APLICAÇÃO DO MÉTODO ÁGIL SCRUM NO DESENVOLVIMENTO DE SOFTWARE DO SISTEMA "REDE BEM ESTAR" DA PREFEITURA MUNICIPAL DE VITÓRIA (ES).

Silvana Ramos Buzetti[1]

Resumo: O mercado globalizado tem exigido qualidade e produtividade em software. Em resposta, as empresas de tecnologia da informação têm buscado a atualização de conhecimento e a inteligência na aplicabilidade como um diferencial na concorrência. O Manifesto Ágil em 2001, foi um marco inovador para a engenharia de software ao propor maneiras melhores de desenvolver software. A divulgação das metodologias ágeis – *Extreme Programming, Scrum, Feature Driven Development, Kanban* e outros – trouxe um crescimento progressivo de adeptos e ampliou a escolha da melhor abordagem para gerenciar um projeto: clássica ou ágil. O Scrum vem se destacando para o planejamento e gerenciamento de projetos, devido a vantagem de um método leve, empírico e de fácil adequação. Este trabalho buscou responder como o *Scrum* pode auxiliar na distribuição e gerenciamento das atividades do desenvolvimento de software, usando o método estudo de caso com objetivo exploratório. Para tanto, a pesquisa teve como amostras alguns integrantes das equipes envolvidas no projeto e os dados coletados de caráter qualitativo. Os resultados obtidos com as investigações realizadas demonstraram a dificuldade da compreensão e adoção correta dos princípios ágeis e as anomalias ocorridas nesta adaptação do *Scrum*. Conclui-se que esta adaptação não serviu como fator crítico ao sucesso do projeto. Pesquisas futuras podem cooperar na prevenção de anomalias, abordando as adaptações – *Scrumbuts* – que tem sido bastante discutidas pela comunidade *Scrum* questionando a sua legitimidade e eficácia.

Palavras-chave: Gestão de Projetos. Métodos Ágeis. Scrum.

[1] Acadêmico(a) do curso Especialização em Gerência de Projetos de Tecnologia da Informação da Universidade do Sul de Santa Catarina. http://www.unisul.br. Formada em Administração de Empresas.

1 INTRODUÇÃO

A Rede Bem Estar (RBE) é o software de gestão de saúde utilizado no município de Vitória-ES, criado e mantido pela Subsecretaria de Tecnologia de Informação (SUB-TI) da Prefeitura Municipal de Vitória (PMV).

Projeto iniciado em 01/07/2008 que interliga todos os equipamentos de saúde (farmácias, laboratórios de análises clínicas, pronto-atendimento, unidades de saúde, consultórios odontológicos, centros de referência, centro de especialidades, etc.) em um único sistema.

Antes da Rede Bem Estar, Vitória enfrentava os seguintes problemas: anotações ilegíveis; perda de documentos do paciente; rasuras e acesso indevido aos prontuários e dificuldade na geração de indicadores (proposta de melhorias para o sistema de saúde).

O objetivo principal do projeto é unir, no Prontuário Eletrônico do Paciente (PEP) toda a história clínica do paciente, com dados produzidos em formatos diversos (atestado, receita, radiografia, exame laboratorial, registro de visitas domiciliares, etc), em épocas diferentes, feitos por diferentes profissionais, em locais distintos de atendimento. Resolveu os problemas citados e trouxe inúmeros benefícios para o cidadão e gestores municipais.

O sistema conta hoje com 450.000 prontuários e 17.145.000 registros de procedimentos (médicos, odontológicos, laboratoriais, etc).

Como a gestão da saúde não era informatizada e a Secretaria da Saúde preferia a aquisição de um software de terceiros ao risco de empreender um sistema próprio através da SUB-TI. Várias discussões foram necessárias até que a resistência inicial cedeu.

Sobre empreender projetos de software, Martins (2007, p. 1) descreve:

> "Em muitos aspectos a engenharia de software pode ser comparada à engenharia civil. [...]Similarmente, num projeto de construção de software as etapas são: engenharia de sistema, análise de requisitos, projeto técnico, construção e validação.
> Na engenharia de software o projeto começa com uma necessidade de negócio e uma visão da solução, também considerando restrições de tempo e custo. O primeiro passo é entender e limitar o contexto de negócio onde a solução computacional está inserida. Na sequência vem a análise de requisitos (similar à arquitetura na construção civil) que busca entender as necessidades dos clientes e usuários, usando técnicas como construção de protótipos, modelagem estática e dinâmica do sistema, e definição das funcionalidades que o sistema deve prover. O próximo passo é a elaboração do projeto técnico, onde são definidos os elementos da estrutura do sistema, a arquitetura (componentes e módulos) e como estes elementos interagem para prover o comportamento esperado. Com base no que foi especificado, a equipe começa a próxima etapa, que é a construção; nesta etapa o sistema é codificado, integrado e testado. Finalmente vem a fase de validação, quando o sistema é testado pela equipe de desenvolvimento e homologação pelos usuários e clientes."

A engenharia de software é uma ciência recente se comparada a outras. Os fracassos de projetos de software culminaram nas décadas de 60 e 70. Koscianski (2006, p. 191) explica a importância da escolha e aplicação de uma metodologia de desenvolvimento de software:

> Muitas organizações desenvolvem software sem usar nenhum processo. Geralmente isso ocorre porque os processos tradicionais não são adequados às suas realidades. Em particular, as pequenas e médias organizações não possuem recursos suficientes para adotar o uso de metodologias pesadas, e, por esta razão, normalmente não utilizam nenhum processo. O resultado desta falta de sistematização na produção de software é a baixa qualidade do produto final, além de dificultar a entrega do software nos prazos predefinidos e inviabilizar a futura evolução do software.[...]

Isto incentivou a criação de diversas metodologias para gerenciamento de projetos. Hoje, há várias abordagens para gerenciamento de projetos:

> [...]algumas são classificadas como metodologias clássicas e outras como ágeis. Na abordagem clássica o processo é totalmente planejado antes de ser executado e durante a execução a equipe de projeto persegue a execução do plano. Na abordagem ágil há um planejamento parcial no início do projeto, sendo todos os detalhes do planejamento definidos junto com a execução, que normalmente é feita em ciclos.[...] Este método é o mais adequado para projetos de inovação e criação de novos produtos[...].(MARTINS, 2007, p. 3)

Como modelo clássico se tem o cascata (ou seqüencial) e o espiral que não admitem alterações nos requisitos seguindo uma seqüencia rígida de etapas com toda a documentação gerada. Esta forma de desenvolvimento dominou até o início da década de 1990. Alguns pesquisadores criticaram o modelo clássico indicando outra forma de desenvolvimento, o incremental, principalmente para grandes projetos. (KOSCIANSKI, 2006, p. 192).

Como modelo ágil, destacam-se o SCRUM e o XP- EXTREME PROGRAMMING. "A maioria das práticas da XP causa polêmica à primeira vista e muitas não fazem sentido se aplicadas isoladamente." (KOSCIANSKI, 2006, p. 195). Estes modelos possuem pontos em comum: "[...]equipes pequenas, trabalhando com requisitos instáveis ou desconhecidos e utilizando iterações curtas para promover visibilidade para o desenvolvimento.[...]" (KOSCIANSKI, 2006, p. 200).

Cada abordagem tem suas vantagens e desvantagens. "A melhor abordagem para gerenciar um projeto deve ser avaliada pela equipe e pelo gerente de projeto, podendo ser uma metodologia específica ou uma combinação de práticas e processos de todas elas." (Martins, 2007, p. 3).

Neste contexto, este trabalho objetiva apresentar um estudo de caso da equipe no período de 2009 a 2012, descrevendo como o SCRUM pode auxiliar na distribuição e gerenciamento das atividades do desenvolvimento de software, os principais desafios, as adequações, o nível de satisfação da equipe e da gerência de desenvolvimento, visando auxiliar projetos futuros.

