



CENTRO UNIVERSITÁRIO EUROAMERICANO – UNIEURO
PRÓ-REITORIA E PÓS-GRADUAÇÃO, PESQUISA E EXTENSÃO
COORDENAÇÃO DE PÓS-GRADUAÇÃO LATO SENSU
CURSO PÓS-GRADUAÇÃO EM TESTE DE *SOFTWARE*

RENATA ELIZA PINTO FERREIRA
SIBELE ESTEVES RAMOS
VIVIAN CRISTINA LAGARES

SCRUM NO TESTE DE SOFTWARE

Belo Horizonte, Março/2010



RENATA ELIZA PINTO FERREIRA
SIBELE ESTEVES RAMOS
VIVIAN CRISTINA LAGARES

SCRUM NO TESTE DE SOFTWARE

Trabalho de Conclusão de Curso – Monografia,
apresentada como requisito parcial para
conclusão do Curso de MBA em Teste de
Software do Centro Universitário Euroamericano
– Unieuro

Orientadora: Professora Cleziana de Freitas Costa

Belo Horizonte, Março/2010

RENATA ELIZA PINTO FERREIRA
SIBELE ESTEVES RAMOS
VIVIAN CRISTINA LAGARES

SCRUM NO TESTE DE SOFTWARE

Esta monografia foi julgada adequada à obtenção do grau de Especialista em Teste de *Software* e aprovada em sua forma final pelo curso de Pós-Graduação Lato Sensu em Teste de *Software* do Centro Universitário Euroamericano - UNIEURO.

Data de aprovação: 12-03-2010

Banca Examinadora

Professora Especialista Cleziana de Freitas Costa

Centro Universitário UNIEURO

Professora Wilsa Sette Morais Figueiredo

Centro Universitário UNIEURO

Professora Msc. Edna Dias Canedo

Centro Universitário UNIEURO

"Testar igual a minha mãe testa é fácil. Difícil é se tornar um especialista e testar o *software* utilizando técnicas e metodologias."

Renata Eliza

À minha família, em especial minha mãe Eliza e minha Tia Maria pelo intenso incentivo e apoio. Aos meus amigos, que de certa forma compreenderam a minha ausência durante boa parte do curso. À minha super amiga e companheira de curso Vivian Lagares, por todo o companheirismo, reflexões, network e apoio para que vencêssemos juntas essa etapa. E ao Cara Lá de Cima, que mais uma vez me mostrou que quando a gente realmente quer alguma coisa, a gente consegue.

Renata Eliza

A meus pais e familiares pelo apoio de sempre. E a todos aqueles que, direta ou indiretamente, acreditaram e me incentivaram a buscar meus ideais

Sibele Esteves

Ao meu marido, mãe e irmã por sempre me apoiarem e incentivarem em todos os meus projetos e por entenderem as freqüentes ausências. Eles são a minha vida, o meu alicerce! Ao meu padrinho Alysson, pela hospitalidade, dicas, apoio e conselhos profissionais e pessoais. A minha amigona do coração Renata Eliza, por sempre me ajudar, por ter paciência com as nossas diferenças e por me incentivar em todos os momentos do curso.

Vivian Lagares

AGRADECIMENTOS

Agradecemos a orientadora Cleziana Costa pela compreensão e ajuda.

Ao Fabrício Ferrari, ao Denis Ferrari, ao Thúlio Prudente e ao Douglas Matoso pelo compartilhamento da experiência.

A Deus e a todas as pessoas que contribuíram com suas reflexões para o desenvolvimento deste trabalho.

RESUMO

O presente trabalho foi baseado em pesquisa qualitativa, exploratória e descritiva através de estudos de casos de empresas privadas com seguimento em desenvolvimento de *Software* que utilizam a metodologia *Scrum*; e provas de conceito apresentando os resultados, com o objetivo de estudar o Processo Ágil *Scrum*, e sua aplicabilidade no Teste de *Software*. Inicialmente é apresentado o conceito de Processos Ágeis, o Manifesto Ágil e os modelos existentes da Metodologia Ágil. Em seguida, é apresentada a Metodologia *Scrum*, sua origem, ciclo de vida, seus papéis e ferramentas. Além disso, é apresentada a aplicabilidade do *Scrum* no Teste de *Software*, os estudos de caso e as vantagens e as desvantagens da utilização da metodologia.

Palavras-Chave: Processos Ágeis, *Scrum*, Teste de *Software*, Teste, Metodologia Ágil.

ABSTRACT

This study was based on qualitative research, exploratory and descriptive using case studies of private companies focused in developing *software* using the *Scrum* methodology, and proof of concept showing the results aiming to study the Agile Process *Scrum*, and its applicability in *Software* Testing. First is presented the concept of Agile processes, the Agile Manifest and the existing models of Agile Methodology. Then you see the *Scrum* methodology, their origin, life cycle, their roles and tools. Additionally, you receive the applicability of the *Scrum Software* Testing, case studies and the advantages and disadvantages of using the methodology.

Key-words

Agile Processes, *Scrum*, *Software* Testing, Testing, Agile Methodology.

LISTA DE ILUSTRAÇÕES

Figura 01 - O <i>Scrum</i> - (CLARIFICATION, 2009)	13
Figura 02 - Modelo de Gerenciamento Empírico - (CORDEIRO, 2006)	15
Figura 03 - O ciclo de vida - (THAMIEL, 2009)	16
Figura 04 - <i>Sprint Burndown</i> - (GIRL WRITES CODE, 2009).....	21
Figura 05 - Quadro Kanban - (SPARTA CONSULTORIA, 2009)	22
Figura 06 - <i>Planning Poker</i> - (SPRINT PLANNING, 2009)	22
Figura 07 - Tipos de Teste - (DUONG, 2009)	26

SUMÁRIO

1.	INTRODUÇÃO	1
1.1	Problema	1
1.2	Justificativa	1
1.3	Objetivos	2
1.4	Metodologia	2
1.5	Estrutura	3
2.	PROCESSOS ÁGEIS	4
2.1	Definição	4
2.2	O Manifesto Ágil	5
2.3	Conceitos Chave	9
2.4	Modelos	9
3.	SCRUM	13
3.1	A origem	13
3.2	Metodologia	14
3.3	O Ciclo de Vida	16
3.4	Papéis	19
3.5	Ferramentas	21
4.	APLICABILIDADE DO SCRUM NO TESTE DE SOFTWARE	23
4.1	Equipes Metodologia Tradicional x Equipes Metodologia Ágil	23
4.2	Os papéis do Analista de Teste no Time <i>Scrum</i>	23
5.	ESTUDOS DE CASO	28
5.1	Voice Technology	28
5.2	Ci&T	29
5.3	Empresa A	30
5.4	EMBRAS.net	31
6.	VANTAGENS E DESVANTAGENS DO SCRUM	33
7.	CONCLUSÃO	36
8.	REFERÊNCIA BIBLIOGRÁFICA	37

9. ANEXO 40

1. INTRODUÇÃO

1.1 Problema

Em função do fracasso no desenvolvimento de grandes projetos de *software* no início da década de 70, surgiu a necessidade da criação de uma nova metodologia, como uma alternativa às metodologias tradicionais.

Uma nova abordagem para desenvolvimento de *software* tem despertado grande interesse entre as organizações de todo o mundo. Estamos vivendo uma tendência para o desenvolvimento ágil de aplicações devido ao ritmo acelerado de mudanças na tecnologia da informação, pressões por constantes inovações, concorrência acirrada e grande dinamismo no ambiente de negócios (BOEHM, 2006).

Apesar de existir há um bom tempo, apenas recentemente a expressão “Métodos Ágeis” vem se tornando mais popular no Brasil por usar uma abordagem simplificada. No entanto, “ser simples” geralmente é confundido com falta de controle e completa anarquia. Na verdade, ser simples, ter agilidade, é fazer a diferença e, ao contrário do que parece, exige muita disciplina e organização.

A expressão “Métodos Ágeis” está se tornando mais popular no Brasil apenas recentemente, apesar de ser um termo antigo. A simplicidade pode ser confundida com falta de controle, porém, a agilidade quebra esse paradigma, pois necessita de muita organização e disciplina.

Métodos, práticas e técnicas para o desenvolvimento ágil de projetos prometem aumentar a satisfação do cliente (BOEHM, 2003) para produzir alta qualidade de *software* e para acelerar os prazos de desenvolvimento de projetos (ANDERSON, 2003).

1.2 Justificativa

Nosso trabalho tem o enfoque na apresentação da utilização da metodologia *Scrum* no Teste de *Software*, ressaltando benefícios como colaboração, integração e boas práticas, auxiliando na qualidade e sucesso dos projetos.

Além disso, serão apresentados modelos de adoção do *Scrum* em empresas e em equipes, demonstrando assim possíveis problemas de adaptabilidade, já que poderá ser uma grande mudança na forma de se trabalhar.

Serão realizados estudos de viabilidade de adoção do *Scrum*, incluindo a equipe de Teste de *Software*, já que a metodologia veio se opor ao processo cascata de desenvolvimento, que joga os testes sempre para o fim esmagando os prazos e comprometendo a qualidade.

