AVALIAÇÃO DA ADOÇÃO DO *SCRUM* POR EMPRESAS DE *SOFTWARE* QUE COMERCIALIZAM SISTEMAS DE GESTÃO EMPRESARIAL

EVALUATION OF SCRUM ADOPTION BY COMPANIES OF SOFTWARE THAT COMMERCIALIZE BUSINESS MANAGEMENT SYSTEMS

Leandro Henrique Furtado Pinto Silva[1]

Liziane Santos Soares[2]

RESUMO:
Cada vez mais, cresce o número de organizações que buscam soluções de *software* com o intuito de automatizar seus processos de negócio. O segmento de sistemas integrados de gestão empresarial - SIG ou SIGE (em inglês, ERP - *Enterprise Resource Planning*) é bastante expressivo no contexto de desenvolvimento de software. SIGs são sistemas de informação que integram dados e processos de uma organização em um único sistema, envolvendo diferentes funções. Em geral, uma organização que desenvolve um SIG trata esse sistema como uma família de produtos, onde o sistema é vendido para diferentes clientes, podendo o sistema ser configurado e customizado para as diferentes organizações que o adquirem. Este trabalho visa à avaliar o desenvolvimento de software praticado por essas empresas. As soluções de software em questão consistem em produtos prontos, disponíveis para aquisição pelo cliente e estão em constante evolução para atender às necessidades de seus clientes e do mercado. O Scrum consiste em um método ágil de desenvolvimento. Possui como princípio, a entrega rápida do software e apresenta maior ênfase na comunicação verbal do que em formalizações de atividades e artefatos. Desta forma, para atender ao perfil de desenvolvimento das empresas focadas por esse trabalho, o Scrum requer adaptações. O resultado do trabalho consiste na proposição de um processo adaptado a partir do Scrum. As adaptações propostas para o Scrum visam a tratar os desafios identificados no tipo de desenvolvimento foco deste trabalho. As adaptações visam melhorar a capacidade do Scrum original para lidar, principalmente, com a gerência de requisitos e alteração, de forma que o escopo do software não seja comprometido. Além disso, as adaptações visam melhorar a capacidade do Scrum em realizar a gestão de configuração, tendo em vista que os sistemas em questão estão em constante evolução.
PALAVRAS-CHAVE: Processo de *Software; Scrum;* Gerência de Alteração; Gerência de Requisitos.

ABSTRACT:
Increasingly, there are more organizations looking for software solutions to automate their business processes. The segment of integrated enterprise management systems - SIG or SIGE (Enterprise Resource Planning) is very expressive in the context of software development. SIGs are information systems that integrate data and processes of an organization into a single system, involving different

[1] Cursa especialização em Engenharia de Software pela Pontifícia Universidade Católica de Minas Gerais, graduado em Sistemas de Informação pela Universidade Federal de Viçosa Campus Rio Paranaíba. Professor substituto na Universidade Federal de Viçosa Campus Rio Paranaíba. Currículo: http://lattes.cnpq.br/4534176583993364.

[2] Doutoranda em Ciências da Computação pela Universidade Federal de Minas Gerais, mestra e graduada em Ciência da Computação pela Universidade Federal de Viçosa. . Professora da Universidade Federal de Viçosa Campus Rio Paranaíba. Currículo: http://lattes.cnpq.br/6696452917505803.

functions. In general, an organization that develops a GIS treats this system as a family of products, where the system is sold to different customers, and the system can be configured and customized for the different organizations that acquire it. This work aims to evaluate the software development practiced by these companies. The software solutions in question consist of ready-made products, available for purchase by the customer and are constantly evolving to meet the needs of their customers and the market. Scrum is an agile development method. It has, as a principle, the fast delivery of software and presents a greater emphasis on verbal communication than on formalizations of activities and artifacts. Thus, to meet the development profile of companies focused on this work, Scrum requires adaptations. The result of the work is the proposition of a process adapted from Scrum. The adaptations proposed for Scrum aim to address the challenges identified in the type of development focus of this work. The adaptations aim to improve the original Scrum's ability to handle, mainly, the management of requirements and change, so that the scope of the software is not compromised. In addition, the adaptations aim to improve Scrum's ability to perform configuration management, given that the systems in question are constantly evolving.
KEYWORDS: Software Process; Scrum; Change Management; Requirements Management.

01 – INTRODUÇÃO

Cada vez mais, cresce o número de organizações que buscam soluções de *software* com o intuito de automatizar seus processos de negócio. Diante disto, as empresas que comercializam soluções de *software* vêm buscando processos que aliem a entrega mais rápida do produto ao cliente e uma maior qualidade do produto.

O processo de *software* consiste em um conjunto estruturado de atividades, que juntas serão necessárias para o desenvolvimento de *software*. Dentro do processo de *software*, algumas etapas são essenciais para o progresso e desenvolvimento do mesmo: Especificação, Projeto, Validação e Evolução. Segundo Sommerville (2007), nesse contexto, há também os modelos de processos de *software*, que são uma representação abstrata dos processos de *software*. Esses modelos apresentam a descrição de um processo a partir de uma perspectiva particular. São exemplos de modelos de processos de *software*: modelo cascata, desenvolvimento evolucionário e engenharia de *software* baseada a componentes e entre outros.

Grande parte dos principais processos de *software* têm como foco principal o desenvolvimento de sistemas ainda não implementados, ou seja, que partem de requisitos iniciais estabelecidos pelo cliente e requerem todas as fases de um projeto de *software* para serem construídos. Entretanto, várias empresas oferecem soluções prontas de *software*, as quais são adaptadas conforme as necessidades de cada cliente e também evoluídas conforme as necessidades do mercado.

O segmento de sistemas integrados de gestão empresarial - SIG ou SIGE (em inglês, ERP - *Enterprise Resource Planning*) é bastante expressivo no contexto de desenvolvimento de *software*. SIGs são sistemas de informação que integram dados e processos de uma organização em um único sistema, envolvendo diferentes funções (tais como finanças, contabilidade, recursos humanos, etc) e níveis da organização (nível operacional, gerencial, apoio à decisão). Em geral, uma organização que desenvolve um SIG trata esse sistema como uma família de produtos, onde o sistema é vendido para diferentes clientes, podendo o sistema ser configurado e customizado para as diferentes organizações que o adquirem.

