UNIVERSIDADE FEDERAL DE SANTA CATARINA
CAMPUS ARARANGUÁ

Paula Alicia Lessa Paulo

USO DE METODOLOGIA ÁGIL EM UMA EMPRESA DE MEIO DE PAGAMENTO PÓS-PAGO: ESTUDO DE CASO NA EMPRESA KOIN

Araranguá, julho de 2014.

Paula Alicia Lessa Paulo

USO DE METODOLOGIA ÁGIL EM UMA EMPRESA DE MEIO DE PAGAMENTO PÓS-PAGO: ESTUDO DE CASO NA EMPRESA KOIN

Trabalho de Conclusão de Curso submetido à Universidade Federal de Santa Catarina, como parte dos requisitos necessários para a obtenção do Grau de Bacharel em Tecnologias da Informação e Comunicação.

Araranguá, julho de 2014.

Paula Alicia Lessa Paulo

USO DE METODOLOGIA ÁGIL EM UMA EMPRESA DE MEIO DE PAGAMENTO PÓS-PAGO: ESTUDO DE CASO NA EMPRESA KOIN

Este Trabalho de Conclusão de Curso foi julgado aprovado para a obtenção do Título de Bacharel em Tecnologias da Informação e Comunicação, e aprovado em sua forma final pelo Curso de Graduação em Tecnologias da Informação e Comunicação.

Araranguá, julho de 2014.

Prof. Vilson Gruber, Dr.
Coordenador do Curso

Banca Examinadora:

Prof.ª Luciana Bolan Frigo, Dr.ª
Orientadora

Prof.ª Eliane Pozzebon, Dr.ª

Prof. Eugênio Simão, Dr.

Dedico este trabalho a minha família que é a base de tudo que sou hoje. Ao meus irmãos por serem meus heróis e por sempre me apoiarem desde o início do curso.

AGRADECIMENTOS

Agradeço primeiramente a Deus por me iluminar nesta caminhada. A minha professora-orientadora por me incentivar a não desistir nos momentos mais difíceis e sempre buscar aperfeiçoamento no meu trabalho. Aos meus pais que nos momentos de estresse sempre me confortaram com suas doces palavras para que tivesse calma e paciência nos momentos difíceis. Em especial ao meus grandes amigos Ellen e Leandro e por fim, a todos meus amigos que estiveram comigo nesta jornada.

"Sonhos determinam o que você quer. Ação determina o que você conquista".

Aldo Novak

RESUMO

Devido a grande competitividade entre as organizações a busca por entregar produtos com qualidade e na velocidade que o mercado atualmente exige, faz com que muitas empresas adotem métodos que dizem suprir estas necessidades. Os métodos ágeis prometem atender tais requisitos valorizando as pessoas e as interações entre elas, ao invés de focar nos processos e nas ferramentas como as metodologias clássicas. E por estes motivos, a empresa Koin resolveu adotar no seu processo de desenvolvimento o Scrum, com o intuito de melhorar a produtividade e a qualidade dos seus produtos.
Esta monografia tem como objetivo documentar e analisar os impactos ocasionados pela implantação de princípios ágeis, nos processos e nas pessoas envolvidas acerca dos fatores que influenciam positivamente ou negativamente na implantação.

ABSTRACT

Due to the high competitiveness between organizations Search by delivering quality products and the speed that the market currently demands, causes many companies to adopt methods they say meet these needs. Agile methods promise meet such requirements valuing people and the interactions between them, rather than focusing on the processes and tools such as classical methodologies. And for these reasons, the company decided Koin adopt in its development process Scrum, in order to improve productivity and quality of their products.
This monograph focuses on documenting and analyzing the impacts caused by the deployment of agile principles, processes and the people involved about the factors that influence positively or negatively on the implementation.

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

SUMÁRIO

1 INTRODUÇÃO

Na busca por lucro, as empresas desenvolvedoras de software procuram por uma metodologia que venha aperfeiçoar a entrega de seus produtos com qualidade. Com as metodologias ágeis a entrega do produto na qual o cliente adquiriu é reduzida em relação aos processos tradicionais de desenvolvimento, além disso traz proveito para todos os envolvidos pois estão estruturados, criando um software com mínimo de recursos desperdiçados(FADEL; SILVEIRA, 2010).

Os métodos ágeis foram criados principalmente para organizações em que os requisitos especificados pelo cliente fossem alterados durante o processo de criação do software. A metodologia ágil quando inserida em uma organização é capaz de introduzir interatividade entre o cliente e a equipe de desenvolvimento. Desta forma, o cliente tem envolvimento direto com a equipe, podendo alterar ou propor novas soluções ao software durante seu processo de desenvolvimento (SOMMERVILLE, 2007).

Esta modificação resulta aumento de produtividade da equipe, consequentemente, acelera a entrega do produto para o cliente, proporcinando assim sua satisfação (MILLER; SY, 2009). Com essas mudanças que os métodos ágeis são capazes de introduzir em uma organização, empresas que adotam esse método vem ganhando espaço e se destacando no mercado desenvolvedor de software (MILLER; SY, 2009).

O Manifesto Ágil é a base de todos os métodos ágeis, valorizando os indivíduos e interações mais que processos e ferramentas, entre outros. A proposta da cultura ágil é fornecer uma gestão que seja mais adaptável e empreendedora. Empresas ligadas ao setor de tecnologia possuem este aspecto da agilidade ainda mais evidente e, portanto a adoção de uma cultura ágil se torna uma necessidade.

Este trabalho aborda um estudo de caso, a ser realizado na empresa Koin. Segundo o site da empresa,a Koin é uma empresa desenvolvida no segmento chamado "Fin Tec", ou seja, empresas cujas atividades unem serviços financeiros e tecnologias. Formada por profissionais das áreas de *e-commerce*, meios de pagamento e finanças, a Koin opera sob uma plataforma proprietária de alta tecnologia e inovação(KOIN, 2014).

1.1 JUSTIFICATIVA E DEFINIÇÃO DO PROBLEMA

A palavra agilidade vem sendo utilizada e gerando mudanças, pois a maioria das empresas que adotaram esta metodologia obtiveram vantagens em produzir um software de forma acelerada e de qualidade, atender as necessidades do cliente, aumentar a produtividade e possibilitar uma excelente visualização do desenvolvimento tornando o processo mais previsível (COHN, 2011). Equipes ágeis são aquelas que são capazes de acompanharem as mudanças que pode ter um grande impacto sobre o projeto e o produto final.

Durante algum tempo foi observada uma rápida tranformação tecnológica, cultural, organizacional, social e econômica nos sistemas das organizações. Estas transformações passaram a exigir das empresas mudanças culturais e comportamentais, fazendo com que suas equipes buscassem uma forma diferente de atingir seus objetivos. Uma metodologia quando inserida em uma organização tem presente poder sobre a transformação estrutural e organizacional de uma empresa. Em um ponto estrutural, as organizações se tornam ágeis para melhor concorrer com o mercado, e já no ponto organizacional é capaz de fazer com que a empresa busque sempre se atualizar em relação a aumentar sua produtividade e fazer com que seus clientes estejam sempre satisfeitos (ALMEIDA; MARçAL; KOVALESKI, 2004). Diante da teoria apresentada o presente trabalho tem como foco responder a seguinte pergunta:

A adoção de uma metodologia ágil realmente traz benefícios ao processo de desenvolvimento de software melhorando o desempenho das equipes, aumentando a produtividade e a qualidade do produto final gerando lucros mais significativos?.

Serão analisadas as seguintes hipóteses:

- A metodologia ágil melhora a produtividade da equipe;
- A metodologia ágil melhora a comunicação entres os membros da equipe facilitando a disseminação do conhecimento e a identificação dos problemas;
- A metodologia ágil, sob o ponto de vista do cliente, entrega um produto de qualidade mais rapidamente do que com outras metodologias;

1.2 OBJETIVOS

- **Geral**

  Avaliar o desempenho de uma equipe empresarial após a implantação do método ágil Scrum, com o objetivo de demonstrar os resultados obtidos.

Para alcançar o objetivo geral será necessário cumprir um conjunto de etapas delimitadas como objetivos específicos:

- **Específicos**

  1. Descrever o Manifesto Ágil;
  2. Apresentar algumas metodologias ágeis e suas particularidades.
  3. Analisar a aplicabilidade da metodologia Scrum na empresa Koin.
  4. Analisar o impacto gerado na adoção da metodologia Scrum e as mudanças que ocorreram com a aplicação deste método.

1.3 METODOLOGIA

A etapa inicial deste trabalho é composta pelo embasamento teórico sobre as características das metodologiaa tradicionais e ágeis, em especial o Scrum, utilizado como base para o estudo de caso na empresa Koin.

Para validar as hipóteses levantadas foram aplicados questionários na equipe de desenvolvimento da empresa. O objetivo destes questinários é analisar como era o processo antes da adoção do Scrum e coletar informações sobre os benefícios após sua utilização.

1.4 ORGANIZAÇÃO DO TRABALHO

Este trabalho, além da introdução está organizado em mais quatro capítulos. **O Capítulo 2** apresenta as definições dos processos de desenvolvimento de software abordando os métodos tradicionais e os métodos ágeis. Contribuindo com o tema de metodologia ágil serão apresentadas algumas ferramentas que auxiliam na adoção desta.

O Capítulo 3 tem como objetivo apresentar a metodologia Scrum adotada pela empresa Koin. Serão mostradas suas principais características. **O Capítulo 4** apresenta um estudo de caso realizado com o intuito de validar a aplicação da metodologia Scrum na empresa Koin. Este estudo mostra os principais resultados alcançados pela empresa após uma mudança cultural e organizacional.

O Capítulo 5 apresenta as considerações finais e algumas propostas para trabalhos futuros.

2 PROCESSO DE DESENVOLVIMENTO DE SOFTWARE

A engenharia de software surgiu juntamente com a crise do software em 1968, quando a maioria dos projetos sofriam atrasos, apresentavam custos maiores do que o planejado, tornando assim software de difícil manutenção e sem nenhuma credibilidade. Novos métodos foram criados para acompanhar a demanda no processo de criação a grandes sistemas de softwre (SOMMERVILLE, 2003).

A partir deste cenário, surgiu a necessidade de tornar o desenvolvimento de software como um processo estruturado, planejado e padronizado, para que as necessidades fossem atendidas e os gastos fossem compensados. Para estas necessidades foram criadas as metodologias de desenvolvimento (RUAS, 2011).

Metodologia de desenvolvimento é um conjunto de práticas recomendadas para o desenvolvimento de Software, sendo que essas práticas, geralmente, passam por fases que são subdivisões do processo para ordená-lo e melhor gerenciá-lo (SOMMERVILLE, 2003).

O processo de software é uma ordem de atividades que leva a produção do software, na qual quatro atividades essenciais são comuns (SOMMERVILLE, 2011):

1. Especificação de software - onde clientes e engenheiros de software definem as restrições e o que será desenvolvido;

2. Desenvolvimento de software - onde o software é criado;

3. Validação de software - processo em que o cliente verifica se o que foi feito esta de acordo com suas necessidades;

4. Evolução do software - o software obtém melhorias de acordo com a necessidade de mudança dos clientes.

Para se obter sucesso no projeto é necessário que todas as pessoas tenham conhecimento sobre o que será feito em cada etapa e que o projeto utilize métodos que façam com que as atividades envolvidas durante sua criação seja feita de forma planejada e clara (SOMMERVILLE, 2011).

2.1 CICLO DA VIDA DO DESENVOLVIMENTO DE SOFTWARE

> "Quando o processo envolve a elaboração de um produto, algumas vezes nos referimos a ele como ciclo de vida. Assim, o processo de desenvolvimento de software pode ser chamado de ciclo de vida do software, pois ele descreve a 'vida' do produto de software desde a conceppção até a implementação, entrega, utilização e manutenção"(PFLEEGER, 2004, p. 37).