Os riscos e vantagens da utilização da metodologia *Scrum* vão ser listados, bem como os resultados obtidos da análise das empresas que utilizam o *Scrum*.

1.3 Objetivos

1.3.1 Objetivo Geral

O objetivo desse trabalho é descrever o Processo Ágil *Scrum*, e analisar como o mesmo pode ser aplicado no Teste de *Software*.

1.3.2 Objetivos Específicos

- Identificar conceitos de Processos Ágeis e da metodologia *Scrum*;
- Analisar a aplicabilidade do *Scrum* no Teste de *Software*;
- Descrever os papéis da equipe de Teste no Time *Scrum*;
- Investigar em empresas que utilizam o *Scrum*, informações relativas à implantação, vantagens, desvantagens e aplicação da metodologia no Teste de *Software*;
- Descrever as vantagens e riscos inerentes a utilização da Metodologia *Scrum*.

1.4 Metodologia

Como metodologia de pesquisa para reunir informações foi utilizada pesquisa Bibliográfica, pesquisa em empresas que utilizam a metodologia, provas de conceito com a apresentação de resultados, a *Internet* para obter conhecimento sobre o que havia de novo sendo desenvolvido e quais as exigências do mercado, a fim de conhecer profundamente a metodologia *Scrum*.

1.5 Estrutura

O trabalho é dividido segundo a estrutura a seguir:

- O capítulo 2, intitulado “Processos Ágeis”, abrange a definição e características dos Processos Ágeis.
- O capítulo 3, intitulado “*Scrum*”, conceitua a metodologia.
- O capítulo 4, intitulado “Aplicabilidade do *Scrum* no Teste de *Software*”, descreve como a metodologia pode ser aplicada no Teste de *Software*.
- O capítulo 5, intitulado “Estudos de Caso”, relata entrevistas realizadas com profissionais de empresas de desenvolvimento de *software* que utilizam o *Scrum*.
- O capítulo 6, intitulado “Vantagens e Desvantagens”, lista as vantagens e as desvantagens provenientes da utilização da metodologia no Teste de *Software*.
- Finalmente, o capítulo 7 conclui o trabalho mostrando a relevância do mesmo.

2. PROCESSOS ÁGEIS

2.1 Definição

Agilidade envolve disciplina e trabalho coordenado e pode tratar de projetos de vários segmentos e portes.

Agilidade é a habilidade de criar e responder a mudanças com respeito ao resultado financeiro do projeto em um turbulento ambiente de negócios. Agilidade é a habilidade de balancear flexibilidade com estabilidade. (Highsmith, Jim. *Agile Project Management*, 2002)

Parte do princípio que os projetos devem ser desenvolvidos com uma estrutura e organização necessários. Estruturas e organizações reduzem a criatividade e a flexibilidade de suportar mudanças, poucas estruturas e organizações trabalham a ineficiência e resulta em esforços maiores que os necessários.

Em função do fracasso no desenvolvimento de grandes projetos de *software* no início da década de 70, surgiu a necessidade da criação de uma nova metodologia, como uma alternativa às metodologias tradicionais.

Os métodos de desenvolvimento conhecidos como “Ágeis” (em inglês *Agile Modeling*, ou AG) possuem o objetivo de reduzir o ciclo de vida do *software* (e por consequência acelerar o seu desenvolvimento) através do desenvolvimento de versões mínimas. As funcionalidades são integradas por processos iterativos baseados na interatividade com o cliente e testes em conjunto e paralelo ao ciclo de desenvolvimento.

Segundo *Scott W. Ambler*, Metodologia Ágil “é uma metodologia baseada na prática para modelagem eficaz de *software*”.

É um conjunto de práticas, onde os princípios e valores aplicados por profissionais de *software* no dia a dia são seguidos como guia. Não é um processo que define detalhadamente como criar um modelo, mas provê conselhos.

Em 1990 o desenvolvimento de *software* ágil evoluiu bastante em resposta a inconformidade dos métodos tradicionais, caracterizados por uma pesada regulamentação e gerenciamento através do modelo cascata de desenvolvimento.

O processo originou-se da visão de que o modelo cascata era burocrático, contraditório e lento.

Duas motivações podem ser citadas como principais para a criação da Metodologia Ágil:

- O objetivo primário de um projeto de *software* é o próprio *software*, e não um grande conjunto de documentação sobre ele;
- Um artefato é criado primordialmente para permitir a comunicação e a troca de informações entre a equipe (um modelo de classes é criado para passar para a equipe a hierarquia imaginada pelo projetista, assim como um diagrama de casos de uso é criado para a equipe ter uma visão do mecanismo da funcionalidade envolvida) e permitir a discussão e refinamento do modelo do sistema. Assim, se um artefato não está passando informação relevante ao projeto, ele não cumpre seu objetivo básico.

A criação dos Processos Ágeis não implica no abandono das metodologias tradicionais e das práticas de engenharia de *software*.

2.2 O Manifesto Ágil

O Manifesto Ágil é uma declaração de princípios que fundamentam o desenvolvimento ágil de *software* (WIKIPEDIA, 2009).

Em 2001 foram apresentadas práticas efetivas para que fosse possível a utilização dos Processos Ágeis no gerenciamento de projetos. Essas práticas foram compiladas através do consenso de grandes pensadores da comunidade de desenvolvimento de *softwares* e foram denominadas “Manifesto Ágil”.

Os autores do manifesto são: Kent Beck, Mike Beedle, Arie van Bennekum, Alistair Ockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, Jon Kern, Brian Marick, Robert C. Martin, Steve Mellor, Ken Schwaber, Jeff Sutherland e Dave Thomas.

O manifesto ágil cita doze princípios ágeis, vistos a seguir.

2.2.1 A prioridade é satisfazer ao cliente através de entregas contínuas e freqüentes

Como a principal prioridade é por um produto final de qualidade, foram feitas pesquisas para definir quais as características durante a fase de desenvolvimento, que foram comuns nos processos que obtiveram sucesso. O resultado desta pesquisa revelou que o quanto antes se entregasse uma versão, mesmo que com poucas funcionalidades, do sistema, melhor seria a qualidade final, pois o sistema começaria a evoluir mais cedo. Sendo que essas versões deveriam ser atualizadas e entregues num período curto de tempo.

Um Processo Ágil é aquele que entrega cedo e constantemente. Tenta-se entregar o primeiro sistema já com algumas funcionalidades desenvolvidas e funcionais em algumas semanas do início do projeto, sendo que novos pacotes de funcionalidades continuam sendo desenvolvidos e integrados ao módulo anterior, no decorrer de algumas semanas. Isso deve ocorrer constantemente. É de opção do cliente, escolher se vai implantar o sistema ainda não todo completo ou vai apenas testá-lo para definir novas funcionalidades ou encontrar não conformidades com aquilo que é necessário.

2.2.2 Receber bem as mudanças de requisitos, mesmo em uma fase avançada do projeto

Indica atitude. Os participantes dos Processos Ágeis não temem mudanças, pois assumem que as mudanças são indicadores de que o time está aprendendo sobre o sistema, possibilitando saber se vai satisfazer o mercado.

2.2.3 Entregas com freqüência, sempre na menor escala de tempo

A idéia é entregar *softwares* funcionais regularmente em um curto período de tempo. Documentações e papéis não contam como entregas, pois não satisfazem à necessidade atual do cliente.

2.2.4 As equipes de negócio e de desenvolvimento devem trabalhar juntas diariamente

Um processo ágil precisa ser guiado constantemente, portanto é necessária a interação freqüente de Clientes, Desenvolvedores e Partes Interessadas, pois todos fazem parte do time.

2.2.5 Manter uma equipe motivada fornecendo ambiente, apoio e confiança necessários

Os indivíduos são parte fundamental do processo, como citados nos quatro valores das Metodologias Ágeis, portanto é fundamental para o sucesso do projeto que todos estejam motivados, seja com o apoio de outros membros da equipe ou material e equipamento. Indivíduos motivados trabalham melhor.

2.2.6 A maneira mais eficiente da informação circular através de uma conversa face-a-face

Conversação e Comunicação são valores básicos dos Processos Ágeis. Documentos e papéis são supérfluos, apenas escritos quando extremamente necessitados e significantes. A maneira padrão de troca de informações é a conversação.

2.2.7 Ter o sistema funcionando é a melhor medida de progresso

30% do projeto estão concluídos, quando 30% das funcionalidades necessárias estiverem funcionando. O volume de documentos e qual fase do processo se está não determinam o progresso atual do projeto.

2.2.8 Processos Ágeis promovem o desenvolvimento sustentável

A velocidade do desenvolvimento nos Processos Ágeis deve ser sempre constante. De nada adianta começar um projeto desenvolvendo de maneira rápida e essa velocidade não ser mantida, pois isso irá iludir o cliente, causando a ele uma impressão de falsa velocidade. Desenvolver em velocidades diferentes cansa mais os

desenvolvedores, fazendo-os ficarem estressados isso diminui a qualidade final do produto, quebrando um dos principais valores dos Processos Ágeis.