Este tipo de desenvolvimento apresenta algumas características específicas, pois será focado na evolução constante de um *software* já existente. Desta forma, o processo de *software* a ser usado para tal desenvolvimento deve lidar com essas características específicas do desenvolvimento, provendo atividades especiais para lidar principalmente com a gerência de requisitos e gerência de alteração.

Esse modelo de negócio de *software* contempla empresas de diferentes portes. Mas, especificamente no caso de empresas menores, processos tradicionais com alto nível de formalização não são indicados. Nesse tipo de contexto, onde há limitações de recursos financeiros e de pessoal, é indicado um processo que apresente certo nível de agilidade.

Segundo Pressman (2010), a definição de *framework* é estabelecida através da designação de um pequeno número de atividades de enquadramento que sejam aplicáveis a todos os projetos de *software*, independentemente da sua dimensão ou complexidade. Nesse contexto, o *Scrum* é, atualmente, um *framework* de processo ágil e pode ser usado no contexto de empresas que visam um desenvolvimento mais eficiente e a entrega mais rápida do produto. Trata-se de um processo contendo número menor de papéis, artefatos e regras podendo ser adaptado para oferecer atividades específicas que lidem com necessidades inerentes a um determinado tipo de desenvolvimento. No caso de empresas que comercializam sistemas de gestão empresarial, uma questão a ser investigada é se o *Scrum* pode ser adaptado para oferecer não só agilidade, mas ênfase nas

características específicas desse tipo de desenvolvimento como gerência de requisitos, gerência de alteração e gerência de configuração.

02 – PROCESSO DE *SOFTWARE*

Um processo de *software* é um conjunto de passos que conduz à produção de um produto de *software*. Esses processos são complexos e, assim como todos os processos intelectuais e criativos, dependem de avaliação. Apesar de existir vários processos de *software* diferentes, algumas atividades são comuns a todos eles (SOMMERVILLE, 2007):

- **Especificação de *Software*** - Refere-se a funcionalidade do *software* e as restrições sobre suas operações.
- **Projeto e Implementação de *Software*** - O *software* deve atender às especificações para que seja produzido.
- **Validação de *Software*** - O *software* deve ser validado para garantir que ele faça o que o cliente necessita.
- **Evolução de *Software*** - O *software* deve estar em constante evolução para atender as necessidades mutáveis do cliente.

Um modelo de processo de *software* é a representação abstrata de um processo de *software*. Cada modelo atua como um processo sob determinada perspectiva e, assim, fornece apenas informações parciais em relação a esse processo. São exemplos de modelos de processo de *software*:

- **O Modelo Cascata** - Um modelo de processo de *software* sequencial, no qual o desenvolvimento é visto como um fluir constante para frente (assimilando a uma cascata), através das fases de e análise de requisitos, projeto, implementação, testes (validação), integração e manutenção de *software,* como representa a figura 1.
- **Desenvolvimento Evolucionário** - Modelo de processo de *software* que se baseia na ideia de desenvolvimento de uma implementação inicial, o qual é apresentada ao cliente que faz seus comentários. Posteriormente, a versão é ajustada e reapresentada ao cliente. A partir disso vão sendo geradas novas versões e

refinando-se por meio de várias versões até que seja desenvolvido um sistema adequado, como representa a figura 2.

• **Engenharia de *Software* Baseada em Componentes** - O modelo orienta-se em uma abordagem orientada ao reuso, que depende de uma grande base de componentes de *software* reusáveis e algum *framework* de integração desses componentes. A figura 3 ilustra o arcabouço do desenvolvimento em questão.

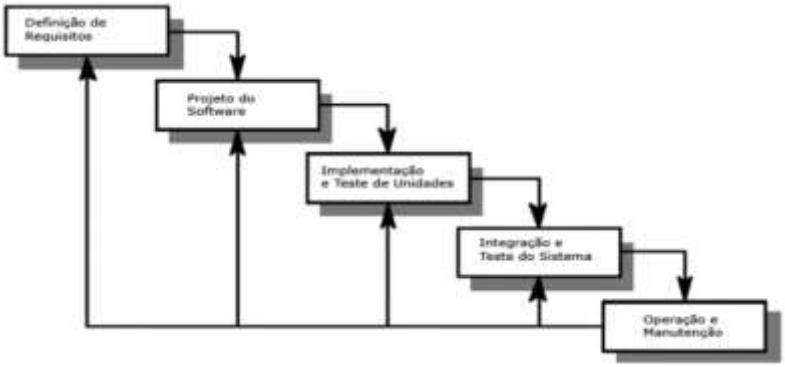

Figura 1: O modelo cascata. Fonte: Morais (2012).

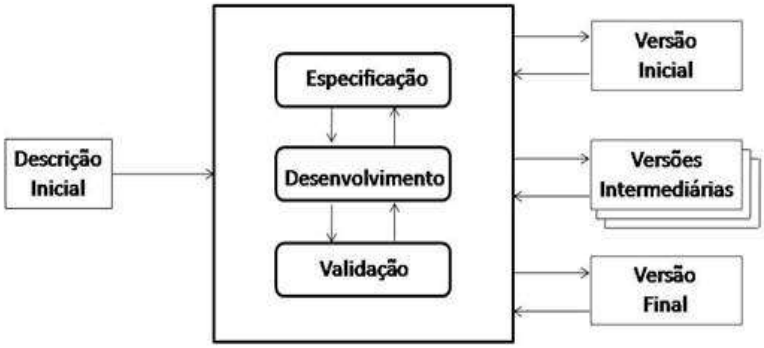

Figura 2: O desenvolvimento Evolucionário. Fonte: Sommerville (2007).

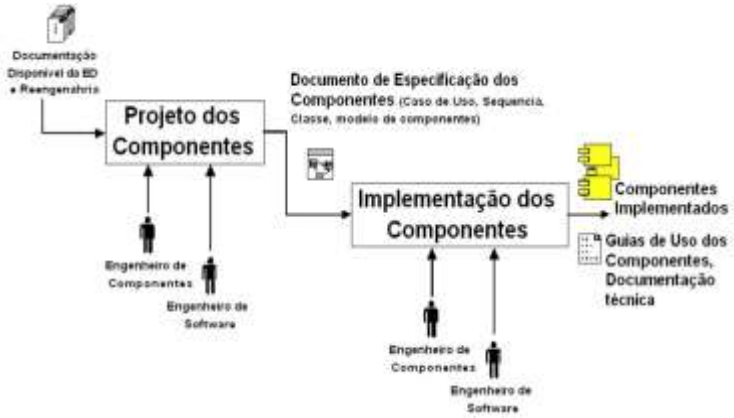

Figura 3: Engenharia de *Software* Baseada em Componentes. Fonte: Adaptado de Sommerville (2007).