O ciclo de vida do software é uma sequência de atividades que ocorre durante o processo de criação do software. Segundo (LEITE, 2000) dependendo do modelo de processo algumas atividades também podem estender-se por mais de uma fase do ciclo de vida. O autor identifica três fases presentes no ciclo de vida do software que serão descritas a seguir, são elas as fases de: definição, desenvolvimento e operação.

Na fase de **definição** é feito o levantamento das necessidades do cliente para que o projeto venha a ser desenvolvido. É de extrema importância o cliente oferecer detalhes sobre o sistema, sendo responsabilidade do analista, responsável por esta fase na maioria das organizações, verificar o que será viável construir ou não no software. Nesta fase são repassadas algumas informações como arquitetura, restrições do sistema, cabendo ao analista de sistemas verificar o que será importante ou não.

Sendo assim, ao final desta fase é feito um planejamento do que será desenvolvido, serão definidos o prazo de entrega, o custo do projeto, além da escolha do modelo de processo de desenvolvimento. A fase de definição deve ser feita de forma minuciosa para que resultado esperado saia de forma esperada.

A fase de **desenvolvimento** inclui pôr em prática todas as atividades propostas na fase de definição. Nesta fase são realizadas atividades como a codificação e verificação da interface junto ao cliente.

A fase de **operação** é onde é realizada a implantação do software no seu ambiente de utilização.

2.2 MODELOS DE PROCESSO DE SOFWARE

Os modelos de processo de software trazem benefícios para a organização pois ela é responsável pelo gerenciamento e planejamento das

atividades que serão realizadas. Os modelos quando bem aplicado dentro de uma organização faz com que seja mais possivel entender todos os passos. Um modelo é uma proposta que determina as atividades e as pessoas envolvida em cada uma delas (SOMMERVILLE, 2011).

Nesta sessão serão demonstrados alguns modelos de processo, são eles modelo cascata e modelo de desenvolvimento incremental. Existem vários tipos de modelos de processo sendo cada um com suas características. Os modelos variam de acordo com a organização e conforme o produto a ser desenvolvido.

2.2.1 Modelo Cascata

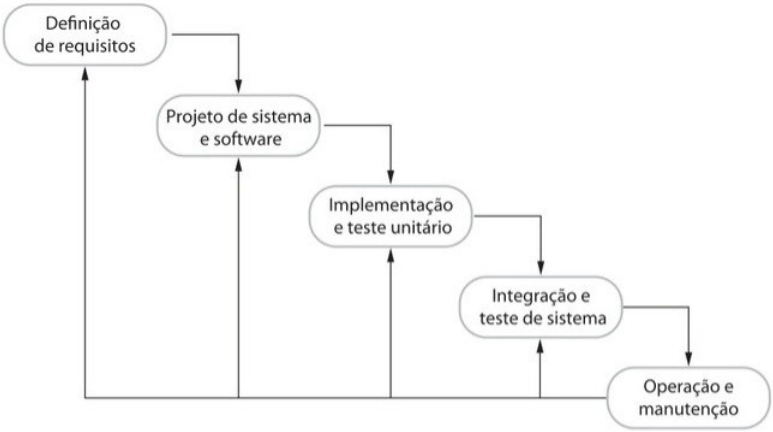

Figura 1 – Modelo Cascata
Extraído de (SOMMERVILLE, 2011).

Conforme ilustado na Figura 1, para (SOMMERVILLE, 2011) os principais estágios do modelo em cascata são:

1. Análise e definição de requisitos - são definidas metas, restrições do sistema, detalhes e funcionamento do sistema.

2. Projeto de sistema de software - são separados os requisitos para hardware e software, através de uma arquitetua geral do sistema. O projeto de software possui as descrições essencais do sistema de software e seus relacionamentos.

3. Implementação e teste unitário - Durante este estágio o projeto de software é desenvolvido em partes do programa, para que cada parte seja verificada a fim de atender o que foi especificado.

4. Integração e teste de sistema - Unidades individuais do programa são integradas e testadas garantindo assim se os requisitos estão sendo atendidos, logo após o teste o sistema é entregue ao cliente.

5. Operação e manutenção - É a fase que mais dura no estágio do ciclo de vida, pois o sistema é colocado no ambiente na qual ele será utilizado, e a manutenção vem pra corrigir erros caso ocorra.

Entendendo a forma como foi escrita por (SOMMERVILLE, 2011) o modelo cascata é aquele que deve ser usado quando os requisitos do projeto estiverem bem definidos e durante o processo, não haja alteração. No final de cada etapa do modelo é desenvolvida uma documentação, gerando assim um feedback para a fase seguinte. Esta documentação faz com que o gerente possa ter controle do projeto como um todo.

O Modelo cascata considera as atividades fundamentais do processo: especificação, desenvolvimento e as fases de validação e evolução que são tratados separadamente. Neste modelo não se dá inicio a fase seguinte antes que a anterior tenha sido terminada. As vantagens do modelo cascata consistem na documentação produzida em cada fase e sua aderência a outros modelos de processos de engenharia. Seu maior problema é a divisão inflexível do projeto em estágios distintos. Os compromissos devem ser assumidos no estágio inicial do processo, o que torna difícil reagir às mudanças de requisitos. Este modelo pode ser usado quando os requisitos forem bem compreendidos e houver pouca probabilidade de mudança radical durante o desenvolvimento do software (PRESSMAN, 2011).

2.2.2 Desenvolvimento Incremental

O desenvolvimento incremental é baseado na idéia de desenvolver algo inicialmente, divulgar aos comentários dos usuários e proceder por meio de inúmeras versões até que um sistema adequado seja desenvolvido. As fases de desenvolvimento incremental como ilustra a figura 2 é composta por atividades de especificação, desenvolvimento e validações são divididos, sem separar estas duas atividades e entre todas estas atividades existe um feedback. Para criar um software de forma incremental, é mais fácil de corrigi-lo caso haja mudanças durante seu desenvolvimento (SOMMERVILLE, 2011).

O modelo incremental entrega uma quantidade de versões, chamadas de incrementos que oferecem na medida que o software vai sendo desenvolvido, algo para o cliente. Quando utilizado este modelo, na maioria das vezes o primeiro incremento são atendidas as necessidades básicas e prioritárias. O resultado desta avaliação é feito um planejamento para o incremento seguinte. Este processo é feito diversas vezes após a divulgaçao de cada incremento, até que seja entregue o produto final (PRESSMAN, 2011).

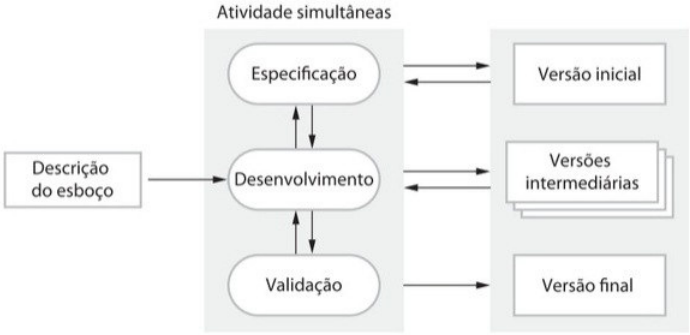

Figura 2 – Desenvolvimento Incremental
Extraído de (SOMMERVILLE, 2011).

Segundo (SOMMERVILLE, 2011) o desenvolvimento incremental possui três vantagens importantes comparado ao modelo em cascata:

1. O custo para tratar mudanças exigidas pelo cliente é reduzido e a documentação que é feita neste modelo é bem menor quanto ao modelo cascata.

2. Os clientes tem disponibilidade em visualizar algo que já foi implementado no projeto de forma fácil, podendo interferir no que já foi feito através de sugestões como por exemplo alterações ou melhorias no projeto. Existe um bloqueio com os clientes em compreender o que foi feito no projeto através de documentação.

3. Caso alguma parte do software ainda não tenha sido implementada é possível entregar para o cliente um software executável, em funcionalidade, esse software entregado inicialemte os cliente já podem utiliza-lo, anteriormente com o modelo mascata isto não era possível.

O desenvolvimento incremental é uma mistura de abordagem dirigida a planos ou ágil. Na abordagem dirigida a planos, os incrementos são verificados inicialmente e na abordagem ágil eles são identificados logo no início mas o desenvolvimento dos próximos depende do progresso e das prioridades do cliente (SOMMERVILLE, 2011).

Na próxima sessão será descrita sobre metodologia ágil que possui como base o desenvolvimento incremental.

2.3 METODOLOGIA ÁGIL

As metodologias ágeis são tratadas como métodos de desenvolvimento flexíveis em um ambiente onde existe mudança dos requisitos frequentemente e os resultados amostrados ao cliente devem ser feitos em determinado espaço de tempo. Esta metodologia propõe separar o desenvolvimento em ciclos menores que interagem entre si em determinadas semanas, oferecendo no final de cada ciclo uma versão para o cliente que venha agregar valor para o projeto (DANTAS, 2003)

O principal conceito da metodologia agil é a interação de pessoas e não processos ou algoritmos. Além disso, existe a preocupação de gastar menos tempo com documentação e mais com a implementação. Uma característica das metodologias ágeis é que elas são adaptativas ao invés de serem preditivas. Com isso, elas se adaptam a novos fatores decorrentes do desenvolvimento do projeto, ao invés de procurar analisar previamente tudo o que pode acontecer no decorrer do desenvolvimento (SOARES, 2004).

Assim, os desenvolvedores podem acompanhar a mudança dos requisitos no início de cada ciclo, além de ter uma retroalimentação contínua do cliente, reduzindo assim os riscos do projeto.

Enquanto as metodologias tradicionais de desenvolvimento mantêm o foco na geração de documentação sobre o projeto e no cumprimento rígido de processos, a proposta ágil é concentrar as atenções no desenvolvimento em si e nas relações entre os participantes (MUNDIM et al., 2002).

As metodologias ágeis trabalham com constante *feedback*, o que permite adaptar rapidamente a eventuais mudanças nos requisitos. Alterações essas que são, muitas vezes, críticas nas metodologias tradicionais, que não apresentam meios de se adaptar rapidamente às mudanças. Um outro ponto positivo das metodologias ágeis são as entregas constantes de partes operacionais do software. Desta forma, o cliente não precisa esperar muito para ver o software funcionando e notar que

não era bem isso que ele esperava (SOARES, 2004).

2.3.1 História da Metodologia ágil

A história da metodologia ágil deu inicio em 2001 quando dezessete homens reunidos dentre eles representantes de alguns métodos ágeis como XP e Scrum, em um reunião resolveram construir software de alta qualidade. O acordo comum resultante da reunião foi criado o Manifesto Ágil, na qual eles definiram conceitos e principios da metodologia ágil (HIGHSMITH, 2001).

Ágil é uma forma se produzir software, na qual processos são tratados de forma ágil e incremental, na maioria das vezes são projetos imprevisiveis e que possuem alteração durante o processo de desenvolvimento. Os processos são executados em ciclos curtos, na qual o cliente esta sempre presente, obtendo assim um *feedback* rápido do que já foi desenvolvido, fazendo com que a entrega de um produto pronto e de qualidade seja entregue para o cliente de forma rápida (STEFFEN, 2012).

O Manifesto ágil não despreza os processos e ferramentas, a documentação, a negociação de contratos ou o planejamento, eles apenas ficam em segundo lugar quando comparado com os indivíduos e interações, com o software funcionando, com a colaboração com o cliente e as respostas rápidas a mudanças e alterações (SOARES, 2004 apud LIBARDI; BARBOSA, 2010).