2.2.9 Atenção contínua a excelência técnica e a um bom projeto aumentam a agilidade

Alta qualidade é a chave para a agilidade. Para manter rápido o desenvolvimento é necessário deixar o *software* o mais limpo e o mais robusto possível. Isso porque os times de Processos Ágeis estão comprometidos em desenvolver códigos da melhor qualidade.

2.2.10 Simplicidade é essencial

O caminho a ser escolhido para desenvolver e projetar é sempre o caminho mais simples e consistente que leve ao resultado final esperado. A escolha do caminho mais simples implica em dizer que caso alguma mudança ocorra – e elas vão ocorrer – será mais fácil alterar o que já foi feito.

2.2.11 As melhores arquiteturas, requisitos e projetos provêm de equipes organizadas

Decisões são tomadas por toda a equipe e não por apenas um indivíduo. Isso caracteriza uma equipe organizada, pois todos trabalham juntos para o sucesso do projeto. Responsabilidades são compartilhadas por todos os membros da equipe, mostrando que todos estão envolvidos no processo de um projeto.

2.2.12 Em intervalos regulares, a equipe deve refletir sobre como se tornar mais eficaz

Para uma equipe ágil melhorar é uma constante máxima. Sempre são feitas reuniões com a equipe a fim de encontrar novos pontos onde o time pode ser melhorado para satisfazer as necessidades atuais do meio, que está em constante mudança. O meio está sempre mudando e para a equipe continuar ágil é necessário que ela mude de acordo.

2.3 Conceitos Chave

Alguns defendem que além dos itens citados no manifesto, deve-se levar em consideração os seguintes conceitos chave:

- Indivíduos e iterações ao invés de processos e ferramentas;
- *Software* funcional ao invés de documentação detalhada;
- Colaboração do Cliente ao invés de negociação de contratos;
- Responder às mudanças ao invés de seguir um plano.

2.4 Modelos

Devido ao grande interesse da comunidade de desenvolvimento de *software*, tem crescido a demanda e com isso surgido uma considerável quantidade de métodos ágeis nos últimos anos, como:

2.4.1 *Scrum*

É um modelo de desenvolvimento ágil de *software* e não uma fórmula mágica que resolverá todos os problemas obtidos na sua empresa no processo de desenvolvimento. Não é baixada da Internet e nem precisa ser instalado, pois é na verdade uma *framework*. Hoje em dia no Brasil e na Europa o *Scrum* é a metodologia ágil mais utilizada pelas pequenas, médias e grandes empresas. O seu funcionamento acontece da seguinte forma: atividades de monitoramento e *feedback*, com reuniões rápidas e diárias com suas equipes de poucos integrantes, visando a identificação e correção de qualquer problema no processo de desenvolvimento. É uma mudança cultural na equipe de trabalho e não quer dizer que ao utilizá-lo, o método em si fará com que o seu cronograma seja cumprindo a risca, isso vai depender de toda a equipe técnica envolvida no processo. Um dos pontos fortes dessa metodologia é a maneira de se relacionar em equipe, ou seja, a parceria em times de profissionais cada um com sua especialidade têm o mesmo grau de comprometimento para a conclusão e o sucesso do projeto.

2.4.2 *Extreme Programming (XP)*

É um modelo de desenvolvimento ágil de *software*, mais utilizado pelas pequenas e médias empresas, tornou-se popular na década de 90 nos Estados Unidos e atualmente vem crescendo e sendo adotado com bastante frequência no Brasil. O processo é composto por quatro atividades: Planejamento, Projeto, Codificação e Teste e por quatro diretrizes: *Feedback*, Comunicação, Simplicidade e Coragem. Um dos pontos propostos por essa metodologia é execução de reuniões de pé, para que cada membro da equipe possa descrever as atividades realizadas no dia anterior de forma clara, rápida e objetiva. Essa metodologia é muito utilizada em projetos cujos requisitos do projeto são sempre alterados. O cliente deve sempre se fazer presente e cabe a ele apontar as prioridades do projeto e dessa forma definir a ordem que as atividades serão executadas.

2.4.3 *Feature Driven Development (FDD)*

É uma metodologia ágil com grande eficiência e aderência a requisitos, muito utilizada em *softwares* de tamanho médio e grande. Tornou-se conhecida em 1997, definida por duas fases que são: Concepção e Planejamento e a de Construção, é uma metodologia que agrada tanto o cliente, como os gerentes e desenvolvedores. A FDD chama a atenção por algumas características peculiares:

- Resultados úteis a cada duas semanas ou menos;
- Blocos bem pequenos de funcionalidade valorizada pelo cliente, chamados "*Features*";
- Planejamento detalhado e guia para medição;
- Rastreabilidade e relatórios com incrível precisão;
- Monitoramento detalhado dentro do projeto, com resumos de alto nível para clientes e gerentes, tudo em termos de negócio;
- Fornece uma forma de saber, dentro dos primeiros 10% de um projeto, se o plano e a estimativa são sólidos;

O lema da FDD é: "Resultados frequentes, tangíveis e funcionais."

2.4.4 *Dynamic Systems Development Method (DSDM)*

É uma metodologia de desenvolvimento de *software* onde o grande referencial é construir *softwares* de forma que satisfazem às restrições de prazos apertados através de prototipagem incremental. Utilizado por pequenas equipes e suporta alteração de requisito durante o processo de desenvolvimento, criada em 1994, onde os princípios básicos são:

- Iteração (duas a seis semanas);
- *Releases* frequentes;
- Qualidade total;
- Adaptabilidade a mudança de requisitos.

2.4.5 *Adaptive Software Development (ASD)*

É uma metodologia utilizada em sistemas grandes e complexos, definida por ciclos de três fases que são: Colaboração, Especulação e Aprendizado, cada ciclo dura de quatro a oito semanas. Criado em 1997, se baseia na idéia de que Sistema Complexo é igual a Resultados Imprevisíveis. Suporta mudanças frequentes e a abordagem é: *Do it wrong the first time: erre cedo, corrija cedo, não potencialize mal-entendidos.*

2.4.6 *Família Crystal/Clear*

É um conjunto de metodologias, uma para cada projeto definido de acordo com a necessidade de cada um, voltada para projetos pequenos de no máximo seis desenvolvedores. Seu autor é *Alistair Cockburn* (IBM - anos 90). Essas metodologias possuem como foco o desenvolvimento dividido em períodos curtos que são chamados iterações. Essas iterações duram em média de duas a quatro semanas. Ao final de cada período de iteração é possível ter um *software* já com partes funcionais, possível de ser usado. Seguem processos iterativos e sucessivas entregas ao cliente, dessa forma, desde cedo o cliente pode verificar se o que está sendo construído atende as suas necessidades e expectativas.

O *Scrum*, particularmente tem tido um crescimento visível na indústria de *software*, e concentra-se no planejamento e acompanhamento do projeto.

3. SCRUM

3.1 A origem

Scrum é o nome de uma jogada do *rugby*; um jogo britânico com oito jogadores em cada time. Acontece na disputa pela bola em casos de penalidades ou faltas.

Os jogadores dos dois times, um de cada lado, formam um único bloco de pessoas encaixadas pelos ombros e pelos braços. A bola é jogada no meio desse bloco. Neste momento um time tenta empurrar o outro até que a bola saia entre as pernas dos jogadores e jogo continue.



Figura 01 - O *Scrum* - (CLARIFICATION, 2009)

O ponto crucial dessa jogada é o trabalho em equipe, onde se um falhar na formação, o outro time ganha bola e domina a situação.

Em 1986, *Hiroataka Takeuchi* e *Ikujiro Nonaka* publicaram um estudo intitulado de "*The New Product Development Game*" (Harvard Business Review, Janeiro-Fevereiro 1986). Nesse estudo, eles compararam equipes pequenas, de alto desempenho e multidisciplinares à jogada "*Scrum*" do *Rugby*.

Takeuchi e *Nonaka* descobriram que utilizando esse tipo de equipe, obteriam melhores resultados. Desenvolveram então, o *Scrum* voltado para gerenciar o desenvolvimento de produtos em empresas de fabricação de automóveis e produtos de consumo.

Cada elemento deste por si não traz velocidade e flexibilidade. Mas tomados como um todo, formam características que podem produzir um poderoso conjunto de dinâmicas que fazem a diferença. (TAKEUCHI, NONAKA, 1986, p. 137).

Para eles, existem os seguintes fatores de sucesso para o *Scrum*:

- Instabilidade Intrínseca
- Times Auto-Organizados
- Fases de Desenvolvimento Sobrepostas
- Multi-Aprendizado (Multi-Nível e Multi-Funcional)
- Controle Sutil
- Transferência de Conhecimento Organizacional

Em 1993, *Jeff Sutherland*, *John Scumniotales*, e *Jeff McKenna*, incorporando o estilo de gerenciamento observado por *Takeuchi* e *Nonaka*, criaram, documentaram e implantaram o *Scrum* na empresa *Easel Corporation*. Dessa vez, voltado para gerenciar o desenvolvimento de projetos de *software*.

Mais tarde, em 1995, *Ken Schwaber* formalizou a definição de *Scrum* e ajudou a implantá-lo em desenvolvimento de *software* em todo o mundo.

