FACULDADE DE TECNOLOGIA DE SÃO PAULO

Tecnologia em Processamento de Dados

MARCUS VINÍCIUS AUGUSTO

DESENVOLVIMENTO DE SOFTWARE

COM APOIO DE PRÁTICAS SCRUM

São Paulo – SP

2011

MARCUS VINÍCIUS AUGUSTO – R.A. 052921-4

DESENVOLVIMENTO DE SOFTWARE

COM APOIO DE PRÁTICAS SCRUM

Monografia apresentada ao curso de Processamento de Dados como requisito parcial para a obtenção do título de Tecnólogo em Processamento de Dados.

Orientadora: Prof. Me. Vânia Franciscon Vieira

São Paulo – SP

2011

Para minha família,

Ovídio, Marília e Diego

AGRADECIMENTOS

Primeiramente agradeço a Deus por me sustentar em cada ciclo e sempre me auxiliar, especialmente nesta etapa.

Aos meus pais Ovídio e Marília, meu irmão Diego e minha namorada Tainá, por acreditarem em mim, mostrar-me a importância em continuar e pelo apoio que me dão.

A orientadora: Prof. Me. Vânia Franciscon Vieira, pela orientação, varias correções, propostas e auxilio na elaboração desta monografia para que ela ficasse melhor possível.

A Faculdade de Tecnologia de São Paulo como um todo, pelo empenho do corpo docente na formação acadêmica, profissional e pessoal do individuo.

Aos meus amigos e amigas que, de forma direta ou indireta, contribuíram para que hoje estivesse aqui realizando este trabalho.

Muito Obrigado!

RESUMO

Este estudo busca o conhecimento de diversas fontes bibliográficas e compila conceitos fundamentais sobre práticas SCRUM, respondendo a questões como: o que é, para que serve, como funciona, como utilizar, utilizar ou não, vantagens e possíveis desvantagens, visando proporcionar ao leitor a capacidade de decidir se é viável ou não adotar SCRUM em uma especifica necessidade.

O publico alvo deste estudo são profissionais que desenvolvem software e não usam metodologia ou usam uma metodologia que não está funcionando, este estudo aborda o tema de forma prática e focando a área de desenvolvimento, pois é onde o SCRUM se aplica mais efetivamente, também citará o SCRUM em outras partes do processo de desenvolvimento de software, mas não tão incisivamente.

Digamos que a intenção deste estudo é apresentar o SCRUM como opção para melhorar o processo de desenvolvimento de software e garantir uma melhor qualidade do produto final, abordando de forma que possa servir como um manual de consulta, um pequeno guia funcional, prático, objetivo, que possa ser absorvido com uma boa curva de aprendizado, mostrando como as práticas SCRUM podem ser positivas no desenvolvimento de software.

Palavras Chave: Scrum, Ágil, Metodologia, Empírico.

ABSTRACT

This study seeks knowledge from various literature sources and compiles basic concepts of Scrum practices, answering questions like: what is, what it does, how it works, how to use, or not use, advantages and possible disadvantages, in order to provide the reader with the ability to decide whether it is feasible or not adopt a specific need in SCRUM.

The target audience of this study are professionals who develop software and do not use methods or use a methodology that is not working, this paper addresses the topic in a practical, focusing on the development area, where it is most effectively apply SCRUM also cite SCRUM in other parts of the software development process, but not as sharply.

Let's say that the intent of this study is show SCRUM as an option to improve the software development process and ensure a better quality of the final product, covering so that it can serve as a reference manual, a pocket guide, functional, practical, objective that can be absorbed with a good learning curve, showing how practices can be positive in SCRUM software development.

Keywords: Scrum, Agile, Methodology, Empirical.

LISTA DE ILUSTRAÇÕES

LISTA DE ABREVIATURAS E SIGLAS

CMMI - Capability Maturity Model Integration

ISO - International Organization for Standardization

OOPSLA - Object-Oriented Programming, Systems, Languages & Applications conference

PDCA – Plan, Do, Check, Act

PO – Product Owner

ROI - Return on Investment

SUMÁRIO

1. INTRODUÇÃO

É perceptível que a inclusão e a simples presença de boas práticas ou metodologias podem ajudar o processo de desenvolvimento de software e sua gestão.

No mercado há uma deficiência na efetivação de boas práticas de apoio na hora do desenvolvimento de softwares e na gestão deste desenvolvimento, independente se por equipes boas com profissionais treinados, diplomados, certificados na área de atuação, recursos humanos capazes e capacitados, assim como em recursos humanos não preparados ou empresas estruturadas e não estruturadas, infra-estrutura de hardware e software compatíveis com o que se precisa para desenvolver a solução e mesmo assim, em diversos ambientes há dificuldades desproporcionais durante todo o processo de desenvolvimento.

Prazos estourados, equipe desmotivada, produtividade baixa, falta de parâmetros, retrabalho, stress, fazem parte do cotidiano e da cultura de diversos ambientes de desenvolvimento.

O manifesto ágil que surgiu em 2011 tem a visão de conseguir amenizar alguns destes transtornos que ocorrem. Devido ao seu foco nos indivíduos e a forma iterativa como é executado, isso transforma o processo de desenvolvimento da solução que o cliente comprou e faz com que o desenvolvimento seja extremamente vantajoso para todos que participam do processo. Conseguir isso investindo em conhecimento: melhor ainda. Motivando o time, abrindo o raciocínio para uma nova maneira de pensar, de fazer, de agir. Pode ser, e realmente é, uma solução viável, vantajosa, competitivamente falando produtiva e lucrativa.

2. OBJETIVOS

O objetivo é apresentar um cenário geral do SCRUM, demonstrar os procedimentos realizados pelas práticas Scrum, com a preocupação de verificar sua importância no atual cenário de gestão de projeto de software.

3. PROCESSOS EMPÍRICOS

Controlar um processo empiricamente é como aprender dirigir um carro: não precisa traçar um destino inicialmente e chegar em linha reta até o final. Aprender a

dirigir está muito mais relacionado com pequenas correções de rota até a chegada final, partindo do principio que nem todas as características do produto são conhecidas na análise e que provavelmente os requisitos mudarão com o passar do tempo.

É típico adotar a abordagem de modelagem definida (teórica) quando os mecanismos subjacentes pelos quais um processo opera é razoavelmente bem entendido. Quando o processo é muito complexo para ser definido, a abordagem empírica é a escolha apropriada (Babatunde A. Ogunnaike, W. Harmon Ray, 1992)

Em processos empíricos os processos são complexos ou com muita incerteza em que seus detalhes ainda são desconhecidos, é difícil estimar tempos de execução ou mesmo definir atividades a serem realizadas na gestão.

Tomemos como exemplo para ilustrar melhor,s a figura 1, um processo empírico usado na metodologia PDCA – Plan, Do, Check, Act, que é focada em resolver problemas, identificando a causa raiz e a melhor solução:

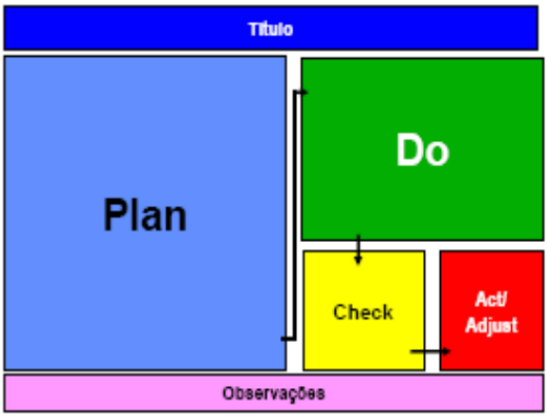

Figura 1 – Fluxo PDCA formato A3

O ciclo PDCA, conforme figura 2, pode ser todo incluído em uma folha A3 sendo que 80% do tempo é utilizado na compreensão do problema.