A exigência do mercado globalizado por qualidade de produtos e serviços de TI é consolidado pelo leque de certificações disponibilizadas. Em geral, as empresas já apresentam suas certificações e metodologias nos seus sites, como forma de atestar a qualidade em seus processos.

A busca pela qualidade criou meios para aprimorar processos, capacitar gerentes e exige da formação do profissional de TI conhecimentos relacionados a negócios.

Recentemente, no Brasil, a divulgação da metodologia ágil SCRUM, com participação de empresas brasileiras e multinacionais, sugere um crescimento progressivo de adeptos.

O Manifesto Ágil, criado em 2001, descreve a essência de um conjunto de abordagens para desenvolvimento de software criado ao longo da última década. "Os engenheiros de software devem ser ágeis o suficiente para responder a um ambiente de negócios mutante." (PRESSMAN, 2001, p.59).

Um relato de alerta e cautela:

> Mas o próprio Alberto reconhece que nem todos os projetos são adequados para os métodos ágeis. "No caso de sistemas maiores, como um ERP (sistema de gestão), as metodologias tradicionais são mais indicadas", afirma o gerente. Desde o surgimento das metodologias ágeis, fanáticos de ambos os lados tentam provar qual dos métodos é o melhor. Mais maduro, o mercado agora vê que cada caso é um caso e tenta tirar o melhor proveito de ambos os métodos.[...]
>
> A questão é polêmica — existem empresas que usam metodologias tradicionais em projetos de ciclo de vida curto, com sucesso — e será difícil chegar a um consenso. Mas poucos profissionais discordam que o sucesso depende mesmo da boa definição dos requisitos e de uma comunicação sem falhas entre a área de negócios e o departamento de tecnologia. (COMPUTERWORLD, 2009).

Diante do exposto, não é tarefa fácil optar por uma metodologia que se adapte ao projeto em questão e sem ter ainda montada uma equipe.

O analista de sistemas responsável pelo projeto, através de sua experiência em outra empresa privada, sugeriu a adoção do framework de gerenciamento de projeto SCRUM, pois se tratava de um projeto novo, inovador, complexo, com o desafio de demonstrar de forma rápida (em ciclos de 30 dias no geral) e satisfatória (com qualidade) o produto que estava sendo construído e entregue em partes (de forma incremental e iterativa) para a Secretaria de Saúde que não acreditava no sucesso do empreendimento devido a sua complexidade. Para a equipe que seria constituída, exigiria dela flexibilidade nas adaptações às mudanças de requisitos (funcionalidades solicitadas), já que elucidar o novo e inovador é imprevisível. Estas características se encaixaram na abordagem ágil SCRUM e a influência de experiências positivas ratificou a escolha. A proposta foi aceita, apesar de nunca ter sido empregada em outros projetos da SUB-TI.

A equipe com papéis definidos, segundo o SCRUM, formou-se em 2009 e, o protótipo inicial implantado apenas em uma Unidade de Saúde. De 2011 a 2012 a equipe foi renovada e realizada a expansão da utilização da Rede Bem Estar para toda a rede do município (30 unidades de saúde). Entre 2013 a 2015 a equipe continuou quase a mesma, realizando manutenções e implementações ainda com base no modelo de gestão SCRUM.

Ao optar pelo novo enfoque em gestão de projetos, o gerenciamento ágil, em detrimento do gerenciamento clássico, sabendo que nem todo projeto se adapta a gestão ágil, buscarei responder como o SCRUM pode auxiliar na distribuição e gerenciamento das atividades do desenvolvimento de software do sistema "Rede Bem Estar" da Prefeitura Municipal de Vitória (ES).

A realização da presente pesquisa é de objetivo exploratório, usando o método estudo de caso, para o sistema Rede Bem Estar. O campo de pesquisa é a PMV - Prefeitura Municipal de Vitória (ES), tendo como amostra alguns participantes (cinco) das equipes que desenvolveram o projeto no período de 2009 a 2012. Os dados coletados de caráter qualitativo serão obtidos através de questionário. Os questionários serão enviados por e-mail aos participantes. A análise e interpretação dos dados serão na forma textual. Além disso, serão realizadas pesquisas bibliográficas, que permitem que se tome conhecimento de material relevante, tomando-se por base o que já foi publicado em relação ao tema, de modo que se possa delinear uma nova abordagem sobre o mesmo, chegando a conclusões que possam servir de embasamento para pesquisas futuras.

2 DESENVOLVIMENTO DE SISTEMAS

Analisando a linha histórica da Engenharia de Software, constata-se o seu esforço e evolução ao acompanhar a necessidade das empresas para atender ao mercado. Desde a década de 60 vem aprendendo com os seus fracassos. Inicialmente, a ênfase técnica, formulou os modelos de desenvolvimento de software: os Modelos em Cascata, os Modelos em Espiral e os processos iterativos. Na ênfase gerencial, o SEI – *Software Engineering Institute* – inovou, em 1986, com a publicação do SW-CMM – *Software Capability Maturity Model*. (BECK, 1999; COHEN et al, 2003 apud DIAS, DIAS, p. 9).

"[...] o SW-CMM reúne as melhores práticas de engenharia (desenvolvimento de software) e de gerenciamento de projetos, além de apontar o caminho para o aprimoramento dos processos de construção de software nas organizações." (SEI, 1995; PAULK, 2001 apud DIAS, 2005, p. 9).

Na década de 90 surgiu uma nova competitividade na economia que "[...] baseia-se principalmente na construção de competências essenciais para a aquisição de conhecimentos e de inovação." (HAMEL, 1995; PRAHALAD, 1995 apud DIAS, 2005, p.8).

A área de desenvolvimento de software também estava nessa transformação silenciosa, pois os métodos clássicos de desenvolvimento não se adequavam a nova realidade confrontada por alguns consultores. Por exemplo, Mary Poppendieck e Bob Charette perceberam que as práticas do *Lean Manufacturing* utilizadas na produção industrial japonesa, podiam ser aplicadas ao desenvolvimento de software e elaboraram o *Lean Programming*. (AGUANNO, 2005, p. 57, tradução nossa).

Kent Beck e Ron Jeffries estavam envolvidos em um projeto fracassado, que para reerguê-lo acabaram desenvolvendo um novo método, o *Extreme Programming*. De forma similar Alistair Cockburn, decidiu entrevistar equipes de desenvolvimento da IBM e construiu um processo de melhores práticas e lições aprendidas, resultando na criação do método *Crystal*. Estes novos métodos, criados independentemente, comungam ao abordarem alguns desses novos valores e princípios básicos descobertos. (AGUANNO, 2005, p. 59, tradução nossa).

Dentre estes e outros consultores, desenvolvedores e simpatizantes, reuniram-se em 2001 criando a Aliança Ágil e, também, documentaram, assinaram e publicaram o Manifesto Ágil. Eis a declaração:

> Estamos descobrindo maneiras melhores de desenvolver software, fazendo-o nós mesmos e ajudando outros a fazerem o mesmo. Através deste trabalho, passamos a valorizar:
> Indivíduos e interações mais que processos e ferramentas
> Software em funcionamento mais que documentação abrangente
> Colaboração com o cliente mais que negociação de contratos
> Responder a mudanças mais que seguir um plano
> Ou seja, mesmo havendo valor nos itens à direita,
> valorizamos mais os itens à esquerda. (AGILEMANIFESTO, 2001).

Aguanno (2005, p. 17, tradução nossa) faz alguns questionamentos cruciais: ”[...] O que é que eles oferecem que é diferente dos métodos tradicionais? Há algo realmente *novo* nestes métodos? [...]”. Dias (COHEN et al, 2003 apud 2005, p.3) aponta a resposta:

> Por fim, ressalta-se que apesar dos modelos em Espiral e iterativo permitirem uma maior agilidade ao desenvolvimento de software, estes ainda são criticados por não conseguirem promover respostas às mudanças em uma velocidade adequada à realidade dos negócios. O formalismo e a manutenção de uma documentação exaustiva típicos destes modelos, somados ao foco no planejamento e no controle do gerenciamento de projetos são considerados empecilhos à verdadeira agilidade no desenvolvimento de software.