Em 1996, *Schwaber* e *Sutherland* apresentaram o *Scrum* na *Object-Oriented Programming, Systems, Languages and Applications (OOPSLA)*, uma conferência anual sobre programação, sistemas, linguagens e aplicações.

O *Scrum* junta conceitos do pensamento *Lean* de desenvolvimento iterativo e incremental. O pensamento *Lean* é um conjunto de conceitos e procedimentos, que tem como objetivo simplificar o modo como uma organização acaba com os desperdícios.

Atualmente, o *Scrum* vem se enquadrando junto às necessidades dos projetos de *software*, com a proposta de tornar os projetos mais flexíveis e ágeis, se tornando uma das mais utilizadas metodologias da atualidade.

3.2 Metodologia

O *Scrum* é uma metodologia ágil para gestão e planejamento de projetos de *software*. É um processo de desenvolvimento iterativo e incremental que pode ser aplicado a qualquer produto ou até mesmo no gerenciamento de uma atividade complexa, ou seja, sempre que um grupo de pessoas necessite trabalhar em conjunto para atingir um objetivo comum.

Mike Cohn definiu o Scrum em 100 palavras:

Scrum é um processo ágil que permite manter o foco na entrega do maior valor de negócio, no menor tempo possível.

Isto permite a rápida e contínua inspeção do software em produção (em intervalos de duas a quatro semanas).

As necessidades do negócio é que determinam as prioridades do desenvolvimento de um sistema. As equipes se auto-organizam para definir a melhor maneira de entregar as funcionalidades de maior prioridade.

Entre cada duas a quatro semanas todos podem ver o real software em produção, decidindo se o mesmo deve ser liberado ou continuar a ser aprimorado por mais um "Sprint". (SCRUM TRAINING & AGILE TRAINING FROM SCRUMMASTER MIKE COHN, 2009)

O Scrum aplica o modelo de gerenciamento empírico.

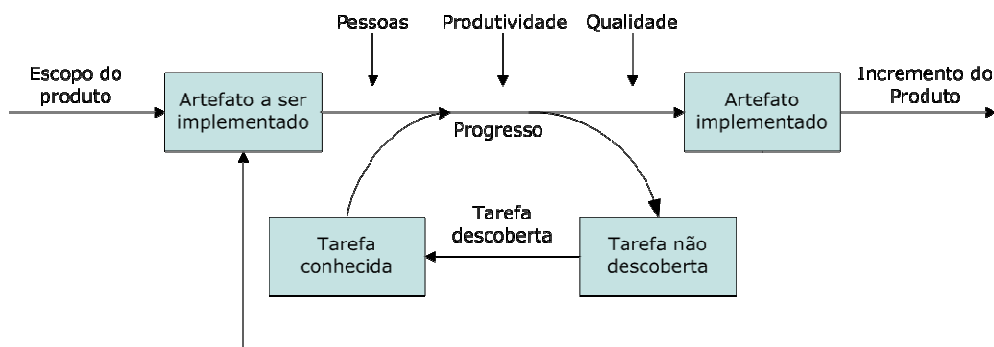


Figura 02 - Modelo de Gerenciamento Empírico - (CORDEIRO, 2006)

Três pilares sustentam qualquer implementação de controle de processos empíricos.

O primeiro pilar é a transparência, que visa garantir que aspectos do processo que afetam o resultado devem ser visíveis para aqueles que gerenciam os resultados.

O segundo pilar é a inspeção. Todos os aspectos do processo devem ser inspecionados com uma frequência suficiente para que qualquer situação não programada no processo seja detectada.

O terceiro pilar é a adaptação. A partir da inspeção, que um ou mais aspectos do processo estão fora dos limites aceitáveis e que o produto resultante será inaceitável, ele deverá ajustar o processo ou o material sendo processado.

O *Scrum* inspeciona a condição das tarefas e empiricamente determina o que deve ser feito para produzir o resultado esperado. Parte do princípio de que nem todas as características do produto são conhecidas na análise e que provavelmente os requisitos mudarão com o passar do tempo.

O *Scrum* é um processo que controla a desordem resultante de necessidades e interesses. É uma forma de aumentar a comunicação e maximizar a cooperação e também detectar e remover qualquer impedimento que atrapalhe o desenvolvimento de um produto, não há prática de engenharia prescrita e as equipes se auto-organizam.

3.3 O Ciclo de Vida



Figura 03 - O ciclo de vida - (THAMIEL, 2009)

3.3.1 Sprints

Os *Sprints* são ciclos (ou iterações) periódicos que podem variar de duas a quatro semanas de duração. Durante o período estabelecido para o *Sprint* são realizadas as atividades pré-definidas no *Product Backlog*.

É uma parte importante do ciclo de vida do *Scrum*. Cada *Sprint* deve ser conduzido com muita atenção e cuidado.

3.3.2 Sprint Planning

É uma reunião realizada no começo de cada *Sprint*, visa traçar a meta de trabalho para cada ciclo.

3.3.3 *Sprint Backlog*

São todas as tarefas que devem ser desenvolvidas durante cada *Sprint*, extraídas do *Product Backlog*.

Tem como objetivo dar visibilidade ao Projeto.

Deve ser visível e de fácil acesso aos membros do *Scrum Team*. É fundamental que não sofra nenhuma alteração durante a iteração.

As atualizações devem ser registradas em tempo real e podem ser apresentadas através de um quadro chamado “Kanban”.

3.3.4 *Product Backlog*

É o que o cliente deseja implementar. É uma lista que contém todas as tarefas a serem realizadas, incluindo a prioridade de cada uma. As tarefas devem ser descritos de forma clara e simples.

Definidas as prioridades, cabe ao *Scrum Team* determinar quais os itens serão implementados no próximo *Sprint*.

Cada item do *Product Backlog* é dividido em uma ou mais tarefas do *Sprint Backlog*. Essa divisão auxilia na distribuição do trabalho entre os membros da equipe.

Não existe nenhuma obrigatoriedade do *Product Backlog* estar completo já no começo do projeto. Ele deve começar com os itens que são mais importantes em um primeiro momento. O *Product Backlog* cresce e muda à medida que se aprende mais sobre o produto e seus usuários.

O *Product Owner* deve estar priorizando e atualizando o *Product Backlog* constantemente, sempre em função da visão do produto.

3.3.5 *Daily Scrum*

É uma breve reunião realizada diariamente, com duração máxima de 15 minutos.

O objetivo é compartilhar e disseminar conhecimento sobre as atividades realizadas no dia anterior, identificar os problemas, os riscos e priorizar as tarefas que serão realizadas naquele dia. Aconselha-se que ocorra pela manhã.

Durante o *Daily Scrum*, cada membro do *Scrum Team* deve responder às seguintes três perguntas:

- O que você fez ontem?
- O que você fará hoje?
- Há algum impedimento no seu caminho?

As respostas dessas perguntas auxiliarão o *Scrum Team* na compreensão do status atual do *Sprint*. A reunião é importante também, pois faz com que cada membro da equipe assuma compromissos perante os demais.

Os problemas impeditivos identificados durante o *Daily Scrum* devem ser tratados rapidamente pelo *Scrum Master*.

3.3.6 *Sprint Review*

É uma reunião realizada ao final de cada *Sprint*. O projeto é avaliado e o *Scrum Team* apresenta os resultados obtidos durante o *Sprint*. Só devem ser apresentados os itens que estiverem 100% prontos, através de uma demonstração funcional.

Deve ter a duração máxima de duas horas. Os slides e vídeos de apresentação são dispensados em função da informalidade da reunião.

O *Product Owner* tem o direito de aceitar ou rejeitar o *Sprint*.

Qualquer necessidade de mudança, como a incorporação de novas tarefas ao *Product Backlog*, devem ser tratadas em momento oportuno e priorizadas novamente.

3.3.7 *Sprint Retrospective*

É uma reunião realizada ao final de um *Sprint*. O objetivo é garantir um processo de melhoria contínua.

O *Sprint Retrospective* visa identificar o que funcionou, o que precisa ser melhorado e quais ações serão tomadas para melhorar. Ou seja, avaliar e aprender com a experiência visando aumentar a produtividade.

Nessa reunião é obrigatória a participação do *Scrum Team*. O *Product Owner* depende de um convite da equipe para acompanhá-la. Cabe ao *Scrum Master* registrar todas as informações e o time prioriza a ordem ideal de mudança.

Aconselha-se que o tempo de duração seja de 15 a 30 minutos.

3.4 Papéis

3.4.1 *Product Owner*

O *Product Owner* é o procurador do cliente, sua função consiste em representar e defender os seus interesses. Está sob sua responsabilidade também, criar, manter e priorizar a lista de tarefas do projeto.

É ele quem define as metas de trabalho do *Scrum Team* para cada *Sprint*. A equipe se compromete a executar o conjunto de atividades estabelecidas e o *Product Owner* se compromete a não trazer novos requisitos durante o *Sprint*.

É válido ressaltar que o *Product Owner* não é o chefe do *Scrum Team*, não tem poder de decisão sob aspectos técnicos e não pode impor datas e quantidades de itens que a equipe deve trabalhar em cada *Sprint*.