• **O *Rational Unified Process* (RUP)** - Um exemplo de modelo de processo que foi derivado do trabalho sobre a *Unified Modeling Language* (UML) e do Processo Unificado de Desenvolvimento de *Software* associado. O RUP é um modelo de processo genérico e moderno, organizado em fases que aborda elementos de todos os modelos de processo de *software* e é usualmente descrito a partir de três perspectivas: (Kruchten, 2004).

1. Uma perspectiva dinâmica, que mostra as fases do modelo no decorrer do tempo.

2. Uma perspectiva estática, que mostra as atividades desenvolvidas durante o processo.

3. Uma perspectiva prática, que sugere as boas práticas a serem usadas durante o processo.

O RUP, ilustrado na figura 4, é um modelo formado por fases que identifica quatro fases discretas no processo de *software,* sendo elas (WAZLAWICK, 2004):

• **Concepção:** Etapa na qual o analista busca as primeiras informações do sistema a ser desenvolvido.

• **Elaboração:** Nessa etapa, busca desenvolver um entendimento do domínio do problema.

• **Construção:** Etapa que está intimamente relacionada com projeto, programação e teste do sistema.

• **Transição:** Ocorre após o último ciclo iterativo, quando depois de pronto, o sistema será implantado na empresa.

Esses modelos genéricos de processo são extensivamente usados nas práticas atuais de engenharia de *software*. Eles influenciam diversos processos de *software* e, muitas vezes, observamos a influência de mais de um desses modelos em um determinado processo, principalmente nos desenvolvimentos de sistemas de grande porte (SOMMERVILLE, 2007).

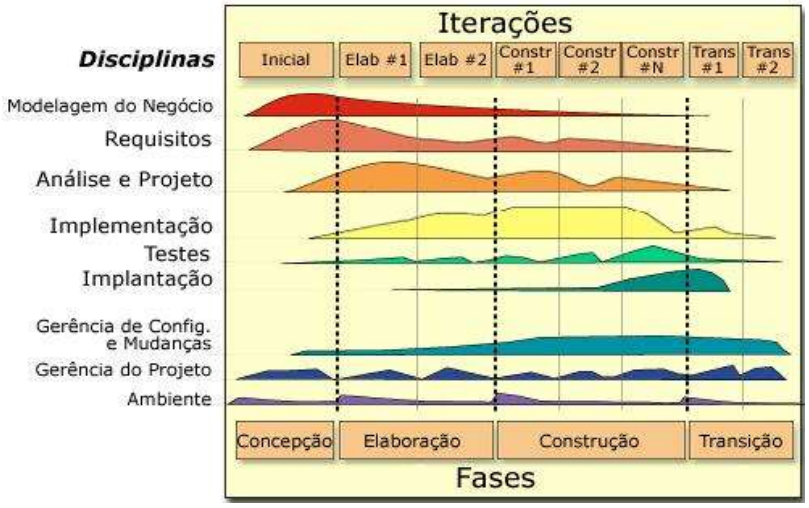

Figura 4: Esquema de Funcionamento do Processo de *Software.* Fonte: KROLL, P., & KRUCHTEN, P. (2003).

2.1 – Metodologias Ágeis

Os modelos ágeis de desenvolvimento de *software* vão seguir uma sistemática distinta em relação aos modelos prescritivos. Uma vez que não apresentam uma “receita de bolo”, com fases a serem executadas, mostram uma abordagem focada nos valores humanos e sociais.

Outro ponto de destaque é que apesar de os métodos ágeis serem frequentemente mais leves, não é certo abstraí-los como modelos de processos simples. O foco não é apenas a simplicidade, mas sim o constante objetivo maior no resultado do que no processo.

Uma filosofia ágil para a engenharia de *software* destaca quatro pontos chaves (PRESSMAN, 2010):

1. A importância de equipes auto organizadas que possuem controle sobre o trabalho que realizam.
2. Comunicação e cooperação entre os integrantes da equipe, os profissionais e seus clientes.
3. Um reconhecimento de que as alterações oriundas no projeto representam uma oportunidade.

4. Uma ênfase na entrega rápida de *software*s que satisfaçam ao cliente. Os modelos ágeis de processo foram projetados para atender a cada um desses tópicos.

Atualmente, um conjunto significativo de modelos é considerado ágil, sendo muitos deles diferentes entre si. Dentre os modelos ágeis mais conhecidos e representativos, temos: FDD (*Feature Driven Development*), DSDM(*Dynamic Systems Development Method*), XP (*eXtreme Programming*), *Cristal Clear*, ASDO (*Adaptive Software Development*) e *Scrum*, que é foco desse trabalho. (WAZLAWICK, 2013).

03 – REQUISITOS DE *SOFTWARE*

Requisitos de sistema são especificações dos serviços fornecidos pelo sistema e suas restrições operacionais. Os requisitos refletem as necessidades dos clientes. Certos problemas que aparecem no decorrer do processo de engenharia de requisitos são resultantes da falta de uma clara distinção entre esses diferentes níveis de descrição. Segundo Sommerville (2007), existe a distinção entre eles usando o termo requisitos de usuário para requisitos abstratos e de alto nível e requisitos de sistema para a descrição mais detalhada do que o sistema deve fazer.

Dentro dos requisitos de sistema de *software*, existe a classificação entre requisitos funcionais, requisitos não funcionais e requisitos de domínio (WAZLAWICK, 2004).

- Requisitos funcionais: são as operações que constituirão a funcionalidade do sistema.
- Requisitos não funcionais: são as restrições sobre as funções que o sistema oferece.
- Requisitos de domínio: são os requisitos vindos do domínio do sistema e que afetam as características e as restrições desse domínio.

3.1 – Gerência de Requisitos e Alteração

O Gerenciamento de requisitos é o processo para entender e controlar as alterações dos requisitos de sistema. Esse processo deve iniciar assim que uma versão inicial do documento de requisitos esteja pronto, mas o planejamento das mudanças de requisitos deve começar durante o processo de levantamento de requisitos (SOMMERVILLE, 2007).

