Aplicação de Metodologias Ágeis para Desenvolvimento de Software: um Estudo de Caso na Empresa *Alliance Software*

Juliano da Conceição[1], Sidnei Renato Silveira[2]

[1]Curso de Bacharelado em Sistemas de Informação , [2]Departamento de Tecnologia da Informação – CESNORS (Centro de Educação Superior Norte do RS) Frederico Westphalen – RS – Brasil - UFSM – Universidade Federal de Santa Maria

conceicao_jc@hotmail.com, sidneirenato.silveira@gmail.com

Resumo. *Este artigo apresenta um estudo de caso sobre metodologias ágeis de desenvolvimento de software na empresa Alliance Software, localizada em Palmeira das Missões - RS. Inicialmente, são abordados os métodos atuais de desenvolvimento empregados nas empresas. O estudo de caso teve como objetivo, definir, aplicar e coletar os resultados do emprego de metodologias ágeis no desenvolvimento de software nestas empresas.*

Palavras-Chave: *Metodologias Ágeis, Desenvolvimento de Software, Engenharia de Software.*

Abstract. *This paper presents a case study on agile methodologies, in the Alliance Software company, in Palmeira das Missões - RS. Initially, are addressed the current development methods applied in enterprises. The case study aims to define, enforce and collect the results of the use of agile methodologies in software development these companies.*

Keywords: *Agile methodologies, Software Development, Software Engineering.*

1. Introdução

O principal objetivo deste trabalho foi o de realizar um estudo de caso envolvendo as empresas de desenvolvimento de *software* de Palmeira das Missões - RS, visando identificar as metodologias de desenvolvimento de *software* empregadas e propor a utilização de metodologias ágeis.

Com o passar dos anos, as empresas de desenvolvimento de *software* começaram a ganhar espaço em Palmeira das Missões - RS, realizando um trabalho importante para a comunidade. Entretanto, atualmente elas encaram um grande desafio, que é o de conciliar o desenvolvimento de *software* com os requisitos solicitados pelos clientes, já que, segundo levantamento realizado pelos autores deste trabalho, elas não seguem um modelo padronizado de desenvolvimento. A não aplicação de um modelo padronizado ocasiona atraso na entrega dos produtos e, na maioria das vezes, o produto solicitado pelo cliente não reflete o desejado, devido à má interpretação dos requisitos.

Neste contexto, este trabalho teve, por objetivo, auxiliar as empresas de desenvolvimento de *software* localizadas em Palmeira das Missões - RS, apresentando

métodos de desenvolvimentos ágeis, para que as mesmas tenham mais agilidade e facilidade no decorrer de suas atividades, visando evitar atrasos e falhas na entrega dos produtos de *software* para os clientes.

Inicialmente, foi realizado um diagnóstico envolvendo os processos de desenvolvimento de *software* aplicados pelas empresas de Palmeira das Missões – RS, bem como o estudo de conceitos, técnicas e ferramentas para a aplicação de metodologias ágeis nestes processos. A partir destes estudos, foram definidas uma ou mais técnicas ágeis para serem aplicadas em uma destas empresas, visando realizar um estudo de caso da aplicação destas metodologias.

Neste sentido, o presente artigo está organizado da seguinte forma: a seção 2 apresenta o referencial teórico, abordando conceitos de Engenharia de *Software* e algumas metodologias de desenvolvimento de *software*. A terceira seção apresenta alguns trabalhos relacionados, visando compor o estado da arte. Na seção 4 é detalhado o estudo de caso realizado apresentando, também, algumas informações sobre as empresas de desenvolvimento de *software* localizadas em Palmeira das Missões - RS. Encerrando o artigo, são apresentadas as considerações finais e as referências empregadas.

2. Referencial Teórico

Esta seção apresenta alguns conceitos sobre as áreas envolvidas no desenvolvimento deste trabalho, tais como Engenharia de *Software* e algumas metodologias ágeis para o desenvolvimento de *software*.

2.1 Conceitos de Engenharia de *Software*

O conceito de Engenharia de *Software* foi incialmente proposto em 1968, em uma conferência organizada para discutir o que foi então chamado de "crise do *software*". A crise do *software* resultava indiretamente da introdução de um novo *hardware* de computador baseado em circuitos integrados. A aplicação de circuitos integrados fez das aplicações de computador, consideradas até então não realizáveis, propostas viáveis. O software resultante era maior e mais complexo que sistemas anteriores de *software* (SOMMERVILLE, 2007).

Praticamente todos os países hoje em dia, dependem de sistemas complexos baseados em computadores. Infraestruturas e serviços nacionais contam com sistemas baseados em computadores, e a maioria dos produtos elétricos inclui um computador e um *software* de controle. A manufatura e a distribuição industriais estão completamente automatizadas, assim como os sistemas financeiros. Portanto, produzir e manter o software dentro de custos adequados e com qualidade é essencial para o funcionamento da economia nacional e internacional (SOMMERVILLE, 2007).

Para Sommerville (2007), a Engenharia de *Software* é um ramo de engenharia, cujo foco é o desenvolvimento dentro de custos adequados de sistemas de *software* de alta qualidade.

A Engenharia de *Software* é uma tecnologia em camadas, que envolvem ferramentas, métodos, processo e foco na qualidade. Qualquer abordagem de

engenharia (inclusive de *software*) deve estar fundamentada em um comprometimento organizacional com a qualidade. A gestão da qualidade total Seis Sigma[1] (GYGI; DECARLO; WILLIAMS, 2008) e filosofias similares promovem uma cultura de aperfeiçoamento contínuo de processos, e é esta cultura que, no final das contas, leva ao desenvolvimento de abordagens cada vez mais efetivas na Engenharia de *Software*. A pedra fundamental que sustenta a engenharia do *software* é o foco na qualidade (PRESSMAN, 2011).

2.2 Metodologias de Desenvolvimento de *Software*

Uma metodologia de desenvolvimento é o conjunto de práticas recomendadas para o desenvolvimento de *software*. Essas práticas podem ser subdivididas em fases para ordenar e gerenciar o processo (SOMMERVILLE, 2007).

Originalmente, modelos de processo prescritivo foram propostos para trazer ordem ao caos existente na área de desenvolvimento de *software*. A história tem demonstrado que esses modelos tradicionais proporcionaram uma considerável contribuição quanto à estrutura utilizável no trabalho de engenharia de *software* e fornecem um roteiro razoavelmente eficaz para apoiar as equipes de *software* (PRESSMAN, 2011).

Para Pressman (2011), os modelos de processo de desenvolvimento de *software* mais utilizados são:

- **Modelo Cascata:** O modelo cascata, algumas vezes chamado ciclo de vida clássico, sugere uma abordagem sequencial e sistemática para o desenvolvimento de *software*, começando com o levantamento e necessidades por parte do cliente, avançando pelas fases de planejamento, modelagem construção, emprego e culminando no suporte contínuo do *software* concluído;
- **Modelo Incremental:** O modelo incremental aplica sequências lineares, de forma escalonada, à medida que o tempo vai avançando. Cada sequência linear gera "incrementos" (entregáveis/aprovados/liberados) do *software* de maneira similar aos incrementos gerados por um fluxo de processos evolucionários;
- **Modelos de Processo Evolucionário:** Modelos evolucionários são iterativos. Apresentam características que possibilitam desenvolver versões cada vez mais completas de *software*. Os dois modelos mais comuns em processos evolucionários são a prototipação onde, frequentemente, o cliente define uma série de objetos gerais para o *software*, mas não identifica, detalhadamente, os requisitos para funções e recursos; e o modelo espiral, que é um modelo de processo de *software* evolucionário que acopla a natureza iterativa da prototipação com os aspectos sistemáticos e controlados do modelo cascata;
- **Modelos Concorrentes:** A modelagem concorrente se aplica a todos os tipos de desenvolvimento de *software*, fornecendo uma imagem precisa do estado atual

[1] *Seis Sigma* é uma metodologia para solução de problemas que minimiza erros e maximiza valores. (GYGI; DECARLO e WILLIAMS, 2008)

de um projeto. Em vez de limitar as atividades, ações e tarefas da engenharia de *software* a uma sequência de eventos, ela define uma rede de processos.

2.3 Metodologias Ágeis

O termo Metodologias Ágeis foi criado em 2001, quando dezessete especialistas em processos de desenvolvimento de *software*, representando os métodos Scrum (COHN, 2011), e*Xtreme Programming* (XP) (BECK, 2008) e outros, criaram a Aliança Ágil e a denominaram Manifesto Ágil (AGILE MANIFESTO, 2015).

Para Pressman (2011), metodologias ágeis se desenvolveram em um esforço para sanar fraquezas reais e perceptíveis da Engenharia de *Software* convencional. O desenvolvimento ágil oferece benefícios importantes, no entanto, não é indicado para todos os projetos, produtos, pessoas e situações, também não é antítese da prática de Engenharia de *Software* consistente[2] e pode ser aplicado como uma filosofia em geral para todos os trabalhos de *software*.

Processos de desenvolvimento rápido de *software* são projetados para criar *software* útil rapidamente. Geralmente, eles são processos iterativos nos quais a especificação, o projeto, o desenvolvimento e o teste são intercalados. O *software* não é desenvolvido e disponibilizado integralmente, mas em uma série de incrementos e, cada incremento inclui uma nova funcionalidade do sistema (SOMMERVILLE, 2007).

Entre as metodologias ágeis existentes, este trabalho aborda conceitos de XP, SCRUM, *Lean* e *Kanban*, que são discutidas nas próximas seções.

2.3.1 *eXtreme Programming* (XP)

Para Beck (2008), a XP é uma maneira leve, eficiente, de baixo risco, flexível, previsível, científica e divertida de desenvolver *software*. Tem, como principais diferenças em relação às outras metodologias, o *feedback* contínuo, derivado dos ciclos curtos e a abordagem Incremental de planejamento, que gera rapidamente um plano geral que vai evoluir com o decorrer do projeto.

A XP é a combinação de uma abordagem colaborativa, livre de desconfianças, com um conjunto de boas práticas de engenharia de *software* que são eficientes por si só, individual e independentemente do contexto. Cada uma dessas práticas contribui para o aumento da qualidade do *software* e ajuda a garantir que o produto final agregue valor e atenda as necessidades do negócio (PRIKLADNICKI; WILLI e MILANI, 2014).

A XP foi desenvolvida para funcionar em projetos com times de dois a dez programadores, que não sejam severamente restringidos pelo ambiente computacional existente e, nos quais, boa parte da execução de testes possa ser feita em uma fração de dia (BECK, 2008).

---

[2] Engenharia de *Software* consistente ou convencional engloba processos, métodos e ferramentas que possibilitam a construção de sistemas complexos baseados em computador dentro do prazo e com qualidade (PRESSMAN, 2011)

A XP assume que a volatilidade dos requisitos existe e, em vez de tentar eliminá-la, trata o desenvolvimento do *software* a partir de uma abordagem flexível e colaborativa, na qual desenvolvedores e clientes fazem parte de uma única equipe que tem o propósito de produzir software de alto valor agregado (PRIKLADNICK; WILLI e MILANI, 2014).

A XP envolve um conjunto de regras e práticas constantes no contexto de quando de atividades metodológicas: planejamento, projeto, codificação e testes. A Figura 1 ilustra o processo XP e destaca alguns conceitos e tarefas-chave associadas a cada uma das atividades metodológicas (PRESSMAN, 2011).

Figura 1: O Processo da XP (Pressman, 2011)

2.3.2 *Scrum*

O termo *Scrum* surgiu em um artigo publicado por Hirotaka Tekeuchi e Ikujiro Nonaka na *Harvard Business Review* de 1986. Nesse artigo, intitulado "*The new new product development game*" ("O novo novo game do desenvolvimento de produtos"), Takeuchi e Nonaka descreveram uma abordagem holística, na qual equipes de projeto são compostas por pequenas equipes multifuncionais, trabalhando com sucesso rumo a um objetivo comum, que os autores compararam à formação do *Scrum* do *Rugby* (PHAM; PHAM, 2012).