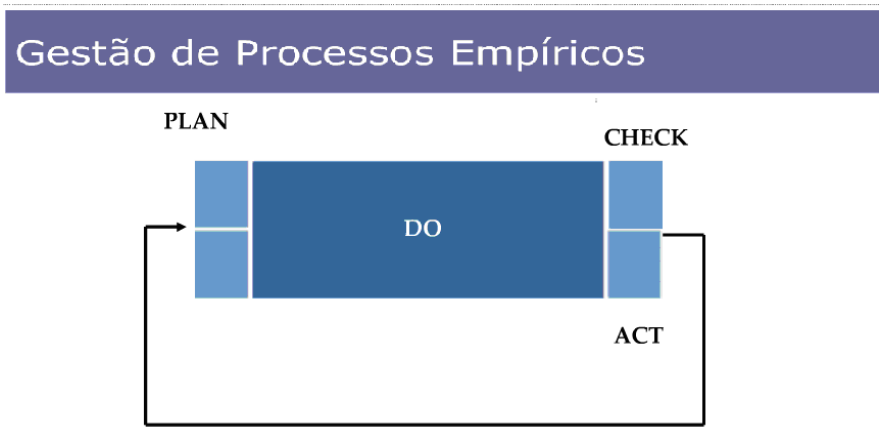

Figura 2 – Fluxo PDCA

Agora veja analogamente, na figura 3, como Sprints Scrum funcionam:

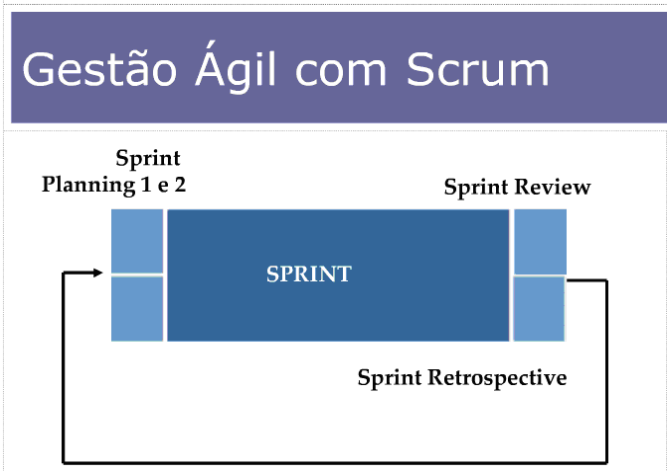

Figura 3 – Fluxo de um Sprint Scrum

Analogia entre SCRUM e PDCA

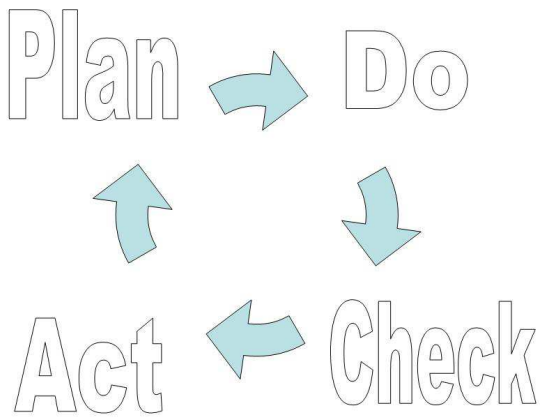

Figura 4 – Ciclo PDCA

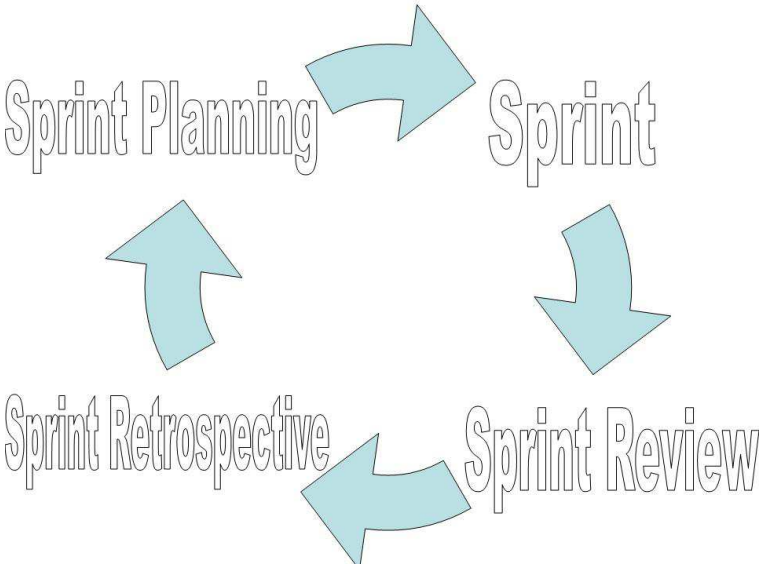

Figura 5 – Scrum baseado em PDCA

4. METODOLOGIAS ÁGEIS

Para ilustrar melhor o que é Scrum é necessário abordar pontos da metodologia ágil. Metodologias ágeis têm sido apontadas como uma alternativa às abordagens tradicionais para o desenvolvimento de software. As metodologias tradicionais, conhecidas também como pesadas ou orientadas a planejamentos, devem ser aplicadas apenas em situações em que os requisitos do sistema são estáveis e requisitos futuros são previsíveis. Entretanto, em projetos em que há muitas mudanças, em que os requisitos são passíveis de alterações, onde refazer partes do código não é uma atividade que apresenta alto custo, as equipes são pequenas, as datas de entrega do software são curtas e o desenvolvimento rápido é fundamental, não pode haver requisitos estáticos, necessitando então de metodologias ágeis. Além disso, o ambiente das organizações é dinâmico, não permitindo então que os requisitos sejam estáticos.

Processos orientados a documentação para o desenvolvimento de software são, de certa forma, fatores limitadores aos desenvolvedores e muitas organizações não possuem recursos ou inclinação para processos pesados de produção de software. Por esta razão, as organizações pequenas acabam por não usar nenhum processo. Isto pode levar a efeitos desastrosos na qualidade do produto final, além de dificultar a entrega do software nos prazos e custos pré-definidos. Em particular, o modelo Clássico ou Seqüencial será apresentado como exemplo de metodologia tradicional.

O termo Metodologias Ágeis tornou-se popular em 2001 quando dezessete especialistas em processos de desenvolvimento de software representando os métodos Scrum (Schwaber, 2002), Extreme Programming (XP) (Kent Beck, 1999) e outros, estabeleceram princípios comuns compartilhados por todos esses métodos. Foi então criada a Aliança Ágil e o estabelecimento do Manifesto Ágil (Agile Manifesto, 2004).

Os conceitos chave do Manifesto Ágil são:

- Indivíduos e interações ao invés de processos e ferramentas.
- Software executável ao invés de documentação.
- Colaboração do cliente ao invés de negociação de contratos.
- Respostas rápidas a mudanças ao invés de seguir planos.

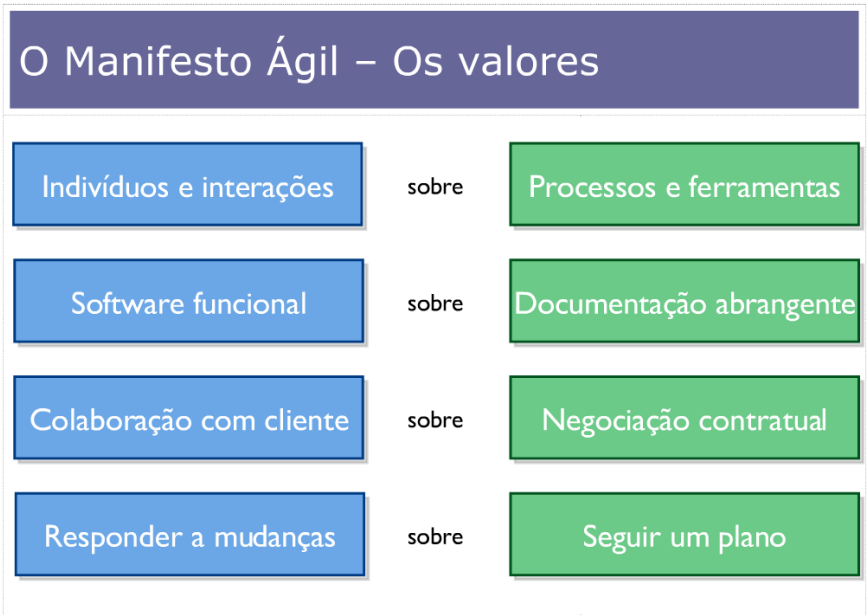

Figura 6 – Valores do manifesto Ágil

O Manifesto Ágil não rejeita os processos e ferramentas, a documentação, a negociação de contratos ou o planejamento, mas simplesmente mostra que eles têm importância secundária quando comparado com os indivíduos e interações, com o software estar executável, com a colaboração do cliente e as respostas rápidas a mudanças e alterações. Esses conceitos aproximam-se melhor com a forma que pequenas e médias organizações trabalham e respondem a mudanças.