2.1 SCRUM

Segundo Martins (2007, p. 252), o artigo “O jogo do desenvolvimento de novos produtos” escrito por Takeuchi e Nonaka sobre as 10 melhores práticas de empresas japonesas, introduziu o termo Scrum:

> [...] usado para a reunião de jogadores, no jogo de Rugby, quando eles se organizam em círculo para planejar a próxima jogada. É uma forma de mostrar que o projeto deve ser conduzido em pequeno ciclos, mas com uma visão de longo prazo, que é ganhar o jogo. (MARTINS, 2007, p. 252)

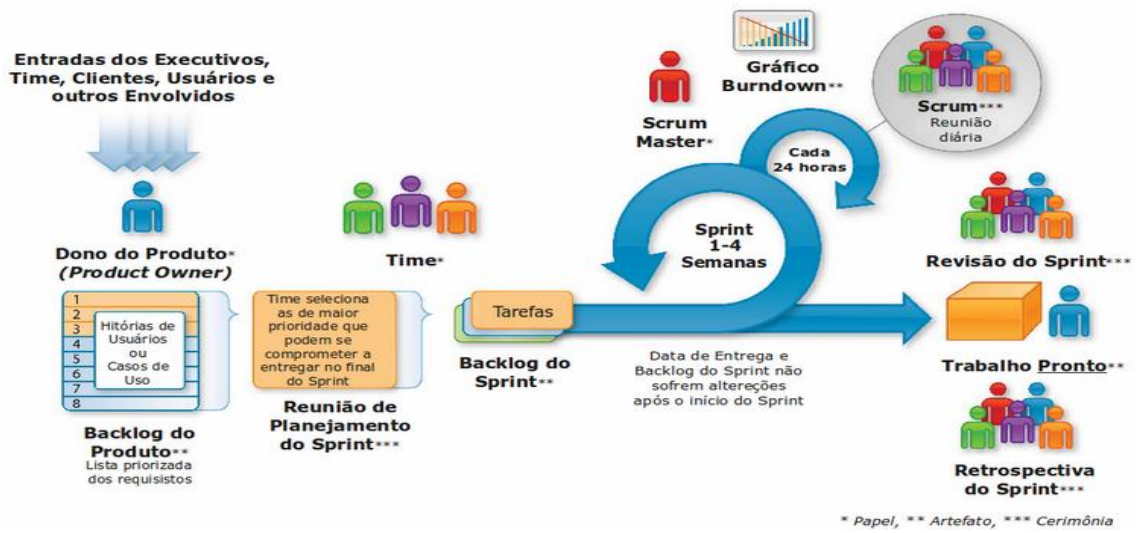
Em seu site, Schwaber (2005, nossa tradução), narra que em 1995, Ken Schwaber e Jeff Sutherland formalizaram o Scrum. Para eles processos complexos exigem controle de processos empíricos. Schwaber fundou e presidiu a Aliança Ágil e depois a Aliança Scrum. Eles têm propagado os valores e princípios ágeis bem como o SCRUM para aqueles que almejam recuperar a sua profissão, a dominância de mercado e excelência profissional. Relata ainda:

> Ágil é agora utilizado mais do que o modelo cascata nas organizações de desenvolvimento, e em 2009, 86% de todo o desenvolvimento Ágil foi baseado em Scrum. Será que isso significa que Scrum é superior? Não, isso simplesmente significa que Scrum é simples e bem explicado, e fácil para as pessoas entenderem como uma comunidade e equipes Scrum. (SCHWABER, 2015, nossa tradução).

“Scrum é um processo bastante leve para gerenciar e controlar projetos de desenvolvimento de software e para criação de produtos. [...] segue as filosofias iterativa e incremental. [...] ele é adaptativo e empírico.” (MARTINS, 2007, p. 253).

“O framework Scrum consiste em um conjunto formado por Times Scrum e seus papéis associados, Eventos com Duração Fixa (Time-Boxes), Artefatos e Regras.” (SCHWABER, 2009, p. 3).

Figura 1 - Visão resumida do Scrum



Fonte: Agile For All (tradução nossa)

Primeiramente, tem-se o Dono do Produto que representa o cliente e demais interessados no projeto. Em uma lista chamada Backlog do Produto, o Dono do Produto descreve os requisitos e funcionalidades iniciais, priorizados de acordo com o quanto de valor cada item gera para o negócio do cliente. Em seguida é realizada a Reunião de Planejamento do Sprint com o Dono do Produto, o Scrum Master e o Time. A finalidade desta reunião é o Time selecionar os itens que se compromete a entregar, ou seja, o Backlog do Sprint. Após isto a iteração (Sprint) começa não podendo sofrer alterações na Data de Entrega ou no Backlog do Sprint. Uma Reunião Diária entre Scrum Master e o Time é realizada para manter as informações atualizadas e disponíveis sobre o progresso (gráfico Burndown Chart) do Time. No final do Sprint, o Time faz a Revisão do Sprint apresentando o produto ao Dono do Produto e outros *stakeholders*. E, finalmente, o Scrum Master faz a Retrospectiva do Sprint com o Time e o Dono do Produto para avaliar o processo de trabalho e demais providências. (MARTINS, 2007, p. 253).

2.1.1 PAPÉIS NO SCRUM

O Time Scrum é composto pelo Scrum Master, pelo Product Owner e pelo Time.

O Scrum Master é o educador, o treinador e o responsável pelo Time Scrum e a organização a aderirem aos valores do Scrum, às práticas e às regras. Ele pode ser um membro do Time, mas jamais o Dono do Produto. Cabe a ele identificar e designar o Dono do Produto (Product Owner).

O Dono do Produto é uma pessoa responsável pelo Backlog do Produto e por determinar a prioridade de cada item e mantê-lo visível para todos. Ele representa os interesses do(s) cliente(s).

O Time é composto por desenvolvedores interdisciplinares com disposição ao desapego por especialidades e status, que compartilham tudo e se esforçam em descobrir *como* transformar o Backlog do Produto em incrementos de funcionalidades entregáveis. O Scrum Master ou outra pessoa não diz como fazer, por isto o Time é auto-organizável. A quantidade de membros sugerida é 7. Menos de 5 a interação e produtividade caí e mais do que 9 aumenta a necessidade de coordenação. (SCHWABER, 2009, p. 6-9).

2.1.2 BACKLOG DO PRODUTO

"O Product Owner é o responsável pelo Backlog do Produto, por seu conteúdo, por sua disponibilidade e por sua priorização. [...] ele está constantemente mudando para identificar o que o produto necessita para ser apropriado, competitivo e útil." (SCHWABER, 2009, p. 16).

Um exemplo de um Backlog do Produto, de acordo com Kniberg (2007, p. 10):

ID	**Imp**	**Nome**	**Nota**	**Teste**	**Estimativa**
1	30	Deposito	Diagrama Seqüência UML	Fazer login e depositar R$10.	5

ID: uma identificação única; **Imp**: a pontuação de importância dessa estória para o Dono do Produto; **Nome**: uma breve descrição para a estória; **Nota**: qualquer informação útil necessária; **Teste**: breve roteiro; **Estimativa**: a unidade é pontos por estória e geralmente corresponde mais ou menos a "relação homem/dias". Por exemplo, com três pessoas trancadas em uma sala levará aproximadamente quatro dias então a estimativa inicial é de 12 pontos por estória.

No Guia do Scrum, Schwaber (2009, p. 17) afirma:

> Os itens do Backlog do Produto possuem os atributos de descrição, prioridade e estimativa. A prioridade é determinada por risco, valor e necessidade. Há diversas técnicas para dar valor a esses atributos.
> O Backlog do Produto é ordenado por prioridade. O Backlog do Produto de mais alta prioridade leva a atividades de desenvolvimento imediatas. Quanto mais alta sua prioridade, mais urgente ele é, mais se pensou sobre ele e há mais consenso no que diz respeito ao seu valor. O Backlog de mais alta prioridade é mais claro e possui mais informações detalhadas do que o Backlog de prioridade mais baixa.[...]