2.3.2 Conceitos e Princípios da Metodologia ágil

O manifesto ágil quando assinado em 2001, não era uma novidade de desenvolvimento ágil, esta idéia começou a ser materializada na década de 1990. É importante ressaltar que os métodos ágeis foram criados no estímulo de vencer as fraquezas apreendidas da engenharia de software convencional. O que diferencia os métodos ágeis das tradicionais é que os métodos ágeis foca nas pessoas ao invés de processos e algoritmos como também frisa em gastar menos tempo com documentação e maior com implementação (PRESSMAN, 2006 apud JUNIOR; YANZER, 2011).

Para melhor compreender os conceitos de desenvolvimento ágil, o Manifesto ágil define os **valores** e **princípios** acordados (BECK et al., 2001):

Os valores do Manifesto Ágil são:

1. Indivíduos e interações mais que processos e ferramentas;
2. Software em funcionamento mais que documentação abrangente;
3. Colaboração com o cliente mais que negociação de contratos;
4. Responder a mudanças mais que seguir um plano;

Além dos valores o manifesto ágil é composto por princípios que, quando seguidos corretamente, levam a melhores resultados no processo de desenvolvimento de software. O Manifesto Ágil segue os seguintes princípios:

1. Nossa maior prioridade é satisfazer o cliente, através da entrega adiantada e contínua de software de valor.
2. Aceitar mudanças de requisitos, mesmo no fim do desenvolvimento. Processos ágeis se adequam a mudanças, para que o cliente possa tirar vantagens competitivas.
3. Entregar software funcionando com freqüencia, na escala de semanas até meses, com preferência aos períodos mais curtos.
4. Pessoas relacionadas à negócios e desenvolvedores devem trabalhar em conjunto e diáriamente, durante todo o curso do projeto.
5. Construir projetos ao redor de indivíduos motivados. Dando a eles o ambiente e suporte necessário, e confiar que farão seu trabalho.
6. O Método mais eficiente e eficaz de transmitir informações para, e por dentro de um time de desenvolvimento, é através de uma conversa cara a cara.
7. Software funcional é a medida primária de progresso.
8. Processos ágeis promovem um ambiente sustentável. Os patrocinadores, desenvolvedores e usuários, devem ser capazes de manter indefinidamente, passos constantes.
9. Contínua atenção à excelência técnica e bom design, aumenta a agilidade.
10. Simplicidade: a arte de maximizar a quantidade de trabalho que não precisou ser feito.

11. As melhores arquiteturas, requisitos e designs emergem de times auto-organizáveis.

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

Nas próximas sessões serão descritas sobre as metodologias ágeis Lean e XP e no Capítulo 3 deste trabalho será detalhado o Scrum.

2.3.3 Lean

De acordo com o Lean Institute Brasil, "Lean é uma estratégia de negócios para aumentar a satisfação dos clientes através da melhor utilização dos recursos. A gestão Lean procura fornecer consistentemente valor aos clientes com os custos mais baixos (propósito) através da identificação de melhoria dos fluxos de valor primários e de suporte (processos) por meio do envolvimento das pessoas qualificadas, motivadas e com iniciativa (pessoas). O foco da implementação deve estar nas reais necessidades dos negócios e não na simples aplicação das ferramentas lean (LIB, 1998)".

A metodologia Lean no final de 1940 foi criada pelo responsável do Sistema Toyota de Produção, Taiichi Ohno. Ele verificou que o mercado japonês não apresentava tamanho suficiente, pela grande quantidade de carros que estavam sendo produzidos. Ao observar este cenário de muita produção, sem mercado para alcançar esta demanda, determinou o elemento principal da metodologia Lean, a eliminação de desperdícios. Taiichi Ohno afirma que tudo que não introduza valor ao cliente é desperdício, como por exemplo um carro sem ser produzido, esperar, dentre outros, ele acredita que é possivel produzir um produto de forma rápida sem haver desperdícios. Assim descreve diferentes maneiras de se identificarem os desperdícios no processo de desenvolvimento de um produto (POPPENDIECK; POPPENDIECK, 2007).

Lean possui sete princípios, na qual todos eles propõem entrega de uma forma mais ágil e de baixo custo (FILHO, 2008). Esses princípios serão resumidos a seguir:

1- Eliminar desperdícios: O desperdício pode acontecer de diversas maneiras, desta forma serão, descritos alguns desperdícios na qual se é possível observar e elimina-lo.

a) Funcionalidades do sistema, por exemplo tarefas incompletas, requerem esforço para serem executadas e não agregam valor ao software.

b) Excesso de processos requer recurso e amplifica-se tempo para finaliza-las destas tarefas. A criação de documentação deixa o processo muito maior, leva tempo para produzir, desatualiza com facilidade, podendo assim ser perdida. A documentação é vista como desperdício pois não acrescenta valor ao software. Processos complexos aumentam a quantidade de documentação, por isto também sendo um desperdício.

c) Antecipar funcionalidades é um desperdício porque faz com que o software se torne difícil sem gerando assim mais código, mais esforços com testes e mais integrações.

d) Troca de tarefas é uma forma de desperdício porque altera a forma mental na qual já esta sendo utilizada durante o processo contribuindo assim com a diminuição da produtividade.Incluir desenvolvedores em mais de um projeto é um desperdíos pois as necessidades um projeto não levam em conta a situação dos outros.

Esperar gera atraso, com isso desperdício em esperar por requisitos, por testes, por aprovação ou por feedback retardam o fluxo do desenvolvimento ou a identificação de problemas.

Defeitos são desperdícios pois o custo para manutenção deles custa tempo. À medida que o projeto evolui, a complexidade do código aumenta, com isso, a localização e a remoção de um defeito torna-se mais difícil.

2 - Amplificar o aprendizado: as lições devem ser retiradas das experiências vividas pela equipe e adicionadas ao processo, fazendo com que as com que as dificuldades passadas sejam fonte de conhecimento e contribuam para o amadurecimento da equipe e do processo. O uso de um processo definido engessa o aprendizado.

Neste cenário é identificado como melhoria no ciclo do processo em que primeiro deve-se fazer a identificação do problema,localizar a sua causa, criar uma solução, implementar, verificar os resultados e se adaptar à nova realidade (DEMING, 1986 apud FILHO, 2008).

3 - Adiar comprometimentos e manter a flexibilidade: Ambientes que possuem muitas incertezas, não é possivel prever as mudanças. Adiar decisões faz com que escolhas sejam estabelecidas por mais experiência e conhecimento adquiridos no decorrer do processo. Para adiar decisões durante a construção de sistemas é importante que a equipe crie a capacidade de absorver mudanças tratando os planejamentos como estratégias para atingir um objetivo e não como comprometimentos. Assim, mudanças serão observadas como oportunidades para aprender e atingir as metas.

4 - Entregar rápida: A entrega rápida de algo funcional, permite que o cliente e desenvolvedor possa melhorar e aprender quanto ao que foi pedido através de um retorno rápido, atualizado e confiável. Essa iteração entre desenvolvedor e cliente caminha através de um processo em que o cliente pode melhor especificar suas necessidades que já podem ser encaminhadas para a proxima fase desse ciclo feito de forma veloz.

5 - Valorizar a equipe: A equipe de desenvolvimento responsável pela criação do produto final, que é entregue o software ao clinte, possui entendimento sobre todos os detalhes do software. A partir destas caracteristicas a equipe precisa ser valorizado, sendo assim motivada em contribuir para a melhora do processo, obtendo assim um comprometimento com o processo.

6- Adicionar segurança ao software: A equipe deve conter soluçoes que deixe o software seguro, para manter um produto de qualidade. Com testes automatizados é possivel manter um sistema mais suscetivel a mudanças, é um meio de se segurança e aumentar os niveis de qualidade do software, em vez de perder tempo para encontrar erros, o teste é um esforço que previne isto.

7 - Otimizar o todo: Quando se desenvolve grandes soluções é necessário obter uma análise global do sistema, com isso um software otimizado faz com que a concentração desse esforço resulte em um produto consistente, com qualidade, satisfazendo assim a necessidade do cliente. Para identificar essa evolução, lean recomenda a utilização de métricas para poder avaliar esta troca vantajosa.

A implantação da metodologia ágil Lean é capaz de mudar toda a estrutura organizacional de uma empresa, por isso sua utilização depende da empresa aceitar suas idéias, a mudança é feita não somente pelas e sim pelo processo como um todo (FILHO, 2008).

2.3.4 XP

A programação Extrema (XP) é a mais conhecida nas metologias ágeis, no método XP diferentes desenvolvedores criam versões do sistema que podem ser testadas, integradas, tudo isso em apenas um dia. No processo do XP o cliente é envolvido na fase de definir os requisitos, visto como participante da equipe e pode argumentar como qualquer integrante. Os clientes descrevem suas necessidades em forma de história, na história são especificadas suas necessidades. Estas histórias posteriormente são organizadas como atividades para a equipe de de-

senvolvimento que cria entregas baseadas nestas histórias(requisitos) (SOMMERVILLE, 2011). A prática de desenvolvimento ágil XP possui alguns princípios e práticas (SOMMERVILLE, 2011). Estas práticas serão resumidas a seguir:

1. Planejamento incremental – as especificações dos clientes são memorizadas em cartões de história, que são estabelecidas por tempo e prioridade para serem desenvolvidas. A equipe de desenvolvimento separa estas histórias em atividades.

2. Pequenos *Releases* – inicialmente são desenvolvidos um grupo de requisitos úteis, que agregue valor ao negócio. Releases do sistema ao passar do tempo vao adicionando funcionalidade ao primeiro release.

3. Projeto simples – O projeto é feito para acatar as necessidades atuais e não mais que isso;

4. Desenvolvimento *test-first* – Um framework de teste inicial automatizado é utilizado para descrever os testes para uma nova utilidade antes mesmo de algo funcional ser executado.

5. Refatoração – Todos os criadores tem de refatorar o código assim que constatar melhorias para conserva-lo de maneira simples e de fácil utilização.

6. Programação em Pares – São dois desenvolvedores trabalhando um trabalha e o outro verifica, os pares são trocados frequentemente

7. Propriedade coletiva – Os desenvolvedores que trabalham em par devem entender e ser responsáveis por todo o código e podem alterar qualquer coisa do código.

8. Integração e contínua – Quando uma tarefa é finalizada ela é adicionada no sistema completo, e logo após é realizado testes de unidade nesta tarefa.

9. Ritmo sustentável – Horas a mais não são tomadas como algo positivos, no final são considerados a diminuição da qualidade do código.

10. Cliente no local – O cliente está sempre presente com a equipe de desenvolvimento, sendo considerado como um integrante da equipe e por repassar os requisitos do sistema para implementação.

O XP apresenta quatro valores que guiam o desenvolvimento sendo eles: a comunicação, a coragem, o feedback e a simplicidade (SOMMERVILLE, 2011). Além dos valores explicitados anteriormente, (FILHO, 2008) inclui o valor Respeito. Os valores são descritos abaixo:

Comunicação: A maioria dos problemas que ocorrem entre uma equipe é a falta de comunicação entre a equipe de desenvolvimento e o cliente. No XP mantem esta comunição como por exemplo práticas de teste de unidade e programação em pares

Simplicidade: É possivel escolher uma forma que possa aperfeiçoar o funcionamento do processo de desenvolvimento da equipe. O XP é um método econômico que desenvolve funcionalidades simples, e assim que for surgindo necessidades ele é adaptado conforme surgirem alterações. Desta forma não se tenta prever as necessidades que serão que ser desenvolvidas futuramente.