A maioria das metodologias ágeis nada possuem de novo. O que as diferencia das metodologias tradicionais são o enfoque e os valores. A idéia das metodologias ágeis é o enfoque nas pessoas e não em processos ou algoritmos. Além disso, existe a preocupação de gastar menos tempo com documentação e mais com a implementação.

Uma característica das metodologias ágeis é que elas são adaptativas ao invés de serem preditivas. Com isso, elas se adaptam a novos fatores decorrentes do desenvolvimento do projeto, ao invés de procurar analisar previamente tudo o que pode acontecer no decorrer do desenvolvimento. Essa análise prévia é difícil e apresenta alto custo, além de tornar-se um problema caso não se queira fazer alterações nos planejamentos. Por exemplo, para seguir estritamente o planejamento, pode ser necessário que a equipe trabalhe sobre pressão e faça muitas horas extras, o que prejudica a qualidade do software.

5. SCRUM

O Scrum foi formalmente apresentado e publicado no OOPSLA 1995, durante os cinco anos seguintes, Mike Beadle e Martine Devos fizeram contribuições significativas e posteriormente outros grandes nomes do mundo do software, tais como: Kent Beck, Jim Highsmith, Alistair Cockburn, Martin Fowler, Ken Shwaber e Jeff Sutherland.

Em 2001, esse grupo de profissionais veteranos na área de software decidiu se reunir em uma estação de esqui, nos EUA, para discutir formas de melhorar o desempenho de seus projetos.

Embora cada envolvido tivesse suas próprias práticas e teorias sobre como fazer um projeto de software ter sucesso, cada qual com as suas particularidades, todos concordavam que, em suas experiências prévias, um pequeno conjunto de princípios sempre parecia ter sido respeitado quando os projetos davam certo.

Scrum não é um processo ou uma técnica para o desenvolvimento de produtos. Ao invés disso, é um framework dentro do qual se pode empregar diversos processos e técnicas. O papel do Scrum é fazer transparecer a eficácia relativa das suas práticas de desenvolvimento para que se possa melhorá-las, enquanto provê um framework dentro do qual produtos complexos podem ser desenvolvidos, é fundamentado na teoria de controle de processos empíricos, emprega uma abordagem iterativa e incremental para aperfeiçoar a previsibilidade e controlar riscos. Três pilares sustentam qualquer implementação de controle de processos empíricos.

Equipes Scrum são pequenas, figura 7, com aproximadamente sete integrantes e com autogestão.

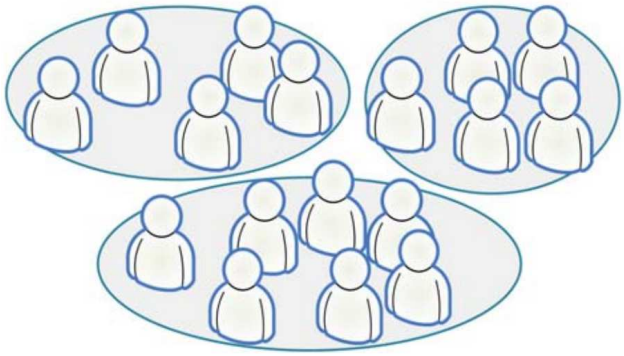

Figura 7 – Grupos Scrum

O trabalho é todo dividido em uma pequena lista de pequenas tarefas concretas e mensuráveis, ordene a lista por prioridades e estime o esforço relativo para cada item.

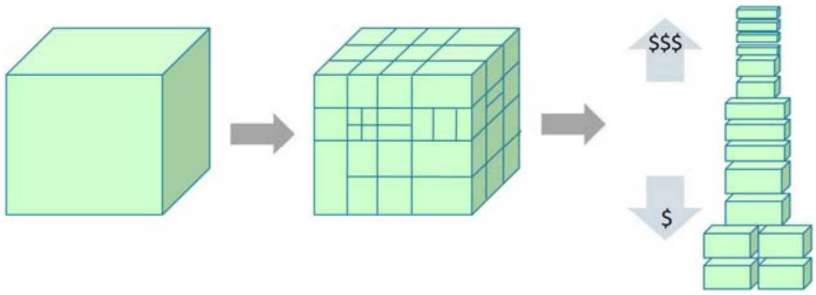

Figura 8 – Divisão do problema

O tempo é dividido em iterações, figura 9, de pequenas datas fixas, usualmente de 1 a 4 semanas, com pedaços de software testados e funcionando ao final de cada iteração.

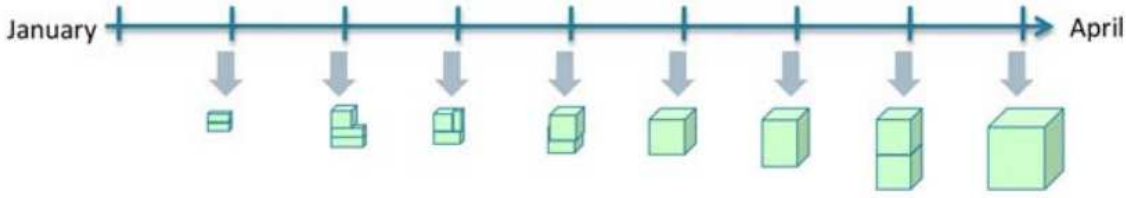

Figura 9 – Divisão das iterações

A melhor forma de ser ágil é construir somente o que o cliente valoriza e não mais que isto. O excesso de formalidade pode limitar o progresso do projeto, mas por outro lado, o caos total, sem a utilização de processos pode impedir que se alcancem os objetivos do projeto. Scrum permite criar produtos melhor adaptados à realidade do cliente de forma ágil. Além do mais, adotar Scrum nos projetos traz grandes benefícios para a equipe como comprometimento, motivação, colaboração, integração e compartilhamento de conhecimento, o que facilita em muito o gerenciamento e sucesso dos projetos.

5.1. ORIGEM DO SCRUM

O Scrum foi criado no início da década de 1990 por Jeff Sutherland, PhD e Ken Schwaber, nos Estados Unidos, trata-se de um conjunto de práticas relativamente novas, largamente usadas no desenvolvimento de software e que estão apresentando resultados satisfatórios em relação às metodologias tradicionais, Scrum pode ser usado no desenvolvimento de outros produtos e gerenciamento de qualquer outro trabalho.

Scrum engloba uma maneira diferente de pensar, planejar, ordenar, agir, fazer, dialogar, revisar... Por se tratar de uma maneira nova de lidar com o desenvolvimento de software, prioriza atitudes que são cruciais para o sucesso do projeto, Scrum se atenta a pequenos detalhes que vão desde trazer o cliente para mais perto do projeto até organizar os sonhos deste cliente em post-it colados em uma lousa para que sejam avaliados, priorizados, mensurados e ordenados, detalhes que aparentemente são caprichos, mas que durante o desenvolvimento da solução que o cliente comprou para o seu negócio faz muita diferença.