Para exercer o papel de *Product Owner*, as seguintes características são necessárias:

- Visão estratégica
- Criatividade
- Persuasão
- Comunicação
- Disponibilidade
- Conhecimento do Negócio

3.4.2 *Scrum Team*

É a equipe responsável pela execução direta das tarefas do projeto. Deve possuir entre cinco e nove componentes, com características multidisciplinares. Na equipe não haverá rótulos como analistas, arquitetos, coordenadores e gerentes. Todos são membros da equipe.

O *Scrum Team* é responsável pela definição das estimativas de tempo e esforços das tarefas, e devem manifestar os impedimentos ao *Scrum Master*.

Os membros da equipe não devem aguardar ordens. Devem tomar iniciativas para a conclusão das tarefas do projeto.

As seguintes características são necessárias:

- Ser auto-gerenciado
- Multidisciplinar
- Comprometido
- Comunicativo
- Resolvedor de Conflitos

3.4.3 *Scrum Master*

É um removedor de obstáculos. Sua função é proteger o *Scrum Team* de influencias externas. Assim a equipe tende a ter um caminho mais fácil para obter sucesso no *Sprint* definido.

Auxilia o *Product Owner* no processo de condução dos requisitos. É sua função também combater a ilusão do comando além de priorizar impedimentos e remove-los.

O *Scrum Master* não é chefe de ninguém, ele deve exercer um papel facilitador, moderador. É ele quem conduz todas as reuniões.

É importante que seja um profundo conhecedor do *Scrum* para que consiga contagiar a equipe com a utilização das boas praticas.

O *Scrum Master* deve ser:

- Responsável
- Comunicativo
- Humilde
- Facilitador
- Organizado
- Influente

3.5 Ferramentas

3.5.1 *Sprint Burndown*

Tem com objetivo mostrar o esforço restante para a conclusão do *Sprint*, bem como mostrar o quão próximo (ou distante) o *Scrum Team* se encontra de atingir a meta.

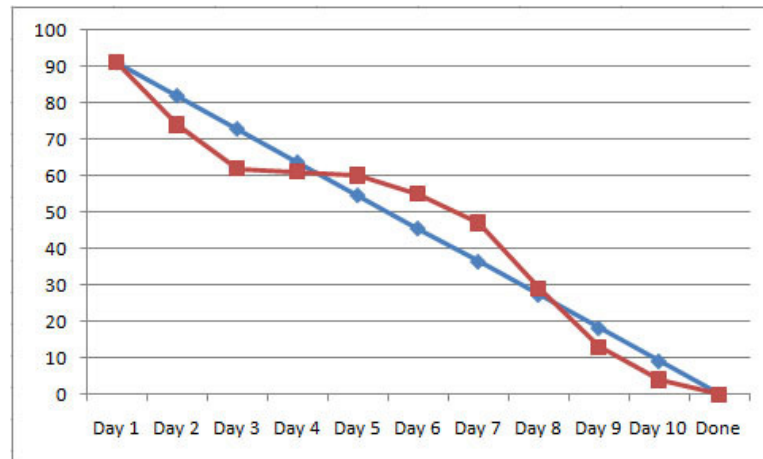


Figura 04 - *Sprint Burndown* - (GIRL WRITES CODE, 2009)

A coluna vertical representa a quantidade de esforço.

A coluna horizontal representa os dias de um *Sprint*.

A Linha Vermelha representa o Fluxo ideal de trabalho. Quando o gráfico está acima da linha vermelha significa que o time está distante de atingir a meta. Quando está em cima da linha vermelha significa que está no fluxo de trabalho ideal. E quando está abaixo da linha vermelha significa que o time está superando as expectativas.

O *Sprint Burndown* deve ser atualizado diariamente.

3.5.2 Quadro Kanban

É um quadro que visa auxiliar o acompanhamento da execução das tarefas.

Viabiliza a visibilidade e a comunicação entre os integrantes da equipe.

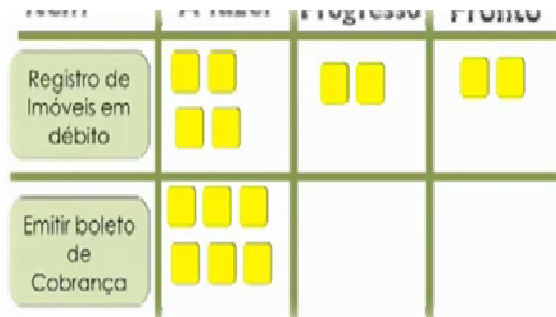


Figura 05 - Quadro Kanban - (SPARTA CONSULTORIA, 2009)

3.5.3 Planning Poker

“Cada membro da equipe recebe um baralho de 13 cartas como mostrado acima. Sempre que uma estória deve ser estimada, cada membro 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 simultaneamente. Dessa forma, cada membro da equipe é forçado a pensar por si próprio ao invés de basear-se na estimativa de outra pessoa.

Se houver uma grande divergência entre duas estimativas, a equipe discute as diferenças e tenta chegar a uma visão comum do trabalho envolvido na estória. Eles podem fazer algum tipo de decomposição de tarefas. Depois disso, a equipe faz novamente a estimativa. Esse processo é repetido até que as estimativas de tempo cheguem a uma convergência, isto é, todas as estimativas sejam aproximadamente a mesma para cada estória” (KNIBERG, 2008, p.45)



Figura 06 - Planning Poker - (SPRINT PLANNING, 2009)

4. APLICABILIDADE DO SCRUM NO TESTE DE SOFTWARE

4.1 Equipes Metodologia Tradicional x Equipes Metodologia Ágil

Na metodologia tradicional, as equipes são divididas por área. Cada área representa uma função distinta e não há uma integração entre as mesmas. No *Scrum* todas as equipes (Configuração/Instalação, Negócios, Teste, Desenvolvimento, entre outros) fazem parte de um mesmo Time.

Com a fusão das equipes no mesmo Time, o *Scrum* permite que todos trabalhem de forma integrada facilitando a comunicação, gerando um ambiente de parceria e um verdadeiro espírito de equipe na empresa. Desta forma, a equipe de Teste não é vista pela equipe de desenvolvimento como adversária, mas sim parceira.

Como as equipes são um único Time no *Scrum*, o conceito do que precisa ser entregue está pronto, pode ser diferente, pois um determinado item pode estar pronto dependendo do conceito de conclusão/término estabelecido pela equipe. O item pronto pode ser: Codificação concluída ou Codificação + Teste concluídos ou Codificação + Teste + Integração concluídas, ou Teste concluídos ou Codificação + Teste + Integração + Regressão. Por isso, é necessário que a equipe defina quando um item vai estar realmente pronto. Para cada item, a definição de pronto pode ser diferente, vai depender do que cada um se propõe.

4.2 Os papéis do Analista de Teste no Time *Scrum*

O Analista de Teste no *Scrum* assume vários papéis como Líder, Arquiteto e Automatizador. Em cada fase do projeto, ele “coloca o chapéu” necessário.

Um dos papéis que o Analista de Teste pode exercer, é o de *Product Owner* (PO). O PO é o responsável pela visão do produto, ou seja, a representação da sua necessidade, é o que deve ser satisfeito ao fim do projeto.

Para a definição dessa visão, o PO, colhe informações com os clientes, usuários final, time, gerentes, *stakeholders* e executivos.

O Analista de Teste participa da fase de definição da visão do produto contribuindo com as seguintes tarefas:

- Colaboração nas reuniões com o cliente para levantamento dos requisitos;
- Participação das reuniões de *Brainstorm*;
- Foco na visão da qualidade do produto;
- Revisa a abordagem de teste;
- Identifica as habilidades de teste necessárias para o projeto.

O *Product Owner* (PO) cria uma lista inicial de necessidades para que a visão do produto seja bem sucedida. Esta lista é chamada de *Product Backlog*. O Analista de Teste participa desta fase realizando as seguintes tarefas:

- Participa de discussões de arquitetura;
- Participa das definições de tecnologias utilizadas;
- Identifica as necessidades do ambiente de teste;
- Identifica restrições tecnológicas;
- Identifica ferramentas de teste necessárias para auxiliar na execução do projeto;
- Utiliza mapa mental (*Mind Map*) para auxiliar no entendimento das funcionalidades/requisitos;
- Identifica os “melhores” analistas de teste para o projeto em questão;
- Identifica a necessidade de suporte do time de suporte/operações.

Estimar é uma atividade do Time no *Scrum*. Como o Analista de Teste faz parte do Time, ele participa também desse processo. Além dele, o *Scrum Master* deve mediar o processo, o *Product Owner* deve estar disponível para esclarecer eventuais dúvidas.

Na primeira parte do *Planning Meeting* o PO define a meta da *Sprint* e divulga para o Time os itens prioritários do *Product Backlog*. O Time deve estimar os itens em tamanho e selecionar o que vai ser feito. O Analista de Teste participa dessa fase realizando as seguintes tarefas:

- Ajuda a garantir que os itens selecionados estão de acordo com a meta do projeto e teste;
- Analisa indicadores de desempenho;

- Revisa a estimativa, caso necessário;
- Gerencia os riscos encontrados;
- Tira dúvidas com o *Product Owner*;
- Define quais serão os tipos de teste (Sistema, Aceitação, Regressão) necessários.