2.1.3 REUNIÃO DE PLANEJAMENTO DO SPRINT

"O planejamento de sprint é uma reunião crítica, provavelmente o evento mais importante no Scrum (na minha opinião, claro). Um encontro de planejamento de sprint mal feito pode bagunçar totalmente um sprint." (KNIBERG, 2007, p. 13).

A reunião de planejamento não deve durar mais que 8 horas. Nas primeiras 4 horas o Dono do Produto apresenta e descreve os itens de maior prioridade (com maior valor para o seu negócio), os mais difíceis e com maior risco para mitigá-los o quanto antes. Nas outras 4 horas a equipe planeja o Sprint. (MARTINS, 2007, p. 263).

Como resultado desta reunião, Kniberg (2007, p. 15) descreve: o objetivo do Sprint; o Sprint Backlog; a Data para Entrega; o local da Reunião Diária.

Kniberg (2007, p. 22) reforça a determinar o Objetivo do Sprint:

> O objetivo do sprint pode parecer bobo e artificial durante o planejamento do sprint, mas freqüentemente se torna útil no meio dele, que é quando as pessoas começam a ficar confusas sobre o que elas deveriam estar fazendo.

Exemplificando os nuances que envolve a importância da Reunião de Planejamento:

> Às vezes, os product owners relutam em despender horas com a equipe fazendo planejamento do sprint. "Pessoal, eu já listei o que eu quero. Eu não tenho tempo para estar na sua reunião de planejamento". Isto é um problema muito grave.
> Escopo e importância são definidos pelo product owner. Estimativa é definida pela equipe. Durante uma reunião de planejamento do sprint, estas três variáveis são refinadas continuamente por diálogo cara-a-cara entre equipe e product owner.
> Normalmente, o product owner inicia a reunião resumindo seu objetivo para o sprint e as estórias mais importantes. Em seguida, a equipe toma a frente e estima o tempo de cada estória, começando pela mais importante.[...]
> Em alguns casos, a estimativa de tempo para uma estória não será aquela que o product owner esperava. Isto poderá fazê-lo mudar a importância da estória. Ou mudar o escopo da estória, que por sua vez fará a equipe re-estimar, etc., etc.
> Este tipo de colaboração direta é fundamental para o Scrum e, de fato, todo o desenvolvimento ágil de software. (KNIBERG, 2007, p. 17)

Ainda nesta questão está implícita a qualidade que não pode ser subestimada e negociada:

> Digamos que o product owner diga "OK pessoal, eu respeito sua estimativa de tempo de seis pontos de estória, mas eu tenho certeza de que vocês podem fazer algum tipo de quebra-galho para isso na metade do tempo se vocês se concentrarem nisso."
> [...] Minha experiência é que sacrificar qualidade interna é quase sempre uma idéia muito, muito ruim. O tempo economizado é largamente superado pelo custo, tanto em curto quanto em longo prazo. Uma vez que se permita que uma base de código se deteriore, é muito difícil recuperar a qualidade mais tarde.
> Ao invés disso, eu tento levar a discussão para o escopo. "Já que conseguir essa feature mais cedo é importante para você, podemos reduzir o escopo de modo que seja mais rápido implementá-la.[...] Uma vez que o product owner tenha aprendido que qualidade interna não é negociável, ele geralmente se especializa em manipular as outras variáveis. (KNIBERG, 2007, p. 18)

2.1.4 BACKLOG DO SPRINT

Na segunda parte da reunião, o Time, irá decompor os itens mais complexos em outros mais detalhados gerando uma lista de tarefas que é o Backlog do Sprint. Cada tarefa deve ser feita em menos de um dia a fim de facilitar a estimativa e o acompanhamento. (MARTINS, 2007, p. 264).

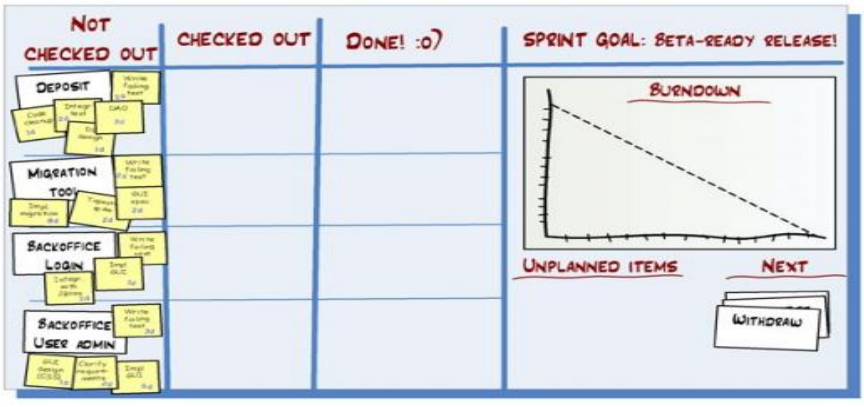
Como o Time é quem decide o que entrará no Sprint e não Dono do Produto, uma tensão estará em jogo: o Dono do Produto negociará com o Time para efetuar trocas de tarefas repriorizando-as ou decompô-as até adequá-las à estimativa do Time. (KNIBERG, 2007, p. 24)

Na Guia do Scrum, Schwaber (2009, p. 18-19), indica:

> A decomposição deve ser suficiente para que mudanças no progresso possam ser entendidas na Reunião Diária. O Time modifica o Backlog da Sprint no decorrer da Sprint, bem como surge Backlog da Sprint durante a Sprint. [...] À medida que se trabalha nas tarefas ou que elas são completadas, as horas estimadas de trabalho restantes para cada tarefa são atualizadas. Quando se verifica que determinadas tarefas são desnecessárias, elas são removidas. Somente o Time pode modificar o seu Backlog da Sprint durante uma Sprint. Somente o Time pode mudar o seu conteúdo ou as suas estimativas. O Backlog da Sprint é um retrato em tempo real altamente visível do trabalho que o Time planeja efetuar durante a Sprint, e ele pertence unicamente ao Time.

Kniberg (2007, p. 47) sugere que o formato mais produtivo para apresentar o Backlog do Sprint é um grande quadro de papel colado na parede para todos acompanharem.

Figura 2 - Backlog do Sprint

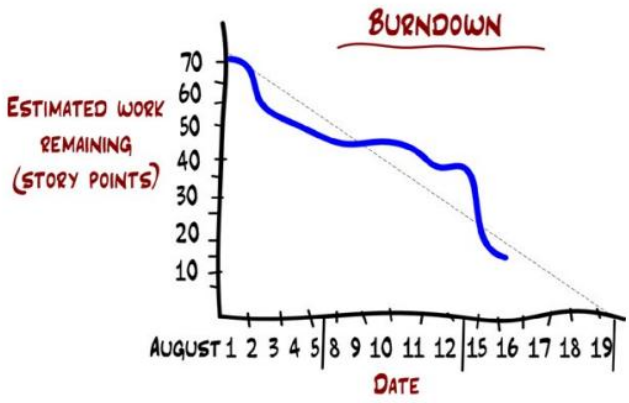


Fonte: Kniberg (2007, p.47)

Not checked out: a fazer; **Checked out**: fazendo; **Done**: feito; **Unplanned**: não planejados; **Next**: próximos. Refere-se aos itens que são completados antes do Sprint.

Burndown: gráfico de tendência do Sprint. Mostra o quanto de trabalho ainda há para fazer.

Figura 3 - Burndown



Fonte: Kniberg (2007, p.50)

2.1.5 SPRINT

De acordo com Martins (2007, p. 261), "Um *Sprint* é um conjunto de atividades de desenvolvimento conduzidas num período de tempo predefinido, chamado de *time box* (caixa de tempo), que normalmente varia de uma a quatro semanas."