Feedback: Os problemas devem ser apresentados previamente para que sejam corrigidos de forma rápida. Toda a oportunidade de alterar ou modificar deve ser descoberta o mais cedo possível para que possa ser adicionada de forma rápida ao produto que está sendo construído.

Coragem: É preciso ter coragem para identificar problemas no projeto, requisitar apoio quando necessário, simplificar um código que já esteja em funcionamente, como também comunicar ao cliente que não será possível implementar um requisito no prazo estimado e, até mesmo, para fazer alterações no processo de desenvolvimento.

Respeito: É necessário respeito entre as pessoas, sendo a peça principal para que os demais valores funcionem. Sem respeito, a comunicação e o feedback não ocorre de forma aceitável e a coragem de um participante da equipe poderá ser nociva aos demais por não estar alinhada com os interesses da equipe. Todos os participantes devem manter o respeito entre si e em relação aos seus trabalhos. A desvalorização alguém sobre a função que exerce ou a qualidade de seu trabalho são formas de falta de respeito que faz com que a equipe não trabalhe em sintonia. Uma forma de respeito de cada um para com a equipe é assumir responsabilidades que serão capazes de cumprir e da equipe para com seus membros confiar que cada um fará o melhor trabalho possível. A metodologia XP considera que os desenvolvedores também são pessoas e preocupa-se com suas necessidades pessoais e profissionais através dos princípios da humanidade e da diversidade, estabelecendo compromissos com o princípio da aceitação da responsabilidade.

A metologia XP é uma metologia ágil de baixo risco, leve, eficiente, entre outras caracteristicas. Ela se diferencia de outras metodo-

logias por obter (BECK, 2000):

- Feedback inicial, concreto e contínuo de ciclos pequenos;
- Utiliza planejamento incremental, que vem rapidamente com o plano geral previstao para evoluir através da vida do projeto;
- Capacidade de programar de forma flexível implementação de funcionalidades, respondendo às novas necessidades de negócio de forma rápida;
- O procedimento de teste é feito de forma automatizada, escrita por programadores e clientes para observar o progresso do desenvolvimento, fazendo assim com o que o software se desenvolva identificando erros, defeitos mais cedo;
- Sua dependência de comunicação oral, testes e código fonte para comunicar a estrutura do sistema e intenção sua dependência de um processo de projeto evolutivo que dura enquanto dura o sistema.

Neste capítulo foi feito um levantamento teórico dos fundamentos ágeis e das métodologias ágeis XP e Lean. No próximo capítulo serão descritos conceitos, práticas, elementos, principais funções e algumas características que compõem a metodologia ágil Scrum, por ser o objeto de estudo desta monografia.

3 SCRUM

Scrum é uma metodologia ágil que pode ser aplicada a quase qualquer projeto,no entanto, a metodologia Scrum é mais comumente usada no desenvolvimento de software. O processo Scrum é adequado para projetos com rápida mudança ou requisitos altamente emergentes. O desenvolvimento de software baseado em scrum progride através de uma série de iterações chamado sprint, que dura de uma a quatro semanas. O modelo Scrum sugere que cada sprint comece com uma breve reunião de planejamento e cada conclusão de sprint, uma revisão (MGS, 2012a).

A metodologia "Scrum é uma estrutura processual (framework) para suportar o desenvolvimento e manutenção de produtos complexos. O Scrum consiste em Equipes associadas a seus papéis, eventos, artefatos e regras. Cada componente dentro do framework serve a um propósito específico e é essencial para o uso e o sucesso do Scrum"(SCHWABER; SUTHERLAND, 2011) .

Scrum é uma maneira de desenvolver software de forma ágil. A equipe Scrum é responsável por detalhar o que vai ser realizado no projeto e escolhe a melhor maneira de solucionar os problemas, em uma reunião de planejamento feita pela equipe responsável é determinada as tarefas que vão ser realizadas na proxima sprint. O Scrum é auto organizado ou seja, todos envolvidos decidem o que será resolvido. Na equipe ágil Scrum existem duas funções específicas que são compostas pelo Scrum Master e o Product Owner(PO) ou Dono do Produto que serão explicados detalhadamente nas próximas seções deste capítulo (MGS, 2012a).

3.1 ELEMENTOS DE APOIO SCRUM

O Scrum possui alguns elementos que compõe esta prática, são eles os cartões e gráficos de acompanhamento. Os cartões, neles estão descritos as especificações de uma determinada funcionalidade do projeto. Os gráficos de acompanhamento demonstram o resultado e o estado do projeto e são renovados com frequência (FILHO, 2008) .

Os elementos de apoio ao Scrum são (FILHO, 2008):

- Backlog do Produto - Ordena todos os cartões de funcionalidade que o produto possua e que ainda vai ser implementado;

- Backlog Selecionado - É implementado na sprint atual um subconjunto de funcionalidades que o cliente especificou a partir do Backlog do Produto, podendo assim não ser modificada durante a Sprint;

- Backlog de Impedimentos: Lista dos obstáculos identificados pela equipe que não pertencem ao contexto do desenvolvimento;

- Gráficos de Acompanhamento: Gráficos que medem a quantidade de trabalho restante (burndown charts) são os preferidos em Scrum. É recomendado fazê-los para várias esferas do projeto: para o produto, para a release e para o sprint.

3.2 PRINCIPAIS FUNÇÕES DO SCRUM

O Time Scrum é formado por Srum Master, Product owner e a Equipe de Desenvolvimento, a Figura 3 ilustra o time Scrum. Eles se auto organizam em suas atividades a fim de obter competências necessárias para concluir o trabalho sem precisar sujeitar-se a ajuda das outras partes da equipe. O modelo de equipe Scrum é delineado para aperfeiçoar a flexibilidade, criatividade e produtividade com a objetividade de entregar um produto pronto e funcional de forma iterativa e incremental (SCHWABER; SUTHERLAND, 2011).

Figura 3 – Funções do Scrum
Adaptado de (RUBIN, 2012).

3.2.1 *Product Owner*

O proprietário do produto(PO) é responsável por priorizar o backlog durante o desenvolvimento Scrum, ele está acima do que mais se aprende sobre o sistema que está sendo construído sobre todos os envolvidos sejam eles usuários, equipe e assim por diante (SCHWABER; SUTHERLAND, 2011).

Ele que trabalha com a equipe de desenvolvimento e é responsável por gerenciar o backlog do produto, transfere as informações para a equipe de desenvolvimento de forma clara e qualquer alteração feita nas prioridades dos itens no backlog do produto é ele o responsável por gerenciar. Ken Schwaber e Jeff Sutherland estabelecem alguns itens determinantes para o PO (SCHWABER; SUTHERLAND, 2011):

- Expressar claramente os itens do backlog do produto;
- Ordenar os itens do backlog do produto para alcançar melhor as metas e missões;
- Garantir o valor do trabalho realizado pelo time de desenvolvimento;
- Garantir que o backlog do produto seja visível, transparente,claro para todos e mostrar o que o time scrum vai trabalhar a seguir;
- Garantir que a equipe de desenvolvimento entenda os itens do backlog do produto no nível necessário.

Entretanto o Product Owner é o Dono do Produto, ou seja, ele é responsável por todo o conhecimento do negócio e conceder ao cliente um produto de valor, assim como também garante a equipe desenvolvedora a compreensão do produto transmitindo a eles os itens priorizados adicionando assim valor ao produto e ao cliente (CRUZ, 2013).

3.2.2 Equipe de Desenvolvimento

A Equipe de Desenvolvimento é composta por arquiteto, programador, testador, administrador de banco de dados e assim por diante. A equipe do scrum é formado por um conjunto de pessoas em prol da criação de um produto utilizável na qual essas pessoas são responsáveis por criar, construir e testar o produto desejado. Ela se auto organiza e

especifica uma melhor maneira de alcançar as metas estabelecidas pelo Product Owner o Dono do Produto e na maioria das vezes é composta por no máximo 9 pessoas para produzir com qualidade um software funcional (SCHWABER; SUTHERLAND, 2011).

De acordo com Ken Schwaber e Jeff Sutherland as Equipes de Desenvolvimento tem as seguintes características:

- A equipe de desenvolvimento é auto-organizada. Ninguém (nem mesmo o Scrum Master) diz a equipe de desenvolvimento como transformar o Backlog do Produto em uma versão funcional e utilizável;
- São multifuncionais, possuindo todas as habilidades necessárias para criar o incremento do Produto.
- O Scrum não reconhece títulos nenhum membro da equipe que não seja o desenvolvedor, independente do trabalho que foi realizado pela pessoa, não há exceções para esta regra.
- Cada integrante da equipe pode ter habilidades em áreas de especialização, mas a responsabilidade de cada área da equipe é de todos da equipe;
- Equipes de Desenvolvimento não contém dedicadas a domínios específicos de conhecimento, tais como teste ou analista de negócios.

3.2.3 *Scrum Master*

O Scrum Master é responsável por toda a equipe, possuindo alguns papéis na equipe de desenvolvimento. O Scrum Master é como um líder, não um gerente (RUBIN, 2012).

De acordo (COHN, 2011) no livro "Desenvolvimento de Software com Scrum - Aplicando métodos ágeis com sucesso"descreve o papel e os atributos de um Scrum Master, na qual serão explicadas seções a seguir.

3.2.3.1 Papel do Scrum Master

O Scrum Master é um líder da equipe e tem autoridade sobre os processos e não pela equipe. O Scrum Master talvez não possa dizer "Você está despedido", mas ele pode dizer "Iremos testar sprints

de duas semanas para o próximo mês"determinando assim as decisões tomadas pela equipe, fazendo com que os membros da equipe tomem decisão por si próprio (COHN, 2011).

Sua autoridade mantém um limite, ele não tem autoridade de fazer com que algum membro faça determinada função mas ele apenas lembrará dos seus objetivos a serem realizados, na qual sua autoridade vai até onde o cliente permitir e, a equipe que lhe concede esta autoridade. No processo ele exerce na atividade que determina para a equipe o que se fazer para garantir um melhor desempenho na próxima sprint, como algo que deu errado durante o processo e que não pode ser entregue. O Scrum Master não é capaz de fazer com algum membro da equipe realize algo que esteja fora do processo, isto vai além de suas funções, sendo assim seu papel pode ser mais dificil do que um gerente de projeto que possui uma posição autoritária constante, o Scrum Master possui uma limitada e restrita autoridade que assegure que o Scrum esteja sendo seguido (COHN, 2011).

3.2.3.2 Atributos do Scrum Master

De acordo com Mike Cohn (COHN, 2011) existem seis atributos comuns entre os melhores Scrum Master na qual ele já trabalhou.

- Responsável - O Scrum Master é responsável por amplificar o rendimento da equipe e por auxiliar os participantes da equipe a utilizar o Scrum, como dito anteriormente ele possui responsabilidade sem ter qualquer autoridade que possa ser útil em seu cumprimento.

- Humilde - Um Scrum Master humilde não coloca suas necessidades em primeiro lugar e auxilia a equipe a atingir seus objetivos. Eles reconhecem o valor de todos os membros da equipe.

- Colaborativo - Um Scrum Master exerce a função colaborativa na equipe a partir de palavras e ações. Ele precisa garantir que todos os envolvidos possam levantar questões para uma discussão aberta e que tem apoio para fazer isso. Quando surgir algum conflito, ele é responsável por estimular uma solução a todos envolvidos, alem de colaborar com este comportamento ele faz com que todos na equipe sejam colaborador uns com os outros para melhor combater comportamentos inapropriados da equipe.