Na segunda parte da *Planning Meeting* o time colhe mais detalhes dos itens do *Select Product Backlog* e os divide em tarefas gerando o *Sprint Backlog*. Após a decomposição, cada membro do Time escolhe as tarefas que deseja executar durante a *Sprint* e estimá-las. O Analista de Teste participa desta fase com a seguinte contribuição:

- Define o nível de regressão de teste de acordo com a priorização dos itens/tarefas do *Sprint Backlog*;
- Atualiza a matriz de teste por funcionalidade;
- Atualiza o mapa mental.

Após a definição da *Sprint*, o quadro de acompanhamento é preparado. Ele pode ser alterado de acordo com a sua realidade e necessidade.

Durante a execução da *Sprint* o *Scrum Master* através da sua capacidade de liderança e colaboração, facilita o trabalho do Time removendo os impedimentos encontrados e garantindo a boa aplicação do *Scrum*.

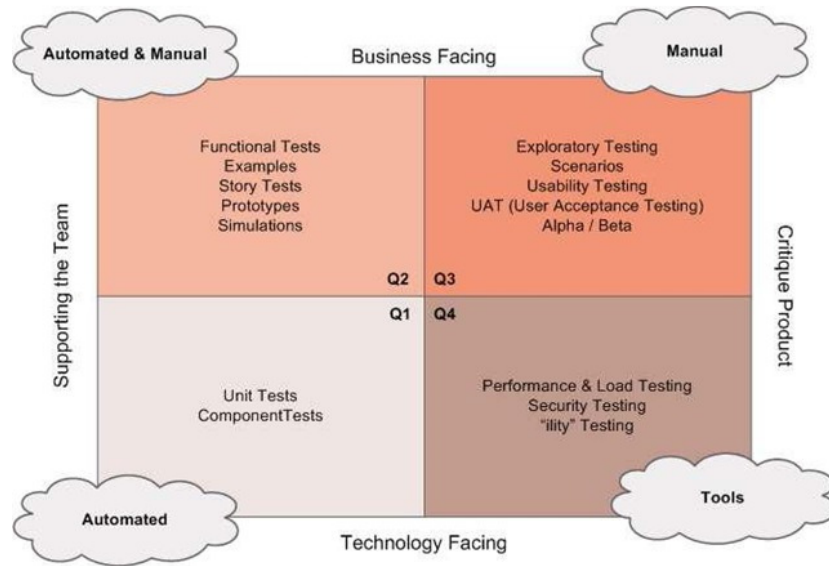


Figura 07 - Tipos de Teste - (DUONG, 2009)

- Q1 – São os testes que focam na arquitetura. A responsabilidade é dos desenvolvedores e os analistas de teste auxiliam na elaboração dos testes unitários automáticos;
- Q2 – São os testes que focam no negócio. A responsabilidade é dos analistas de teste em conjunto com outros envolvidos no projeto (clientes, usuários, etc.). Ajuda no entendimento das funcionalidades;
- Q3 – São os testes que focam no negócio e encontrar defeitos. A responsabilidade é dos analistas de teste;
- Q4 – São os testes que focam na arquitetura, estrutura do *software*. A responsabilidade é dos analistas de teste.

Durante a execução da *Sprint* o Analista de Teste possui o seguinte papel:

- Monta e configura o ambiente e infra-estrutura de teste;
- Executa testes;
- Automatiza casos e tarefas de teste;
- Auxilia os desenvolvedores na elaboração dos testes unitários automáticos;
- Evidencia os resultados;
- Acompanha os defeitos encontrados;

- Desenvolve novas habilidades;
- O Analista de Teste é a pessoa que aprova. Nada é considerado “pronto” em um *Sprint* até que ele diga que está;
- É responsável em ficar focado na meta da *Sprint*.

Na reunião diária (*daily Scrum*) de 15 minutos, o time ganha visibilidade de como está o caminho para a meta e planeja o dia seguinte de trabalho. O *Scrum Master* é o facilitador. O Analista de Teste também participa dessas reuniões contribuindo com as seguintes confecções:

- Relatório de teste atualizado;
- Evidência de teste atualizada;
- Gráfico S-Curve atualizado;
- Lista de defeitos atualizada;
- Lista de impedimentos atualizada;
- Quadro Kanban atualizado.

O Analista de Teste participa de outras duas reuniões. Reunião de Revisão, que possui o propósito de apresentar o que foi feito para o *Product Owner* e convidados e Reunião de Retrospectiva que possui o objetivo de representar o espírito de Inspeção – Adaptação dentro do *Scrum*.

Na reunião de Revisão, é realizada uma apresentação por todo o Time e o *Product Owner* avalia se a meta foi alcançada e faz anotações que podem ser transformar em novos itens para o *Product*.

A reunião de Retrospectiva é facilitada pelo *Scrum Master* onde são apresentadas as lições aprendidas. O Time avalia o que foi bom na última *Sprint*, o que deve ser melhorado e quem está no controle.

O Analista de Teste deve participar de todas as fases do *Scrum*, pois ele faz parte de um Time. Não existe Time de Teste ou Desenvolvimento no *Scrum*, mas sim, Time do *Scrum*.

5. ESTUDOS DE CASO

Para o nosso estudo de caso, realizamos entrevistas com profissionais da área de Teste de *Software* das empresas colaboradoras, com intuito de levantar como ocorreu o processo de implantação do *Scrum* e verificar a existência ou ausência das vantagens e desvantagens propostas em nosso trabalho.

Para idealização dos Estudos de Caso, enviamos para os respectivos colaboradores o questionário base que se encontra no item 9, Anexo.

O objetivo do questionário era visualizar a utilização do *Scrum* nas empresas. Após as respostas, novas comunicações foram realizadas através das ferramentas *Gtalk*, *Skype* e e-mails para refinar a pesquisa e concluir o entendimento.

5.1 Voice Technology

A Voice Technology é uma empresa de pequeno porte do ramo de desenvolvimento de *software*. Desenvolve sistemas para o mercado de Telecomunicações: URAs, Discadores, PABX Virtual, Sistemas de Conferência, entre outros.

O *Scrum* foi iniciado na empresa no final de 2008. O nível de envolvimento da empresa na implantação da metodologia foi baixo, pois a iniciativa surgiu de um gerente de projeto, e apenas alguns *Shareholders* se interessam em conhecer mais sobre o *framework*, e o mesmo não foi amplamente adotado pelos outros departamentos da empresa por comodismo e medo de mudança.

O primeiro projeto da empresa que ousou e adotou a metodologia, teve os seguintes benefícios:

Melhor alinhamento com a equipe de desenvolvimento, graças as participações nas reuniões de planejamento e *review*;

Maior entendimento sobre o sistema, o que ajudou na elaboração dos testes e melhorou o planejamento dos testes, como por exemplo, uma melhor seleção dos testes de regressão a serem realizados.

Diminuição do retrabalho, principalmente devido aos testes realizados durante o desenvolvimento, pois os mesmos davam um *feedback* mais rápido ao desenvolvedor, que poderia corrigir os defeitos, e assim a versão final, entregue ao time de teste era entregue com uma maior qualidade.

No segundo projeto da Voice Technology que adotou o *framework*, o principal benefício foi a efetividade dos testes, pois o profissional responsável pela maioria das tarefas de Teste *Software*, também participava ativamente das tarefas de desenvolvimento e se envolvia com os desenvolvedores e o *Product Owner*, e assim tinha uma visão completa tanto da parte técnica quando da parte de negócio. Podendo assim criar cenários de teste tanto de verificação quanto na validação do sistema.

Um ponto interessante do time de *Scrum*, é que as pessoas costumam exercer variados papéis durante o projeto, isso é muito bom para o time, mas é difícil para encaixar a pessoa em uma grade salarial.

A participação de todos os membros do time em algumas atividades de teste, como a execução, e a comunicação "frente-a-frente" diminuiu bastante a quantidade de retestes, comparando com outros projetos.

Em ambos projetos citados o *Scrum* não trouxe nenhuma influencia negativa. As experiências foram muito boas, e trouxeram ótimos resultados, tanto para as equipes como para os clientes. Um detalhe importante é que mesmo com os projetos não utilizando todas as práticas do *Scrum* e dos princípios ágeis, pudemos considerar a implantação do *Scrum*, como um sucesso, pois se encaixou bem nos projetos e com ele as equipes puderam fazer o seu melhor trabalho e buscar a melhoria contínua.

5.2 Ci&T

A Ci&T é uma empresa de médio porte do ramo de desenvolvimento de *software*. Desenvolve sistemas em .NET, Java e PHP.

O *Scrum* foi iniciado na empresa em meados de 2008. Para adoção da metodologia nos projetos depende de uma análise específica de cada projeto e de cada cliente. E quando adotada, abrange toda a área de desenvolvimento.

A empresa está buscando cada vez mais aplicar o conceito de *Lean It*, gerando mais valor para o cliente, com o menor custo (tempo e recursos). Com isso a empresa está fortemente envolvida em oferecer a metodologia para novos clientes, e aprimorar o processo a cada novo projeto.

Para a área de Teste da empresa, o *Scrum* significou uma grande melhoria na qualidade dos processos. As equipes ficaram mais integradas, os erros nos sistemas foram reduzidos.