Para Prikladnicki, Willi e Milani (2014), o *Scrum* é um *framework* ágil que auxilia no gerenciamento de projetos complexos e no desenvolvimento de produtos. É conhecido como um *framework* que prescreve um conjunto de práticas leves e objetivas, muito utilizadas na área de desenvolvimento de *software*. As práticas do *Scrum* também podem ser utilizadas para projetos de outra natureza, desde que possuam certo grau de complexidade, pois só assim suas práticas de inspeção e adaptação fazem sentido.

O *Scrum* fornece uma maneira para que as equipes trabalhem juntas para desenvolver um produto. O desenvolvimento de produtos, utilizando *Scrum*, ocorre em pequenos pedaços, com cada peça sobre peças criadas anteriormente. A construção de

uma peça pequena de cada vez estimula a criatividade e permite que as equipes correspondam ao *feedback* e mudança, para construir exatamente e somente o que é necessário (SCRUM.ORG, 2015).

O *Scrum* é um *framework* dentro do qual, pessoas podem tratar e resolver problemas complexos e adaptativos, enquanto produtiva e criativamente entregam produtos com o mais alto valor possível (SCRUMGUIDES.ORG, 2015).

O *Scrum* maximiza a entrega de *software* de modo eficaz, adaptando-se à realidade das mudanças. As funcionalidades de maior valor são desenvolvidas antecipadamente, enquanto se reflete sobre a necessidade ou não das menos prioritárias (PRIKLADNICKI; WILLI e MILANI, 2014).

Os princípios do *Scrum* são consistentes com o manifesto ágil e são usados para orientar as atividades de desenvolvimento dentro de processo que incorpora as seguintes atividades estruturais: requisitos, análise, projeto, evolução e entrega. Em cada atividade metodológica, ocorrem tarefas a realizar dentro de um padrão de processo chamado *Sprint*[3]. O trabalho realizado dentro de um *Sprint* é adaptado ao problema em questão e definido, e muitas vezes modificado em tempo real, pela equipe de desenvolvimento *Scrum*. A Figura 2 ilustra o fluxo geral do *Scrum* (PRESSMAN, 2011).

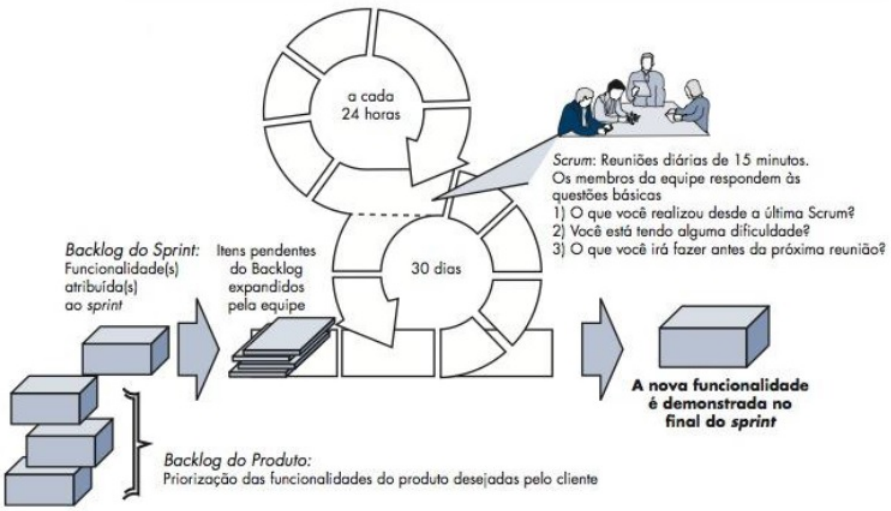


Figura 2: Fluxo geral do *Scrum* (Pressman, 2011)

2.3.3 *Lean Software Development* (LSD)

Segundo o *Lean Institute* Brasil (2015), a metodologia *Lean* é uma filosofia e estratégia de negócios para aumentar a satisfação dos clientes por meio da utilização dos recursos de forma adequada. Procura fornecer, de forma consistente, valor aos clientes com os custos mais baixos, identificando e sustentando melhorias nos fluxos de valor primários e secundários, por meio do envolvimento das pessoas qualificadas, motivadas e com

[3] São os ciclos em que o *Scrum* é dividido, geralmente são ciclos mensais. O Sprint representa um *Time Box* dentro do qual um conjunto de atividades deve ser executado (PRESSMAN, 2011)

iniciativa. O foco da implementação deve estar nas reais necessidades dos negócios e não na simples aplicação das ferramentas *Lean* (LEAN.ORG, 2015).

As práticas *Lean* envolvem a criação de fluxos contínuos e sistemas puxados[4] baseados na demanda dos clientes, a análise e melhoria do fluxo de valor das plantas e da cadeia completa, desde as matérias-primas até os produtos acabados e o desenvolvimento de produtos que efetivamente sejam solução do ponto de vista do cliente (LEAN.ORG, 2015).

A Figura 3 apresenta a pirâmide *Lean*, que provê princípios, técnicas e ferramentas para dar suporte às melhorias de forma balanceada e incremental em cada uma das áreas da organização, cobrindo desde a estratégia de negócio, passando pelo gerenciamento, pela engenharia de *software*, pelo desenvolvimento de produtos até operações (PRIKLADNICKI; WILLI e MILANI, 2014).

O modelo se divide em três partes fundamentais (PRIKLADNICKI; WILLI e MILANI, 2014):

- **Topo (Estratégia):** Apresenta princípios e valores fundamentais oriundos do *Lean* e do Manifesto Ágil como base para a formação de uma cultura ágil que visa à criação de um fluxo de entrega de valor constante, de uma organização onde o aprendizado é implícito e a melhora contínua é o *modus operandi*. Além disso, é no topo da pirâmide que são definidas a estratégia de negócios, a visão e a percepção de valor, elementos que precisam estar claros para todo e qualquer membro da organização;
- **Centro (Gestão):** A aplicação dos princípios, dos valores e da estratégia de negócios é realizada com o auxílio de ferramentas consagradas, como o *Scrum* e o *Kanban*. A visão é desmembrada em objetos específicos e mensuráveis pela criação e priorização de *backlog (colocar uma nota de rodapé com o conceito de backlog)* de estórias que representam o valor de negócios definido. É também nesta camada da organização que se estabelece o planejamento necessário para unir a estratégia à engenharia e possibilitar a entrega real de valor de forma cadenciada e mais previsível. Aqui, estabelece-se o modelo de gestão visual tão fundamental para ambientes ágeis;
- **Base (Engenharia):** A base da pirâmide foca em práticas de engenharia de *software* essenciais para promover a criação de um ambiente seguro, produtivo, flexível e sustentável, permitindo respostas rápidas às mudanças, possibilitando vantagens competitivas à empresa. Essas práticas de engenharia estão na base da pirâmide, pois são a parte mais importante de um processo de adoção ágil, são elas que permitem a entrega contínua e confiável de *software*. Sem elas, um processo de adoção ágil estaria viciado e entregando resultados apenas marginais.

[4] Um sistema puxado ou produção puxada significa reduzir ao máximo o estoque produzindo apenas o que o cliente comprou. No caso do desenvolvimento de software é fazer apenas o que o cliente solicitou. (LEAN.ORG, 2015)

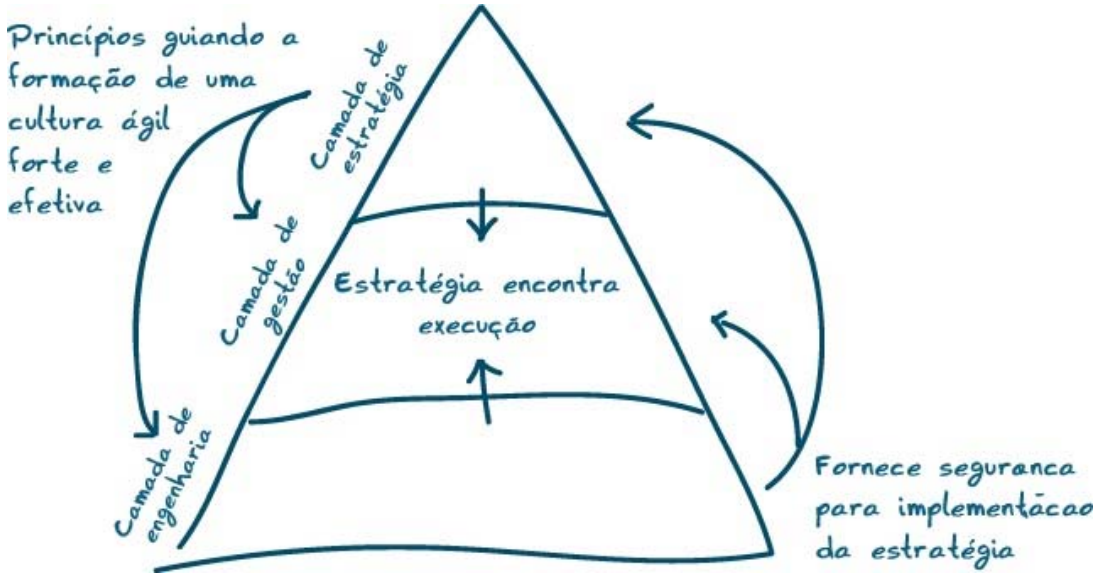


Figura 3: Pirâmide *Lean* (Prikladnick; Willi e Milani, 2014)

2.3.4 *Kanban*

A utilização de um sistema *Kanban* permite um controle detalhado de produção com informações sobre quando, quanto e o que produzir. O método *Kanban* foi inicialmente aplicado em empresas japonesas de fabricação em série e está estreitamente ligado ao conceito de "*just in time*". A empresa japonesa de automóveis *Toyota* foi a responsável pela introdução desse método devido a necessidade de manter um funcionamento eficaz do sistema de produção em série. O Método *Kanban* foi formulado por David J. Anderson e é baseado em práticas *Lean*, visando aperfeiçoar o processo de desenvolvimento de *software* pré-existente. Este método limita o trabalho em progresso, apresentando a evolução de forma visual, tornando os problemas evidentes, proporcionando uma cultura de melhoria contínua (ARRUDA, 2012).

As práticas *Lean* caracterizam a estrutura do *Kanban*. O principal objetivo da metodologia *Kanban* é avaliar o trabalho (WIP *Work in Progress*), visando mostrar quando uma funcionalidade do *software* pode ser arquitetada, codificada, testada, etc., por meio da visualização. O *Kanban* é uma palavra japonesa que significa cartão sinalizador. Por meio de um quadro de atividades, é possível perceber qual a quantidade de esforço que poderá ser adicionada, levando-se em consideração a indicação de limite e de capacidade da equipe e do *software* que está sendo desenvolvido. Desta forma, a equipe pode fazer entregas a qualquer instante para o cliente e o mesmo pode modificar a importância das atividades quando desejar, tornando o desenvolvimento mais transparente, já que é possível visualizar o fluxo de trabalho, sem preocupações com as iterações e estimativas, como em outros métodos (ARRUDA, 2012).

Na área de *software*, ferramentas *Kanban* começaram a ser utilizadas com o surgimento e a progressiva adoção de métodos ágeis, tais como XP e *Scrum*. Nesses métodos, as equipes posicionam quadros em parede visíveis da sala de projetos, preenchendo-os com cartões que sinalizam os itens de trabalho selecionados para uma dada iteração. Normalmente, os cartões são posicionados conforme seu estado presente:

não iniciado, em andamento e finalizado. A Figura 4 mostra o quadro visual utilizado tradicionalmente em projetos ágeis.