Práticas Scrum são relativamente novas no desenvolvimento de software e estão apresentando resultados satisfatórios em relação às metodologias não tão novas, o fato é que Scrum é uma maneira de pensar, uma filosofia diferente, onde as mesmas coisas são feitas de maneira diferente é bastante objetivo, com papéis bem definidos, de fácil adaptação.

5.2. CONCEITO SCRUM

Scrum é bastante objetivo, com papeis bem definidos, de fácil adaptação e sua curva de aprendizado é relativamente baixa.

Segundo o conceituado professor, consultor e um dos papas da administração moderna, Drucker (1998), o que não se pode medir não se pode gerenciar. Seu raciocínio traduz bem a necessidade, cada vez maior, de que os atuais gestores de Tecnologia da Informação têm de se servir de metodologias e indicadores que lhes permitam estabelecer objetivos, monitorar os resultados e verificar, de forma objetiva, como e se as metas propostas foram atingidas.

A experiência tem mostrado que os antigos manuais de procedimentos utilizados no passado já não atendem mais aos requisitos das empresas.

O turbulento ambiente empresarial, que se apóia na tecnologia e vive em constante mutação, exige formas mais ágeis e flexíveis de gerenciamento.

Processos orientados a documentação para o desenvolvimento de software são, de certa forma, fatores limitadores aos desenvolvedores e muitas organizações não possuem recursos ou inclinação para processos pesados de produção de software. Por esta razão, as organizações pequenas acabam por não usar nenhum processo. Isto pode levar a efeitos desastrosos na qualidade do produto final, além de dificultar a entrega do software nos prazos e custos predefinidos. (PRESSMAN, 2007)

As metodologias ágeis surgiram com a proposta de aumentar o enfoque nas pessoas e não nos processos de desenvolvimento. Além disso, existe a preocupação de gastar menos tempo com documentação e mais com resolução de problemas de forma iterativa.

5.3. CARACTERISTICAS SCRUM

O Scrum é um framework iterativo e incremental para desenvolvimento de produtos e gerenciamento de qualquer trabalho, seu principal objetivo é entregar funcionalidades com o mais alto valor de negócio para o cliente. Dentre as características desta metodologia destacam-se por ser um processo ágil para o gerenciamento e controle de projetos que envolve práticas de engenharia existentes, além de abordar o desenvolvimento de sistemas e produtos onde os requisitos sofrem constantes mudanças, Scrum visa otimizar a comunicação do time e favorecer a cooperação é escalável para pequenos projetos e grandes corporações, substitui o gerenciamento empírico e processos de controle, por feedback em l*oops* de inspeção e adaptação, distribui funcionalidades em Sprints de 30 dias, é escalável para projetos longos, largos e distribuídos e Suporta CMMI Nível 3 e ISO9001 (FUTADO, 2007)

6. PERSONAGENS DO SCRUM

6.1. Product Owner

O Product Owner (PO), é a voz que representa o cliente e usuários, é responsável por definir os itens que irão compor o Product Backlog e priorizar as funcionalidades que irão trazer maior valor de negócio ao produto (ROI) para o cliente através da Sprint Planing, alem de aceitar e mensurar o quanto foi entregue do produto o final de cada Sprint. É ele quem vai levantar os requisitos, planejar releases e clarear ao máximo a visão do time de desenvolvimento sobre os reais objetivos do software.

O PO deve fazer a ligação direta entre a visão do software pelos conceitos e entendimentos que o cliente possui e o conhecimento que o time precisa ter para realizar as atividades necessárias, objetivando o resultado que o produto final deve atingir. Vale lembrar que, para maximizar o ROI, o PO abstrai os principais valores que irão dar retorno imediato ao cliente. O time de desenvolvimento vai, preferencialmente, trabalhar os itens de maior retorno de investimento no início do projeto e não no meio da construção do mesmo ou no final.

Uma habilidade importantíssima que o PO deve ter é, além de ser uma pessoa extremamente acessível, ter uma ótima comunicação para com todos os stakeholders. E ao mesmo tempo, esta comunicação deve ser constante. Se o projeto não possui um PO com as habilidades necessárias, certamente vai ocorrer em seu andamento problemas de inter-relacionamento entre os stakeholders e isso compromete 100% do possível sucesso do mesmo. Muitas vezes é discutido na comunidade Scrum o porquê do PO levar o status de “chato” da equipe. É muito simples de responder: por ter sempre o foco voltado na entrega do produto funcional. No entanto, ele deve saber lidar com essa responsabilidade, fazendo com que ela não seja um ponto negativo no processo. E este é um grande desafio. (www.ges.blog.br, 2010)

6.2. Scrum Master

O Scrum master cuida para que o time possua condições de executar suas atividades. Atua como um facilitador do Daily e torna-se responsável por remover obstáculos que sejam levantados pela equipe durante as reuniões. O papel do Scrum master é tipicamente exercido por um gerente de projeto ou um líder técnico, mas em

principio pode ser qualquer pessoa da equipe é responsável pelo sprint, fazendo reuniões diárias com o time para rastrear e eliminar "blocks" que prejudicam o desenvolvimento das atividades. o Scrum Master não é o melhor técnico, ele é melhor com pessoas.

É responsável por garantir que a equipe Scrum se adere aos valores do Scrum, práticas e regras. O Scrum Master ajuda o Time a adotar Scrum, seja treinando, seja levando-a a ser mais produtiva e produzir produtos de maior qualidade, ajuda o Time a compreender a auto-gestão e interdisciplinaridade. No entanto, o Scrum Master não gerencia a equipe Scrum, pois a equipe Scrum é de auto-organizada.

O Scrum Master pode ser um membro da Equipe, por exemplo, um desenvolvedor executar tarefas Sprint. No entanto, isso muitas vezes leva a conflitos quando o Scrum Master tem que escolher entre remover impedimentos e executar tarefas. O Scrum Master nunca deve ser o Product Owner (Liberato, 2011).

6.3. O Time

O Time é uma equipe composta por desenvolvedores para tornar o Product Backlog em funcionalidades potencialmente entregáveis a cada final de sprint. Basicamente o time decide como as tarefas serão executadas e quem irá executar. Compromete-se a entregar o Sprint backlog no final da iteração. Membros da equipe geralmente têm habilidades especializadas, tais como controle e processos de qualidade, análise de negócios, arquitetura, design de interface do usuário, ou design de banco de dados.

O time tem que de abordar as exigências e transformar em um produto utilizável. Pessoas que se recusam a codificar por exemplo, porque são arquitetos ou designers não se adaptam bem a um time Scrum onde todos contribuem, mesmo que isso exija aprender novas habilidades ou lembrar as antigas. Não há títulos em Times Scrum, e não há exceções a esta regra.

O Time faz sua autogestão, cada membro da equipe aplica sua especialidade para todos os problemas. A sinergia nos resultados melhora a eficiência global de toda a equipe. O tamanho ideal para uma equipe é de sete pessoas, mais ou menos dois. Quando há menos de cinco Membros da equipe, há menos interação e, como resultado menor ganho de produtividade. Além do mais, o time poderá encontrar limitações de conhecimento durante partes da Sprint e ser incapaz de entregar um parte pronta do

produto. Se houver mais de nove membros, exigirá mais coordenação. Equipes grandes geram muita complexidade para um processo empírico de gerir. No entanto, há alguns Times de sucesso que excedeu o limites superior e inferior dessa faixa de tamanho. O Product Owner e o Scrum Master não são incluídos nessa conta. A composição da equipe pode mudar no final de um Sprint. Toda vez que o Time é mudado, a produtividade obtida com a auto-organização é diminuída. Cuidados devem ser tomados quando mudar a composição da equipe.(www.scrum.org, 2011).

7. ARTEFATOS DO SCRUM

7.1. Estórias

Estórias são pequenas orações que descrevem as funcionalidades que o cliente precisa, sintaxe: Eu como <usuário-perfil-Quem> posso/gostaria/necessito/devo <funcionalidade-requisito-o que> para <retorno-por que>.