A este propósito, a Guia do Scrum (2009, p. 10-11) orienta:

> [...] Durante a Sprint, o ScrumMaster garante que não será feita nenhuma mudança que possa afetar a Meta da Sprint. Tanto a composição do time quanto as metas de qualidade devem permanecer constantes durante a Sprint. As Sprints contêm e consistem na reunião de Planejamento de Sprint, o trabalho de desenvolvimento, a Revisão da Sprint e a Retrospectiva da Sprint. As Sprints ocorrem uma após a outra, sem intervalos entre elas.[...]
> Dica: Se o time sentir que se comprometeu com mais do que podia, ele se encontra com o Product Owner para remover ou reduzir o escopo do Backlog do Produto selecionado para a Sprint. Se o time sentir que sobrará tempo, ele pode trabalhar junto ao Product Owner para selecionar mais do Backlog do Produto.

Sobre o cancelamento de Sprint:

> As Sprints podem ser canceladas antes que o prazo fixo da Sprint tenha acabado. Somente o Product Owner tem a autoridade para cancelar a Sprint,
> embora ele possa fazê-lo sob influência das partes interessadas, do Time ou do Scrum Master.[...]
> Quando uma Sprint é cancelada, todos os itens do Backlog do Produto que estejam completados e "feitos" são revisados. Eles são aceitos se representarem um incremento potencialmente entregável. Todos os outros itens do Backlog do Produto são devolvidos ao Backlog do Produto com suas estimativas iniciais. Assume-se que qualquer trabalho realizado nesses itens é perdido. Cancelamentos de Sprints consomem recursos, já que todos têm que se reagrupar em outra reunião de Planejamento de Sprint para iniciar uma nova Sprint. Cancelamentos de Sprints são frequentemente traumáticos para o Time, e são muito incomuns. (GUIA DO SCRUM, 2009, p.11-12)

2.1.6 REUNIÃO DIÁRIA

Trata-se de uma reunião fundamental de inspeção e adaptação, de curta duração, 15 minutos, sempre no mesmo horário e local para manter as pessoas de pé, focadas nas seguintes perguntas: O que ele realizou desde a última reunião diária; O que ele vai fazer antes da próxima reunião diária; Quais obstáculos estão em seu caminho.

É dever do Scrum Master garantir que o Time realize sempre esta reunião. Outras pessoas podem participar, porém não estão autorizadas a falar e interferir. (GUIA DO SCRUM, 2009, p. 16)

2.1.7 REVISÃO DA SPRINT

É a apresentação da funcionalidade resultante da Sprint da seguinte forma:

> A reunião inclui ao menos os seguintes elementos. O Product Owner identifica o que foi feito e o que não foi feito. O Time discute sobre o que correu bem durante a Sprint e quais problemas foram enfrentados, além de como esses problemas foram resolvidos. O Time então demonstra o trabalho que está pronto e responde a questionamentos. O Product Owner então discute o Backlog do Produto da maneira como esse se encontra.[...] (GUIA DO SCRUM, 2009, p. 14)

2.1.8 REPROSPECTIVA DA SPRINT

Conforme descrito na Guida do Scrum (2009, p. 15):

> Após a Revisão da Sprint e antes da próxima reunião de Planejamento da Sprint, o Time Scrum tem uma reunião de Retrospectiva da Sprint. Nessa reunião, com duração fixa em três horas, o ScrumMaster encoraja o Time a revisar, dentro do modelo de trabalho e das práticas do processo do Scrum, seu processo de desenvolvimento, de forma a torná-lo mais eficaz e gratificante para a próxima Sprint.

2.1.9 PRONTO

Todos devem entender a definição de pronto! Não cabe presumir! A definição é:

> Um incremento completamente “pronto” inclui toda a análise, projeto, refatoramento, programação, documentação e testes para o incremento e todos os itens do Backlog do Produto no incremento. Os testes incluem testes de unidade, de sistema, de usuário e de regressão, bem como testes não-funcionais como de performance, de estabilidade, de segurança e de integração. “Pronto” inclui também qualquer internacionalização necessária. (GUIA DO SCRUM, 2009, p. 21)

2.1.10 PLANEJAMENTO DA VERSÃO PARA A ENTREGA

"Tipicamente, o planejamento de release é para nós uma tentativa de responder à questão "quando, no pior caso, nós seremos capazes de entregar a versão 1.0 deste novo sistema". (KNISBERG, 2007, p. 71)

2.1.11 REGRAS

Todas as descrições anteriores são as regras que fazem o elo entre os eventos com duração fixa, os papéis e os artefatos do Scrum, fundamentados nos pilares transparência, inspeção e adaptação. (GUIA DO SCRUM, 2009, p.3-4).

3 EXPERIMENTO

A presente pesquisa contou com a participação de cinco respondentes que atuaram nas equipes (em geral de 5 pessoas) no período de 2009 a 2012, delimitado quanto a aplicação do Scrum. Destes respondentes, o Scrum Master e o Product Owner estiveram do início até o fim do período. Os outros três foram os desenvolvedores, com larga experiência profissional (pleno e sênior) e com prática de Scrum anteriormente. O instrumento de pesquisa, dois questionários, foi enviado por meio de e-mail. Algumas perguntas foram encaminhadas exclusivamente para o Scrum Master e o Product Owner. Todos contactados responderam.
A lógica do primeiro questionário seguiu a seqüência dos eventos do Scrum para mapear a presença dos papéis, artefatos e regras. Em outro questionário o foco estava na auto-organização e na interdisciplinaridade que foram tratados separadamente na intenção de obter uma atenção exclusiva e não exaustiva, destes temas polêmicos na comunidade Scrum.

3.1 ANÁLISES DOS RESULTADOS

Analisando as respostas, pode-se dizer, que a maioria, afirmou as conclusões a seguir. Não houve treinamento de Scrum. O Time Scrum teve apenas o Scrum Master como mentor e treinador, tendo a sua atuação avaliada como satisfatória. Constatou-se a existência dos papéis do Scrum Master, do Product Owner e do Time. O Product Owner representava a "Câmara Gestora Rede Bem Estar", priorizava o Backlog do Produto e esclarecia as dúvidas.
A documentação seguiu o modelo ágil.

As Reuniões de Planejamento do Sprint foram poucas, indicando que o Time e o Product Owner interagiram pouco para determinar o Objetivo da Sprint e a Data de Entrega. Para o Scrum Master, nas poucas Reuniões de Planejamento do Sprint, apenas Objetivo da Sprint era determinado e o Backlog da Sprint nunca foi realizado, ou seja, o Time não planejava a Sprint, não tinha autonomia para decompor, estimar e negociar as tarefas com o Product Owner, sendo isto confirmado pela resposta do Scrum Master na Q18, "Estimadas pelo Scrum Master pelo perfil de cada membro da equipe." e na afirmativa da Q36, "O Scrum Master gerenciava o Time Scrum". O Time reafirmou esta declaração com as suas respostas.

O pequeno período de uma semana de Sprint (Q20, Q21 e Q32), sugere que o trabalho foi bem decomposto pelo Scrum Master, a fim de apresentar a Revisão da Sprint à "Câmara Gestora Rede Bem Estar" em partes funcionais até a conclusão do produto final definido e priorizado por ela. O seu cancelamento era raro. Para acompanhar as tarefas da Sprint foi utilizado o quadro do Scrum com as colunas Backlog, To do, Doing, To be Tested e Done, com os seus respectivos *post its*.

Os *bugs* freqüentemente eram incorporados na próxima Sprint.

O gráfico Burndown de Sprint foi utilizado raramente.

A definição de Pronto era freqüente e respaldada pelo testador (Q30 e Q31).

A Q26 e Q27 demonstraram que a relação da Data de Entrega e a qualidade da entrega das funcionalidades foram um sucesso freqüente.

A Reunião Diária deu-se freqüentemente e com eficácia (Q23 e Q24). Juntamente com a utilização do quadro do Scrum trouxe visibilidade ao andamento das tarefas da Sprint (iteração) e o desempenho da equipe.

A Revisão da Sprint foi pouco utilizada. Não houve Retrospectiva da Sprint.