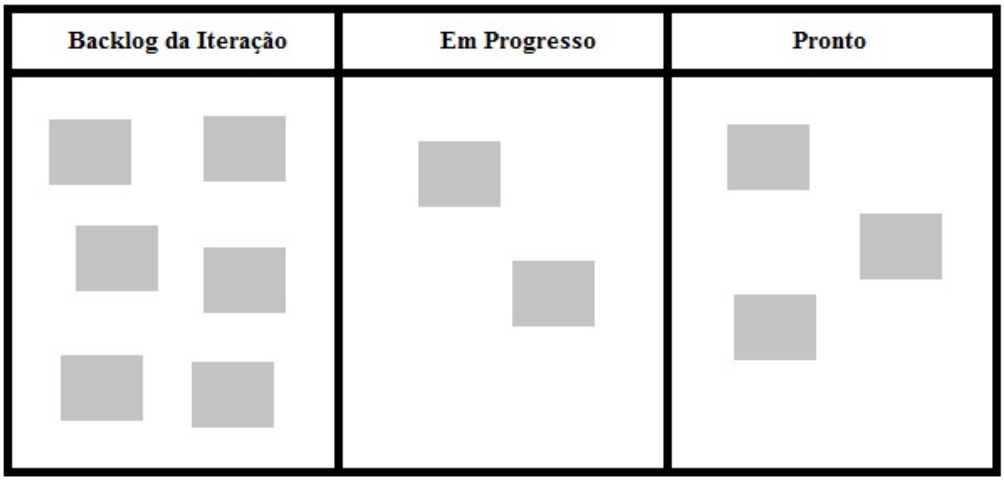


Figura 4: Quadro Tradicional *Kanban* (Adaptado de Prikladnick; Willi e Milani, 2014)

3. Estado da Arte

Nesta seção são abordados alguns trabalhos que serviram como base para o estudo realizado. Após a apresentação das principais características, realizou-se um comparativo entre os trabalhos aqui apresentados e o estudo de caso desenvolvido.

3.1 Desenvolvimento de *Software* através de *eXtreme Programming* e *Scrum*

Soares (2004) apresenta um estudo destacando as vantagens obtidas na aplicação de metodologias ágeis, como o e*Xtreme Programming* e o *Scrum*, em relação à aplicação de metodologias tradicionais no processo de desenvolvimento de *software*. Comparando as metodologias ágeis em relação às tradicionais, tais como o modelo em Cascata, o estudo traz como pontos fortes das metodologias ágeis: a viabilidade de adaptação aos fatores decorrentes do processo de desenvolvimento; o modo com que recebem, avaliam e respondem às mudanças no processo; e maior possibilidade de atender aos requisitos do cliente.

Soares (2004) relacionou diversos estudos acerca de metodologias ágeis, expondo que, apesar do surgimento das mesmas ser recente, a maioria dos resultados iniciais, no que diz respeito à qualidade, confiabilidade, prazos de entrega e custos, são muito superiores aos das metodologias tradicionais. Entre estes estudos foram apresentados dois quadros com curvas de tempo e valores no desenvolvimento de *software*, percebendo-se uma grande diferença na curva de aumento de valores, sendo que nas metodologias ágeis os gastos foram reduzidos em comparação à outra metodologia devido ao tempo de desenvolvimento e aplicação. A justificativa para esta redução é porque as metodologias ágeis incentivam as mudanças no decorrer do processo, objetivando fornecer ao cliente o produto que ele realmente precisa.

De acordo com Soares (2004), o desafio para a aplicação de metodologias ágeis é buscar formas de subtrair seus pontos fracos, dentre eles a falta de análise de riscos,

sem torná-las metodologias pesadas como as tradicionais. Também, pelo fato de serem muito recentes, buscar modos de utilizá-las em empresas de grande porte, visto que sua maior utilização atualmente é feita em empresas que possuem uma equipe de desenvolvimento reduzida (SOARES, 2004).

3.2 Estudo Empírico Acerca do Uso de Metodologias Ágeis

Treccani e Souza (2010) realizaram um estudo empírico em empresas que utilizavam as Metodologias Ágeis para o desenvolvimento de *software*. Os caminhos metodológicos utilizados para a condução do estudo foram o uso da Teoria fundamentada em Dados, para a análise qualitativa dos resultados, os quais foram coletados por meio de entrevistas semiestruturadas com vinte e dois colaboradores de quatro empresas produtoras de *software* nos estados do Pará e Pernambuco, de diversos segmentos, como o da saúde, serviços, universidades, entre outros. As entrevistas foram gravadas, e posteriormente, transcritas para análise.

Os autores identificaram que todas as empresas entrevistadas utilizam o *Scrum* como a base do processo de desenvolvimento, incorporando inclusive outras metodologias, tais como a *eXtreme Programming* e a *Feature Driven Development*. Outras similaridades, envolvendo a totalidade das empresas entrevistadas foram observadas, como a necessidade de treinamento na metodologia (devido à baixa experiência da equipe desenvolvedora nas metodologias ágeis); o desenvolvimento sempre é realizado em pares; definição clara de papéis no desenvolvimento, além da presença de um ambiente de convivência, local este em que todos os colaboradores têm a liberdade de circular livremente.

O resultado relevante encontrado após a análise dos dados é a presença, em todas as empresas pesquisadas, de uma abordagem mais colaborativa das atividades, neste caso a refatoração, a qual consiste em alterar o *software* sem modificar seu comportamento externo, ou seja, realizando apenas modificações na estrutura interna do código-fonte. Entretanto, a literatura traz essa atividade como sendo individual, baseada nas metodologias tradicionais de desenvolvimento de *software*. Ocorre que as metodologias ágeis mudam este conceito, utilizando a ideia de atividades colaborativas em detrimento das individuais (TRECCANI; SOUZA, 2010).

3.3 Aprimorando a Gerência e o Desenvolvimento de Software com Metodologias Ágeis

O trabalho apresentado por Sganderla et. al. (2010) propôs o emprego de metodologias ágeis para o desenvolvimento de uma ferramenta de gerenciamento de processo seletivo de uma Instituição de Ensino Superior. Para gestão e acompanhamento, foram utilizadas as práticas do *Scrum* e para implementação, a metodologia *eXtreme Programming.*

Dentre as principais dificuldades encontradas para o desenvolvimento deste sistema, pode-se destacar a distância geográfica entre os desenvolvedores e o cliente, o prazo curto para desenvolvimento (devido às funcionalidades exigidas pelo cliente), bem como a mudança cultural no modelo de gerenciar e desenvolver o *software*. Todavia, os benefícios trazidos por meio desta experiência superaram as dificuldades: o ambiente de trabalho foi projetado com vistas à colaboração e motivação entre a equipe,

a formação de um time multidisciplinar possibilitou a troca de experiências, houve o aumento da confiança e autoestima dos colaborados no decorrer do processo, entre outros aspectos positivos (SGANDERLA et. al., 2010).

O sistema foi colocado em produção e disponibilizado após 6 meses de processo de desenvolvimento, superando as expectativas. Conseguiu-se não apenas superar os riscos do projeto, como também automatizar vários processos e integrar o novo produto com um outro produto financeiro já utilizado na instituição. Desta forma, concluiu-se que, por meio da aplicação de metodologias ágeis, foi possível otimizar o processo de trabalho, possibilitando a introdução de novas práticas para o desenvolvimento de um software.

3.4 Uso de Metodologias Ágeis no Desenvolvimento de Software

Um estudo de caso realizado por Oliveira (2003) teve, por objetivo, apresentar as metodologias ágeis como alternativa aos processos tradicionais utilizados na Engenharia de *Software*, aplicadas ao desenvolvimento de um Sistema de Embarque da Companhia Siderúrgica de Tubarão, localizada na cidade de Serra - ES.

Após a apresentação das necessidades mínimas requisitadas pelos clientes, a maior dificuldade encontrada foi o prazo para entrega: dois meses para desenvolver uma interface em coletores de dados portáteis para o embarque de bobinas no Terminal de Produtos Siderúrgicos da companhia. As metodologias ágeis escolhidas para o desenvolvimento foram o *Scrum* e o *eXtreme Programming*, devido ao curto prazo para o projeto e pelo fato das mesmas serem complementares.

Após a implementação do sistema, que foi considerado satisfatório e atendeu a todos os requisitos exigidos pelo cliente, concluiu-se que as metodologias baseadas em processos adaptativos têm potencial de apresentar resultados melhores que os resultados apresentados pelas metodologias baseadas em processos preditivos, como também apresentam melhores condições de trabalho e melhores resultados que as metodologias orientadas a processos.

3.5 Estudo Comparativo

O Quadro 1 apresenta algumas características que foram encontradas nos trabalhos estudados, bem como um comparativo com o estudo de caso desenvolvido.

Quadro 1: Comparativo entre os trabalhos estudados (Fonte: dos autores, 2015)

Característica	Trabalho	Trabalho	Trabalho	Trabalho	Estudo de

s	apresentado por Soares (2004)	apresentado por Treccani e Souza (2010)	apresentado por Sganderla et. al. (2010)	apresentado por Oliveira (2003)	Caso desenvolvido
Aplicação do *Scrum*	Sim	Sim	Sim	Sim	Parcialmente
Aplicação de XP	Sim	Sim	Sim	Sim	Não
Aplicação de *Kanban*	Não	Sim	Não	Não	Sim
Aplicação de *Lean*	Não	Não	Não	Não	Não
Aplicação prática em empresas (estudo de caso)	Não	Não	Sim	Sim	Sim

Como pode-se visualizar no quadro 1, todos os trabalhos correlacionados fizeram um estudo sobre as metodologias *Scrum* e XP, sendo que alguns trabalhos foram aplicados em um ambiente real.

O estudo de caso desenvolvido apresenta um diferencial que é a implementação de práticas do *Kanban*, sendo que nos demais trabalhos essa técnica não foi abordada. Além do *Kanban,* foram aplicadas algumas técnicas do *Scrum,* já que este *framework* é muito utilizado (o que se verifica em todos os trabalhos estudados). Como a equipe de desenvolvimento da empresa onde foi aplicado o estudo de caso é pequena, já que possui apenas 4 colaboradores, não foi possível a aplicação do *Scrum* em sua totalidade.

4. Estudo de Caso Desenvolvido

Este trabalho envolveu a realização de um estudo de caso sobre a aplicação de metodologias ágeis para o desenvolvimento de *software* nas empresas de desenvolvimento de *software* sediadas em Palmeira das Missões - RS, propondo que as mesmas apliquem tais metodologias, visando agilizar a forma de trabalho, dentro de custos previsíveis e com qualidade.

A primeira parte deste trabalho envolveu o diagnóstico da forma de desenvolvimento de *software* utilizada pelas empresas, visando identificar as potencialidades e fragilidades dos processos atualmente aplicados. Para a realização deste diagnóstico foram realizadas entrevistas com os profissionais responsáveis pelo desenvolvimento de *software* nas empresas citadas, a partir do instrumento apresentado no Anexo A. No Anexo B apresenta-se o termo de consentimento livre e esclarecido, que foi entregue e assinado por todos os entrevistados.

4.1 Resultados obtidos com a aplicação do questionário

Esta seção apresenta os resultados obtidos com a aplicação do instrumento apresentado no Anexo A. Atualmente existem três empresas de desenvolvimento de *software* sediadas em Palmeira das Missões – RS. O levantamento das empresas existentes foi realizado com base nas informações disponibilizadas na Internet e, também, de forma empírica, pelo conhecimento dos autores que residem na cidade. Cabe destacar que apenas uma das empresas possui cadastro junto à ACAIP (Associação Comercial Agro Industrial e Serviços de Palmeira das Missões - RS):

- Empresa 1: Não utiliza uma metodologia de desenvolvimento de *software* padronizada. Utiliza a linguagem de programação *Delphi*, desenvolvendo *software* na área hospitalar, atuando desde 1988, contando com 14 colaboradores;
- Empresa 2: Utiliza a linguagem de programação *Pascal* (com IDE *Delphi)*. Sua finalidade é voltada para o desenvolvimento de Sistemas de Informação aplicados no comércio em geral, atuando desde 2013, contando com apenas 1 colaborador;
- Empresa 3: Não utiliza uma metodologia padronizada. Como as demais, utiliza a linguagem de programação *Delphi*. Desenvolve Sistemas de Informação aplicados no comércio em geral, atuando desde 2007, contando com 4 colaboradores.