- Comprometido - Mesmo que o Scrum Master não trabalhe em um

projeto em período integral, esta função requer que ele esteja totalmente envolvido com o trabalho. O Scrum Master possui alto nivel de compromentimento em suas sprints e em realizar seus objetivos sem deixar seus objetivos perdurarem por muitos dias para ser finalizado. É claro que isso é inevitável, pois não é todo dia em que se pode concluir uma tarefa. Geralmente se uma equipe verificar que obstáculos existentes não são removidos de forma rápida, a equipe deve lembrar o Scrum Master a importância de se compremeter com a equipe. Uma forma do Scrum Master estar comprometido com a equipe é permanecer nesta função durante todo o projeto, a mudança do Scrum Master no meio de projeto desestrutura toda a equipe.

- Influente - O Scrum Master deve ser influente na equipe de forma em que possa fornecer como por exemplo uma nova prática a ser inserida no projeto sem ter determinada atitude como do tipo "porque eu disse que é assim". Ele tem poder de persuadir um diretor da qualidade a dedicar testadores em tempo integral para o projeto, ou seja como dito no livro todos os Scrum Masters tenham que saber como utilizar sua influência pessoal ele tambem possui um grau de habilidade de politico-corporativo. Este termo politico-corporativo utilizado no livvro, é usado pejorativamente; no entanto um Scrum Master tem que ter noção de quemtoma decisões na empresas, como são tomadas, que acordos existem e assim por diante pode ser um triunfo para a equipe.

- Informado - O Scrum Master além de ter entendimento sobre o processo Scrum ele deve ter conhecimento de mercado, ou qualquer assunto especializado para auxiliar a equipe a seguir com seus objetivos, (LAFASTO; LARSON; LARSON; LAFASTO, 2001, 1989 apud COHN, 2011) estudaram equipes bem-sucedidas e seus líderes e concluiram que "um conhecimento profundo e detalhada de como algo funciona aumenta a chance de o lider ajudar a equipe a levantar as questões técnicas e sutis que devem ser resolvidas". (COHN, 2011) afirma também que um Scrum Master não precisa ter entendimento sobre tudo de marketing ou programação mas deve ter conhecimento de ambos para melhor liderar a equipe.

3.3 PROCESSO SCRUM

De acordo com o Kechi Hirama (HIRAMA, 2012) o processo Scrum é composto por dois ciclos principais. O primeiro ciclo denominado como Sprint com tempo de até 4 semanas, periodo na qual se é desenvolvido determinadas funcionalidades que ao final é entregue as parte interessada do produto. Estas funcionalidades que sao os requisitos do cliente na qual se é chamada de Backlog do Produto, que são argumentadas e definidas para compor o BackLog da Sprint que são as atividas que vão ser realizadas pela equipe Scrum. O segundo ciclo é quando a Sprint já foi iniciada e são efetuadas reuniões diárias(24hrs) para determinar o que foi efetuado no dia anterior, impeditivos que seriam os problemas ao se efetuar a tarefa e privilegiar as tarefas do dia. Pode-se ter uma melhor visualização do processo Scrum através da Figura 4.

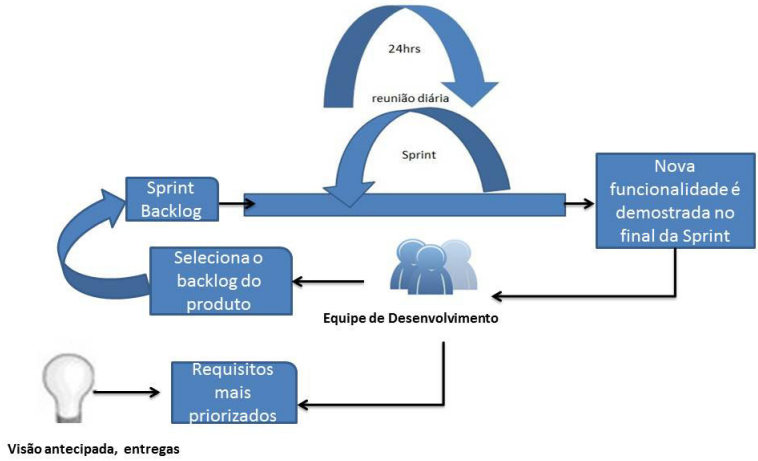

Figura 4 – Visão geral do processo Scrum
Adaptado de: (SCHWABER, 2004).

3.4 *SPRINT*

A Sprint é um trabalho realizado em ciclos curtos com duração de no máximo um mês, na qual cada trabalho realizado em cada ciclo deve resultar em algo funcional para o cliente ou usuário. As Sprints possuem tempo definido para inicio e fim e na maioria das vezes todas possuem a mesma duração, mas durante a sprint não é permitido qualquer alteração com exceção a regra quando se é necessário alterar algo relacionado ao negócio (RUBIN, 2012) .

O início de uma nova sprint é imediatamente após a conclusão da sprint anterior e durante a sprint (SCHWABER; SUTHERLAND, 2011):

- Não são feitas mudanças que podem afetar o objetivo da Sprint;
- A composição da Equipe de Desenvolvimento permanecem constantes;
- As metas de qualidade não diminuem; e,
- O escopo da Sprint pode ser classificado e renegociado entre o Product Owner e a Equipe de Desenvolvimento quanto mais for aprendido.

O sprint backlog é uma lista de tarefas identificadas pela equipe de Scrum para ser concluída durante a sprint. Durante a reunião de planejamento do sprint, a equipe seleciona um determinado número de itens do backlog do produto, geralmente sob a forma de histórias de usuários e identifica as tarefas necessárias para completar cada história de usuário. A maioria das equipes também estimam quantas horas cada tarefa vai levar alguém da equipe para ser concluído.É importante que a equipe selecione os itens e tamanho do sprint backlog. Eles são as pessoas que vão se submeter a realizar aquela tarefa portanto são eles que determinam o que vai ser feito naquela sprint. O sprint backlog é na maioria das vezes mantida como uma planilha (MGS, 2012b).

3.4.0.3 Reuniões de Planejamento da *Sprint*

Uma sprint é composta por três reuniões que são: reuniões de planejamento da sprint, reunião de revisão de sprint e reuniões diárias (BROD, 2013).

O planejamento da sprint é composto por duas partes. As primeiras quatro horas com Product Owner(PO) na qual ele apresente as

prioridades dos itens presentes no Backlog do produto como por exemplo detalhando todas suas informações como finalidade, conteúdo etc. A equipe se compromete com o Product Owner que ele vai fazer o seu melhor. Durante o segundo de quatro horas de reunião de planejamento da Sprint, a equipe planeja o Sprint. As tarefas existentes são inseridas em um Sprint Backlog (SCHWABER, 2004).

Logo após a equipe coloca em prática as tarefas presentes no Backlog da Sprint o Time Scrum determina a meta da Sprint. A meta da Sprint é um objetivo que estará na mente da equipe a fim de alcançala caminho de buscar satisfazer o objetivo da Sprint, ela implementa a funcionalidade e a tecnologia. Caso o trabalho acabe por ser diferente do esperado pela Equipe de Desenvolvimento, então eles colaboram com o Product Owner para negociar o escopo do Backlog da Sprint dentro da Sprint (SCHWABER; SUTHERLAND, 2011).

3.4.0.4 Reuniões diárias

As reuniões diárias existentes no processo Scrum possui um tempo de 15 minutos ou menos. Esta atividade é muitas vezes conhecida como diário stand-up por causa da prática de todos levantar-se durante a reunião para ajudar a promover a brevidade. O Scrum Master é o que realiza esta reunião e cada membro da equipe responde três perguntas (RUBIN, 2012):

- O que foi realizado desde a última reunião diária?
- O que pretende trabalhar até a próxima reunião diária?
- As metas de qualidade não diminuiram?
- Quais são os obstáculos e impedimentos que estão impedindo-lhe de fazer progresso?

Ao final, na qual todos já terem respondido, todos os membros da equipe consegue ter uma melhor visão dos processos que estão acontecendo em direção a meta Sprint e também conseguem discutir questões ou modificações que precisam ser abordados. A reunião diária é essencial para ajudar a equipe de desenvolvimento gerenciar o fluxo rápido e flexível de trabalho dentro de uma sprint (RUBIN, 2012).

A reunião diária não é uma atividade de resolução de problemas, em vez disso, muitas equipes utilizam esta atividade para discorrer sobre os problemas existentes na Sprint, com a finalidade de comunicar

a todos os membros da equipe o estado de Sprint, no intuito de ajudá-los a fazer um trabalho melhor (RUBIN, 2012).

3.5 REVISÃO DA *SPRINT*

A revisão da Sprint é executada no final da sprint onde os membros da equipe colaboram sobre o que foi feito na sprint. (SCHWABER; SUTHERLAND, 2011) afirma que a reunião de revisão inclui os seguintes elementos :

- O product owner identifica o que está "Pronto" e o que não está "Pronto";
- A equipe de desenvolvimento discute o que foi bem durante a Sprint, quais problemas ocorreram dentro da Sprint, e como estes problemas foram resolvidos;
- A equipe de desenvolvimento demonstra o trabalho que está "Pronto" e responde as questões sobre o incremento;
- O product owner discute o backlog do produto tal como está. Ele (ou ela) projeta as prováveis datas de conclusão baseado no progresso até a data; e,
- O grupo todo colabora sobre o que fazer a seguir, e é assim que a reunião de revisão da sprint fornece valiosas entradas para a reunião de planejamento da próxima Sprint.

O resultado da reunião de revisão da Sprint é um backlog do produto revisado que define o provável backlog do produto para a próxima sprint.

3.6 EXECUÇÃO DA *SPRINT*

Uma vez que a equipe scrum termina o planejamento do sprint e acordam com o conteúdo para a próxima sprint,a equipe de desenvolvimento, guiada pelo scrum master, executa todas as tarefas e as ordena no sprint backlog. Os membros da equipe definem seu próprio trabalho em nível de tarefa e, em seguida, auto-organizam de uma forma na qual cada membro sinta-se melhor para alcançar o obejtivo da sprint (RUBIN, 2012).

3.6.1 *Product Backlog*

O product backlog é o coração do scrum. É aqui que tudo começa. O product backlog é basicamente uma lista de requisitos, histórias de elementos que o cliente deseja que seja desenvolvido, descritos utilizando a terminologia do cliente (KNIBERG, 2007).

O product backlog é uma lista que contém todas as funcionalidades requiridas do cliente para o produto, o product owner é responsável por definir todo conteúdo do product backlog. O product backlog é ágil pois ele pode ser modificado juntamente quando for identificado algo que pode adicionado no produto, e é justamente por este motivo que ele nunca esta completo. É no product backlog em que os requisitos estão mais detalhados e são melhores compreendidos (LIBARDI; BARBOSA, 2010).

A ordem para definir a lista é que os itens que estão no topo da lista de ordem mais alta são itens que devem ser melhor compreendidos, devem estar descritos de forma clara, do que os itens de ordem mais baixa. A estimativa que esta relacionado com a história depende da estimativa que foi incluida naquele item, quanto menor a ordem do item na lista, menor vão ser seus detalhes no product backlog (SCHWABER; SUTHERLAND, 2011).

3.6.2 Histórias do *Product Backlog*

Os itens ou histórias do product backlog é composto por uma identificação, nome, importância e estimativa inicial. A identificação é importante para caso haja mudanças no nome, ela serve para controlar. O nome é a descrição do item, que deve ser feito de forma clara para que os desenvolvedores e toda equipe possa entender e diferencia-las dos demais itens. A importância é criada em em forma de pontos, quanto maior a quantidade de pontos. A estimativa inicial é a quantidade de tempo para se implementar aquele item, que pode ser comparado a outros itens, tal estimativa esta relacionado a quantidadede pessoas responsáveis pelo item que descreve o tempo para implementar. A descrição do item é melhor descrita na apresentação da sprint. Por final são apresentados notas que são qualquer outra informação daquele item de forma ágil, bem resumido. O responsável por este documento é o product owner (KNIBERG, 2007).