O escopo é o elemento mais trabalhoso de se estabelecer da maneira correta e um dos mais importantes do ciclo de vida do projeto. Uma mudança muito drástica no escopo pode gerar grandes dores de cabeça à equipe de desenvolvimento. Mudanças no cronograma de atividades podem gerar grandes impactos nos documentos de gerenciamento, custo e riscos do projeto (LIMA, 2013).

Atualmente, no meio empresarial, as ferramentas e soluções de *software* são cada vez mais importantes e essências. Nesse contexto, é a ferramenta que se adapta ao processo da empresa e não o contrário. Diante disso, fica evidente a importância da implementação e gerencia de mudança de requisitos. As principais atividades deste processo são: identificação e registro da necessidade de mudança, análise de impacto e implementação da mudança (NASCIMENTO, 2014).

Um importante fator para identificar os requisitos impactados pela mudança é a rastreabilidade, que nada mais é que a associação entre uma ou mais entidades, sejam elas requisitos ou não. Outro fator fundamental e que ajuda consideravelmente na análise de impacto é possuir a rastreabilidade de forma total, do nível mais alto do negócio até o requisito em nível mais granuloso, e do requisito até os artefatos que o satisfaçam.

Durante o processo de análise da solicitação de alteração, a equipe procura por três tipos de impactos: inclusão, exclusão ou alteração de requisitos. Após encontrar e detalhar todos os requisitos que precisam ser alterados, a equipe tem capacidade de estimar o esforço e o custo da alteração e, também, determinar quais os outros pilares do projeto que serão afetados. Além disso, também é fundamental, verificar se a alteração acarretará novos riscos ou modifica a probabilidade ou impactos de riscos já existentes e identificados.

Por fim, é necessário decidir pela implementação ou não da alteração. Os seguintes fatores deverão ser avaliados: prazo, esforço, custo, impacto da não implementação da alteração, viabilidade do projeto caso a mudança seja aceita e todos os riscos inerentes (NASCIMENTO, 2014).

04 – RESULTADOS

Nesta seção será apresentado os resultados obtidos nesse trabalho, bem como a avaliação das proposições de adaptações no *Scrum* para o nicho envolvido.

4.1 – *Scrum* Original

De acordo com Cordeiro (2007), o *Scrum* pode ser definido como um processo bastante leve para organizar e planejar projetos de desenvolvimento de *software* e criação de produtos. O *Scrum* é uma metodologia ágil e possui uma estrutura iterativa e incremental. Ele se concentra no que de fato, é importante: gerenciar o projeto e criar um produto que acrescente valor para o negócio.

Os esquemas dão ênfase aos marcos do processo e ao ciclo de desenvolvimento que consiste no *Sprint*. Como um dos objetivos deste trabalho consiste na adaptação do *Scrum* para um nicho específico de desenvolvimento, é importante que possamos representar o *Scrum* em uma notação de processo, onde seja possível representar graficamente seus papeis, artefatos e atividades. Desta forma, a partir das diversas referências do *Scrum*, realizamos sua representação na notação BPMN (Figura 5).

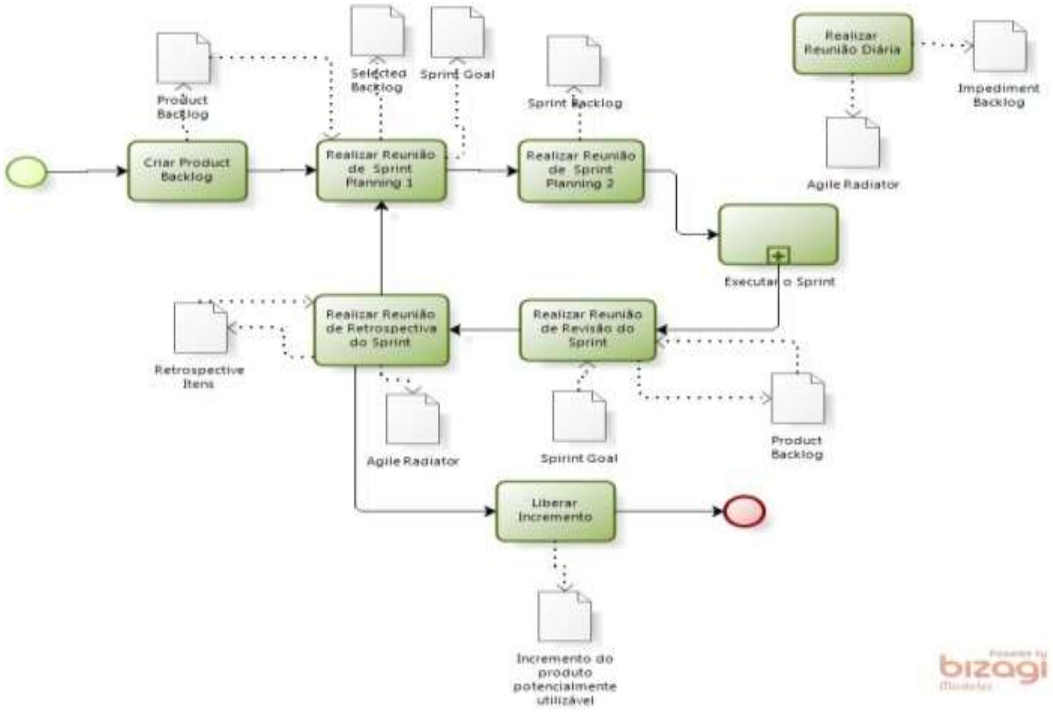

Figura 5: O *Scrum* representado em BPMN. Fonte: autores.

A última síntese sobre o *Scrum*, publicada como *"O guia do Scrum"* (SUTHERLAND; SCHWABER, 2013) tem sido atualizado ao longo do tempo para lidar com a crescente variedade de perguntas apresentada pela comunidade que utiliza este processo. Na atualização mais recente do guia, várias dicas e estratégias relacionadas ao uso do processo foram removidas, sendo mantidas apenas as regras básicas do *Scrum*. Segundo Jocham e Huizer (2011) isso ocorreu porque as versões anteriores do *Scrum* incluíam muitos detalhes sobre o que deveria ser feito ao longo do processo. A ideia por trás das mudanças é que o guia foque no porque fazer determinadas atividades no processo, deixando-o mais amplo e permitindo assim que cada equipe escolha a melhor abordagem de realizar uma determinadas atividade.