4.1.1 Empresa 1

A empresa foi criada em 1988 e conta com 14 colaboradores. A empresa já desenvolveu muitos sistemas desde a sua criação (não informando o número exato) e atende 160 clientes. Além dos sistemas já desenvolvidos, conta com oito projetos em desenvolvimento.

A empresa desenvolve *softwares* na área de saúde (hospitais e clínicas), sistemas para supermercados e lojas e *pet shops*. As principais tecnologias utilizadas no desenvolvimento dos sistemas são: linguagens de programação *Delphi* 2007, e *Delphi* xe7, gerenciador de banco de dados *Firebird* e *Sqllite* e, para manipulação dos dados utiliza a ferramenta *Ibexpert*.

A empresa não utiliza uma metodologia formal para o desenvolvimento de *software* aplicando, como ferramenta de modelagem o *Power Designer*. Segundo o responsável pela equipe de desenvolvimento, a empresa utiliza controle de versão e documentação, apenas para sistemas maiores. A equipe de desenvolvimento não tem conhecimento sobre as metodologias ágeis, mas possui interesse em utilizá-las.

4.1.2 Empresa 2

Esta empresa foi criada em 2013 e conta com apenas um colaborador. Até o momento, foi desenvolvido apenas um sistema de ERP (*Enterprise Resource Planning*). A empresa atende mais de 50 clientes, e até o momento não possui nenhum projeto em desenvolvimento.

Os principais focos da empresa são automação comercial e comércio em geral. As principais tecnologias utilizadas no desenvolvimento dos sistemas são: Linguagem Pascal, IDE Delphi, SGBD (Sistema Gerenciador de Bancos de Dados) *Firebird* e Ferramenta de Modelagem *Power Designer*.

A empresa utiliza técnicas de *eXtreme Programming* para o desenvolvimento de *software*, apresentando os resultados, de tempos em tempos, para os clientes. A modelagem dos sistemas é realizada com o diagrama de MER (Modelo de Entidade e Relacionamento). Segundo o responsável pela empresa, a mesma conta com um sistema de controle de versão SVN (*Apache Subversion*) e os sistemas são documentados no sistema *HELP*. Além disso, a equipe de desenvolvimento conhece as metodologias ágeis *Scrum e eXtreme Programming.*

4.1.3 Empresa 3

A empresa foi criada em 2007 e conta com quatro colaboradores. Até o momento, já foram desenvolvidos cinco sistemas, atendendo aproximadamente 200 clientes. Além dos sistemas já desenvolvidos, conta com dois projetos em desenvolvimento.

Os principais focos da empresa envolvem gestão comercial e governo eletrônico. As principais tecnologias utilizadas no desenvolvimento dos sistemas são: linguagens de programação *Delphi* e *C#*, SGBDs *Postgres* e *Firebird*.

A empresa não utiliza uma metodologia formal para o desenvolvimento de *software* aplicando, como ferramenta de modelagem o diagrama ER (Entidade-Relacionamento). Segundo o responsável pela equipe de desenvolvimento, a empresa conta com um sistema de controle de versão e os sistemas são documentados em pares. A equipe de desenvolvimento conhece as metodologias ágeis *Scrum e Kanban,* mas não as utiliza no desenvolvimento. Entretanto, possui grande interesse em aplicar uma metodologia ágil no desenvolvimento de seus produtos.

4.2 Estudo de Caso

O estudo de caso iniciou pela definição da empresa na qual as técnicas ágeis de desenvolvimento de *software* seriam aplicadas. Esta escolha foi motivada pelo interesse em que a empresa demonstrou no aprendizado das metodologias ágeis, e o tempo que a mesma disponibilizou para os pesquisadores no momento da realização do diagnóstico inicial. Neste sentido, a empresa escolhida para a aplicação o estudo de caso, foi a empresa 3 (*Alliance Software*), pois a empresa mostrou interesse no estudo e aplicação das metodologias ágeis.

A partir dos resultados apresentados na seção 4.1, além do estudo de metodologias ágeis, foram definidas as técnicas ágeis a serem aplicadas: o quadro do *Kanban* e algumas técnicas do *Scrum*, tais como: divisão do trabalho em *Sprints*, reunião de planejamento do *Sprint* e reunião de revisão do *Sprint*, para serem implementadas nas melhorias do *software* da empresa *Alliance Software*. Definiu-se que essas seriam as técnicas a serem empregadas, pois a empresa não possui pessoal suficiente para a aplicação total do *Scrum*, nem para aplicação da metodologia *XP*. O proprietário da empresa optou pela aplicação de algumas técnicas do *Scrum* e do quadro *Kanban*. Esta definição baseou-se no fato de que o desenvolvedor da empresa já possuía

um pouco de conhecimento em *Scrum*, e o quadro do *Kanban* foi escolhido visando proporcionar uma melhoria na organização das tarefas da equipe.

A empresa em questão desenvolve Sistemas de Gestão Comercial, entre eles o *Agill Software de Gestão*, que tem como funcionalidades o controle de estoque, movimentação de produtos com vendas, *backup* automático, relatórios gerenciais, controle financeiro, emissão de boletos bancários, ordem de serviço, escrituração fiscal digital, além de interface com o sistema de governo eletrônico, para emissão de nota fiscal eletrônica "NF-e", nota fiscal de consumidor eletrônica e nota fiscal de serviço eletrônico. O referido sistema foi desenvolvido no ano de 2013 e está em constante atualização. A empresa não utilizava, até a realização deste estudo de caso, nenhum método de desenvolvimento, ou método de organização para a manutenção e/ou atualização dos sistemas.

O estudo de caso foi dividido em 6 passos, sendo eles: 1) definição da empresa onde seria realizado o estudo; 2) reunião com os gestores da empresa e com o grupo de desenvolvedores; 3) apresentação de metodologias e técnicas ágeis para o desenvolvimento de *software*; 4) escolha das técnicas ágeis a serem aplicadas no estudo de caso; 5) acompanhamento da aplicação das técnicas ágeis definidas e 6) discussão dos resultados.

A partir da definição da empresa, o segundo passo foi realizar uma reunião com os gestores da empresa e com o grupo de desenvolvedores, para discutir sobre o dia e horários disponíveis para apresentação, de forma sucinta, de alguns objetivos e técnicas envolvendo as metodologias ágeis para desenvolvimento de *software*. Esta reunião foi realizada no dia 09 de agosto de 2015, com duração de aproximadamente 40 minutos.

O terceiro passo envolveu a apresentação das metodologias e técnicas ágeis estudadas neste trabalho, apresentando os pontos positivos e limitações de cada metodologia. Esta apresentação foi realizada na forma de uma aula, abordando a metodologia *Scrum* e suas ferramentas incluindo, também, a apresentação do quadro *Kanban* e suas funcionalidades. Esta apresentação foi realizada no dia 22 de agosto de 2015 contando com a participação dos proprietários e de dois colaboradores. O conteúdo desta apresentação encontra-se no Anexo E.

Após a realização desta apresentação, verificou-se se que a metodologia *Scrum* não poderia ser aplicada em sua totalidade, pois a empresa não possui pessoal suficiente para a formação de uma equipe *Scrum*, já que a equipe de desenvolvimento conta com apenas 3 colaboradores e um time *Scrum* deveria ter, pelo menos, 5 integrantes. Neste sentido, definiu-se, juntamente com os gestores da empresa, que seriam aplicados o quadro do *Kanban* e a divisão do trabalho em *Sprints*, reunião de planejamento do *Sprint* e reunião de revisão do *Sprint* do *Scrum* no estudo de caso.

No quinto passo do estudo de caso, realizou-se o acompanhamento das atividades da empresa, com a aplicação das técnicas ágeis definidas. Nesta etapa já foi implementado o quadro *Kanban*, onde foi sugerida a organização das tarefas por etapas: *a fazer*, para tarefas a serem iniciadas; *fazendo*, para tarefas que estão em andamento; *feito*, para tarefas terminadas; *checado*, para tarefas que foram testadas e *problemas*, para tarefas que deverão ser refeitas ou corrigidas, como mostra a Figura 5. O quadro foi desenvolvido por um dos autores deste trabalho.

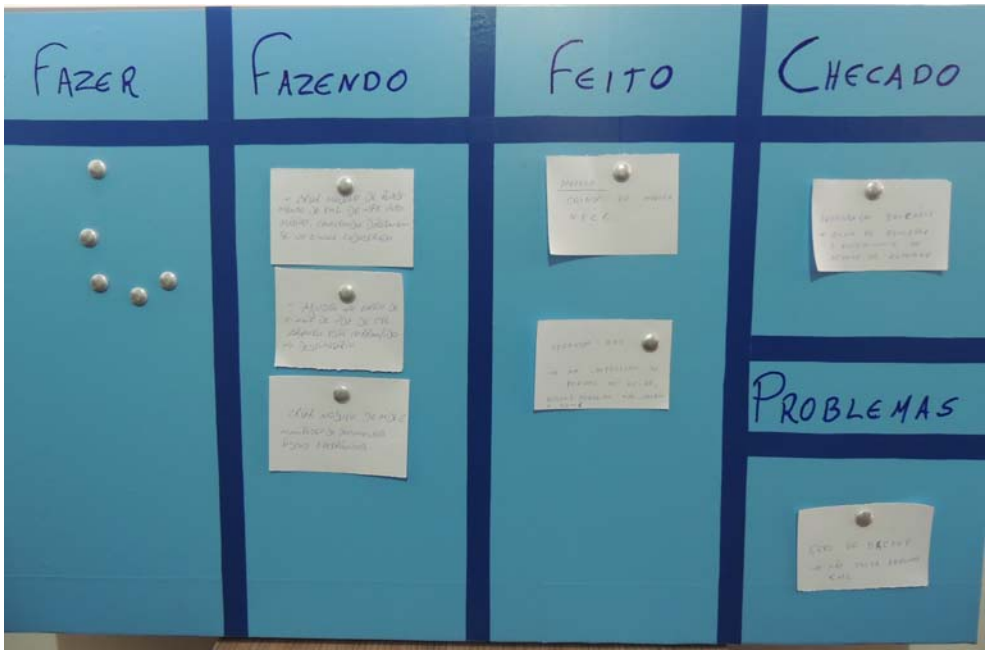


Figura 5: Quadro *Kanban* aplicado na empresa (Fonte: dos autores, 2015)

Após a implementação do quadro, também foram implementadas algumas práticas de *Scrum.* Estas práticas foram implementadas com o início do desenvolvimento de um novo *software* para Manifesto Eletrônico de Frete, que é um sistema baseado na nota fiscal eletrônica, para substituir a sistemática antiga de emissão de documentos em papel, com validade jurídica garantida pela assinatura digital do emitente, simplificando as obrigações dos contribuintes e permitindo, ao mesmo tempo, o acompanhamento em tempo real das operações comerciais pelo Fisco.

Neste sentido, foram realizadas algumas reuniões com a equipe e com os desenvolvedores, ocorrendo a divisão do trabalho em *Sprints* e reunião de planejamento dos *Sprints*, reuniões estas que aconteceram nos dias 31 de agosto e 5 de setembro, contando com a participação de 4 colaboradores da empresa. O projeto foi dividido em 3 *sprints* (desenvolver, revisar e ajustar). Entretanto, o projeto acabou parando no meio do caminho, devido a problemas de saúde enfrentados pelos proprietários da empresa, além de um acidente com um dos colaboradores da empresa, que acabou se machucando e ficando 90 dias de atestado médico. A empresa não possui um prazo de entrega deste novo sistema, pois possui um sistema adaptado em funcionamento para resolver este problema. A intenção é desenvolver um sistema novo, para no futuro evitar problemas com atualizações.

Na última etapa do estudo de caso – análise e discussão dos resultados – utilizou-se um questionário (Anexo C), aplicado com os participantes do estudo na empresa, além das observações realizadas durante o estudo.