As ocorrências das alterações de requisitos foram "regular" prevalecendo sobre a opção "constante" e "crítica". O seu impacto na produtividade considerado "moderado" entre as opções de "pouco significativa" e "crítica" (Q48 e Q49).

O segundo questionário concentrou-se na experiência do Time com a auto-organização e a interdisciplinaridade tentando "desvelar conteúdos implícitos, dimensões contraditórias e mesmo aspectos silenciados" não captados no primeiro questionário (LÜDKE; ANDRÉ, 1986 apud GIL, 2002, p. 134).

Apesar de o Scrum Master afirmar que houve a auto-organização identificando o nível como *self-organization*, todos do Time não se contradisseram ao negarem, pois escolheram o nível *Top-down* e não marcaram as características principais encontradas na auto-organização (A, B, D, F e G).

Importa esclarecer que não houve especialistas no Time (dba, programador específico, webdesigner), sendo assim, as opções de rotatividade foram de desenvolvedor, testador e suporte ao usuário. Segundo o Scrum Master, quando necessário a especialização de dba ou webdesigner, recorria-se a estes profissionais fora do Time que atendem a todas as demandas do setor.

Infere-se que o Time confirmou o interesse, a habilidade generalista e a facilidade para aplicar a interdisciplinaridade (B, D, Q6), porém, ninguém do Time marcou a característica "Os membros aprenderam novas habilidades através da rotatividade de trabalho" (C) indicando a fixação de função dos membros do Time.

4 CONCLUSÃO

Primeiramente, apresentaram-se as características e funcionamento do Scrum, sucedido pelas adaptações captadas pelos questionários. Atualmente, as adaptações do Scrum ao ambiente empresarial são conhecidas pelo termo Scrum-but (Scrum, mas) e discutidas acaloradamente pela comunidade Scrum (os dogmáticos e os pragmáticos), que indaga se ao efetuar um Scrum-but a equipe não pode tirar o máximo benefício proposto pelo Scrum.

Pham (2011, p.159) orienta que as adaptações ao ambiente deveriam ser encorajadas, mesmo correndo o risco de realizar Scrum-buts bons e ruins que são sutis nas diferenças. O dogmatismo pode tornar-se contraproducente.

Apello (2009, tradução nossa), ressalta que a retrospectiva proporciona o aprendizado essencial para diferenciar Scrum-buts bons e ruins e a boa compreensão do espírito ágil.

Cohn (2009, tradução nossa), discute a importância da auto-organização como princípio ágil, porém ressalta que não significa livre de todo controle de gestão, sendo exercido de forma sutil e indireta. Pham (2011, p.139) corrobora que o fator fundamental para o sucesso de qualquer projeto tradicional, ágil ou Scrum é o entrosamento da equipe.

A adaptação realizada transpareceu a falta de interdisciplinaridade e a fraca auto-organização devida à estrutura organizacional de "comando e controle", já que havia um coordenador (Scrum Master) e seus subordinados (Time). Do mesmo modo, a ausência da Retrospectiva da Sprint impossibilitou a equipe discutir, sugerir sobre as adaptações e o estímulo a idéias.

A equipe não considerou o Scrum aplicado como fator crítico para o sucesso do desenvolvimento do projeto. Apontou "A qualidade técnica da equipe", "A cooperação entre os membros da equipe" e "Acompanhamento gerencial" como fatores determinantes. Isto se deve a quebra do Scrum, respaldada pela Scrum CheckList de Kniberg (2009) e por

Sutherland (2009) que propõe o exercício da apreciação e da aceitação de opiniões cognitivas divergentes para transcender as restrições organizacionais e visões particulares. Trata-se de uma mudança lenta e dificílima a qualquer organização e pessoa ao genuíno Scrum!

Conclui-se, que a adaptação descrita expôs a dificuldade da compreensão e adoção correta dos princípios ágeis e a anomalia na distribuição e gestão das tarefas. Tais fatos não impediram o sucesso, contudo não atribuíram o ganho ao Scrum. Resultado da falta de maturidade organizacional a metodologia Ágil. Projetos futuros podem estender a utilização do framework e, face ao exposto, buscar a maturidade organizacional prevenindo as anomalias apontadas.

METHOD OF APPLICATION AGILE SCRUM IN THE SYSTEM SOFTWARE DEVELOPMENT "REDE BEM ESTAR" OF PREFEITURA MUNICIPAL DE VITÓRIA (ES).

ABSTRACT: The global market has required quality and productivity software. In response, information technology companies have sought to update knowledge and intelligence in applicability as a differential in the competition. The Agile Manifesto in 2001, was an innovative framework for software engineering to propose better ways of developing software. Disclosure of Agile – *Extreme Programming, Scrum, Feature Driven Development, Kanban* and others – brought a progressive growth of fans and expanded choosing the best approach to managing a project: classical or agile. *Scrum* has been outstanding for planning and project management, because the advantage of a light method, empirical and easy adjustment. This study sought to answer how *Scrum* can assist in the distribution and management of software development activities using the case study method with exploratory objective. Therefore, the research was to sample some team members involved in the project and the data collected from qualitative.

The results of the investigations have shown the difficulty of understanding and correct adoption of agile principles and anomalies that occur in this adaptation of *Scrum*. It is concluded that this adaptation did not serve as critical to the success of the project. Future research can cooperate in preventing the anomalies, addressing the adaptations – *Scrumbuts* – that have been discussed for quite *Scrum* community questioning its legitimacy and effectiveness.

Keywords: Project Management. Agile Methods. Scrum.

REFERÊNCIAS

AGILEFORALL. **Intro to Agile**. Disponível em :
< http://www.agileforall.com/intro-to-agile/ >. Acesso em: 4 jun. 2015.

AGILEMANIFESTO. **Manifesto for Agile Software Development**. Disponível em :
<http://www.agilemanifesto.org>. Acesso em: 4 jun 2015.

AGUANNO, Kevin. Chapter One: Introduction. In: _____. **Managing Agile Projects**. 1 ed. Canada: Multi-Media Publications Inc., 2005, p. 17.

AGUANNO, Kevin. Chapter Two: The Roots of Agility. In: _____. **Managing Agile Projects**. 1 ed. Canada: Multi-Media Publications Inc., 2005, p. 57-59.

APELLO, Jurgen. **ScrumButs Are the Best Part of Scrum**. Disponível em:
<http://noop.nl/2009/09/scrumbuts-are-the-best-part-of-scrum.html>. Acesso em: 9 jul. 2015.

COHN, Mike. **Succeeding with Agile:** Leading a Self-Organizing Team. Disponível em:<http://www.informit.com/articles/article.aspx?p=1382538>. Acesso em: 9 jul. 2015.

COMPUTERWORLD. **Metodologias de desenvolvimento**: qual a mais adequada?Disponível em:<http://computerworld.com.br/gestao/2009/08/05/metodologias-de-desenvolvimento-qual-a-mais-adequada>. Acesso em: 24 maio 2015.

DIAS, Marisa Villas Bôas. **Um novo enfoque para o gerenciamento de projetos de desenvolvimento de software**. 2005. 212 f. Dissertação (Mestre em Administração)-Universidade de São Paulo, São Paulo, 2005.

DOI, Takuo. **How to Build a Self-Organizing Team**. Disponível em:
<https://www.scrumalliance.org/community/articles/2015/april/how-to-build-self-organizing-team>. Acesso em: 9 jul. 2015.

GIL, Antonio Carlos. **Como elaborar projetos de pesquisa**. 4. ed. São Paulo: Atlas, 2002.

KNIBERG, Henrik. **Scrum Checklist**. Disponível em:
< https://www.crisp.se/gratis-material-och-guider/scrum-checklist>. Acesso em: 9 jul. 2015.

KOSCIANSKI, André; SOARES, Michel dos Santos. **Qualidade de software**: aprenda as metodologias e técnicas mais modernas para o desenvolvimento de software. 2.ed. São Paulo:Novatec Editora, 2007.

MARTINS, José Carlos Cordeiro. **Técnicas para gerenciamento de projetos de software**. 1.ed. Rio de Janeiro:Brasport, 2007.