5.3 Empresa A

A Empresa A, é uma empresa de grande porte e desenvolve sistemas customizados em C#.

O *Scrum* iniciou na empresa no final de 2008, e atualmente abrange a área de desenvolvimento da empresa.

A empresa apoiou com os recursos necessários, mas a implantação e execução foi de total responsabilidade da equipe. As equipes são compostas por 6 profissionais.

O Teste está envolvido na construção (TDD) e na documentação do *software*. Os artefatos de descrição dos casos de uso só são mantidos durante a construção, depois de implementados os testes passam a ser a documentação do sistema.

O grande benefício que o *Scrum* trouxe para a equipe de Teste de *Software* a possibilidade de melhorar as técnicas e os procedimentos por *Sprint*.

Como ponto negativo, como nem toda a equipe estava preparada pra trabalhar com *Scrum*, os primeiros *Sprints* foram mais demorados e tiveram mais re-trabalho.

O nome da empresa é fictício. O colaborador optou pela não divulgação do nome da empresa.

5.4 EMBRAS.net

A EMBRAS.net é uma empresa de médio porte que desenvolve nas linguagens Delphi e PHP o SIAP, um ERP voltado para Prefeituras, Câmaras e Autarquias Municipais.

O *Scrum* iniciou na empresa, há aproximadamente 09 meses, com a implantação de algumas práticas.

Hoje, abrange os departamentos de Desenvolvimento (e Manutenção) de *Software*, Qualidade (Teste, Homologação e Documentação) e também com participação do departamento de Suporte Técnico que prioriza as tarefas, fazendo assim o papel do *Product Owner*.

Antes da implantação do *Scrum*, houve uma resistência natural. Toda a equipe de desenvolvimento interna e empresas terceirizadas foram treinadas para que conhecessem as práticas da metodologia, com isso também foi passado o conhecimento para outros departamentos como o de Qualidade (Teste, Homologação e Documentação) e o de Suporte, para que pudessem participar do processo.

Atualmente, existem 4 times de *Scrum*, sendo 2 times internos e 2 times terceirizados, compostos em média por 5 integrantes cada.

O Teste se utiliza de *Sprints* sincronizados com o desenvolvimento, e tem a missão de homologar ou reprovar tudo que foi desenvolvido pelo desenvolvimento, antes de chegar ao cliente, dando um *feedback* para os desenvolvedores para que os mesmos possam corrigir o mais rápido possível.

Após a homologação, entra o trabalho do documentador que inicia a atualização dos documentos: descritivo do sistema (para fornecer ao departamento Comercial material sempre atualizado), novidades da versão (lista de melhorias e correções de *bugs* que foram realizados no *Sprint* de cada sistema homologado), *help* e manual do sistema (para auxílio ao usuário) e documento interno de treinamento (este documento contém todas as implementações concluídas no *Sprint* por sistema, e é utilizado para a realização do treinamento interno do departamento de Suporte, um dia antes da liberação dos sistemas para os clientes). Deste modo, quando o cliente recebe a nova versão do sistema, o Suporte já está preparado para ajudá-lo.

Antes do *Scrum*, o Teste funcionava de acordo com o modelo cascata. Não existia entrosamento entre o departamento de Qualidade (Teste, Homologação e Documentação) e o departamento de Desenvolvimento (e Manutenção) de *Software*, além disso a comunicação possuía muitas falhas.

Com o *Scrum*, os departamentos passaram a ter uma padronização nos eventos, uma sincronização entre as tarefas de cada equipe e uma interatividade muito maior. Assim, ganharam produtividade e qualidade, diminuíram situações emergenciais, além de terem criado um processo simples, que é conhecido e envolve toda a empresa.

O *Scrum* não influenciou negativamente em nenhum aspecto.

A metodologia não é utilizada em sua totalidade, mas são usadas boas práticas que atendem a necessidade da empresa.

Basicamente o *Scrum* organizou as entregas com a configuração dos *Sprints*, melhorou o foco de trabalho com o *Sprint Backlog* e tornou mais assertivas as entregas dos "pacotes" de *softwares* aos clientes.

6. VANTAGENS E DESVANTAGENS DO SCRUM

Todas as metodologias possuem vantagens e desvantagens e com o *Scrum* não seria diferente.

Uma das grandes vantagens da metodologia *Scrum* é a comunicação que existe entre todos os envolvidos da equipe, pois todos participam das decisões, fazendo com que a comunicação aconteça de forma clara e objetiva. Outro ponto positivo é a forma de se trabalhar com divisões de tarefas, isso faz com que todas as pessoas envolvidas, participem de todas as tomadas de decisões e resoluções de problemas, e vale ressaltar, que além da equipe de desenvolvimento, o cliente participa também de forma ativa desde o começo do projeto, o que colabora ainda mais com a propagação da informação.

A participação do cliente traz fortes benefícios e de certa forma se destaca também no *Scrum*, pois o contato com os desenvolvedores, além de permitir a divulgação do que será de fato produzido, colabora com o *feedback* do cliente junto a empresa, que acontece de forma rápida e eficaz, colaborando e muito com o seguimento do processo. Dessa forma, a organização estabelece uma relação de confiança e credibilidade junto ao seu cliente, pois, se todos participam com empenho, o resultado desse esforço com certeza gera produtos funcionais e com excelente qualidade, e a sincronia da equipe gera uma aceleração no tempo de desenvolvimento, e com isso se ganha tempo; e tempo é dinheiro, quando falamos de desenvolvimento de *software*.

O *Scrum* também se destaca por ter uma forma de trabalhar bem flexível e de fácil adaptação, podendo ser aplicada em diversos ambientes. Além do fato de trabalhar de forma transparente e eficiente, trabalha com constantes alterações de escopo. A forma como trabalhada, faz com que essas alterações não interferem de forma ativa nos custos do projeto, uma vez, que todas as mudanças são definidas e discutidas constantemente com o cliente, evitando surpresas na conclusão das tarefas. Isso porque o *Scrum* trabalha de forma totalmente diferenciado no quadro de atividades, ou seja, todo o processo é dividido em pequenos ciclos, chamados de *Sprint*, e caso venha ocorrer mudança de escopo ou até mesmo, mudanças na equipe de trabalho, a alteração e a inclusão de um novo membro na equipe acontece de forma dinâmica, uma vez que os pequenos ciclos tem um tempo curto de duração, isso,

facilita qualquer tipo de mudança que venha acontecer. Essa vantagem do *Scrum* faz com que, seja uma metodologia indicada para processo de desenvolvimento de *software* que constantemente são alteradas, e essas alterações não influencia de forma impeditiva na conclusão do processo.

Além dessas vantagens, é uma metodologia de fácil automatização, isso influencia diretamente em custos e prazos, e colabora de forma efetiva no controle das atividades a serem desenvolvidas. Devido ao efetivo controle, a participação do gerente de projeto, torna-se de suma importância, principalmente no gerenciamento de conflitos e resolução de quaisquer problemas que venha de certa forma, atrapalhar ou até mesmo impedir a conclusão do processo, ou até mesmo de alguma atividade.

Devido ao fato do *Scrum* suportar diversas mudanças, essa vantagem, pode ser vista de certa forma, como uma desvantagem, isso porque, para suportar mudanças constantes, é necessário trabalhar de forma bem exigente, tendo a frente um gerente de postura firme, para tomar todas as providências que venha ser necessária para suprir os obstáculos que a equipe venha encontrar. Sabemos o quão isso é difícil, pois gestão de pessoas acaba sendo classificado pelas organizações, como ponto de grande dificuldade, ainda mais na metodologia *Scrum*, onde, cada membro da equipe tem liberdade de expressão, isso pode causar certos desentendimentos entre os gestores, dessa forma, deve-se, gerenciar com muita autoridade de decisão e flexibilidade, sendo capaz de adaptar e aceitar bem as mudanças, e manter o controle de todos os conflitos, respeitando as opiniões de todos os envolvidos, além de garantir o entrosamento da equipe, inclusive, permitir que os todos os membros sejam autônomos o suficiente em suas tarefas, a ponto de terem em alguns momentos liberdade para tomar suas próprias decisões. Para que isso acontece de forma tranquila o gerente, no caso o *Scrum Master* deve confiar em sua equipe, pois, ser rigoroso demais, pode influenciar e muito no fracasso do projeto. Dessa forma, exige um acompanhamento bem rigoroso do gerente tanto qualitativamente como quantitativamente.

Para se trabalhar de forma adequada no *Scrum*, é necessário que se tenha mão-de-obra especializada e não simplesmente, micro gestão. E mão-de-obra especializada custa caro.

Além disso, não podemos deixar de dizer que se trata de uma metodologia inovadora e flexível, causando nas pessoas certa resistência ao novo.

Outro ponto, considerado negativo, é a informalidade que o *Scrum* trabalha a documentação, pois não tem a preocupação em documentar, o foco que prevalece, é o que será desenvolvido.