Verificou-se que a empresa, por meio da aplicação do quadro do *Kanban*, obteve uma organização e praticidade maior com relação aos hábitos anteriores, já que a mesma não possuía nenhum tipo de organização. Os desenvolvedores constataram que as atividades de melhorias do sistema, nas quais o quadro *Kanban*, foi utilizado (tais como: Integração Bancária, Módulo de Boletos, Módulo da Nota Fiscal Consumidor

Eletrônica, Módulo de Manifesto de Documentos Ficais Eletrônicos e Módulo Automático de Recebimento de XML da Nota Fiscal Eletrônica) ficaram mais práticas de serem implementadas, além de possibilitar a correção de *bugs*. Na Figura 6, um dos autores deste trabalho durante a apresentação do quadro *Kanban* aos colaboradores da empresa.

Figura 6: Apresentação da evolução das atividades com o Quadro *Kanban*

(Fonte: dos autores, 2015)

Por meio do questionário de validação do estudo de caso (Anexo C), pôde-se constatar que o estudo de caso alcançou pontos positivos, tendo-se em vistas as respostas satisfatórias. Com relação à pergunta "Você considera que a adoção de técnicas ágeis no desenvolvimento de *software* trouxe benefícios para a empresa?", o proprietário da empresa respondeu: "sim, benefícios como um controle maior sobre as tarefas, sobre o ciclo de liberação de nova versão, permitindo que esse ciclo seja cumprido nas datas estabelecidas".

Analisando a resposta da segunda questão, "Com relação à apresentação sobre as metodologias ágeis (aula ministrada pelos autores deste trabalho) você considera que ela foi esclarecedora e apresentou de forma adequada a temática?", o proprietário da empresa acredita que apresentação foi adequada, pois "...o aluno mostrou um profundo conhecimento sobre as técnicas ágeis apresentadas e propostas, permitindo assim um fácil entendimento sobre as mesmas".

Com relação à questão "Dentre as técnicas ágeis escolhidas para aplicação na sua empresa, qual(is) você considera mais positiva(s)?", a resposta foi o "Quadro *Kanban,* o quadro *Kanban* nos possibilitará um acompanhamento visual das tarefas e

projetos em andamento, permitindo que eventuais problemas possam ser corrigidos em tempo hábil para não prejudicar o ciclo da versão".

Além das perguntas, no final do questionário, havia um espaço para sugestões e/ou comentários sobre o estudo de caso desenvolvido. O proprietário da empresa destacou: "Como sugestão para trabalhos futuros, coloco minha empresa à disposição para a conclusão deste trabalho". Neste sentido, destaca-se o acompanhamento do sistema que não foi concluído, tendo-se em vista os problemas destacados anteriormente.

A partir de reuniões realizadas com os gestores da empresa, bem como com os resultados obtidos por meio do instrumento de validação, acredita-se que os resultados foram satisfatórios, apesar de um dos projetos estar parado. A organização do quadro *Kanban*, e a divisão de tarefas do *Scrum*, mostraram-se como ferramentas adequadas para a organização e agilidade no desenvolvimento de *software*. A empresa pretende continuar aplicando as técnicas do *Scrum*, a partir do próximo ano.

5. Considerações Finais

Este trabalho foi dividido em duas partes. Na primeira parte foram desenvolvidos o referencial teórico, o estado da arte e o levantamento sobre as empresas de desenvolvimento de *software* de Palmeira das Missões – RS. Já na segunda parte foi definida a empresa onde o estudo de caso foi realizado (*Alliance Software*), apresentadas as metodologias ágeis e definidas algumas técnicas, as quais foram aplicadas na prática. O Anexo D apresenta o termo de autorização da empresa para o desenvolvimento deste estudo de caso.

Acredita-se que os objetivos foram alcançados, mesmo existindo algumas dificuldades. Entre elas, destacam-se os problemas de saúde com um familiar dos proprietários da empresa, e o afastamento de um colaborador devido a um acidente. Estes incidentes não permitiram que o estudo de caso tivesse seu escopo completado. Mesmo assim, foi possível aplicar o quadro do *Kanban* e iniciar as atividades voltadas à aplicação da metodologia *Scrum*.

A escolha do quadro *Kanban* foi definida visando obter uma organização das tarefas da equipe, pois a mesma não possuía nenhum método para organizá-las. Já a aplicação de algumas técnicas do *Scrum* foi definido devido ao desenvolvedor da empresa já possuir um pouco de conhecimento referente a essa metodologia, não sendo possível fazer a sua aplicação total, já que a empresa não possui pessoal suficiente para a formação de uma equipe *Scrum* (a equipe de desenvolvimento conta com apenas 3 colaboradores, e um time *Scrum* deveria ter, pelo menos, 5 integrantes).

Uma das possibilidades deste trabalho poderia envolver a definição de um guia ou roteiro para aplicação de técnicas ágeis para desenvolvimento de *software* em empresas de pequeno porte. Entretanto, como optou-se pela metodologia de estudo de caso, o mesmo só se faz possível mediante a aplicação prática em um ambiente real ou controlado. Neste sentido, mesmo com os problemas envolvendo as atividades da empresa, acredita-se que os resultados foram satisfatórios, pois a empresa *Alliance Software* mostrou-se interessada na aplicação das metodologias ágeis, o que a diferencia neste mercado na cidade de Palmeira das Missões – RS.

Como trabalhos futuros, pretende-se continuar com a aplicação de técnicas ágeis, em especial as integrantes da metodologia *Scrum*, para dar continuidade ao desenvolvimento do *software* de Manifesto Eletrônico de Frete.

Referências

AGILE MANIFESTO (2015). **Manifesto for Agile Software Development.** Disponível em:<http://agilemanifesto.org>. Acessado em 03 de abril de 2015.

ARRUDA, L. V. (2012) Desenvolvimento Ágil de Software: uma análise sintética a partir da metodologia Kanban. Anais do **VII CONNEPI – Congresso Norte Nordeste de Pesquisa e Inovação**. Palmas, Tocantins. Disponível em: <http://propi.ifto.edu.br/ocs/index.php/connepi/vii/paper/viewFile/3644/961>. Acesso em 19 de abril de 2015.

BECK, K. (2008) **Programação Extrema (XP) Explicada**. Porto Alegre: Bookman.

COHN, M. (2011) **Desenvolvimento de Software com Scrum**. Porto Alegre: Bookman.

GYGI, C.; DECARLO, N.; WILLIAMS, B. (2008) **Seis Sigma para Leigos**. Rio de Janeiro: Alta Books.

LEAN.ORG (2015). ***Lean thinking:*** O que é? Disponível em:<http://www.lean.org.br>. Acessado em 15 de abril de 2015.

OLIVEIRA, E. S. (2003) **Uso de Metodologias Ágeis no Desenvolvimento de Software.** Universidade Federal de Minas Gerais, 2003. Disponível em: <http://www.cpdee.ufmg.br/~renato/TesesEDissertacoesOrientadas/Monografia-EbenezerSilvaOliveira.pdf>. Acesso em: maio de 2015.

PHAM, A.; PHAM, P. (2012) **Scrum em Ação**. São Paulo: Novatec.

PRESSMAN, R. S. (2011) **Engenharia de Software**. 7. ed. Porto Alegre: Pearson Makron Books.

PRIKLADNICKI, R.; WILLI, R.; MILANI, F. (2014) **Métodos Ágeis Para o Desenvolvimento de Software**. Porto Alegre: Bookman.

SCRUM.ORG. (2015) **What is Scrum?** Disponível em:<https://www.scrum.org>. Acessado em 11 de abril de 2015.

SCRUMGUIDES.ORG. (2015) **What is Scrum?** Disponível em:<http://www.scrumguides.org>. Acessado em 11 de abril de 2015.

SGANDERLA, M. A.; LACERDA, G. (2010) **Melhorando a Gerência e a Construção de Software com Metodologias Ágeis.** Trabalho de Conclusão de Curso – Bacharelado em Sistemas de Informação, Porto Alegre: UniRitter. Disponível em: **<**https://www.uniritter.edu.br/graduacao/informatica/sistemas/downloads/tcc2k10/mauricio_2010_2.pdf>. Acesso em 25 de maio de 2015.

SOARES, M. S. (2004) **Metodologias Ágeis Extreme Programming e Scrum para o Desenvolvimento de Software.** Universidade Presidente Antônio Carlos. Disponível em: <www.spell.org.br/documentos/download/26416>. Acesso em: maio de 2015.

SOMMERVILLE, I. (2007) **Engenharia de Software.** 8. ed. São Paulo: Pearson Addison Wesley.

TRECCANI, P. J. F.; SOUZA, C. R. B. (2010) **Utilização de Metodologias Ágeis no Desenvolvimento de Software: Resultados de um Estudo Empírico.** Universidade Federal do Pará. Disponível em: <http://www.lbd.dcc.ufmg.br/colecoes/eselaw/2010/005.pdf>. Acesso em: maio de 2015.

Anexo A

Roteiro de entrevistas: Gestores das Empresas de Desenvolvimento de Software de Palmeira das Missões – RS

1. Qual o ano de criação da empresa?
2. Quantos colaboradores existem atualmente?
3. Quantos sistemas/projetos já foram desenvolvidos?
4. Quantos clientes/empresas são atendidos?
5. Quantos sistemas/projetos estão em desenvolvimento?
6. Qual é o principal foco da empresa (em que área são desenvolvidos os sistemas)?
7. Quais são as principais tecnologias utilizadas no desenvolvimento dos sistemas (linguagens de programação, sistemas gerenciadores de bancos de dados, etc)?
8. A empresa adota alguma metodologia formal para o desenvolvimento de software?
9. Quais ferramentas e/ou técnicas são utilizadas para realizar a modelagem do software (diagrama E-R, linguagem UML, etc)?
10. Os sistemas desenvolvidos possuem controle de versão?
11. Os sistemas desenvolvidos são documentados?
12. A equipe de desenvolvimento conhece alguma metodologia ágil? Em caso de afirmativo qual(is)?
13. A equipe de desenvolvimento aplica alguma metodologia ágil? Em caso de afirmativo qual(is)?
14. A empresa e/ou equipe de desenvolvimento possui interesse em aplicar metodologias ágeis no processo de desenvolvimento de *software*?

(Fonte: dos autores, 2015)

Anexo B

TERMO DE CONSENTIMENTO LIVRE E ESCLARECIDO

Convidamos o(a) Sr.(a) para participar do projeto **"Aplicação de metodologias ágeis para desenvolvimento de software: um estudo de caso nas empresas de desenvolvimento de software de Palmeira das Missões – RS"**, sob a responsabilidade do acadêmico Juliano da Conceição, do curso de Bacharelado em Sistemas de Informação da UFSM/CESNORS, sob orientação do Prof. Dr. Sidnei Renato Silveira.

Sua participação é voluntária e se dará por meio do preenchimento de um questionário semiestruturado, aplicado pelo acadêmico. Se você aceitar participar, estará contribuindo para a aplicação de metodologias ágeis para desenvolvimento de software: um estudo de caso nas empresas de desenvolvimento de software de Palmeira das Missões – RS.

Se depois de consentir em sua participação o Sr.(a) desistir de continuar participando, tem o direito e a liberdade de retirar seu consentimento em qualquer fase da pesquisa, seja antes ou depois da coleta de dados, independente do motivo e sem nenhum prejuízo a sua pessoa. O (a) Sr. (a) não terá nenhuma despesa e também não receberá nenhuma remuneração. Os resultados da pesquisa serão analisados e publicados, mas sua identidade não será divulgada sendo guardada em sigilo. Para qualquer outra informação, o (a) Sr. (a) poderá entrar em contato com o pesquisador pelo telefone (55) 9643-5460, ou poderá entrar em contato com a UFSM – Universidade Federal de Santa Maria – CESNORS – Centro de Educação Superior Norte RS, procurando a Coordenação do Curso de Bacharelado em Sistemas de Informação, na Linha Sete de Setembro, s/n, Sala 80 (Bloco 6) ou pelo telefone (55) 3744-8964 Ramal 8790.