3.6.3 Como elaborar um *Product Backlog?*

A elaboração de um product backlog é feita de maneira ágil no formato de histórias curtas, na qual são descritas nestas histórias as funcionalidades do sistema. Os itens no topo da lista ordenada do backlog do produto determinam as atividades de desenvolvimento mais imediatas. Quanto maior a ordem (topo da lista) de um item, mais o item do backlog do produto deve ser considerado, e mais consenso existe em relação a ele e ao seu valor. Os itens do backlog do produto de ordem mais alta (topo da lista) devem ser mais claros e mais detalhados que os itens de ordem mais baixa. Estimativas mais precisas são feitas com base na maior clareza e maior detalhe (SCHWABER; SUTHERLAND, 2011).

A figura 5 demostra um exemplo de como se elaborar um product backlog, que é elaborado da seguinte maneira de acordo com (KNIBERG, 2007):

- ID - Uma identificação única, apenas um número com autoincremento. Isso é para evitar que percamos o controle sobre as histórias quando mudamos seus nomes.
- Nome – Um nome curto e descritivo para a história. Por exemplo, "Ver o histórico de transações". Suficientemente claro para que os desenvolvedores e o product owner tenha entendimento claro o bastante para distingui-la das demais histórias. Normalmente um nome curto.
- Importância – A pontuação de importância da história para o product owner. Por exemplo 10. Mais pontos = mais importante.
- Estimativa inicial – As estimativas iniciais da equipe sobre quanto tempo é necessário para implementar aquela historia, se comparada a outras estórias. É estimada através de pontos por e geralmente corresponde mais ou menos a relação homem/dias.
- Como demonstrar – Uma descrição em alto nível de como a história será demonstrada na apresentação da sprint. Uma simples especificação de teste "Faça isso, então faça aquilo e então isso deverá acontecer".
- Notas – quaisquer outras informações, esclarecimentos, etc. Normalmente feita de maneira breve.

PRODUCT BACKLOG (exemplo)					
ID	**Nome**	**Imp**	**Est**	**Como demonstrar**	**Notas**
1	Depósito	30	5	Logar-se, abrir a página de depósito, depositar R$ 10,00, ir para a página do meu saldo e verificar que este aumentou em R$ 10,00.	Precisa de uma diagrama UML de sequência. Não é necessário se preocupar com criptografia por enquanto.
2	Verificar seu próprio histórico de transações	10	8	Logar-se, clicar em "transações". Fazer um depósito. Voltar para transações, verificar se o novo depósito é listado.	Usar paginação para evitar consultas muito grandes ao banco de dados. Projetar de forma similar à página de visualização de usuários.

Figura 5 – Exemplo de *product backlog*
Extraido de (KNIBERG, 2007).

3.6.3.1 *Planning Poker*

Uma das formas mais utilizadas de estimar histórias é denonominado como planning poker, um jogo que define as prioridades dos itens de backlog em um acordo com os membros da equipe de desenvolvimento. O scrum master e o product owner tem possibilidade de ajudar a coordenar estas estimativas, mas é a equipe de desenvolvimento que é responsável por estimar o quanto vai ter de esforço para realizar aquele item (LAYTON, 2012). Esta técnica é iniciada através do PO que expõe os requisitos de recursos para a equipe que normalmente é formada por testadores, desenvolvedores, programadores, todos os responsáveis pelo desenvolvimento do produto. Cada pessoa joga com um baralho de cartas marcadas com os valores possíveis de estimação 1,2,3,5 e 8, em que um 1 indica o menos complexo e 8, o mais difícil (ECKSTEIN, 2013).

3.6.3.2 Definição de "Pronto"

O termo pronto para a equipe scrum significa quando um item de backlog está preparado para ser entregue, esta definição de pronto, deve estar clara para todos os envolvidos na equipe. Para a equipe Scrum quando um item está pronto, quer dizer que o trabalho foi con-

cluido. Este significado de pronto deve estar claro também na reunião de planejamento quando selecionam os itens que serão desenvolvidos. A finalidade de cada sprint é entregar uma parte funcional do produto, que seja útil e que esteja preparada para ser utilizável, assim o product owner pode adicionar ao incremento anterior na qual eles devem funcionar em conjunto. Com um time scrum responsável é possivel ampliar critérios mais rigorosos de alta qualidade (SCHWABER; SUTHERLAND, 2011).

3.6.4 *Backlog Sprint*

A sprint backlog é uma lista dos itens do product backlog a equipe se compromete a entregar , além da lista de tarefas necessárias para entregar os itens do product backlog (MGS, 2012c).

O sprint backlog ocorre logo após a reunião de planejamento da sprint, mas antes da primeiro reunião diária do Scrum. Existens algumas ferramentas para fazer o sprint backlog, na qual o autor cita dois exemplo como Jira, Excel ou um quadro de tarefas pregado na parede (KNIBERG, 2007). O quadro das tarefas funciona da seguinte maneira, são incluidos colunas do tipo "Esperando pra ser testado"ou "Cancelado", colunas escritas da maneira mais simples possivel. A decisão dos itens que serão incluidos na Sprint, é uma das principais atividades da reunião de planejamento que especificamente é copiar itens do backlog do produto para a sprint backlog, demonstrado na Figura 6 (KNIBERG, 2007).

3.6.4.1 Quadro de Tarefas

Ao praticar Scrum, podemos fazer com que o sprint backlog se torne visível, colocando-o em um quadro de tarefas. Os membros da equipe atualizam o quadro de tarefas continuamente durante todo o sprint caso alguém pense em uma nova tarefa é escrito um novo cartão e colocado no quadro de tarefas(MGS, 2012d). A Figura 7 ilustra um quadro de tarefas.

Cada linha no quadro scrum é um item de trabalho composto no product backlog. Durante a reunião de planejamento da sprint, a equipe separa os itens do Product Backlog que podem finalizar durante o próximo sprint. Cada item do product backlog é transformado em vários itens de sprint backlog. Cada uma delas está representada por

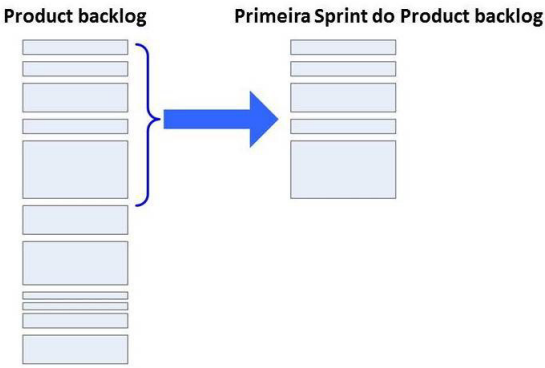

Figura 6 – Transição do *product backlog* para a *sprint backlog*
Adaptado de (KNIBERG, 2007).

um cartão de trabalho que é inserida no Scrum board. Cada cartão tarefa começa no quadro de tarefas Scrum na coluna "fazer". As colunas que geralmente usadas em um quadro de tarefas são (MGS, 2012d):

- História: A descrição dos itens são exibidas nesta coluna;
- Para fazer: Nesta coluna estão todos os cartões que não estão na coluna "Concluído"ou "em processo";
- Atividades em desenvolvimento: Qualquer cartão que estiver sendo trabalhado fica nesta coluna e é o programador que escolhe como trabalhar com ele e ele tem como move quando ela estiver pronta para ser iniciada, na maioria das vezes isso é resolvido durante a reunião diária;
- Para Testar: Nesta coluna estão os cartões com as tarefas para se testar. Então, se há um cartão para se testar o testador descreve os erros que foram achados ou não naquele cartão;
- Pronto: cartões são acumulados nesta coluna quando estiverem prontos. Eles são removidos no final da Sprint. Às vezes, removidas alguns ou todos durante uma sprint, caso existam muitos cartões. Opcionalmente, usam as seguintes colunas em um quadro de tarefas Scrum, dependendo da equipe, a cultura, o projeto e outras considerações:

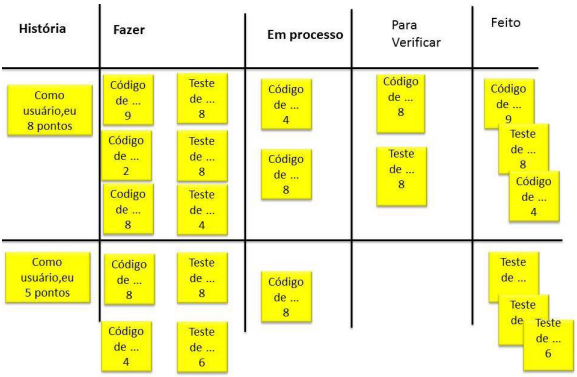

Figura 7 – Quadro de Tarefas
Adaptado de (MGS, 2012d).

Neste capítulo foi feito um levantamento teórico da metodologia ágil Scrum, suas práticas, principais funcionalidades, dentre outros elementos que compõe esta metodologia. No próximo capítulo será descrito o estudo de caso na empresa Koin sobre o uso da metologia ágil Scrum.

4 ESTUDO DE CASO NA EMPRESA KOIN

Este capítulo apresenta o estudo de caso realizado na empresa Koin, cujo objetivo é documentar e analisar a metodologia ágil Scrum, através da aplicação de questionários feito com 11 pessoas.

4.1 SOBRE A EMPRESA KOIN

A Koin foi criada em 2013 e opera sob uma plataforma proprietária de alta tecnologia e inovação (KOIN, 2014).

A empresa trabalha com meios de pagamento diferenciado, um modelo de compras online em que proporciona aos seus clientes a experiência de pós pago, ou seja, o cliente só paga após receber o produto. O cliente que quiser utilizar este modelo de pagamento, integra sua loja com a Koin. A forma como é feita a integração loja-comprador é ilustrada na figura 8.

A equipe conta com aproximadamente 40 funcionários, entre eles marketing, recursos humanos, sac, equipe de desenvolvimento entre outras.

O estudo de caso descrito neste tcc foi realizado na equipe de desenvolvimento que é formada por desenvolvedores, testadores, gerente e arquiteto de software.

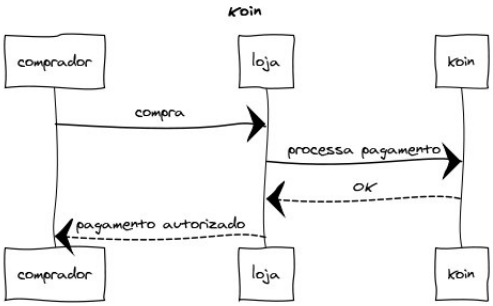

Figura 8 – Guia de Integração Koin
Extraido de: (KOIN, 2014).

4.2 O PROCESSO DE ADOÇÃO DO SCRUM

Antes de adotar o Scrum a empresa não utilizava nenhuma metodologia ágil. O líder da equipe de desenvolvimento era responsável por coletar as informações com o cliente e repassá-las para a equipe. Ele que determinava as tarefas a serem desenvolvidas, já que obtia maior conhecimento de todo sistema, determinando assim o que deveria ser desenvolvido, alterado ou retirado do sistema. O líder da área de tecnologia passava as tarefas, de acordo com solicitações pouco estruturadas dos stakeholders da organização. Houve casos em que a solicitação foi passada por telefone, sem nenhuma documentação e notificação específica para a atividade.

O líder da equipe estimulava as pessoas a concluírem suas tarefas atendendo os prazos, porém dificilmente estes prazos eram cumpridos. Devido a essa desordem organizacional e o aumento de pessoal especializado em cada área, se fez necessário a adoção de uma metodologia que minimizasse estes problemas. A adoção de um método ágil apareceu como uma possível solução a estes problemas.