Na representação do *Scrum* optamos por incluir uma maior variedade de detalhes, provenientes das diferentes referências sobre o *Scrum* já citadas anteriormente. No diagrama da Figura 6, um dos artefatos é o Agile Radiator também conhecido como *Scrumboard* ou quadro de controle do *Scrum* (MONTAIN, 2015). Ele inclui o quadro de tarefas a serem realizadas no *Sprint*, o gráfico *Sprint*

Burndown (gráfico de trabalho restante) e também pode ser usado para listar itens não previstos que surgiram durante a realização do *Sprint.*

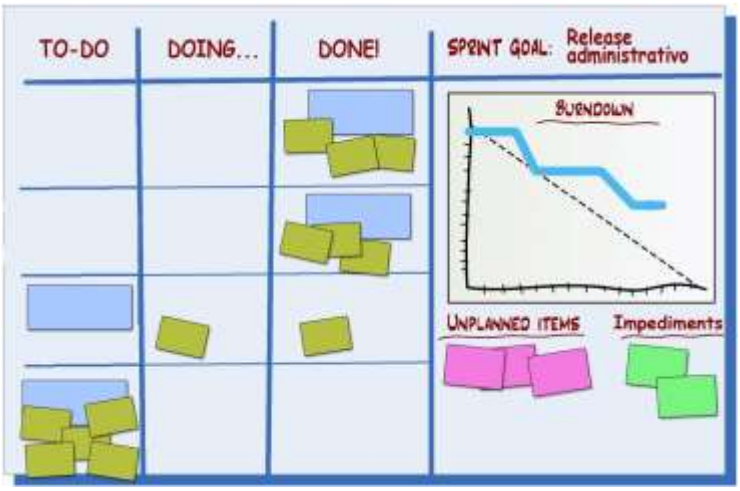

Figura 6: O Agile Radiator. Fonte: AKVG (2011)

É importante ressaltar que os diversos esquemas do *Scrum* disponíveis na literatura, sendo um deles ilustrado na figura 7, podem dar margem a uma interpretação equivocada sobre quais atividades ocorrem em ciclo dentro do processo. Os esquemas enfatizam um ciclo de desenvolvimento de até trinta dias, o que realmente ocorre, mas para cada *Sprint* são realizadas reuniões de *Sprint* Planning 1 e *Sprint Planning* 2. E ao final de cada ciclo são realizadas a Reunião de Revisão do *Sprint* e a Reunião de Retrospectiva do *Sprint*. Desta forma, dentro do processo, existem um conjunto de atividades que ocorrem em ciclos. São elas:

- **Realizar Reunião de *Sprint Planning* 1**
- **Realizar Reunião de *Sprint Planning* 2**
- **Executar o *Sprint***
- **Realizar Reunião de Revisão do *Sprint***
- **Realizar Reunião de Retrospectiva do *Sprint***

Sendo que o ciclo de desenvolvimento propriamente dito, onde os elementos do *Sprint Backlog* são implementados, está relacionado com a execução do subprocesso Executar o *Sprint*. Além disso, nos diversos materiais de referência do *Scrum*, o *Sprint* é descrito como sendo o ciclo de desenvolvimento onde o *Scrum Team* implementa todos os itens selecionados para a *Sprint*, presentes no *Sprint Backlog*. Por se tratar de um *framework* de processo, o *Scrum* não prevê um

conjunto rígido de atividades a serem realizadas durante o ciclo de desenvolvimento, deixando para cada equipe a opção de definir a melhor abordagem de desenvolvimento. Para tal, o time pode realizar projeto, codificação, testes de unidade, de aceitação e, até mesmo, documentação para cada item previsto para ser desenvolvido no *Sprint*. Por isso, optou-se por representar a atividade Execução do *Sprint* como uma macro atividade, isto é, um subprocesso, onde as descrições específicas das atividades que o compõem caberiam a cada equipe que utiliza o *Scrum*.

No diagrama da Figura 5, os papéis do *Scrum* não foram representados para não adicionar uma complexidade desnecessária. Os papéis fundamentais do *Scrum* são: *Product Owner*, *Scrum Master* e *Scrum Team* (equipe de desenvolvimento).

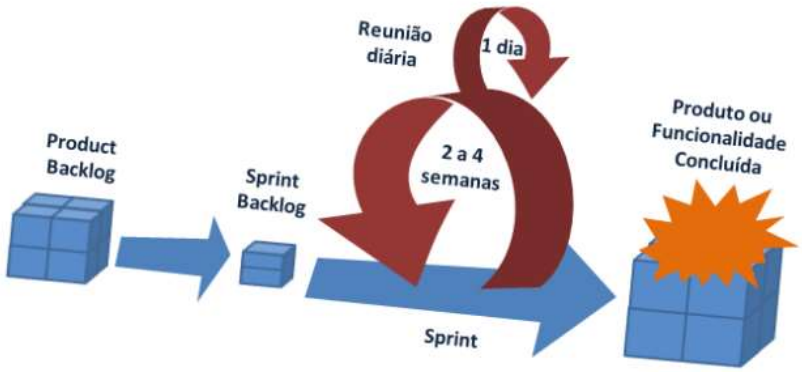

Figura 7: Estrutura básica do *Scrum.* Fonte: Zuliani (2015).

4.2 – *Scrum* Adaptado

Esta seção apresenta como o trabalho de adaptação do *Scrum* foi conduzido. Primeiramente o *Scrum* original foi analisado e representado graficamente para posteriormente, a partir da análise do nicho específico de desenvolvimento, serem feitas as propostas de alterações. Para realizar a representação dos processos, foi utilizada a notação BPMN – *Business Process Management Notation* e a ferramenta *Bizagi BPMN Modeler*.

Um questionário foi desenvolvido com o objetivo de ser usado para analisar o contexto de desenvolvimento das empresas que desenvolvem e

comercializam sistemas de gestão empresarial. O questionário contem doze perguntas relacionadas a questões como: características da equipe, processo usado para desenvolvimento, forma como as solicitações de alterações dos clientes são atendidas, desafios no desenvolvimento realizado entre outros. O questionário foi respondido por profissionais que trabalham em empresas desse segmento. Através dele, foi possível identificar ou confirmar vários desafios vivenciados por uma equipe que trabalha nesse nicho de desenvolvimento, com um produto em constante evolução.

Observa-se, por exemplo, que a pressão exercida pelo cliente influencia diretamente em como uma solicitação de alteração no produto é realizada, como podemos observar na figura 8 as respostas à segunda pergunta do questionário.

Figura 8: Segunda pergunta do Questionário. Fonte: autores.