Consentimento Pós-Informação

Eu, Diego Solano Potrich,
fui informado sobre o que o pesquisador quer fazer e porque precisa da minha colaboração, e entendi a explicação. Por isso, eu concordo em participar do projeto, sabendo que não vou ganhar nada e que posso sair quando quiser. Este documento é emitido em duas vias que serão ambas assinadas por mim e pelo pesquisados, ficando uma via com cada um de nós.

Diego Solano Potrich Data: 02/07/15

Assinatura do participante

Assinatura do Pesquisador Responsável

Anexo B

TERMO DE CONSENTIMENTO LIVRE E ESCLARECIDO

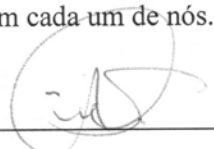
Convidamos o(a) Sr.(a) para participar do projeto **"Aplicação de metodologias ágeis para desenvolvimento de software: um estudo de caso nas empresas de desenvolvimento de software de Palmeira das Missões – RS"**, sob a responsabilidade do acadêmico Juliano da Conceição, do curso de Bacharelado em Sistemas de Informação da UFSM/CESNORS, sob orientação do Prof. Dr. Sidnei Renato Silveira.

Sua participação é voluntária e se dará por meio do preenchimento de um questionário semiestruturado, aplicado pelo acadêmico. Se você aceitar participar, estará contribuindo para a aplicação de metodologias ágeis para desenvolvimento de software: um estudo de caso nas empresas de desenvolvimento de software de Palmeira das Missões – RS.

Se depois de consentir em sua participação o Sr.(a) desistir de continuar participando, tem o direito e a liberdade de retirar seu consentimento em qualquer fase da pesquisa, seja antes ou depois da coleta de dados, independente do motivo e sem nenhum prejuízo a sua pessoa. O (a) Sr. (a) não terá nenhuma despesa e também não receberá nenhuma remuneração. Os resultados da pesquisa serão analisados e publicados, mas sua identidade não será divulgada sendo guardada em sigilo. Para qualquer outra informação, o (a) Sr. (a) poderá entrar em contato com o pesquisador pelo telefone (55) 9643-5460, ou poderá entrar em contato com a UFSM – Universidade Federal de Santa Maria – CESNORS – Centro de Educação Superior Norte RS, procurando a Coordenação do Curso de Bacharelado em Sistemas de Informação, na Linha Sete de Setembro, s/n, Sala 80 (Bloco 6) ou pelo telefone (55) 3744-8964 Ramal 8790.

Consentimento Pós-Informação

Eu, Claudine V. Simons,
fui informado sobre o que o pesquisador quer fazer e porque precisa da minha colaboração, e entendi a explicação. Por isso, eu concordo em participar do projeto, sabendo que não vou ganhar nada e que posso sair quando quiser. Este documento é emitido em duas vias que serão ambas assinadas por mim e pelo pesquisados, ficando uma via com cada um de nós.



Data: 01/07/2015

Assinatura do participante

Assinatura do Pesquisador Responsável

Anexo B

TERMO DE CONSENTIMENTO LIVRE E ESCLARECIDO

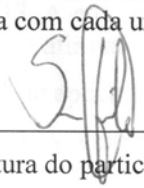
Convidamos o(a) Sr.(a) para participar do projeto **"Aplicação de metodologias ágeis para desenvolvimento de software: um estudo de caso nas empresas de desenvolvimento de software de Palmeira das Missões – RS"**, sob a responsabilidade do acadêmico Juliano da Conceição, do curso de Bacharelado em Sistemas de Informação da UFSM/CESNORS, sob orientação do Prof. Dr. Sidnei Renato Silveira.

Sua participação é voluntária e se dará por meio do preenchimento de um questionário semiestruturado, aplicado pelo acadêmico. Se você aceitar participar, estará contribuindo para a aplicação de metodologias ágeis para desenvolvimento de software: um estudo de caso nas empresas de desenvolvimento de software de Palmeira das Missões – RS.

Se depois de consentir em sua participação o Sr.(a) desistir de continuar participando, tem o direito e a liberdade de retirar seu consentimento em qualquer fase da pesquisa, seja antes ou depois da coleta de dados, independente do motivo e sem nenhum prejuízo a sua pessoa. O (a) Sr. (a) não terá nenhuma despesa e também não receberá nenhuma remuneração. Os resultados da pesquisa serão analisados e publicados, mas sua identidade não será divulgada sendo guardada em sigilo. Para qualquer outra informação, o (a) Sr. (a) poderá entrar em contato com o pesquisador pelo telefone (55) 9643-5460, ou poderá entrar em contato com a UFSM – Universidade Federal de Santa Maria – CESNORS – Centro de Educação Superior Norte RS, procurando a Coordenação do Curso de Bacharelado em Sistemas de Informação, na Linha Sete de Setembro, s/n, Sala 80 (Bloco 6) ou pelo telefone (55) 3744-8964 Ramal 8790.

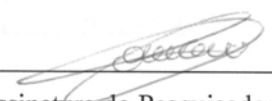
Consentimento Pós-Informação

Eu, SERGINHO OLI, fui informado sobre o que o pesquisador quer fazer e porque precisa da minha colaboração, e entendi a explicação. Por isso, eu concordo em participar do projeto, sabendo que não vou ganhar nada e que posso sair quando quiser. Este documento é emitido em duas vias que serão ambas assinadas por mim e pelo pesquisados, ficando uma via com cada um de nós.



Data: 03/06/2015

Assinatura do participante



Assinatura do Pesquisador Responsável

Anexo C

Questionário de Validação do Estudo de Caso

1.Você considera que a adoção de técnicas ágeis no desenvolvimento de software trouxe benefícios para a empresa?

( ) Sim

( ) Não

Justifique sua resposta:

2. Com relação à apresentação sobre as metodologias ágeis (aula ministrada pelos autores deste trabalho) você considera que ela foi esclarecedora e apresentou de forma adequada a temática?

( ) Sim

( ) Não

Justifique sua resposta:

3.Dentre as técnicas ágeis escolhidas para aplicação na sua empresa, qual(is) você considera mais positiva(s)?

Justifique sua resposta:

4. Apresente sugestões e/ou comentários sobre a realização do estudo de caso e/ou sobre a aplicação de metodologias ágeis no desenvolvimento de *software*

(Fonte: dos autores, 2015)

Anexo D

AUTORIZAÇÃO

A empresa OTT- CONSULTORIA E SISTEMAS DE INFORMATICA LTDA. – ME, com nome fantasia de Alliance Softwares, inscrita no cadastro nacional de pessoas jurídicas sob o CNPJ nº 08.674.787/0001-89, autoriza o acadêmico Juliano da Conceição a desenvolver o seu Trabalho de Graduação em Sistemas de Informação intitulado "**Aplicação de metodologias ágeis para desenvolvimento de software: um estudo de caso nas empresas de desenvolvimento de software de Palmeira das Missões – RS**", aplicando-o em nossa empresa.

Frederico Westphalen-RS, 30 de Novembro de 2015

OTT CONSULTORIA E SISTEMAS
DE INFORMÁTICA LTDA.
CNPJ: 08.674.787/0001-89

Serginho Ott
Sócio Proprietário

Anexo E

APRESENTAÇÃO DO CONTEÚDO DO *SCRUM*

SIN 1022

Gerência de Projetos de Software
Aulas 11 e 12

Como aplicar o SCRUM na prática

Prof. Dr. Sidnei Renato Silveira
sidneirenato.silveira@gmail.com

SCRUM na Prática

SCRUM não é uma metodologia, é um *framework* (ou seja, não existe uma única maneira correta de aplicar o SCRUM)

O melhor e o pior do SCRUM é justamente poder adaptar o processo para situações específicas

O **product backlog** é um dos artefatos mais importantes do SCRUM: é uma lista de requisitos, coisas que o cliente deseja, descritas utilizando a **terminologia do cliente**

SCRUM na Prática

Campos de uma estória

ID: uma identificação única, para evitar a perda do controle sobre as estórias, quando seus nomes são alterados

Nome: um nome curto e descritivo para a estória, suficientemente claro para que os desenvolvedores e o PO entendam sobre o que estamos falando e claro o bastante para distingui-la de outras estórias

Importância: pontuação de importância dessa estória para o PO; mais pontos = mais importante

SCRUM na Prática

Campos de uma estória

Estimativa Inicial: as estimativas iniciais da equipe sobre quanto tempo é necessário para implementar aquela história, se comparada a outras estórias

A unidade utilizada é a de **pontos por estória** e, geralmente, corresponde mais ou menos à relação "homem-dias" ideal

SCRUM na Prática

Campos de uma estória

Por exemplo, se 2 pessoas trabalharem em uma estória, sem distúrbios, em quantos dias apresentarão uma implementação pronta, demonstrável e testada?

Exemplo: com 2 pessoas são necessários 6 dias

6 dias = estimativa inicial de homens-dia 2-6 = 12 pontos por estória

SCRUM na Prática

Campos de uma estória

Como Demonstrar: uma descrição em alto nível de como a estória será demonstrada na apresentação do sprint; uma especificação de teste: "Faça isso, então faça aquilo e então isso deverá acontecer"

Notas: quaisquer outras informações, esclarecimentos, referências a outras fontes de informação, etc.

SCRUM na Prática

ID	Nome	Importância	Estimativa	Como demonstrar	Notas
1	Depósito	30	5	Logar-se, abrir a página de depósito, depositar R$100,00, ir para a página do meu saldo e verificar que este aumentou em RS100,00	Precisa de um diagrama UML de sequência; não é preciso 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 o histórico de transações, verificar se o novo depósito é mostrado	Usar paginação para evitar consultas muito grandes ao BD (projetar de forma similar à página de visualização de relatórios)

SCRUM na Prática

O product backlog pode ser feito em uma planilha eletrônica compartilhada; o PO é o "dono" desta planilha

Campos adicionais de uma estória

Track (trilha/rastro): categorização básica dessa estória, por exemplo: "otimização"

Componentes: base de dados, servidor, cliente (a equipe ou o PO podem identificar quais componentes técnicos estão envolvidos na implementação dessa estória)

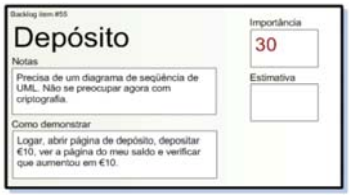
SCRUM na Prática

Campos adicionais de uma estória

Solicitante: o PO pode querer manter o rastro de qual cliente ou o *stakeholder* que solicitou o item, para poder fornecer um *feedback* sobre o progresso desse item

ID do Bug: se houver um sistema separado para registro de erros (*bug tracking*) é útil manter a rastreabilidade da estória com um ou mais erros reportados sobre ela

SCRUM na Prática



SCRUM na Prática

O PO, se tiver uma formação técnica, pode adicionar estórias com jargão técnico de TI; entretanto, ele deve focar-se nos **objetivos de negócio**, deixando que a equipe descubra e decida como resolver as questões técnicas

A descrição técnica deve ser uma **nota**

O product backlog deve estar organizado e fechado **antes** da reunião de planejamento do sprint

SCRUM na Prática

Outras pessoas, além do PO, podem adicionar estórias ao product backlog, mas elas não podem associar uma escala de **importância** (esse é um papel exclusivo do PO)

Outras pessoas, de fora da equipe, também não podem estimar **prazos**

SCRUM na Prática

Planejamento do Sprint

O propósito da reunião de planejamento do sprint é dar à equipe informação suficiente para trabalharem em paz por algumas semanas e para dar ao PO confiança suficiente para deixá-los fazerem isto

Toda a equipe e o PO **devem** estar na reunião de planejamento do sprint, pois as estórias possuem 3 variáveis interdependentes: **escopo, estimativa** e **importância**

SCRUM na Prática

Planejamento do Sprint

Escopo e **importância** são definidos pelo PO

A **estimativa** é definida pela equipe

Durante uma reunião de planejamento do sprint, estas 3 variáveis são refinadas continuamente, por meio do diálogo entre a equipe e o PO

Normalmente, o PO inicia a reunião resumindo seu objetivo para o sprint e as estórias mais importantes

SCRUM na Prática

Planejamento do Sprint

Em seguida, a equipe estima o tempo de cada estória, começando pela mais importante

Tudo no SCRUM é delimitado pelo tempo

Uma reunião de planejamento do sprint deve ter em torno de **4 horas**, com intervalos de 10 minutos a cada hora

SCRUM na Prática

Reunião de Planejamento do Sprint

1) O PO repassa o **objetivo do sprint** e sumariza o product backlog; deve-se definir lugar, data e hora da demonstração dos resultados do sprint

2) A equipe **estima o tempo** e quebra os itens conforme necessário. O PO atualiza as taxas de **importância** conforme necessário. Os itens são esclarecidos. **"Como demonstrar"** é preenchido para todos os itens de maior importância

SCRUM na Prática

Reunião de Planejamento do Sprint

3) A equipe **escolhe as estórias** a serem incluídas no sprint e calcula a velocidade para verificar se o planejamento ficou realista; deve-se definir o horário e lugar para as **reuniões diárias** e complementar a **quebra de estórias em tarefas**

SCRUM na Prática

Sprints Curtos

Permitem que a equipe seja ágil (mude de direção frequentemente)

Entregas mais frequentes

Feedback mais frequente do cliente

SCRUM na Prática

Sprints Longos

A equipe tem mais tempo para ganhar ritmo

A equipe tem mais tempo para se recuperar dos problemas

Menos *overhead* em termos de reuniões de planejamento, apresentações, etc.