Estórias fracionam os requisitos para que seja possível (e mais fácil) estimar o esforço para realizar aquele objetivo. Resumindo, User Stories são descrições simples que descrevem uma funcionalidade e é recomendável que sejam escritas segundo o ponto de vista do usuário.

Estórias devem ser curtas, simples e claras. Deve-se conseguir escrevê-las em um simples e pequeno cartão, se não há espaço para escrevê-la em um cartão é por que devemos refiná-la mais, e as dividir em estórias.

O interessante no uso de estórias é o foco nas necessidades reais e práticas do usuário.

Alguns aspectos importantes podem ser notados no uso estórias:

- Validar se a funcionalidade é necessária antes de incluí-la
- Análise das necessidades reais do usuário
- Ajuda a priorizar o que deve ser feito
- É mais fácil estimar o esforço que será necessário para implementar a funcionalidade

7.2. Critério e testes de aceitação da estória

Os critérios e testes de aceitação da estória devem ser feitos pelo Product Owner ou especialista no negócio de acordo com seus critérios de aceitação.

Os Critérios de Aceitação são representados por uma lista de itens de negócio que expressam formas de usar a funcionalidade implementada em uma estória. O objetivo dessa lista é validar se a estória foi implementada de acordo com o que o PO queria, por isso o nome Critério de Aceitação.

Ao final da sprint, na reunião de revisão, a equipe apresentará estória por estória para o PO e com base nos Critérios de Aceitação, que foram definidos para cada História, que será realizada a apresentação e validação quanto ao funcionamento da mesma.

Incluir os critérios de aceitação como parte da história só vem a agregar vantagens, como:

- Prover material para a equipe pensar em como uma funcionalidade será executada pelo ponto de vista do usuário.
- Eliminar ambigüidades quanto aos requisitos.
- Confirmar que a história está completa e funcionando.
- Garantir maior satisfação do usuário.

O critério de testes de aceitação da estória é a descrição daquilo que será verificado pelo Product Owner para dizer que a funcionalidade atingiu seu objetivo.

7.3. Valor de negócio

Assim que as estórias são definidas elas devem ser priorizados de acordo com o valor de negócio que tem para o cliente, quanto maior o valor de negócio que a estória tem, mais prioritária ela é dentre todas as estórias.

7.4. Product Backlog

O Product Backlog é a lista com todas as funcionalidades/desejos/sonhos do cliente, priorizados e ordenados de acordo com o valor que representam para o negócio. O Product Backlog não precisa estar completo no início de um projeto. Pode-se começar com tudo aquilo que é mais óbvio em um primeiro momento. Com o tempo, o Product Backlog cresce e muda à medida que se aprende mais sobre o produto e seus usuários. Durante o Sprint Planning Meeting, o Product Owner prioriza os itens do Product Backlog e os descreve para a equipe. A equipe então determina que itens sejam capaz de completar durante a Sprint que está por começar. Tais itens são, então, transferidos do Product Backlog para o Sprint Backlog. Ao fazer isso, a equipe quebra cada item do Product Backlog em uma ou mais tarefas do Sprint Backlog. Isso ajuda a dividir o trabalho entre os membros da equipe. Podem fazer parte do Product Backlog tarefas técnicas ou atividades diretamente relacionadas às funcionalidades solicitadas.(improveit,2011)

Backlog Description	Initial Estimate	Adjust-ment Factor	Adjusted Estimate	work remaining until completion						
				1	2	3	4	5	6	7
Title Import				256	209	193	140	140	140	140
Project selection or new	3	0.2	3.6	3.6	0	0	0	0	0	0
Template backlog for new projects	2	0.2	2.4	2.4	0	0	0	0	0	0
Create product backlog worksheet with formatting	3	0.2	3.6	3.6	0	0	0	0	0	0
Create sprint backlog worksheet with formatting	3	0.2	3.6	3.6	0	0	0	0	0	0
Display tree view of product backlog, releases, sprints	2	0.2	2.4	2.4	0	0	0	0	0	0
Sprint-1	13	0.2	15.6	16	0	0	0	0	0	0
Create a new window containing product backlog template	3	0.2	3.6	3.6	3.6	0	0	0	0	0
Create a new window containing sprint backlog template	2	0.2	2.4	2.4	2.4	0	0	0	0	0
Burndown window of product backlog	5	0.2	6	6	6	0	0	0	0	0
Burndown window of sprint backlog	1	0.2	1.2	1.2	1.2	0	0	0	0	0
Display tree view of product backlog, releases, prints	2	0.2	2.4	2.4	2.4	0	0	0	0	0
Display burndown for selected sprint or release	3	0.2	3.6	3.6	3.6	0	0	0	0	0
Sprint-2	16	0.2	19.2	19	19	1.2	0	0	0	0
Automatic recalculating of values and totals	3	0.2	3.6	3.6	3.6	3.6	0	0	0	0
As changes are made to backlog in secondary window, update burndown graph on main page	2	0.2	2.4	2.4	2.4	2.4	0	0	0	0
Hide/automatic redisplay of burndown window	3	0.2	3.6	3.6	3.6	3.6	0	0	0	0
Insert Sprint capability ... adds summing Sprint row	2	0.2	2.4	2.4	2.4	2.4	0	0	0	0
Insert Release capability ... adds summary row for backlog in Sprint	1	0.2	1.2	1.2	1.2	1.2	0	0	0	0
Owner/assigned capability and columns optional	2	0.2	2.4	2.4	2.4	2.4	0	0	0	0
Print burndown graphs	1	0.2	1.2	1.2	1.2	1.2	0	0	0	0
Sprint-3	14	0.2	16.8	17	17	17	0	0	0	0
Duplicate incomplete backlog without affecting totals	5	0.2	6	6	6	6	6	6	6	6
Note capability	6	0.2	7.2	7.2	7.2	7.2	7.2	7.2	7.2	7.2
What-if release capability on burndown graph	15	0.2	18	18	18	18	18	18	18	18
Trend capability on burndown server	2	0.2	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4
Publish facility for entire project, publishing it as HTML web pages	11	0.2	13.2	0	0	13	13	13	13	13
Future Sprints	39	0.2	46.8	34	34	47	47	47	47	47
Release-1				85	70	65	47	47	47	47

Figura 10 – Exemplo de product backlog

7.5. Sprint Backlog

O Sprint Backlog é uma lista de tarefas que o Scrum Team se compromete a fazer em um Sprint. Os itens do Sprint Backlog são extraídos do Product Backlog, pela equipe, com base nas prioridades definidas pelo Product Owner e a percepção da equipe sobre o tempo que será necessário para completar as várias funcionalidades.

Cabe a equipe determinar a quantidade de itens do Product Backlog que serão trazidos para o Sprint Backlog, já que é ela quem irá se comprometer a implementá-los.

Durante um Sprint, o Scrum Master mantém o Sprint Backlog atualizando-o para refletir que tarefas são completadas e quanto tempo a equipe acredita que será necessário para completar aquelas que ainda não estão prontas. A estimativa do trabalho que ainda resta a ser feito no Sprint é calculada diariamente e colocada em um gráfico, resultando em um Sprint Burndown Chart. (improveit, 2011)

7.6. Entregáveis

Independente de metodologia ou área de conhecimento, é considerado uma boa prática de gerenciamento de projetos “fasear” o projeto, definindo de forma clara e objetiva o que tem que ser entregue e quando (FLAVIO,2008). De forma que ao final de cada fase pré estabelecida temos sempre um produto ou “ENTREGA” também conhecido como “empregável”.