Como no *Scrum* se permite mudanças de escopo, isso pode fazer com que, gere projetos duradouros, já que no início do ciclo de desenvolvimento não temos a visibilidade concreta da finalização do projeto. Uma vez que, a definição das tarefas, quando não definidas adequadamente, gera dificuldade para estimar o custo e o andamento do projeto, já que o tempo não é fator principal e sim a conclusão da tarefa.

A forma de trabalhar com equipes pequenas (7 a 9 pessoas), pode também ser vista como desvantagem, pois, pode trazer alguns problemas quando essa equipe vai crescendo e precisa ser novamente dividida, essa nova divisão, pode fazer com que haja uma perda do foco entre os membros da equipe, porém, trabalhar com equipe grande pode causar inúmeras falhas, dessa forma, a divisão pode ser inevitável.

Além de participação, é necessário que o cliente tenha uma visão bem clara do que se deseja, pois a instabilidade pode fazer com que, o projeto não tenha sucesso, portanto o perfil do cliente é de suma importância e nem sempre isso pode ser alcançado facilmente.

7. CONCLUSÃO

Um fator que contribui para o sucesso do *Scrum* é a maneira como se trabalha em equipe, dividindo todos os recursos, em pequenos grupos, sendo sempre equipes pequenas, isso faz com que todos sejam mais participativos e se empenhem melhor em suas atividades. Com a participação ativa de todos, o andamento do processo acontece de forma clara, que de certa forma, gera também, aumento de produtividade.

Devido à flexibilidade, a organização pode ter que realizar alterações na sua estrutura organizacional, para melhor adequar ao funcionamento do *Scrum* já que, a metodologia exige certo nível de formação de todos os envolvidos.

Com a participação do Analista de Teste em um Time do *Scrum*, podemos identificar os seguintes benefícios:

- Integração do time;
- Apoio de quem está desenvolvendo código durante a execução dos testes;
- Apoio de quem está testando código durante a codificação;
- Participação mais direta e ativa do profissional que está testando o *software*;
- Profissionais que estão desenvolvendo código interessados em aprender sobre teste;
- Profissionais que estão testando código interessados em aprender sobre programação;
- Agilidade, interação com testes;
- Acompanhamento de defeitos pelo profissional que está testando o *software*;
- Analistas de Teste deixaram de ser reativos para serem pró-ativos.

Apesar da resistência inicial, no geral as empresas que aderem à utilização de um Processo Ágil como o *Scrum* tem boas experiências com ótimos resultados.

A equipe trabalha mais unida, a qualidade e o retrabalharam reduzem consideravelmente.

8. REFERÊNCIA BIBLIOGRÁFICA

- ALMEIDA, Felipe Rodrigues de. Criando um processo ágil para desenvolvimento de software. Disponível em: http://d.yimg.com/kq/groups/20097703/189506039/name/Como_criar_um_processo_agil.pdf. Acesso em: 01 dez. 2009.
- BECK, Kent. Extreme programming explained: embrace change. Ed. Pearson, EUA, 2000.
- BECK, Kent. Test-Driven Development. The Addison-Wesley Signature Series, 2003.
- BECK, Kent. Test-Driven Development: by Example. The Addison-Wesley Signature Series, 2003.
- CLARIFICATION. Rugdy Scrum. Disponível em: <<http://clarification.files.wordpress.com/2007/09/Scrum.gif>>. Acesso em: 01 dez. 2009.
- COHN, Mike. Succeeding with Agile: Software Development Using Scrum. The Addison-Wesley Signature Series, 2009.
- COHN, Mike. User Stories Applied: For Agile Software Development. The Addison-Wesley Signature Series, 2004.
- CRISPIN, Lisa; GREGORY, Janet. Agile Testing: A Practical Guide for Testers and Agile Team. The Addison-Wesley Signature Series, 2009.
- DERBY, Esther; LARSEN, Diana; SCHWABER, Ken. Agile Retrospectives: Making Good Teams Great. Pragmatic Bookshelf, 2006.
- DUONG, Luu. Functional Testing Tools. Disponível em: <http://luuduong.com/blog/>. Acesso em: 01 out. 2009.
- FUSCO, Camila. Scrum, A Nova Gestão de Projetos, mar, 2008. Disponível em: <http://governanca.wordpress.com/2008/03/18/Scrum-a-nova-gestao-deprojetos>. Acesso em 27/01/2009.
- GIRL WRITES CODE. Sprint Burndown. Disponível em: <http://www.invisible-city.com/sharon/uploaded_images/SampleBurndownChart.jpg>. Acesso em: 01 dez. 2009.
- GOMES, Handerson. Porque usar Scrum, Nov, 2008. Disponível em: <http://webinsider.uol.com.br/index.php/2008/11/06/porque-usar-Scrum/>. Acesso em 27/01/2009.
- KNIBERG Henrik, Scrum and XP from the Trenches, Ed. C4Media, EUA, 2007.
- KNIBERG, Henrik. Scrum and XP from the Trenches: How we do Scrum. Enterprise Software Development Series, 2007.

KNIBERG, Henrik. Scrum e XP direto das trincheiras: Como nós fazemos Scrum. In: KNIBERG, Henrik. Scrum e XP direto das trincheiras. Eua: Infoq, 2008. p. 45.

Metodologia de Desenvolvimento Ágil Scrum: O Caso XMPM. Manaus: Benq Mobile, 2006. p. 11 - 11.

MORAES, Rodrigo da Silveira. Metodologia de Desenvolvimento Ágil de Software

PRIOR David, Scrum Meets Waterfall, mar, 2008. Disponível em: <http://www.Scrumalliance.org/articles/93-Scrum-meets-waterfall>. Acesso em 29/01/2009

QUEZADA, Gustavo. Papel do “Time de Teste” em Projetos SCRUM. Disponível em: <http://gustavoquezada.blogspot.com/>. Acesso em: 01 out. 2009.

SATO, Danilo; GOLDMAN, Alfredo. Desenvolvimento de Software Lean: Curso de Verão 2007 - IME/USP. Disponível em: <<http://www.agilcoop.org.br/>>. Acesso em: 01 out. 2009.

SCHWABER, Ken; BEEDLE, Mike. Agile Software Development With Scrum, Ed. Addison Wesley, EUA, 2001.

SCRUM – Abordagem Teórica e Exemplos de Aplicação. 2008. Monografia de Pós Graduação em projeto e Desenvolvimento de Sistemas de Informação, UNIFIEO, Osasco, 2009.

SCRUM TRAINING & AGILE TRAINING FROM SCRUMMASTER MIKE COHN. An Introduction to Scrum. Disponível em: <http://www.mountangoatsoftware.com/system/asset/file/58/RedistributableIntroToScrum.ppt>. Acesso em: 01 dez. 2009.

SOMMERVILLE, Ian. Engenharia de Software, São Paulo: Ed. Pearson, 2007.

SPARTA CONSULTORIA. Ciclo de Vida do Scrum. Disponível em: <<http://www.spartaconsultoria.com.br/home/index.php/video-aulas/5-ciclo-do-Scrum>>. Acesso em: 01 dez. 2009.

SPRINT PLANNING. Planning Poker Cards. Disponível em: <http://www.sprintplanning.com/planningpoker_cards_1.gif>. Acesso em: 01 dez. 2009.

SURAVARAPU, Srinivas, Implementing Scrum: 8 Questions to Answer Before You Begin, mar, 2008. Disponível em: <http://www.Scrumalliance.org/articles/99-implementing-Scrum--questions-to-answer-before-you-begin>. Acesso em 29/01/2009

TELES, Vinícius Manhães. Extreme Programming: Aprenda como encantar seus usuários desenvolvendo software com agilidade e alta qualidade. São Paulo: Novatec, 2004.

THAMIEL, Thiago. Ciclo de Vida do Scrum. Disponível em: <<http://thiagothamiel.files.wordpress.com/2009/07/Scrum.jpg>>. Acesso em: 01 dez. 2009.

The Agile Alliance - Manifesto for Agile Software Development. Disponível em: <http://agilemanifesto.org/>. Acesso em 21/11/2008

WATTS, Geoff, Production Support and Scrum, mar, 2008. Disponível em: <http://www.Scrumalliance.org/articles/91-production-support-and-Scrum>. Acesso em 29/01/2009

Wells, Don. Extreme Programming: A Gentle Introduction, fev, 2006. Disponível em: <http://www.extremeprogramming.org/index.html>. Acesso em: 17 out. 2009

9. ANEXO

QUESTIONÁRIO BASE

SOBRE A SUA EMPRESA:

1. Existe alguma restrição em revelar o nome da empresa?
2. Qual o porte da empresa?
3. Qual a linguagem de programação?
4. Qual o tipo de *software* desenvolvido?
5. Qual o seu cargo e formação?

SOBRE O *SCRUM*:

1. Quando o *Scrum* iniciou na sua empresa?
2. Qual a abrangência do *Scrum* (quais departamentos)?
3. Qual o nível de envolvimento da empresa na implantação do *Scrum*?
4. Qual o tamanho da equipe do *Scrum*?
5. Qual é o papel do teste no *Scrum* implantado na sua empresa?
6. Quais os benefícios o *Scrum* trouxe para o teste?
7. O *Scrum* influenciou negativamente em algum aspecto?
8. Já existe algum case de sucesso do *Scrum* na sua empresa?