SCRUM na Prática

Reunião de Planejamento do Sprint

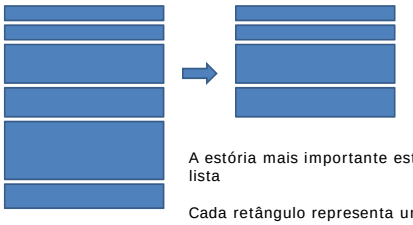
O objetivo do sprint deve ser descrito em termos de **negócio**, não em termos técnicos (termos em que as pessoas de fora da equipe possam entender)

O objetivo do sprint deve responder à pergunta: "Por que nós estamos fazendo este sprint?"

Uma das principais atividades da reunião de planejamento do sprint é **decidir quais estórias incluir no sprint** (do product backlog para o sprint backlog)

SCRUM na Prática

Product Backlog *Sprint Backlog*



A estória mais importante está no topo da lista

Cada retângulo representa uma estória; o tamanho representa o tamanho em pontos de estória

SCRUM na Prática

Reunião de Planejamento do Sprint

O PO pode não ficar satisfeito com as estórias que foram incluídas no sprint: pode mudar a importância; pode dividir a estória

Como a equipe decide que estórias incluir no sprint?

1) No sentimento/instinto: funciona bem como pequenas equipes e sprints curtos

2) Estimativas usando cálculos de velocidade

SCRUM na Prática

Estimativas usando cálculo de velocidades

1) Definir a velocidade estimada

2) Calcular quantas estórias podem ser adicionadas sem exceder a velocidade estimada

Velocidade: medida da quantidade de trabalho realizado

Técnicas: "tempo de ontem", "fator de foco", "*planning poker*"

SCRUM na Prática

Estimativas usando cálculo de velocidades

"Tempo de ontem": olhar para o histórico da equipe: qual foi a velocidade nos sprints mais recentes?

Exemplo: sprint de 3 semanas (15 dias)

Pessoa 1: 15 dias disponíveis
Pessoa 2: 13 dias disponíveis
Pessoa 3: 15 dias disponíveis
Pessoa 4: 7 dias disponíveis

50 homens-dia disponíveis

SCRUM na Prática

Estimativas usando cálculo de velocidades

"Fator de foco": estimativa de como a equipe é focada; um fator de foco baixo significa que a equipe espera ter muitas interferências ou percebe que suas próprias estimativas de tempo são muito otimistas

A melhor maneira para determinar um fator de foco razoável é considerar o último sprint (ou melhor ainda, a média de alguns sprints anteriores)

SCRUM na Prática

Estimativas usando cálculo de velocidades

Fator de foco = velocidade real / homens-dia disponíveis

A velocidade real é a soma das estimativas iniciais de todas as estórias que foram finalizadas no último sprint

Exemplo: último sprint com 3 pessoas em 15 dias:

18 pontos por estória/ 45 homens-dia = 0.4 (40%)

Velocidade estimada:

50 homens-dia X 40% = 20 pontos por estória

SCRUM na Prática

Estimativas usando cálculo de velocidades

Cada **estória deve ser quebrada em tarefas**, para que as estimativas sejam mais fáceis e mais precisas

O PO não precisa estar envolvido na quebra das estórias em tarefas

Estórias são trabalhos que podem ser entregues e que o PO tem que se preocupar; **tarefas** são atividades que não podem ser entregues, ou atividades com as quais o PO não precisa se preocupar

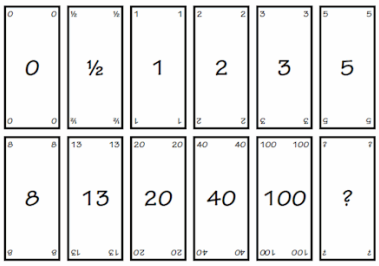
SCRUM na Prática

Estimativas usando cálculo de velocidades

"*Planning Poker*": cada membro da equipe recebe um baralho de 13 cartas; sempre que uma estória deve ser estimada, cada membro da equipe escolhe uma carta que representa a sua estimativa de tempo (em pontos por estória) e coloca-a virada para baixo sobre a mesa

Quando todos os membros da equipe tiverem feito sua estimativa, as cartas são reveladas; dessa forma, cada membro da equipe é forçado a pensar por si próprio, ao invés de basear-se na estimativa de outra pessoa

SCRUM na Prática



SCRUM na Prática

Estimativas usando cálculo de velocidades

Se houver uma grande divergência entre as estimativas, a equipe discute as diferenças e tenta chegar a uma visão comum do trabalho envolvido na estória e faz **novamente a estimativa**

Os membros da equipe devem estimar **a quantidade total de esforço envolvido na estória**, não apenas a sua parte do trabalho

Cartas especiais: 0 (esta estória já está feita ou é tão pequena que leva somente alguns minutos de trabalho), ? (eu não faço nenhuma ideia) e xícara de café (fazer uma pequena pausa)

SCRUM na Prática

Formato mais produtivo para o sprint backlog: um quadro de tarefas pendurado na parede

O quadro com cartões (estórias) e post-its (tarefas) é melhor do que usar uma planilha eletrônica

As pessoas ficam em pé e caminham (ficam alertas por mais tempo)

A repriorização é mais fácil (trocar a posição dos cartões)

SCRUM na Prática

SCRUM na Prática

Algumas Sugestões

Manter a equipe junta

Manter o PO por perto

Reuniões diárias: atualizar o quadro de tarefas

Bibliografia Recomendada

KNIBERG, Henrik. Scrum e XP Direto das Trincheiras: como nós fazemos Scrum. INFOQ, 2007. Série Desenvolvimento de Software Corporativo.

PHAM, Andrew; PHAM, Phuong-Van. Scrum em Ação: gerenciamento e desenvolvimento ágil de projetos de software. São Paulo: Novatec-Cengage Learning, 2011.

SIN 1022

Gerência de Projetos de Software
Aulas 11 e 12

SCRUM

Prof. Dr. Sidnei Renato Silveira
sidneirenato.silveira@gmail.com

Metodologias Ágeis

Ágil é ser rápido, ligeiro. **Eficaz** produz o resultado esperado e **eficiente** produz o resultado esperado, *com qualidade*

As metodologias ágeis têm como características importantes:

- foco nas necessidades do cliente (resultado)
- liderança
- envolvimento das pessoas
- melhoria contínua
- tomada de decisões baseada em análise de dados e informações (controle)

Direitos do Cliente
Segundo Ron Jeffries

Planejamento geral, definindo o que pode ser realizado, quando e a que custo

Acompanhar o andamento do projeto e, principalmente, o progresso do software, passando por testes definidos em conjunto com a equipe

Mudar de ideia, substituir funcionalidades, sem pagar custos exorbitantes

Ser informado sobre mudanças no cronograma, em tempo de escolher e reduzir o escopo

Poder cancelar o projeto a qualquer momento e ainda assim ter um sistema funcionando, refletindo o investimento realizado até o momento

Direitos do Desenvolvedor
Segundo Ron Jeffries

Saber o que é necessário, com declarações claras de prioridade

Produzir trabalho de qualidade, o tempo todo

Pedir e receber ajuda da equipe, superiores e clientes

Fazer e atualizar suas próprias estimativas

Aceitar as suas responsabilidades, ao invés de tê-las impostas

Metodologias Ágeis

Fundamentos do gerenciamento de projetos e desenvolvimento ágil de software:

- Manifesto Ágil (4 valores e 12 princípios) – mais voltado ao desenvolvimento de software www.agilemanifesto.org

- Declaração de interdependência – focaliza mais o gerenciamento de projetos

Metodologias Ágeis

O manifesto ágil estabelece que devemos valorizar:

1) indivíduos e interações ao invés de processos e ferramentas

2) software funcional ao invés de documentação extensiva

3) colaboração com o cliente ao invés de negociação de contrato

4) respostas à mudança ao invés de seguimento de um plano

Metodologias Ágeis

Princípios do manifesto ágil (1/4):

1) nossa prioridade máxima é satisfazer o cliente por meio da entrega antecipada e contínua de software com valor

2) abraçar as mudanças de requisitos, mesmo quando o desenvolvimento já está avançado; os processos ágeis usam as mudanças para trazer vantagens competitivas ao cliente

3) Entregar software funcional frequentemente, indo de algumas semanas a alguns meses, com preferência para a escala de tempo mais curta

Metodologias Ágeis

Princípios do manifesto ágil (2/4):

4) pessoas ligadas ao negócio e ao desenvolvimento devem trabalhar juntas diariamente, ao longo do projeto

5) construir projetos em torno de indivíduos motivados; dê a eles o ambiente e suporte necessários e confie que eles realizarão o trabalho

6) o método mais eficiente e efetivo de transmitir informações para e dentro de uma equipe de desenvolvimento é conversando cara a cara

Metodologias Ágeis

Princípios do manifesto ágil (3/4):

7) software funcional é a primeira medida de progresso

8) processos ágeis promovem desenvolvimento sustentável; os patrocinadores, desenvolvedores e usuários devem ser capazes de manter um ritmo constante indefinidamente

9) atenção contínua à excelência técnica e um bom projeto melhoram a agilidade

Metodologias Ágeis

(1/2)
Declaração de Interdependência (DOI) da Gestão de Projetos Ágeis (http://pmdoi.org) estabelece que:

Somos uma comunidade de líderes de projeto que tem sido altamente bem-sucedida em entregar resultados. Para alcançar tais resultados:

- aumentamos o retorno do investimento

- entregamos resultados confiáveis, engajando os clientes em interações frequentes e propriedade compartilhada

- esperamos incertezas e gerenciamos levando-as em conta, por meio de iterações, antecipação e adaptação

Metodologias Ágeis

(2/2)
Declaração de Interdependência (DOI) da Gestão de Projetos Ágeis (http://pmdoi.org) estabelece que:

- promovemos criatividade e inovação, reconhecendo que os indivíduos são a fonte última de valor e criamos um ambiente em que eles fazem a diferença

- impulsionamos o desempenho por meio do compromisso do grupo em obter resultados e da responsabilidade compartilhada pela eficácia do grupo

- melhoramos a eficácia e a confiabilidade por meio de estratégias, processos e práticas

SCRUM

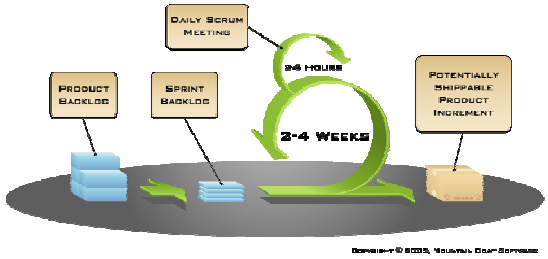
Desenvolver software é uma atividade **criativa;** não é um bom candidato a processos pré-definidos