Antes do Scrum, cada membro da equipe exercia uma função que era dividida por área de conhecimento, logo, cada profissional atacava somente o desenvolvimento de suas especialidades. Caso houvessem problemas durante o processo de desenvolvimento, havia ajuda mútua, trabalhando-se em conjunto com o objetivo de concluir as atividades estabelecidas pelo líder.

4.3 SCRUM NA KOIN

A metodologia scrum está em funcionamento há aproximadamente 5 meses. No início da implantação do Scrum na Koin, foi determinada para duas pessoas da equipe a função de *Product Ownner* e *Scrum Master*, respectivamente.

Com a adoção do Scrum, a Koin aderiu as reuniões diárias, reunião de planejamento e por último a reunião de revisão em que se é discutido sobre a sprint atual. Como também, sugestões e melhorias para a próxima sprint. Na Koin, o scrum master tem possibilidade em alterar itens do backlog, pois o Product Owner e o Scrum Master trabalham em conjunto.

Para avaliar os efeitos da adoção do Scrum, após 5 meses de uso, foram aplicados dois questionários com a equipe de desenvolvimento, que é formada por 11 membros, entre eles o product owner, scrum mas-

ter, estagiários, desenvolvedores, CTO e gerente de desenvolvimento.

4.4 PRIMEIROS EFEITOS APÓS A ADOÇÃO DO SCRUM

Uma mudança abrangente, como a inserção do método scrum, gera um grande impacto na estrutura organizacional da empresa. Por exemplo, um desenvolvedor ao fazer a transição para o scrum deve se familiarizar em trabalhar com pequenas tarefas e entregar algo funcional em cada sprint (COHN, 2011). Este momento de mudança é importante pois a equipe deve ser treinada para aderir a este método (COHN, 2011). Na Koin apenas 30% (Figura 9) da equipe foi treinada formalmente, sendo que os demais integrantes da equipe receberam informações sobre o processo do product owner, do scrum master e um desenvolvedor.

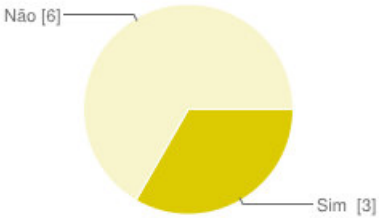

Figura 9 – Resultado sobre treinamento para trabalhar com o Scrum na Koin

A transição para o Scrum fez com que pessoas trabalhassem de maneira em que não estão habituadas, podendo assim agir, de forma negativa a este novo método, e consequentemente tornando-se insatisfeitos, quando não, totalmente resistentes em relação a mudança (COHN, 2011). Na empresa Koin não houve esta resistência pois, toda a equipe conhecia os possíveis benefícios da adoção da metodologia ágil, como mostram as figuras 10 e 11.

A maior parte da equipe aprova o uso do Scrum na empresa, como mostra a figura 12.

Com base na pesquisa, 67% dos entrevistados acharam muito bom o desempenho da equipe após a adoção do Scrum, conforme ilustra a figura 13. Isto foi comprovado através da declaração de um membro da equipe de desenvolvimento na qual afirmou que antes do scrum as atividades não possuíam prazos, consequentemente, tarefas inacabadas

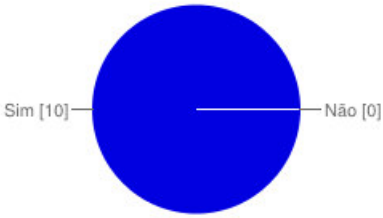

Figura 10 – Você já ouviu falar em metodologias ágeis?

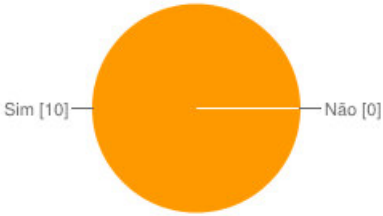

Figura 11 – Você conhece o Scrum?

ou passadas para serem desenvolvidas posteriormente. Com a inserção da metodologia scrum as atividades estão sendo melhor desempenhadas pela equipe por terem seus prazos definidos, acompanhados e cumpridos na maior parte das vezes.

Existe um feedback constante do cliente, fazendo com que seja desenvolvido apenas o necessário. Este envolvimento do cliente possibilita priorizar apenas os itens mais importantes, aumentando assim a produtividade em todos os envolvidos (COHN, 2011). Na opinião da equipe sobre métodos ágeis 90% afirmaram que há aumento na produtividade da equipe(figura 14). Com base nos resultados do questionário foi possível observar que houve aumento da produtividade da equipe, pois, antes de adotar este método, as especificações dos requisitos nem sempre eram bem compreendidas, diminuindo assim a produtividade da equipe, que levava muito tempo para interpretá-los.

Segundo os membros da equipe de desenvolvimento, os clientes estão muito mais satisfeitos com a adoção do Scrum, conforme ilustra a figura 15. Percebeu-se que com os requisitos especificados da forma correta, houve aumento na produtividade e na satisfação dos clientes,

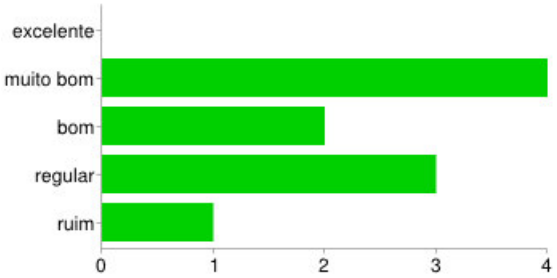

Figura 12 – Qual sua percepção sobre o uso do Scrum na Koin?

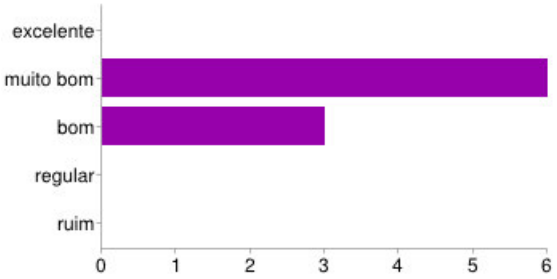

Figura 13 – Qual a sua percepção sobre o desempenho da equipe após a adoção do Scrum?

que passaram a obter produtos funcionais e de qualidade em pouco tempo. Deste modo, os cliente podem interagir com a equipe de desenvolvimento sobre melhorias, ou alterações durante o processo de criação do software.

De acordo com o questionário aplicado ao CTO(diretor de tecnologia) os fatores que indicam o sucesso do Scrum na empresa pôde ser observado de diferentes formas e tem diferentes áreas da empresa. Por exemplo, para áreas de negócio o sucesso foi demonstrado pelo aumento na quantidade de tarefas atendidas em cada sprint. Para a gestão da empresa a melhoria foi identificada pelo aumento da produtividade da equipe como um todo sem a necessidade do aumento de recursos. Para a gestão de TI, a melhoria foi identificada no aumento de conhecimento de diferentes módulos do sistema dentro da equipe de desenvolvimento e a facilidade de identificação de impeditivos. Para os usuários a melho-

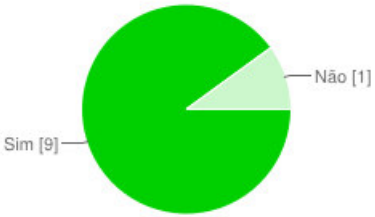

Figura 14 – Na sua opinião, os métodos ágeis aumentam a produtividade do time?

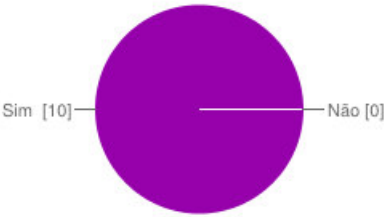

Figura 15 – Na sua opinião, o método ágil aumentou a satisfação do cliente?

ria foi demonstrada na diminuição do tempo de atendimento dos bugs e registros de não conformidade. É importante ressaltar que não obtivemos acesso aos números e dados que comprovam estas informações que foram repassadas tanto pelo diretor de tecnologia quanto pelos demais membros da equipe.

4.5 AS FERRAMENTAS ADOTADAS

A Koin utiliza a ferramenta Jira para gerenciar seu projeto. De acordo com o questionário aplicado, o product owner afirmou que a Koin optou por utilizar o Jira por ser uma ferramenta de mercado, notoriamente reconhecida como prática para o controle de projetos e desenvolvimento de software. O JIRA possui componentes focados para a metodologia Scrum e é altamente configurável para a criação de fluxos de trabalho, sem necessidade de programação. Com base na avaliação

dos entrevistados quanto a adoção da ferramenta Jira na Koin, 60% acharam regular, 20% bom, 10% muito bom e apenas 10% acharam excelente, conforme ilustrado na Figura 16.

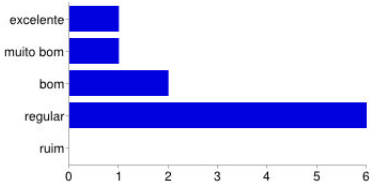

Figura 16 – Qual a sua avaliação da ferramenta Jira adotada pela empresa?

A ferramenta Jira já era utilizada antes do Scrum. Ela era aplicada apenas como bug tracking, ou seja, para coletar bugs do projeto.

Hoje com a adesão do scrum ela é usada também como Scrum Board, onde é depositado o product backlog e a sprint em execução naquele determinado momento. O Scrum board no Jira, é como um quadro de tarefas, só que de forma virtual. Todos tem acesso ao scrum board e podem visualizar os itens do product backlog e mover algum item da sprint que esteja na coluna "desenvolvimento"para a coluna "concluido". A intenção de utilizar o Jira foi para que todos da equipe pudessem trabalhar em conjunto e visualizar as funcionalidades que irão ser executadas durante aquela sprint. Conforme ilustra a Figura 17, em relação a aceitação de um quadro de tarefas físico utilizar o Jira(virtual), 56% dos entrevistados acham não seria útil a adoção de um quadro físico na Koin, maior parte da equipe está contente com a ferramenta adotada.

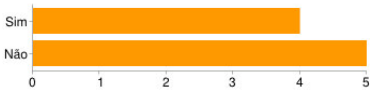

Figura 17 – Você acha que a adoção de um quadro físico seria mais útil do que a versão virtual (Jira)?

4.6 CONSIDERAÇÕES FINAIS SOBRE A UTILIZAÇÃO DO SCRUM NA KOIN

A adoção do Scrum na Koin teve impacto positivo. Na época em que foi adotada tal metodologia, alguns estavam em momento de aprendizado na empresa, as reuniões diárias fizeram com que estas pessoas pudessem ter maior entendimento do projeto, pois na reunião eram discutidos detalhes do que cada um estava esta desenvolvendo ou testando.

Na reunião de planejamento feito entre o product owner e a equipe de desenvolvimento os detalhes eram mais especificados, e qualquer dúvida sobre algum item não entendido pela equipe era exposto de uma forma mais clara durante esta reunião. Antes da adoção havia desorganização neste aspecto e consequentemente o produto final não saía como planejado, acarretando na insatisfação do cliente. Os stakeholders envolvidos perceberam que a adesão do scrum obteve resultados positivos porque a cada sprint concluída a equipe entregava um produto executável sem precisar esperar todas as funcionalidades em uma grande versão final.

A área de tecnologia inicialmente era dividida por área de conhecimento, então, cada profissional atacava somente o desenvolvimento de suas especialidades. Com a metodologia scrum, cada profissional passou a desenvolver novos recursos com mais flexibilidade, o que resultou num alinhamento maior com as funcionalidades existentes do produto final.