Abaixo é apresentado o diagrama com o processo do *Scrum* adaptado (Figura 9) para o nicho envolvido nesse trabalho. As atividades marcadas com a cor verde já fazem parte do *Scrum* original e as demais são atividades inseridas e/ou adaptadas.

Também é apresentada a descrição de cada atividade, papel e artefato que compõem o processo resultante da adaptação do *Scrum* para o desenvolvimento de sistemas em questão. O processo adaptado abrange atividades específicas do *Scrum* e outras que foram inseridas no processo a fim de deixá-lo mais aderente ao nicho de desenvolvimento. Visando ainda atender a esse propósito, foram realizadas adaptações na definição de algumas atividades

específicas. Considerando o desenvolvimento realizado em empresas que atuam no segmento de sistemas integrados de gestão empresarial e também que essas empresas evoluem seus sistemas ao longo do tempo para atender às customizações solicitadas por seus clientes, trata-se então de um produto já pronto e em constante evolução. É importante notar que questões relacionadas com a gerência de requisitos, gerência de alteração e gerência de configuração são essenciais para se manter a integridade do produto e do seu escopo.

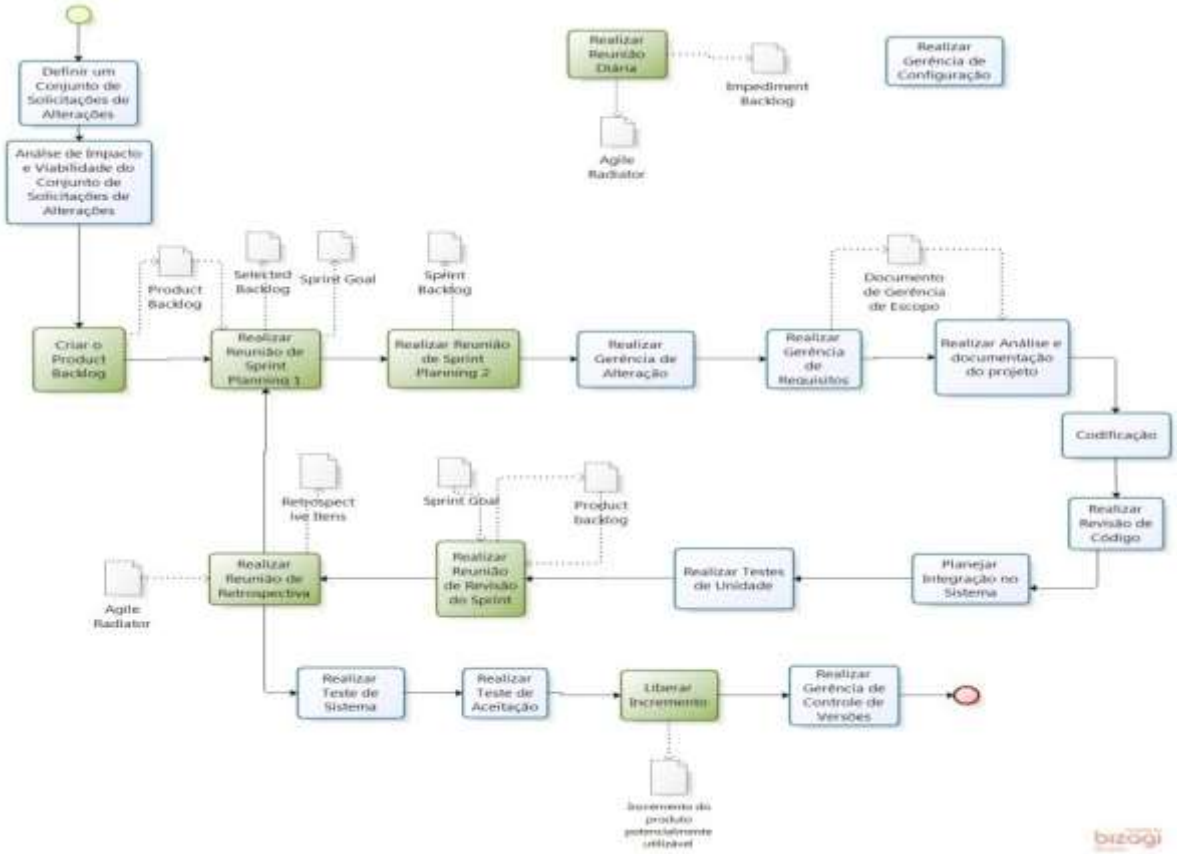

Figura 9: O processo *Scrum* Adaptado. Fonte: autores.

4.2.1 – *Atividades*

Definir o Conjunto de Solicitações de Alteração – Com base nas solicitações dos clientes, realizadas periodicamente, o objetivo dessa atividade é definir o conjunto de atividades a serem executadas em uma execução do *Scrum*. Assim, a equipe ou até mesmo um de seus membros deve ser responsável por

definir o conjunto de solicitações que serão analisadas e atendidas durante uma execução do *Scrum* e que podem dar origem a uma nova versão do produto.

Justificativa - Neste caso o produto já existe, então é necessário estabelecer o que deve ser desenvolvido em uma execução do *Scrum*, ou seja, estabelecer o que exatamente terá o papel de projeto a ser realizado através da execução do processo. Como todo o desenvolvimento a ser realizado consiste na produção de alterações no sistema, e essas solicitações são recebidas periodicamente pela empresa, então é necessário estabelecer o que será atendido posteriormente e o que será atendido de imediato, formando o *Product Backlog* do *Scrum*. Muitas vezes, as solicitações são tratadas como um projeto pela maioria das equipes de desenvolvimento, como mostra os resultados obtidos na aplicação do questionário. Mas as diferentes alterações podem implicar em diferentes proporções de trabalho. Desta forma, nem todas podem ser consideradas um projeto ao se utilizar o *Scrum*, lembrando que um *Sprint* do *Scrum* deve ter duração fixa de duas a quatro semanas. Nem sempre é possível se ter um conjunto de alterações onde cada uma gaste exatamente o tempo de um ciclo ou de uma execução do *Scrum* para ser implementada. Por isso, a definição prévia de um conjunto de alterações é necessária para criar a elaboração do *Product Backlog*.