Modelo de controle de processo **empírico**; o desenvolvimento **nem sempre será repetitivo**

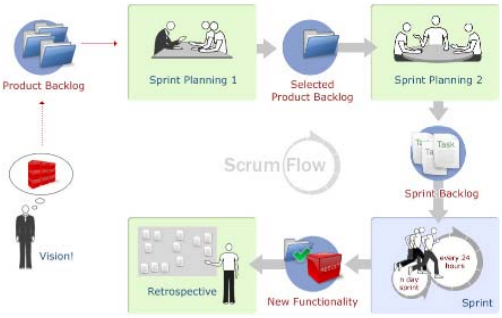
Processos definidos: ambiente controlado

Processos empíricos: processos complexos e imprevisíveis; visibilidade, inspeção e adaptabilidade; comando/controle => liderança/colaboração

SCRUM



SCRUM



Origem do Scrum

Equipes de projeto são compostas por pequenas equipes multifuncionais, trabalhando com sucesso rumo a um objetivo comum (os autores Takeuchi e Nonaka compararam estas equipes à formação *scrum* do *rugby*)

O principal objetivo do *scrum* é, ao final de iterações curtas, ver a demonstração de código funcional ao invés de gráficos de Gantt

Para isto são necessárias a inspeção e adaptação constantes

O *Scrum* pode ser adaptado para situações específicas

Equipe Scrum

- *Scrum Master*

- *Product Owner*

- Equipe de desenvolvimento

Geralmente as equipes de *Scrum* são emprestadas de diferentes departamento aos quais pertencem

Algumas empresas, mesmo adotando o *Scrum*, ainda possuem o gerenciamento de projetos mais tradicional, setor de qualidade, análise de sistemas, etc.

Equipe Scrum

- *Scrum Master:* tem como função primária remover qualquer impedimento à habilidade de uma equipe de entregar o objetivo do *sprint; o Scrum Master* não é o líder da equipe, já que equipes *Scrum* são auto-organizadas, mas atua como um *firewall* entre a equipe e qualquer influência desestabilizadora

O *Scrum Master* deve assegurar que a equipe esteja utilizando corretamente as práticas de *Scrum*, motivando-os e mantendo o foco na meta do *sprint*

Equipe Scrum

O *Scrum Master* deve:

- permitir que a equipe seja auto-gerenciável
- garantir a eficiência da comunicação
- garantir e auxiliar as práticas do SCRUM
- remover impedimentos
- ser o escudo da equipe frente a problemas externos
- facilitar as reuniões

Equipe Scrum

- *Equipe de desenvolvimento:* com todas as habilidades necessárias (coleta de requisitos, projeto, codificação e teste) para construir o produto de software

- *Product Owner:* um projeto *Scrum* inicia com o PO, que é responsável por obter informações dos *stakeholders*, para elaborar uma lista de requisitos e criar um *backlog* de produto

Equipe Scrum

A equipe de desenvolvimento deve:

- definir a meta do *sprint*
- estar comprometida com o trabalho
- estar comprometida com a qualidade
- colaborar com outros membros
- estimar itens de *backlog* e garantir o esforço necessário para implementação
- participar de reuniões diárias
- indicar impedimentos

Características do Scrum

- clientes se tornam parte da equipe de desenvolvimento (os clientes devem estar genuinamente interessados nos resultados do projeto)

- entregas frequentes de funcionalidades 100% desenvolvidas

- planos frequentes de mitigação de riscos desenvolvidos pela equipe

- problemas não são ignorados e ninguém é penalizado por reconhecer ou descrever qualquer problema não visto

- *sprints* para corrigir *bugs* são feitos de tempos em tempos

- o *Burn Down Chart* é um dos artefatos mais importantes do *Scrum*

Projeto Scrum

Um projeto *Scrum* é iniciado com o *Product Owner* (PO) que é responsável por obter informações dos *stakeholders* (ou usuários que os representem) para elaborar uma **lista de requisitos** e criar um *backlog* de produto

Backlog: log (resumo histórico) de acumulação de trabalho em um determinado período de tempo

O *backlog* de produto é uma lista de requisitos priorizada, que pode incluir aspectos do negócio a tecnologias, questões técnicas e correções de *bugs*

Projeto Scrum

O PO deve:

- definir a visão do produto
- gerenciar o retorno do investimento (ROI)
- apresentar ao time/equipe os requisitos necessários para a entrega do produto
- priorizar cada requisito de acordo com seu valor para o cliente
- gerenciar novos requisitos e priorizações
- planejar releases
- atuar como interface perante os vários clientes do projeto
- garantir que especialistas sobre o negócio estejam disponíveis

Projeto Scrum

O *product backlog* é o coração do *Scrum*: é basicamente uma lista de requisitos, estórias, coisas que o cliente deseja, descritas utilizando a terminologia do cliente (podem ser chamadas de estórias ou itens do *backlog*)

Os requisitos de usuário para o *backlog* de produto do *Scrum* costumam ser coletados como estórias de usuários curtas, durante um *workshop* de 1 ou 2 dias, anteriormente à **reunião de planejamento de *releases*** (planejamento de lançamentos) e à **reunião de planejamento de *sprints***

Projeto Scrum

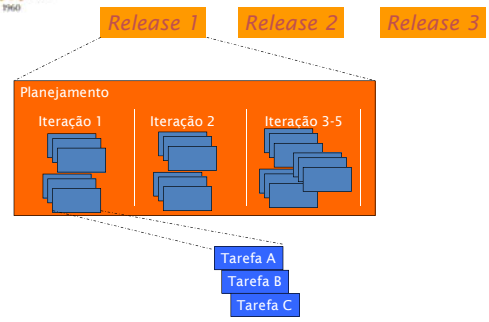
Um ***release*** é formado por múltiplas iterações

Cada iteração pode ser planejada com a mesma caixa de tempo

Stories são colocadas dentro de cada caixa, até estarem completas

O tamanho da caixa representa a velocidade planejada

Projeto Scrum

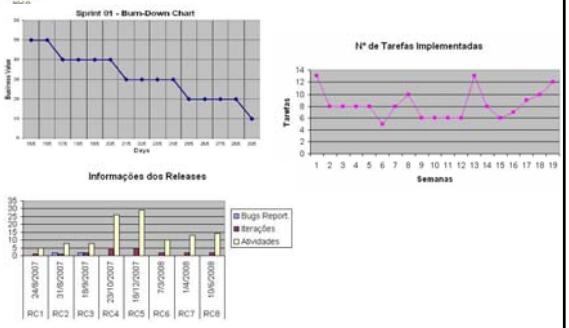


Projeto Scrum

Tamanho ➡ *Velocidade* ➡ *Duração*

Tamanho *300 pontos* ➡ *Velocidade* *20 pontos/Iteração* ➡ *Duração* *300/20=15 Iterações*

Projeto Scrum



Projeto Scrum

Campos de uma estória (itens do backlog de produto):

- Objetivos
- Tarefas
- Itens a verificar

- Tipo de interação: (simples, média, complexa)
- Número de regras de negócio:
- Número de entidades de dados:
- CRUD *(create, read, update* e *delete)*:

Projeto Scrum

Campos de uma estória (itens do backlog de produto):

- Objetivos: Permitir que um cliente faça o seu cadastro no *site*, sem ter que ligar para o serviço de atendimento
- Tarefas: Projeto e codificação da GUI, projeto e codificação da camada de aplicação, projeto e codificação da camada de negócio, projeto e codificação da camada de acesso aos dados, etc.
- Itens a verificar: verificar se o CPF do cliente é válido, verificar se o CEP do cliente existe

- Tipo de interação: (simples, média, complexa)
- Número de regras de negócio: 2
- Número de entidades de dados: 1
- CRUD *(create, read, update* e *delete)*: (Ler e excluir – complexidade simples, criar – média e atualizar/alterar – alta)

Projeto Scrum

O *Product Owner* precisa aprender a respeito do produto antes da reunião de planejamento (quanto mais o PO conhecer o produto, mais poderá ajudar a equipe)

O objetivo do **planejamento de releases** é permitir que a equipe de *Scrum* identifique todos os lançamentos que o produto de software deve ter, com um agenda de entregas prováveis

Normalmente, o planejamento de releases deve durar 4 horas no caso de uma equipe de *Scrum* com *sprints* de 4 semanas

Projeto Scrum

Além do planejamento de releases, a equipe de *Scrum* também deve passar por alguns **planejamentos de sprints**

Normalmente, a reunião de **planejamento de sprints** deve durar em torno de 8 horas, no caso de *sprints* de 4 semanas (ou 4 horas para *sprints* de 2 semanas)

Durante a 1a parte da reunião, o PO percorrerá os requisitos, na forma de estórias do usuário, para decidir, com o *feedback* da equipe, quais deveriam fazer parte de quais *sprints* e quais são seus objetivos

Projeto Scrum

A 1a reunião serve, principalmente, para responder à questão **"O quê?"**

Durante a 2a parte da reunião de planejamento de *sprints*, que focaliza o **"como"**, a equipe de desenvolvimento tentará identificar tarefas (*tasks*) a partir das estórias previamente escolhidas e deduzir quanto tempo (em horas) a equipe levará para transformar essas tarefas em incrementos de produtos potencialmente entregáveis

Projeto Scrum

No *Scrum* os projetos são divididos em ciclos chamados de *sprints* (tipicamente mensais)

O *sprint* representa um tempo definido dentro do qual um conjunto de atividades deve ser executado

Metodologias ágeis de desenvolvimento de software são iterativas, ou seja, o trabalho é dividido em iterações que, no *Scrum*, são chamadas de *sprints* e duram de 2 a 4 semanas

Projeto Scrum

No início de cada *sprint* faz-se um *sprint planning meeting* (reunião de planejamento) no qual o PO (que representa os *stakeholders*) prioriza todos os itens do *product backlog* e a equipe seleciona as funcionalidades que ela será capaz de implementar durante o *sprint* que se inicia

As funcionalidades alocadas em um *sprint* são transferidas do *product backlog* (lista de funcionalidades do projeto) para o *sprint backlog* (lista de funcionalidades que serão implementadas no *sprint*)

Projeto Scrum

Diariamente em uma *sprint*, a equipe faz uma breve reunião de no máximo 15 minutos, com todos os participantes em pé, chamada *daily scrum*

O objetivo é cada integrante dizer o que fez no dia anterior, o que pretende fazer no dia que se inicia e se existe algum impedimento que está atrapalhando o seu trabalho

Projeto Scrum

A equipe se reúne todos os dias, não para inspecionar o estado do projeto e, sim, inspecionar o progresso da equipe rumo ao objetivo do *sprint* (o *daily scrum* não é uma reunião de *status*)

O registro do progresso da equipe rumo ao objetivo do *sprint* é realizado por meio de um gráfico de *burndown* (ou gráfico de consumo)

Projeto Scrum

Antes do final de cada *sprint* a equipe deve se reunir com o PO – revisão de *sprint* organizada pelo SM (reunião cronometrada, que dura 4 horas – *sprints* de 4 semanas ou 2 horas – *sprints* de 2 semanas)

A reunião serve para:
- que a equipe de Scrum e o PO discutam o que foi e o que não foi feito
- demonstrar para o PO o que foi feito e obter o seu *feedback*
- obter atualizações do PO relativas a quaisquer mudanças na direção do mercado ou do produto

Projeto Scrum

Finalmente, faz-se uma *sprint* retrospectiva para identificar o que funcionou bem, o que pode ser melhorado e que ações serão tomadas para melhorar e a equipe parte para o planejamento do próximo *sprint*

Ao final de um *sprint*, a equipe apresenta as funcionalidades implementadas em uma *sprint review meeting*

Bibliografia Recomendada

PHAM, Andrew; PHAM, Phuong-Van. Scrum em Ação: gerenciamento e desenvolvimento ágil de projetos de software. São Paulo: Novatec-Cengage Learning, 2011.