5 CONSIDERAÇÕES FINAIS E PROPOSTAS PARA TRABALHOS FUTUROS

O uso de uma metodologia ágil em uma organização pode levar muitos benefícios tanto para a organização quanto para o cliente. Entretanto, cabe a organização avaliar a metologia ágil que mais irá se adaptar com os processos e com a cultura organizacional assim como, a forma de trabalho e interação entre os membros da equipe. Independente da tecnologia utilizada, o sucesso de qualquer metodologia aplicada numa organização, parte das pessoas que irão trabalhar sob tal metodologia e na firmeza da organização em assegurar que a metodologia seja aplicada com sucesso.

Para isso, é necessária uma transformação cultural, que deve ser disseminada pela organização através de treinamento, incentivo e avaliação de aderência. Se as pessoas não utilizarem a metodologia adotada de forma plena, o processo de trabalho não será consistente e resultará em prejuízos para a organização. Não adianta ter a melhor ferramenta para controlar e gerir uma metodologia de trabalho se as pessoas não estão trabalhando aderente às boas práticas.

Os objetivos deste trabalho foram plenamente atingidos, pois foi possível confirmar os benefícios da implantação da cultura ágil no setor de desenvolvimento da empresa Koin. Com os resultados dos questionários, pode-se evidenciar que as hipóteses inicialmente levantadas foram confirmadas. Houve uma melhora na comunicação entre os membros da equipe, principalmente com as reuniões diárias, que permitem a troca de experiências e rápida identificação e resolução de problemas. A melhora na comunicação e a documentação adequada, das necessidades e tarefas a serem realizadas, também geraram uma melhora na produtividade e uma redução de retrabalho. E por fim, sob a ótica do cliente, o produto e os serviços tem sido entregues com melhor qualidade e em menor tempo.

Desta forma, a empresa Koin pode ainda ampliar os benefícios obtidos estendendo a cultura ágil para outros setores da empresa. Além disso, como trabalho futuro pretende-se aprofundar os conhecimentos a respeito da implantação dos testes ágeis e automatizados, pois no momento o setor de testes pode explorar melhor os valores e princípios ágeis, assim como, adotar ferramentas que auxiliem neste processo.

REFERÊNCIAS

ALMEIDA, S. de; MARçAL, R. F. M.; KOVALESKI, J. L. Metodologias para avaliação de desempenho organizacional. *XXIV Encontro Nac. de Eng. de Produção*, p. 7, 2004.

BECK, K. *Extreme Programming Explained: Embrace Change*. Addison-Wesley, 2000. (An Alan R. Apt Book Series). ISBN 9780201616415. <http://books.google.com.br/books?id=G8EL4H4vf7UC>.

BECK, K. et al. *Manifesto for Agile Software Development*. 2001. <http://agilemanifesto.org/>. Acessado em 20 de junho de 2014.

BROD, C. *Scrum Guia Prático para Projetos Ágeis*. Novatec Editora, 2013. ISBN 9788575223765. <http://books.google.com.br/books?id=oO-RAwAAQBAJ>. Acessado em 15 de junho de 2014.

COHN, M. *Desenvolvimento de Software com Scrum - Aplicando métodos ágeis com sucesso*. [S.l.]: Bookman, 2011.

CRUZ, F. *Scrum e Pmbok - Unidos no Gerenciamento de Projetos*. [S.l.]: BRASPORT, 2013.

DANTAS, V. F. *UMA METODOLOGIA PARA O DESENVOLVIMENTO DE APLICAÇÕES WEB NUM CENÁRIO GLOBAL*. Dissertação (Mestrado) — Universidade Federal de Campina Grande - Centro de Ciência e Tecnologia, 2003.

DEMING, W. E. *Out of the Crisis: Quality, Productivity and Competitive Position*. Cambridge University Press, 1986. ISBN 9780521305532. <http://books.google.com.br/books?id=4qw8AAAAIAAJ>.

ECKSTEIN, J. *Agile Software Development with Distributed Teams: Staying Agile in a Global World*. Addison-Wesley, 2013. (Dorset House eBooks). ISBN 9780133492392. <http://books.google.com.br/books?id=mVgUAAAAQBAJ>.

FADEL, A. C.; SILVEIRA, H. M.

Metodologias ágeis no contexto de desenvolvimento de software: XP, Scrum e Lean — UNICAMP – Universidade Estadual de Campinas, 2010.

FILHO, D. L. B. *Experiências com desenvolvimento gil*. Dissertação (Mestrado) — Instituto de Matemática e Estatística da Universidade de São Paulo, 2008.

HIGHSMITH, J. *History: The Agile Manifesto*. 2001. <http://agilemanifesto.org/history.html>. Acessado em 28 de junho de 2014.

HIRAMA, K. *Engenharia De Software - Qualidade E Produtividade Com Tecnologia*. [S.l.]: Elsevier - Campus, 2012.

JUNIOR, C. H.; YANZER, A. Aplicação de métodos ágeis em um processo de desenvolvimento de software. junho 2011.

KNIBERG, H. *Scrum e XP direto das Trincheiras - Como nós fazemos o Scrum*. [S.l.]: C4Media, InfoQ.com, 2007.

KOIN. *Quem Somos? Profissionais Koin*. 2014. <http://www.koin.com.br/home/index>. Acessado em 20 de julho de 2014.

LAFASTO, F. M. J.; LARSON, C. E. *When team work best: 6,000 team members and leaders tell what it takes succeed*. [S.l.]: Sage Publication, Inc, 2001.

LARSON, C. E.; LAFASTO, F. M. J. *Teamwork: What must go right/what can go wrong*. [S.l.]: SAGE Publications, 1989.

LAYTON, M. C. *Agile Project Management For Dummies*. John Wiley & Sons, 2012. (–For dummies). ISBN 9781118222140. <http://books.google.com.br/books?id=zLuB_sYpAt4C>.

LEITE, J. *Notas de aula de Engenharia de Software*. http://www.dimap.ufrn.br/ jair/ES/c2.html: [s.n.], 2000. Acessado em 30 de maio de 2014.

LIB. *O que é - Lean Thinking*. 1998. <http://www.lean.org.br>. Acessado em 27 de junho de 2014.

LIBARDI, P. L. O.; BARBOSA, V.

Métodos Ágeis — Universidade Estadual de Campinas – UNICAMP, 2010.

MGS. *Scrum*. 2012. <http://www.mountaingoatsoftware.com/agile/scrum>. Acessado em 01 de junho de 2014.

MGS. *Sprint Backlog*. 2012. <http://www.mountaingoatsoftware.com/agile/scrum/sprint-backlog/>. Acessado em 29 de junho de 2014.

MGS. *Sprint planning meeting*. 2012. <http://www.mountaingoatsoftware.com/agile/scrum/sprint-planning-meeting/>. Acessado em 20 de junho de 2014.

MGS. *Task Boards*. 2012. <http://www.mountaingoatsoftware.com/agile/scrum/task-boards/>. Acessado em 26 de junho de 2014.

MILLER, L.; SY, D. Agile user experience sig. *Conference on Human Factors in Computing Systems*, 2009.

MUNDIM, A. P. F. et al. Aplicando o cenário de desenvolvimento de produtos em um caso prático de capacitação profissional. *Gestão & Produção*, v. 9, 2002.

PFLEEGER, S. L. *Engenharia de Software - Teoria e Prática*. [S.l.]: Prentice Hall - Br, 2004. 37 p.

POPPENDIECK, M.; POPPENDIECK, T. *Lean Software Development*. [S.l.]: Addison Wesley, 2007.

PRESSMAN, R. S. *Engenharia de Software*. [S.l.]: McGraw-Hill, 2006.

PRESSMAN, R. S. *Engenharia de Software - Uma Abordagem Profissional*. [S.l.]: AMGH Editora, 2011.

RUAS, S. G. *GERENCIAMENTO ÁGIL DE PROJETOS EM TA*. 2011. <http://www.techplus.com.br/itech/noticias/artigos/01.pdf>.

RUBIN, K. S. *Essential Scrum: A Practical Guide to the Most Popular Agile Process*. [S.l.]: Addison-Wesley Professional, 2012.

SCHWABER, K. *Agile Project Management with Scrum*. Pearson Education, 2004. (Developer Best Practices). ISBN 9780735637900. <http://books.google.com.br/books?id=6pZCAwAAQBAJ>.

SCHWABER, K.; SUTHERLAND, J. *Guia do Scrum: Um guia definitivo para o Scrum: As regras do jogo*. Outubro 2011. <https://www.scrum.org/Scrum-Guide>. Acessado em 19 de junho de 2014.

SOARES, M. D. S. Comparação entre metodologias Ágeis e tradicionais para o desenvolvimento de software. *INFOCOMP Journal of Computer Science*, v. 3, p. 6, 2004.

SOMMERVILLE, I. *Engenharia de software*. [S.l.]: Addison Wesley, 2003.

SOMMERVILLE, I. *Engenharia de Software*. [S.l.]: Pearson Addison Wesley, 2007.

SOMMERVILLE, I. *Engenharia de Software*. [S.l.]: PEARSON, 2011.

STEFFEN, J. B. *O que são essas tais de metodologias Ágeis ?* January 2012. <https://www.ibm.com/developerworks/community/blogs/rationalbrasil/?lang=en>. Acessado em 02 de junho de 2014.

APÊNDICE A – Questionário sobre análise da adoção do scrum na empresa Koin I

A.1 QUESTIONÁRIO

Entrevistado:

Cargo:

Ativa na empresa desde:

Você já ouviu falar em metodologias ágeis?

() Sim () Não

Você conhece o Scrum?

() Sim () Não

Você conheceu o Scrum onde:

() na universidade () na empresa () na internet () em conferência

Você recebeu treinamento para adotar o Scrum nas suas atividades diárias?

() excelente ()muito bom ()bom ()regular ()ruim

Qual sua percepção sobre o uso so Scrum?

()excelente ()muito bom ()bom ()regular ()ruim

Qual a sua percepção sobre o desempenho da equipe após a adoção do Scrum:

()excelente ()muito bom ()bom ()regular ()ruim

Você avalia as ferramentas adotadas, como o Jira:

()excelente ()muito bom ()bom ()regular ()ruim

Você acha que a adoção de um quadro físico do Kanban seria mais útil do que a versão virtual (Jira):

()excelente ()muito bom ()bom ()regular ()ruim

Sugestão ou comentários:

APÊNDICE B – Questionário sobre análise da adoção do scrum na empresa Koin II

B.1 QUESTIONÁRIO

Como era antes de adotar o Scrum?

Como eram passadas as tarefas para a equipe?

Porque surgiu a ideia de utilizar o Scrum?

A idéia de utilizar o Scrum foi porque foi visto que muitas empresas que utilizam esta metodologia tem sucesso?

Você acha que a equipe se deu bem com o Scrum por ser uma empresa de pequeno porte?

Como era feita a coleta de requisitos dos cliente, suas especificações? como por exemplo: incluir alguma funcionalidade no sistema, algo que o cliente quisesse que fosse feito ou alterado no sistema.

Como eram passadas as tarefas para a equipe?

Como eram feitas as especificações das regras de negócio?

Porque resolveram utilizar o Jira?

As pessoas antes de utilizar o Scrum cada um tinha sua função, como por exemplo alguém que cuida do BD, só pegava funções de DBA, ou sempre foi feito na forma em que todos participam independente da funcionalidade que irá ser feita?

Tem algo em que você acha que pode ser inserido, alguma tecnologia que iria acelerar o processo de desenvolvimento da equipe

Existe alguma documentação que comprove as melhorias com a adoção do SCRUM? Caso não tenha como você afirma as melhorias, baseados em quais informações?