Análise de Impactos e viabilidade do Conjunto de Solicitações de Alterações – A partir do conjunto de alterações definido na atividade anterior, é realizada uma análise inicial dos impactos diretos da mudança no sistema já pronto e, assim, decidir pela viabilidade das alterações. As alterações que não forem viáveis são retiradas do conjunto e outras alterações viáveis podem ser inseridas até que se chegue a um conjunto de alterações viáveis para serem desenvolvidas na execução do *Scrum*.

Justificativa – No nicho desse trabalho, o desenvolvimento é guiado pelas solicitações a serem realizadas pelos clientes. De acordo com as respostas à sétima pergunta do questionário, a pressão exercida pelo cliente impacta bastante em como cada solicitação será atendida. É importante que seja inserida uma atividade para gerenciar o atendimento dessas solicitações de forma que elas sejam realizadas com critérios, os quais devem preservar a integridade do produto e manter ser escopo. Alterações que não sejam viáveis no produto existente não devem ser

realizadas, mesmo que tenham sido solicitadas pelo cliente. Cabe à equipe apresentar as devidas justificativas para o cliente.

Criar o *Product Backlog* – Atividade já presente no *Scrum* original consiste em uma lista de todas as alterações a serem realizadas. Tais alterações podem consistir na implementação de novas funcionalidades ou adaptações de algumas funcionalidades já existentes. Podem ser parte do *ProductBacklog* as tarefas técnicas ou atividades diretamente relacionadas às funcionalidades solicitadas. Há também a necessidade de estabelecer e manter o acordo entre o cliente e a equipe do projeto nos requisitos variáveis da solicitação (SCHWABER, 2004).

Realizar Reuniões de *Sprint Planning* 1 – É uma reunião de planejamento do *Sprint*. Nela o *Product Owner* deve preparar o *Product Backlog* previamente a realização da reunião, priorizando os itens mais importantes para o negócio. Há a seleção dos itens do *Product Backlog* que o time faz o comprometimento de fazê-los. O *Product Owner* estabele o objetivo da *Sprint* (*Sprint Goal*), além disso, o time faz uma estimativa inicial do tamanho dos itens selecionados (SCHWABER, 2004).

Realizar Reuniões de *Sprint Planning* 2 – Essa reunião irá ocorrer imediatamente o término do *Sprint* Planning 1. Será de responsabilidade do *Product Owner* estar disponível para o time questionar sobre o *Product Backlog*. O time deve, assim, decidir entre si como os itens selecionados serão implementados. O time realiza um planejamento do próprio trabalho, refina a análise realizada no *Selected Backlog*, define as tarefas para se atingir cada item do *Selected Backlog* e gera o *Sprint Backlog* (SCHWABER, 2004).

Realizar a Gerência de Alteração – A atividade visa gerenciar as alterações a serem realizadas no produto, visando estabelecer um relacionamento entre as mudanças e o sistema já pronto, definindo, assim, o mecanismo para o gerenciamento de diferentes versões destes produtos, controlando as mudanças impostas, aditando e relatando as mudanças realizadas (PRESSMAN, 2010). Assim na atividade deverá se analisar se a solicitação de mudança é válida, quais requisitos serão afetados, estimar esforço e custo, identificar os riscos oriundos do conjunto de alterações, analisar a viabilidade da mudança e decidir pela sua

implementação ou não e, por fim, descrever a mudança dos requisitos (NASCIMENTO, 2014).

Justificativa – As alterações dentro dos projetos de desenvolvimento de sistemas geram, com frequência, dores de cabeça para as equipes, entretanto quando bem gerenciadas podem resultar em fator de sucesso no projeto, visto que os objetivos principais são atender as necessidades dos clientes e preservar o escopo (NASCIMENTO, 2014).

Realizar a Gerência de Requisitos – Far-se-á a identificação, organização e a documentação dos requisitos do conjunto de solicitações de alteração. Segundo o guia Mps.br o entendimento dos requisitos será obtido juntamente com os fornecedores de requisitos. A rastreabilidade bidirecional dentre os produtos de trabalho é estabelecida e também mantida, além disso é objetivo dessa atividade é revisar em planos e produtos de trabalho do projeto. Tudo isso irá identificar e corrigir possíveis inconsistências relacionadas aos requisitos (SOFTEX, 2013).

Justificativa – A equipe deverá obter o entendimento amplo dos requisitos relativos às alterações juntamente com o cliente e o *ProductOwner*. O propósito da atividade será gerenciar os requisitos do produto e dos componentes do projeto e, com isso, identificar incoerências entre os requisitos, os planos do projeto e os produtos de trabalho. A grande dificuldade, apontada no questionário, em identificar possíveis efeitos colaterais no sistema causados pela alteração realizada, embasa a atividade.

Realizar Análise e Documentação do Projeto – A atividade define a forma como planejar, construir e atualizar a estrutura do subsistema ou componente. Deverá construir um documento formal, além de atualizar todos os diagramas inerentes ao sistema (WAZLAWICK, 2004).

Justificativa – A atividade se faz necessária em virtude do nicho de desenvolvimento praticado pelas equipes de desenvolvimento abordadas nesse trabalho, onde os sistemas encontram em constante evolução. Assim, uma boa documentação proporcionará maior confiabilidade e facilidade aos futuros conjuntos de alterações a serem realizados.

Codificação – A atividade tem por objetivo realizar a implementação do código-fonte a fim de atender às solicitações de alterações dos clientes prevista no *Sprint Backlog*. No *Scrum*, cada *Sprint* é dedicado ao desenvolvimento de cada um dos incrementos do produto e, dessa forma, a codificação é uma das atividades necessárias ao processo de desenvolvimento de *software*, assim como a análise, projeto e validação (SOMMERVILLE, 2007).

Justificativa – Sommerville (2007), diz que a atividade de codificação faz parte do processo de desenvolvimento de *software* e não é diferente no *Scrum*, onde a atividade é realizada com intuito de desenvolver as tarefas previstas no *Sprint Backlog*.

Revisar Código – A atividade visa revisar o código para verificar a implementação.

Justificativa – A revisão de código-fonte é um complemento importante de outros mecanismos de qualidade, como a compilação, a integração e o teste (KRUCHTEN, 2004).

Planejar Integração no Sistema – O Objetivo da atividade é planejar a integração das alterações a serem implementadas na execução do *Scrum* com o sistema já pronto.

Justificativa – As dificuldades em identificar possíveis efeitos colaterais causados pela alteração, mostrados na pergunta de número 3 do questionário presente nesse trabalho, juntamente com dificuldade em projetar a solução para realizar as alterações, apontada pela pergunta de número 13 do questionário presente no Apêndice II, embasam a atividade de Planejar a Integração no Sistema.