7.7 Burndown chart

Burndown chart, figura12, é um gráfico que demonstra diariamente o andamento do sprint, deve ser atualizado todo dia antes da daily meeting:

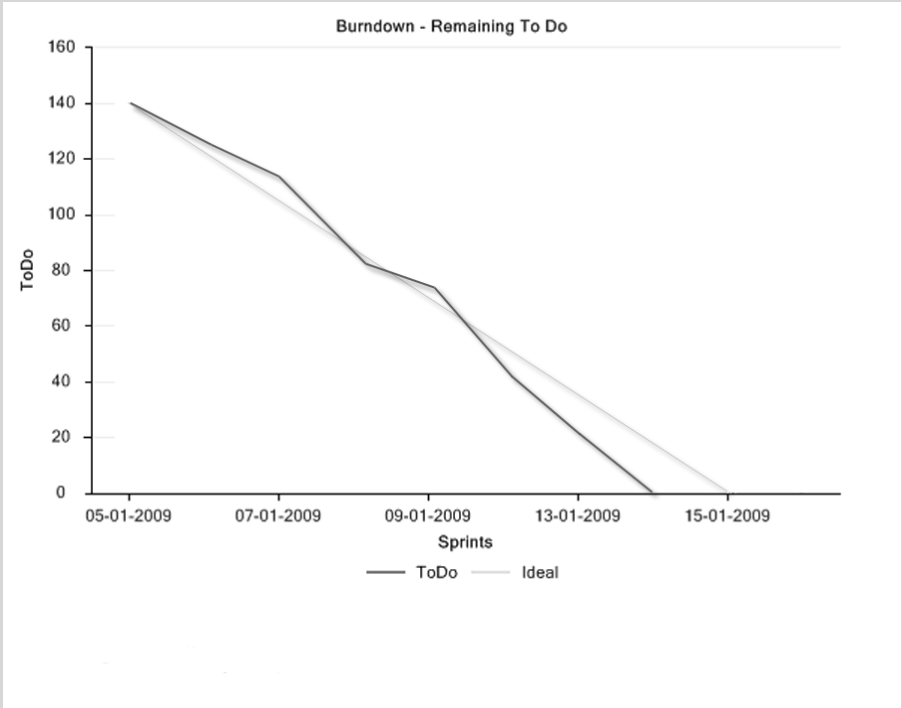

Figura 11 – Burndown Chart

- eixo horizontal (x): quantidade de dias de uma sprint, por exemplo 20 dias
- eixo vertical (y): quantidade de horas definidos para o sprint, capacidade do time

7.8. Quadro Scrum

Métodos Ágeis pedem transparência do processo de desenvolvimento de software e segundo Schwaber (2011), a agilidade e a transparência são a base para qualquer desenvolvimento.

O quadro de tarefas Scrum, figura 13, é parte da Gestão à Vista. A Gestão à Vista, é uma ferramenta com forte apelo à comunicação organizacional e visual, pois transmite a mensagem muitas vezes sem a necessidade de palavras, somente com a utilização de símbolos e cores, de modo que todos conseguem receber a mensagem, muitas vezes de uma forma lúdica.

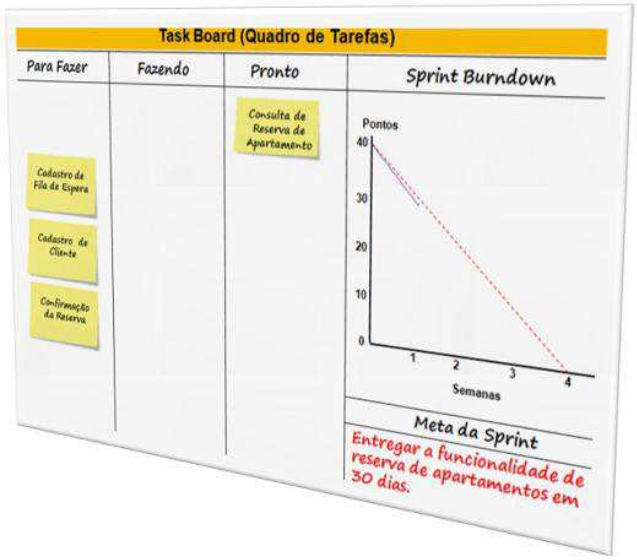

Figura 12 – Task Board - Gestão a vista com Scrum

A Gestão à Vista tem como objetivo disponibilizar as informações necessárias de uma forma simples, visual, transparente e de fácil assimilação, buscando tornar mais fácil o trabalho diário e também a busca pela melhoria da qualidade.

Segundo Rildo (2011) a gestão a vista torna possível a divulgação de informações para um maior número de pessoas simultaneamente e ajuda a estabelecer a prática de compartilhamento do conhecimento e transparência como parte da cultura organizacional.

8. RITOS DO SCRUM

8.1. Planning meeting

Ocorre no inicio de cada Sprint e deve durar 8h, na primeira parte o PO apresenta sua visão do produto, detalhe das estórias, tira dúvidas e conclui a estimativa junto com o time de algum item do product backlog, na segunda parte o time define quais itens do product backlog irão fazer parte do selected product backlog e detalham horas para cada tarefa a ser executada para cada item selecionado formando assim o Sprint backlog

8.2. Daily meetings

Reunião diária de 15 minutos onde cada membro do time deve responder

1 - O que foi feito desde a última Daily Meeting? (O que fiz desde a ultima reunião?)

2 - O que se pretende fazer até a próxima Daily Meeting (O que pretendo fazer até a próxima reunião?)

3 - Há algum impedimento? (Estou tendo algum impedimento?)

com essa reunião o time ganha visibilidade de como está o caminho para a meta e planeja o dia seguinte de trabalho

8.3. Retrospective meeting

Restrospective meeting é a ultima cerimônia do scrum, realizado ao final de uma Sprint ou Release, é uma reunião de lições aprendidas:

- O que foi bom?
- O que deve ser melhorado?
- O que melhoramos?

8.4. Sprint review meeting

É uma apresentação de resultado dos Sprint, deve ser realizada ao final de cada sprint, tem como propósito mostra o que foi feito ao Product Owner e convidados, O time realiza esta apresentação, que deve ser feito no formato de demo. Esta apresentação do Sprint deve ter no máximo 2h e deve ser informal. O produt owner avalia se a meta do Sprint foi alcançada e faz anotações que poderão se transformar em novos itens para o Product Backlog.

8.5. Release Planning Meeting

Planejar o trabalho para a geração de um entregavel, de acordo com uma condição de satisfação real, clara e prioritária

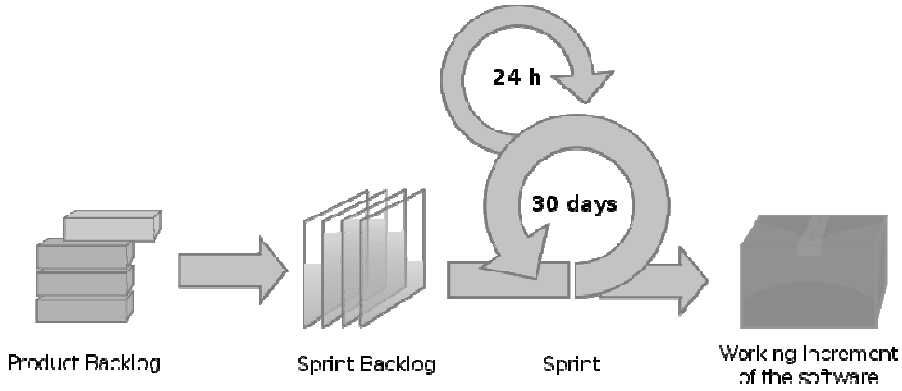

Figura 13 – Iterações Scrum

9. ITERAÇÕES SCRUM

O método baseia-se ainda, conforme Schwaber (2002), em princípios como: equipes pequenas de, no máximo, 7 pessoas; requisitos que são pouco estáveis ou desconhecidos; e iterações curtas. Divide o desenvolvimento em intervalos de tempos de, no máximo 30 dias, também chamadas de Sprints. Este método não requer ou fornece qualquer técnica ou método específico para a fase de desenvolvimento de software, apenas estabelece conjuntos de regras e práticas gerenciais que devem ser adotadas para o sucesso de um projeto. As práticas gerenciais do Scrum são: Product Backlog, Daily Scrum, Sprint, Sprint Planning Meeting, Sprint Backlog e Sprint Review Meeting. A seguir uma breve descrição do Product Backlog e do Sprint, que são as práticas relacionadas à área de requisitos no método ágil Scrum.

