citado na página 25.

ALLIANCE, S. *Who is the Scrum Alliance?* Acesso em: 13/09/15: http://www.scrumalliance.org/, 2015. Citado na página 26.

BASSI, D. *Planejamento agil de projetos*. Rio de Janeiro: Engenharia de Software Magazine, 2009. Citado na página 20.

BECK, e. a. *AGILE MANIFESTO*. Acesso em: 12/09/15: http://www.agilemanifesto.org/, 2015. Citado na página 24.

CALCADO, V. L. X. d. S. *Influência da Utilizacão de Processo Unificado, Testes e Metricas na Qualidade de Produtos de Software*. Brasilia: Universidade de Brasilia, 2007. Citado na página 21.

FREIRE, A.; KON, F.; TORTELI, C. *XP south of the equator: An experience implementing XP in Brazil*. [S.l.]: Springer, 2005. 10–18 p. Citado 2 vezes nas páginas 15 e 30.

HIGHSMITH, J. *Agile Software Development Ecosystems*. Addison-Wesley Pearson Education: Foreword by Tom De Marco., 2002. Citado na página 25.

ILIEVA, S.; IVANOV, P.; STEFANOVA, E. *Analyses of an Agile Methodology Implementation*. [S.l.]: Conference. IEEE, Washington, DC, 2004. 326-333 p. Citado 2 vezes nas páginas 15 e 30.

JUNG, C. *Metodologia para pesquisa e desenvolvimento: aplicada a novas tecnologias, produtos e processos*. Rio de Janeiro: Axcel Books, 2004. Citado na página 16.

KOSCIANSKI, A.; SOARES, M. S. *Qualidade de Software. Aprenda as metodologias e técnicas mais modernas para o desenvolvimento de software*. [S.l.]: 2ª Edição, 2007. Citado na página 26.

LOFTUS, C.; RATCLIFFE, M. *Extreme programming promotes extreme learning?* [S.l.: s.n.], 2005. 311-315 p. Citado 2 vezes nas páginas 15 e 29.

MAIA, L. T. *Um Estudo sobre Aplicacão de Técnicas de Inteligência Artificial e Engenharia de Software à Construcão de um Sistema de Supervisão e Controle*. Brasilia: Universidade de Brasilia, 2007. Citado na página 19.

POPPENDIECK, M.; POPPENDIECK, T. *Implementando o desenvolvimento LEAN: do conceito ao dinheiro*. [S.l.]: Porto Alegre: Bookman, 2011. Citado na página 29.

PRESSMAN, R. S. *Software Engineering: A practitioner's approach*. McGraw-Hill: 8ª Ed., 2015. Citado 3 vezes nas páginas 19, 21 e 22.

SBROCCO, J.; MACEDO, P. *Metodologias Ágeis - Engenharia de Software sob medida*. [S.l.]: Ética, 2012. Citado 2 vezes nas páginas 15 e 29.

SCHWABER, K. *Agile Project Management with Scrum*. 1ª Edição: Microsoft Press, 2004. Citado 2 vezes nas páginas 25 e 26.

SCHWABER, K.; BEEDLE, M. *Agile Software Development with Scrum*. [S.l.]: Prentice Hall, 2002. Citado 2 vezes nas páginas 15 e 21.

SEABRA, J. *Aplicando as Melhores Praticas do PMBok no Planejamento para a Adaptacão e Implantacão do Rational Unified Process*. Brasilia: Unicesp, 2013. Citado na página 23.

SEABRA, J. *UML: Uma ferramenta para o design de software*. Rio de Janeiro: Ciência Moderna, 2013. Citado 2 vezes nas páginas 19 e 20.

SOARES, M. d. S. *Comparacão entre Metodologias Ageis e Tradicionais para o Desenvolvimento de Software*. Lavras: Infocomp – Revista de Ciência da Computacão, 2004. Citado na página 21.

SOMMERVILLE, I. *Software Engineering*. UK: Pearson Education: 8ª Ed. Harlow, 2008. Citado 5 vezes nas páginas 15, 19, 21, 22 e 23.

SVENSSON, S.; HöST, R. *Views from an organization on how agile development affects its collaboration with a software development team*. [S.l.]: Springer Verlag, Berlin, 2005. 487–501 p. Citado 2 vezes nas páginas 15 e 29.

VASCONCELOS, A. M. L. d. et al. *Introducão à Engenharia de Software e à Qualidade de Software*. Lavras - MG: Universidade Federal de Lavras – UFLA/FAEPE, 2006. Citado 3 vezes nas páginas 20, 22 e 23.

WELLS, D. *Extreme Programming: A gentle introduction*. [S.l.]: http://www.extremeprogramming.org, 2009. Citado na página 28.