MITTAL, Nitin. **Self-Organizing Teams**: What and How. Disponível em:
< https://www.scrumalliance.org/community/articles/2013/january/self-organizing-teams-what-and-how >. Acesso em: 9 jul. 2015.

PHAM, Andrew; PHAM Phuong-Van**. Scrum em ação**: gerenciamento e desenvolvimento Ágil de projetos de software. São Paulo: Novatec Editora, 2011.

SCHWABER, Ken. **Guia do Scrum**. Disponível em :
< http://www.scrumguides.org/download.html>. Acesso em: 5 jun. 2015.

SCHWABER, Ken. **Messagem from Ken**. Disponível em :
<http://www.controlchaos.com>. Acesso em: 4 jun. 2015.

SUTHERLAND, Jeff**. Scrum in Church**: Saving the World One Team at a Time. Disponível em:< http://jeffsutherland.com/SutherlandScruminChurchAgile2009.pdf>. Acesso em: 9 jul. 2015.

APÊNDICE A – Questionário A

QUESTIONÁRIO A

Seção 1 - Introdução

Esta pesquisa tem por objetivos identificar os fatores críticos de sucesso na abordagem de gerenciamento de projeto Scrum no desenvolvimento de software da Rede Bem Estar. Os dados aqui coletados serão utilizados única e exclusivamente em pesquisa acadêmica, sem qualquer finalidade comercial. Muito obrigada! (DIAS, 2005, p.175).

Silvana Ramos Buzetti - sil.vix@ig.com.br

Seção 2 – Qualificação do Respondente

Você já participou de algum projeto de desenvolvimento de software que utilizasse Método Ágil Scrum para a execução e/ou gerenciamento do projeto antes da RBE?

( ) Sim ( ) Não

Há quanto tempo você trabalha com desenvolvimento de software?

( ) 1 a 3 anos ( ) 4 a 7 anos ( ) 7 a 10 anos ( ) mais de 10 anos

Você já gerenciou uma equipe de desenvolvimento de software com mais de 3 pessoas?

( ) Sim ( ) Não

Seção 3 – Caracterização do Respondente

1. Qual era o seu cargo?

( ) Coordenador/Gerente ( ) Programador/Analista/Técnico ( ) Área da saúde

2. Quando você ingressou na equipe da Rede Bem Estar (mês/ano)?

3. Até quando você participou da equipe da Rede Bem Estar (mês/ano)?

Seção 4 – Caracterização do Scrum

1. Você recebeu treinamento sobre o Scrum para este projeto?

( ) Sim ( ) Não

2. Qual era seu Papel no Scrum?

( ) Scrum Master ( ) Product Owner ( ) Time

3. O Scrum Master forneceu ajuda para a adoção dos valores, práticas e regras do Scrum de forma:

( ) satisfatória ( ) insatisfatória ( ) Nenhuma

4. Havia o Product Owner?

( ) Sim ( ) Não

5. Havia o Backlog do Produto?

( ) Sim ( ) Não

6. O Product Owner priorizava os itens do Backlog do Produto?

( ) Sim ( ) Não

7. O Product Owner esclarecia todas as dúvidas?

( ) Sim ( ) Não

8. Havia a Reunião de Planejamento do Sprint?

( ) Não ( ) raramente ( ) pouco ( ) com freqüência

9. A duração da Reunião de Planejamento do Sprint:

( ) menos 8hr ( ) 8 hr ( ) mais 8hr

10. O Scrum Master sempre participava da Reunião de Planejamento do Sprint?

( ) Sim ( ) Não

11. O Product Owner sempre participava da Reunião de Planejamento do Sprint?

( ) Sim ( ) Não

12. Todo o Time sempre participava a Reunião de Planejamento do Sprint?

( ) Sim ( ) Não

13. Na Reunião de Planejamento do Sprint o Objetivo do Sprint era definido?

( ) Não ( ) raramente ( ) pouco ( ) com freqüência

14. Na Reunião de Planejamento do Sprint o Backlog da Sprint era definido?

( ) Não ( ) raramente ( ) pouco ( ) com freqüência

15. Na Reunião de Planejamento do Sprint a Data de Entrega era definida?

( ) Não ( ) raramente ( ) pouco ( ) com freqüência

16. O Time é quem decidia os itens do Backlog da Sprint?

( ) Não ( ) raramente ( ) pouco ( )com freqüência

17. O Time era auto-organizável?

( ) Sim ( ) Não

18. Como as tarefas do Backlog do Produto eram estimadas?

19. Como o Backlog da Sprint era apresentado?

( ) Planilha Excel ( ) Quadro de papel na parede ( ) Outro? _____

20. Qual o período de tempo do Sprint?

( ) 1 semana ( ) 2 semanas ( ) 3 semanas ( ) 4 semanas ( ) mais de 4 semanas

21. Houve cancelamento de Sprint?

( ) raramente ( ) pouco ( ) com freqüência

22. O gráfico Burndown de Sprint era usado?

( ) Não ( ) raramente ( ) pouco ( ) com freqüência

23. Havia a Reunião Diária?

( ) Não ( ) raramente ( ) pouco ( ) com freqüência

24. A Reunião Diária, em sua opinião, era eficaz?

( ) Sim ( ) Não

25. Havia a Revisão da Sprint?

( ) Não () raramente ( ) pouco ( ) com freqüência

26. A Data da Entrega era cumprida:

( ) Não ( ) raramente ( ) pouco ( ) com freqüência

27. A entrega das funcionalidades era um:

( ) sucesso ( ) sucesso parcial ( ) insucesso

28. Havia a Retrospectiva da Sprint?

( ) Não ( ) raramente ( ) pouco ( ) com freqüência

29. A Retrospectiva da Sprint, em sua opinião, era eficaz?

( ) Sim ( ) Não

30. Havia no Time uma pessoa responsável pelos testes?

( ) Sim ( ) Não

31. A definição de pronto era cumprida?

( ) Não ( ) raramente ( ) pouco ( ) com freqüência

32. As correções de bugs foram incorporadas no decorrer de outras Sprints?

( ) Não ( ) raramente ( ) pouco ( ) com freqüência

33. O Scrum Master incentivou o autogerenciamento do Time?

( ) Sim ( ) Não

34. O Scrum Master incentivou a interdisciplinaridade do Time?

( ) Sim ( ) Não

35. O Scrum Master incentivou o Time a ser mais produtivo?

( ) Sim ( ) Não

36. O Scrum Master gerenciava o Time Scrum?

( ) Sim ( ) Não

37. O Product Owner gerenciava o Time Scrum?

( ) Sim ( ) Não

38. Quantos participantes havia no Time?

39. De forma geral a opinião/sugestão do Time era respeitada pelo Scrum Master?

( ) Não ( ) raramente ( ) pouco ( ) com freqüência

40. De forma geral a opinião/sugestão do Product Owner era respeitada pelo Time Scrum?

( ) Não ( ) raramente ( ) pouco ( ) com freqüência

41. O Time manteve o bom ânimo com o projeto?

( ) oscilante ( ) freqüente

42. Os novos participantes do Time se adaptavam rapidamente ao Scrum?

( ) Sim ( ) Não

43. Os novos participantes do Time causaram declínio na produtividade do desenvolvimento?

( ) Sim ( ) Não

44. Você se adaptou fácil ao Scrum?

( ) Sim ( ) Não

45. Sobre a forma de adaptação do Scrum aplicada para este projeto?

( ) discordo totalmente ( ) discordo parcialmente

( ) concordo parcialmente ( ) concordo totalmente

46. Quais obstáculos foram encontrados pelo Time Scrum?

47. Você atribui ao Scrum um fator crítico para o sucesso do desenvolvimento de software da Rede Bem Estar?

48. Sobre a ocorrência de alterações na especificação dos requisitos:

( ) pouca ( ) regular ( ) constante ( ) crítica

49. As alterações na especificação dos requisitos impactavam na produtividade do desenvolvimento:

( ) pouco significativa ( ) moderada ( ) crítica

50. A documentação seguiu o padrão:

( ) modelo clássico ( ) modelo ágil

51. Marque os fatores que cooperaram para o sucesso do projeto:

1. A qualidade técnica da equipe ( )
2. A cooperação entre os membros da equipe ( )

3. Baixa rotatividade na equipe ( )
4. A presença do testador na equipe ( )
5. Clareza na especificação dos requisitos ( )
6. A participação dos clientes sempre que solicitados ( )
7. Processo de trabalho bem definido ( )
8. Flexibilidade no prazo de entrega ( )
9. No geral, havia pouca pressão por parte da gerencia e do cliente ()
10. As tarefas foram bem distribuídas na equipe ( )
11. Apoio gradual dos clientes devido a satisfação de suas expectativas ( )
12. Acompanhamento gerencial ( )

Fonte: Elaboração da autora, 2015.