O ciclo do Scrum é baseado em um série de iterações bem definidas, cada uma com duração de 2 a 4 semanas, chamadas ***Sprints***. Antes de cada *Sprint*, realiza-se uma **Reunião de planejamento** (***Sprint Planning Meeting***) onde o time (equipe) de desenvolvedores tem contato com o cliente (***Product Owner***) para priorizar o trabalho que precisa ser feito, selecionar e estimar as tarefas que o time pode realizar dentro da *Sprint.* A próxima fase é a **Execução da *Sprint***. Durante a execução da *Sprint*, o time controla o andamento do desenvolvimento realizando **Reuniões Diárias Rápidas** (***Daily Meeting***), não mais que 15 minutos de duração, e observando o seu progresso usando um gráfico chamado ***Sprint Burndown***. Ao final de cada *Sprint*, é feita uma revisão no produto entregue para verificar se tudo realmente foi implementado.

Ao final da *Sprint*, deve-se realizar uma **Reunião de Revisão** (***Sprint* Review**), onde o time demonstra o produto gerado na *Sprint* e valida se o objetivo foi atingido. Logo em seguida, realiza-se a **Reunião de Retrospectiva** (***Sprint Retrospective***), uma reunião de lições aprendidas, com o objetivo de melhorar o processo/time e/ou produto para a próxima *Sprint*. Scrum torna-se ideal para projetos dinâmicos e suscetíveis a mudanças de requisitos, sejam eles novos ou apenas requisitos modificados. No entanto, para aplicá-lo, é preciso entender antes os seus papéis, responsabilidades, conceitos e artefatos das fases de seu ciclo.

Scrum é bastante objetivo, com papéis bem definidos, de fácil adaptação e ainda, sua curva de aprendizado é relativamente baixa. Segundo Schwaber (2004), o Scrum não é um processo previsível, ele não define o que fazer em toda circunstância. O Scrum é usado em trabalhos complexos nos quais não é possível prever tudo o que irá

ocorrer e oferece um *framework* e um conjunto de práticas que torna tudo visível. Isso permite aos praticantes do Scrum saber exatamente o que está acontecendo ao longo do projeto e fazer os devidos ajustes para manter o projeto se movendo ao longo do tempo visando alcançar os seus objetivos. Logo, o Scrum não vai dizer exatamente o que fazer, não irá resolver todos os seus problemas, mas com certeza os problemas serão mais facilmente identificados. Por ser um *framework*, irá servir como um guia de boas práticas para atingir o sucesso. Entretanto, as decisões de quando e como usá-lo, quais táticas e estratégias seguir para obter produtividade e realizar as entregas ficam por conta de quem estiver aplicando. O conhecimento das suas práticas permite a aplicação das mesmas de forma variada e este é um dos aspectos positivos do Scrum, a adaptabilidade.

Vale ressaltar que as práticas do Scrum podem ser aplicadas em qualquer contexto onde pessoas precisem trabalhar juntas para atingir um objetivo comum. Scrum é recomendado para projetos de outras áreas além de software e principalmente para projetos de pesquisa e inovação.

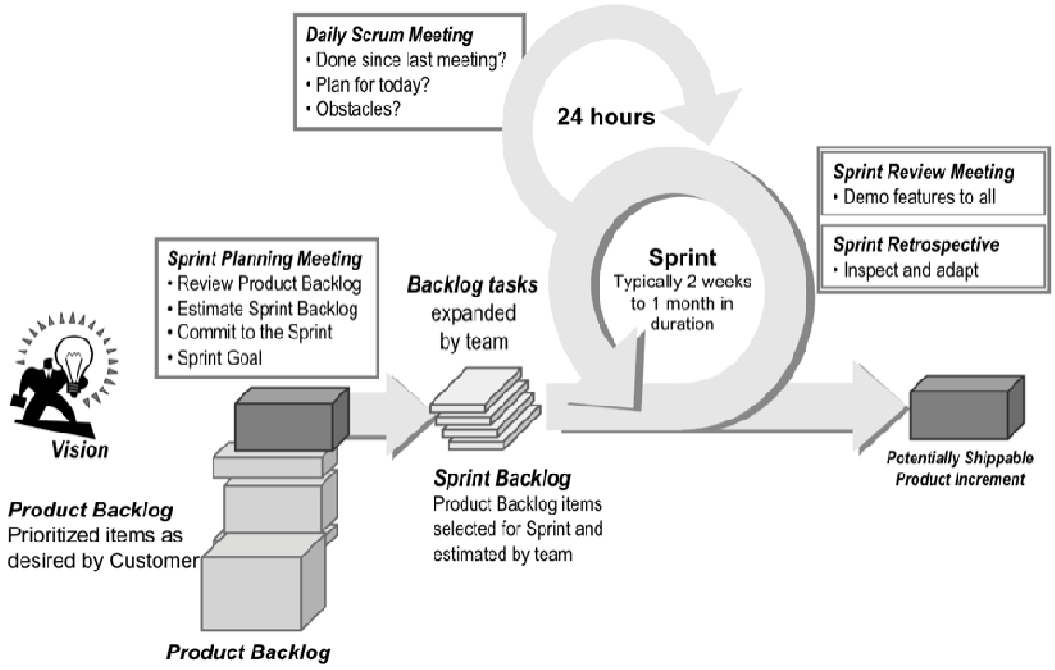

Figura 14 – Iterações Scrum com detalhes

9.1. NokiaTest

Nokia Test é um teste que foi desenvolvido pela Nokia para saber se um time está de fato fazendo Scrum. Trata-se de um teste simples e objetivo com 8 perguntas simples que tem como finalidade determinar se você está fazendo desenvolvimento iterativo incremental e se está realmente aplicando Scrum (NOKIA TEST) :

Suas iterações estão com duração limitada para 4 semanas ou menos?
Há software testado e funcionando no final de uma iteração?
As iterações começam antes das especificações terem sido concluídas?
O time sabe quem é o seu Product Owner?
Existe um Product Backlog priorizado por valor de negócio?
O Product Backlog possui estimativas criadas pelo time?
O time gera gráficos de burndown e conhece sua velocidade?
O time está protegido de gerentes de projetos e outras pessoas que possam interferir com seu trabalho?
Após o Nokia test, Jeff Surtherland co-criador do Scrum estendeu o Nokia Test com um sistema de pontos:

Pergunta 1: Iterações
Nenhuma iteração
Iteração > 6 semanas
Comprimento variável < 6 semanas
Duração da iteração fixa 6 semanas
Duração da iteração fixo 5 semanas
Comprimento da interação fixo de 4 semanas ou menos

Pergunta 2: Teste
Sem testadores dedicados no time
Testes unitários
Testes de funcionalidades
Funcionalidades testadas assim que concluída
Software passa os testes de aceitação
Software é implantado

Pergunta 3: Especificação Agile

Não há requisitos

Grande documentação de requisitos

Pobres histórias de usuário

Bons requisitos

Boas histórias de usuário

Apenas o suficiente, apenas em especificações de tempo

User stories boa amarrada com as especificações quando necessário

Pergunta 4: Product Owner

Nenhum Product Owner

Product Owner que não entende Scrum

Product Owner que atrapalha o time

Product Owner não envolvidos com o time

Product Owner com product backlog claro estimada pela equipe antes da Sprint Planning Meeting

Product Owner com o roteiro e datas de entrega, com base na velocidade do time

Product Owner que motiva equipe

Pergunta 5: Product Backlog

Sem Product Backlog

Múltiplos Product Backlog

Product Backlog único

Product Backlog claramente especificados e priorizados pelo ROI antes da Sprint Planning meeting

Product Owner tem uma versão do burndown com data do entregável baseado na velocidade do time

Product Owner pode mensurar ROI com base na receita real, o custo por ponto da história, ou outras métricas

Pergunta 6: Estimativas

Product Backlog não estimado

Estimativas não produzidas pela equipe
Estimativas não produzida pelo poker planning
Estimativas produzidas pelo poker planning e pela equipe
Estimativa de erro < 10%

Pergunta 7: Sprint Burdown Chart
Não há gráficos de Burndown
Gráfico Burndown não atualizado pela equipe
Burndown chart em horas / dias não respondendo por trabalhos em curso (tarefas parcialmente queimadas)
Gráfico Burndown só queima quando a tarefa é feita
Gráfico Burndown só queima quando a história é feita
Além disso, a equipe sabe sua velocidade ? Sim / Não
Além disso, O Product Owner tem um plano de Lançamento baseado em velocidade conhecida ? Sim / Não

Pergunta 8: Ruídos da equipe
Líder do Projeto gerente ou atrapalha equipe
Product Owner atrapalha equipe
Gerente, Projeto Líderes ou Chefes de equipa diz às pessoas o que fazer
Tem líder de projeto e regras Scrum
Nenhum ruído na equipe, apenas regras Scrum

Pergunta 9: Equipe
Tarefas atribuídas aos indivíduos durante Sprint Planning
O membro da equipe não tem nenhuma sobreposição em sua área de especialização
Nenhuma liderança - um ou mais membros da equipe designam-se como autoridade direta
Equipe não tem a competência necessária
Equipe empenham-se coletivamente para concluir o sprint e backlog
Membros da equipe freqüentemente tem impedimentos durante o sprint
Equipe está em estado de hiperprodução

De acordo com as respostas selecionadas gera-se o seguinte gráfico com possíveis configurações:

Abaixo um Nokia Test com resultado não satisfatório

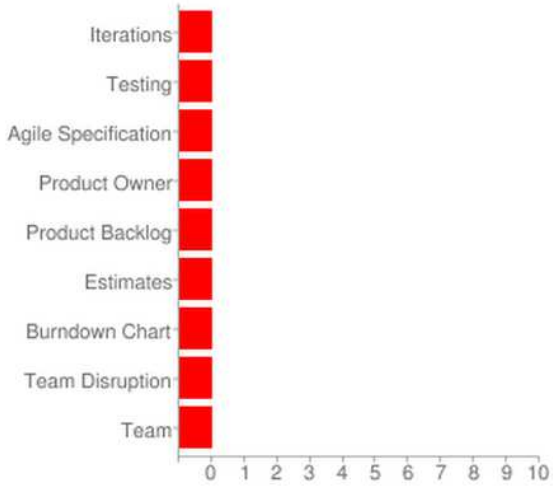

Figura 15 – Nokia Test com resultado não satisfatório

Figura 16 – Nokia Test com resultado satisfatório

9.2. Comunicação

Boa parte dos problemas no desenvolvimento de projetos reside principalmente na comunicação – refletindo nos demais. Quando presentes, mecanismos de comunicação ineficientes são utilizados contribuindo para a falta de compreensão e colaboração por parte dos envolvidos.

A figura 18 promove uma comparação entre a efetividade de comunicação e a "riqueza" do canal de comunicação. Dois mecanismos são evidenciados: os que não possuem perguntas e respostas e os que possuem. Este primeiro tipo propõe pouca interação e não permite a colaboração e troca necessária para o alcance do objetivo proposto. A utilização de papel para descrever e comunicar o problema tem importante cunho "documentacional", além de poder ser compartilhado entre diversas pessoas.

Por outro lado, mecanismos que promovem e facilitam a comunicação, os mecanismos interativos, são compostos por perguntas e respostas - como duas pessoas falando ao telefone ou discutindo via email, são notoriamente mais eficientes. E ainda torna-se mais eficiente se tivermos a presença física na mesma sala dos diversos envolvidos. Para completar, um quadro branco auxiliando na dinâmica das discussões, pode expandir em muito as possibilidades de entender e se fazer entendido.

Figura 17 – Temperatura da comunicação

Comunicação "face-to-face" é uma das formas de se resolver grande parte dos problemas citados e, metodologias ágeis, como o SCRUM, pregam incessantemente esta prática.(Devmedia, 2011)

10. CONCLUSÃO

O uso de Scrum no desenvolvimento de projetos traz vantagens consideráveis no processo de gestão e no desenvolvimento de um produto. O fato de práticas Scrum priorizar a entrega de resultados com maior valor para o cliente primeiramente e de usar prática enxutas somam pontos a favor do uso do Scrum para o cliente pois ajuda no cumprimento de prazos, e em curto prazo o cliente já tem o que é mais importante para seu negócio.

A utilização de Scrum é ideal para equipes pequenas e em projetos onde o desenvolvimento é incerto de requisitos, Scrum implica em comunicação constante, agilidade, mudanças, energia,

É necessário mudança de cultura na equipe que adota Scrum, por exemplo criar grupo de estudos dentro da equipe para aprofundar conhecimento sobre diversos assuntos, e depois desenvolver um treinamento para toda a equipe a fim de nivelar o conhecimento.

Definitivamente o uso de práticas Scrum ajudam entregar o projeto dentro do prazo e orçamento previsto, Scrum mostra-se ideal para projetos dinâmicos e suscetíveis a mudanças de requisitos, sejam eles novos ou apenas requisitos modificados.

A curva de aprendizado de práticas Scrum é consideravelmente fácil sendo ideal para equipes que ainda não utilizam metodologia ou utilizam práticas que não estão apresentando bons resultados.

REFERÊNCIAS BIBLIOGRAFICAS

OGUNNAIKE, Babatunde A.; **RAY**, W. Harmon; **Process Dynamics, Modeling, and Control**,
Oxford University Press, 1992

DRUCKER, Peter; **A profissão de administrador**
Editora Pioneira, São Paulo, 1998

PRESSMAN, Roger S.; **SANTOS**, Jose Carlos Barbosa dos. **Engenharia de software**. São Paulo: Makron, 2006. 1056 p.

SCHWABER, K.; **BEEDLE**, M. **Agile Software Development with Scrum**, NJ, Prentice-Hall, 2002.

SCHWABER, K., **Agile Project Management With Scrum**,
Microsoft, 2004.

WOODWARD, E.; **SURDEK,** S.; **GANIS,** M.. **A Practical guide to Distribted Scrum**,
IBM Press, 2010

NOKIA TEST. Disponivel em "http://www.cedur.se/nokia_test2.html". Acesso em 10/11/2011.

SCRUM BUT... TEST. Disponível em "http://antoine.vernois.net/scrumbut/". Acesso em 11/11/2011.

AGILE MANIFESTO. Disponível em "http://agilemanifesto.org/" Acesso em 11/11/2011.

GERENCIAMENTO ÁGIL DE PROJETOS COM SCRUM. Disponível em "http://www.slideshare.net/paulofurtado/curso-scrum" Acesso em 15/10/2011

BLOG SOBRE GESTÃO DE EQUIPES E PROJETOS. Disponível em "www.ges.blog.br, 2010" Acesso em 15/10/2011

GERENCIAMENTO DE PROJETOS,
Disponível em "http://marceloliberato.wordpress.com/tag/gerenciamento-de-projetos/" Acesso em 15/10/2011

O GUIA DO SCRUM.
Disponível em "http://www.scrum.org/storage/Scrum%20Guide%202011%20-%20PTBR.pdf" Acesso em 15/10/2011

RILDO SANTOS. Disponível em "http://www.rildosan.com/2011/02/para-dar-transparencia-ao-scrum-use-o.html" Acesso em 15/10/2011

ARTIGO ENGENHARIA DE SOFTWARE 4 – POR QUE SCRUM?
Disponível em "http://www.devmedia.com.br/articles/viewcomp.asp?comp=9884" Acesso em 15/10/2011

FLAVIO, Disponível em http://gfinfo.blogspot.com/2008/04/prince2-deliverable-entregveis.html Acesso em 15/10/2011

KLEBER NARDI, http://www.pmkb.com.br/artigos-mainmenu-25/482-pmbok-x-scrum-como-gerenciar-um-projeto-de-software.html, acesso em 13/11/2011