APÊNDICE B – Questionário B

QUESTIONÁRIO B

Seção 1 - Introdução

Esta pesquisa tem por objetivos identificar os fatores críticos de sucesso na abordagem de gerenciamento de projeto Scrum no desenvolvimento de software da Rede Bem Estar. Os dados aqui coletados serão utilizados única e exclusivamente em pesquisa acadêmica, sem qualquer finalidade comercial. Muito obrigada! (DIAS, 2005, p.175).

Silvana Ramos Buzetti - sil.vix@ig.com.br

Seção 2 – Auto-organização e Multifuncionalidade

Definindo equipes auto-organizadas (self-organizing):

“Um grupo de indivíduos motivados, que trabalham juntos em direção a um objetivo, ter a capacidade e autoridade para tomar decisões e facilmente se adaptar às novas exigências”. MITTAL (2013, tradução nossa).

1. Marque as características auto-organizadas que a equipe vivenciou:

a) Eles puxam o trabalho por si mesmos e não esperam por seu líder para atribuir trabalho. Isso garante um maior senso de propriedade e comprometimento. ( )

b) Eles geriram o seu trabalho (alocação, realocação, estimativa, re-estimação, entrega e retrabalho) como um grupo. ( )

c) Eles ainda precisam de mentoramento e treinamento, mas eles não precisam de "comando e controle". ( )

d) Eles se comunicam mais com o outro, e os seus compromissos são mais freqüentemente com a equipe do que com o Scrum Master. ( )

e) Eles entendem os requisitos e não têm medo de fazer perguntas para obter as suas dúvidas esclarecidas. ( )

f) Eles continuamente melhoraram as suas próprias habilidades e recomendaram idéias e melhorias inovadoras. ( )

g) Os membros ajudavam uns aos outros sem sobrecarregar ninguém e sem ociosidade. ( )

h) O código era coletivo. ( )

Fonte: MITTAL (2013, tradução nossa).

2. Marque o nível de auto-organização da equipe:

( ) Chaos: Há membros da equipe, mas eles não colaboram bem.

( ) Top-down: Um dos membros (um líder) gerencia os outros membros. Apenas o líder tem a responsabilidade.

( ) Interdependent: Os membros compartilham responsabilidades.

( ) Collective: Os membros compartilham responsabilidades e as metas.

( ) Self-organizing: Os membros compartilham responsabilidades e as metas, e eles têm a autoridade para atingir a meta.

Fonte: DOI (2015, tradução nossa).

Definindo multifuncional/interdisciplinar (cross-functional):

“Todos contribuem, mesmo que isso exija aprender novas habilidades ou lembrar-se de antigas. Não há títulos em Times, e não há exceções a essa regra. O Time também não contém subtimes dedicados a áreas particulares como testes ou análise de negócios.” SCHWABER (2009, p. 8).

3. Marque as características multifuncionais que a equipe vivenciou:

a) Cada membro tinha sua especialidade definida na equipe. ( )

b) Houve interesse dos membros na multifuncionalidade. ( )

c) Os membros aprenderam novas habilidades através da rotatividade de trabalho. ( )

d) As habilidades dos membros indicavam um prévio conhecimento técnico generalizado de suas experiências profissionais. ( )

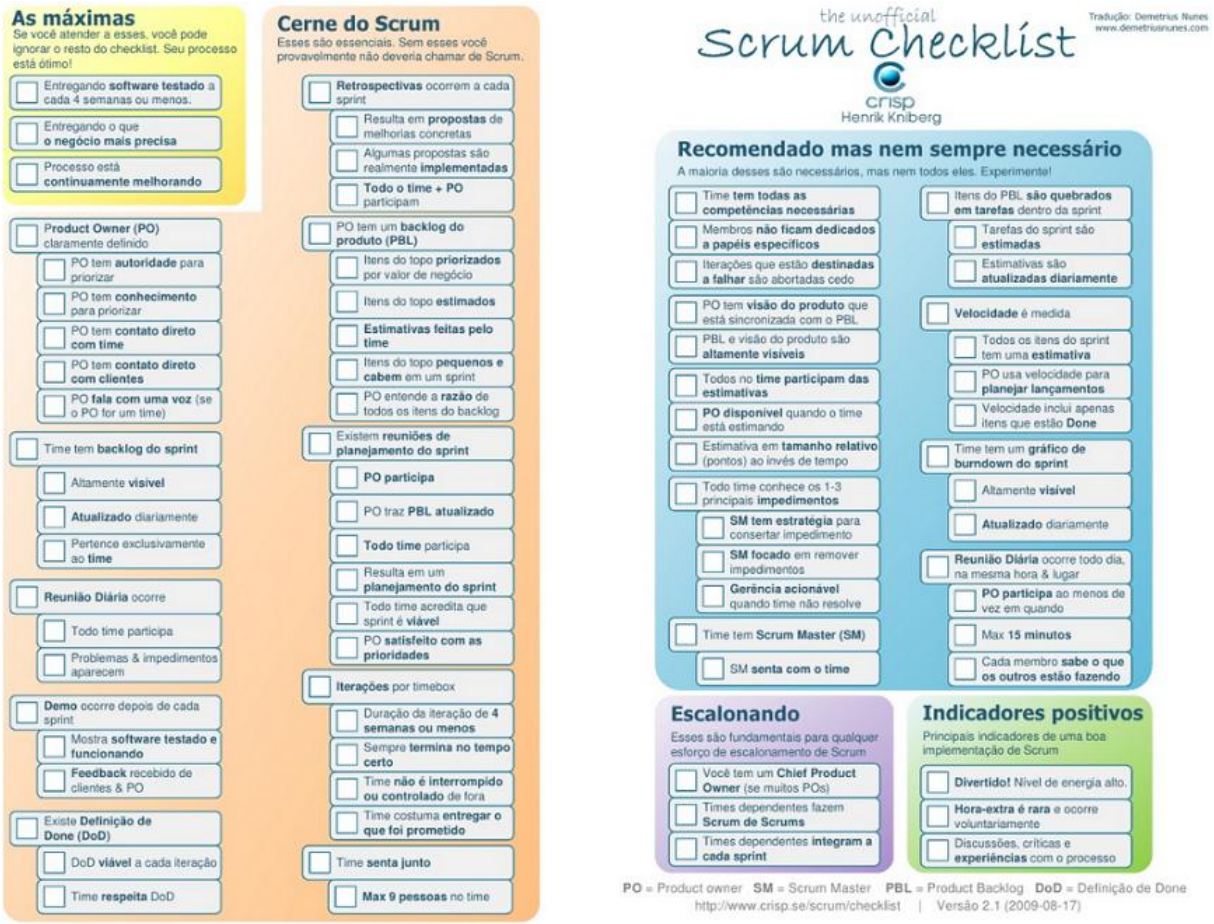
4. A multifuncionalidade proposta pelo Scrum era aplicável ao projeto RBE?

( ) desnecessária ( ) inviável

( ) com razoável facilidade ( ) com razoável dificuldade

Fonte: Elaborada pela autora, 2015.

ANEXO A – Scrum Checklist



Fonte: Kniberg, 2009.