Realizar testes de Unidade – O objetivo é explorar a menor unidade do projeto, com intuito de provocar falhas ocasionadas por deficiências de lógica e de implementação em cada um dos módulos em separado. O teste tem por alvo os métodos dos objetos ou ainda os pequenos trechos de código (NETO, 2014).

Justificativa – As dificuldades apontadas pela pergunta de número 13 do questionário presente nesse trabalho mostram que descobrir possíveis erros causados pela alteração realizada causa grande dificuldade, além disso, segundo a mesma pergunta do questionário, fica claro que a dificuldade em realizar teste na

alteração é ponto preponderante nas equipes que desenvolvem sistemas para esse nicho.

Realizar Reunião de Revisão do *Sprint* – Reunião realizada ao fim de cada *Sprint*, cuja responsabilidade é do *Scrum Master*. O objetivo de tal reunião é o time mostrar ao *Product Owner* e *Stakeholders* o que foi feito (SCHWABER, 2004).

Realizar Reunião de Retrospectiva do *Sprint* – Terá participação do Time, *Scrum Master* e, a critério, também o *Product Owner*. Os membros deverão responder duas questões: O que houve de bom durante o último *Sprint*? O que pode ser melhorado para o próximo *Sprint*?

Realizar testes de Sistema – Visa avaliar o conjunto de alterações realizadas na execução do *Scrum* concomitantes com o *software* já pronto em busca de falhas por meio da utilização do mesmo, como se fosse um usuário final. Desse modo, os testes são executados simulando os mesmos ambientes, com os mesmos dados de entrada e as mesmas condições (NETO, 2014).

Justificativa – Verificar se o produto pronto satisfaz os requisitos e também evitar possíveis efeitos colaterais oriundos do conjunto de alterações.

Realizar testes de Aceitação – Um grupo restrito de usuários do sistema irá testar o mesmo. Eles irão simular operações rotineiras do sistema, com a finalidade de verificar se o comportamento está adequado (NETO, 2014).

Justificativa – Revelar se haverá uma aceitação dos clientes em potencial que serão afetados diretamente pela alteração realizada. Havendo falhas nesse sentido, elas serão identificadas para que possam ser corrigidas pela equipe de desenvolvimento antes da entrega final.

Liberação do Incremento no Projeto – Atividade já presente no *Scrum* original visa a partir dos itens do *Sprint Backlog*, entregar um Incremento no Produto, que representa valor visível para os clientes do projeto. Assim deverá liberar uma nova versão do produto, contendo as alterações realizadas durante a execução do *Scrum* (SCHWABER, 2004).

Realizar Gerência de Controle de Versões – Deverá haver identificação, armazenamento e gerenciamento dos itens de configuração e também de suas versões durante todo o ciclo de vida do *software*. Um histórico deverá ser desenvolvido contendo todas as alterações realizadas nos itens de configuração.

Deverá criar rótulos e ramificações no projeto, além de possibilitar a recuperação de uma configuração em um determinado momento desejado do tempo (PRESSMAN, 2010).

Justificativa – A atividade, a qual fundamental para o desenvolvimento de *software*, apoia as atividades de controle de mudança e a integração contínua com o sistema já pronto. As respostas obtidas no questionário aplicado, nos remete que as equipes possuem um controle especial quanto ao controle das versões geradas aos clientes.

Realizar Gerência de Configuração – O objetivo da atividade é garantir a integridade dos itens de configuração. A gerência de configuração identifica a configuração dos produtos de trabalho selecionados que um determinado ponto no tempo, controla as alterações nos itens de configuração, registra e relata o status do processo de mudança e o de implementação e verifica a aderência aos requisitos específicos (PRESSMAN, 2010).

Justificativa – Com objetivo de garantir a qualidade do *software*, em função das modificações constantes que o mesmo sofre, faz-se necessário o uso da Gerência de Configuração que irá controlar e gerenciar a evolução de um *software* através do controle formal de versão e solicitações de mudanças (KALINOWSKI et al., 2014).

Realizar Reuniões diárias – Tem como objetivo disseminar conhecimento sobre o que foi feito no dia anterior, identificar impedimentos e priorizar o trabalho a ser realizado no dia que se inicia (SCHWABER, 2004).

4.2.2 – *Papéis*

Abaixo são descritos os papeis presentes na definição do *Scrum*. Alguns papeis tiveram sua definição ajustada para realizar a adaptação do *Scrum*.

Product Owner – O papel de *productOwner* deverá ser desempenhado por um membro da equipe que domina bem o escopo do produto já desenvolvido e que acompanha continuamente o produto, assim como as solicitações de alterações de clientes feitas no decorrer da vida do produto. Em geral, esse membro da equipe

pode ser um gerente ou alguém que esteja envolvido com a análise das solicitações de alteração do cliente (SCHWABER, 2004).

Scrum Master – É o responsável por proteger o processo e, assim, assegurar que as práticas sejam utilizadas com disciplina. Além disso, ele é responsável por assegurar o intercalo entre o projeto e a cultura organizacional. Tem por responsabilidade também organizar as reuniões, responder pelo *Scrum Team* perante os demais *Stakeholders*. Sendo assim, o *Scrum Master* exerce um gerenciamento do tipo “coach”, atuando ao lado do *Scrum Team* e acompanha o progresso do trabalho dia a dia (SCHWABER, 2004).

Scrum Team – São os responsáveis por estimar o esforço para os itens presentes no *Backlog* selecionados para o *Sprint*, bem como a expansão dos itens do *SelectedBacklog* para o *Sprint Backlog*. Gerenciam seu próprio trabalho periodicamente, atualiza o *AgileRadiator* e o Burnsown Chart. Outras atribuições do *Scrum Team* são identificar e reportar impedimentos ao *Scrum Master* e desenvolver o projeto (SCHWABER, 2004).

4.2.3 – *Artefatos*

Abaixo são descritos os artefatos presentes na definição do *Scrum* Adaptado. Alguns artefatos tiveram sua definição ajustada e outro incluído para realizar a adaptação do *Scrum*.

Product Backlog – Pilha de requisitos que contém as necessidades de todos os *Stakeholders*, de preferência descrita em linguagem do usuário. Sendo mantida pelo *ProductOwner*, que deve aceitar todas as ideias e reclamações que tenham sentido para o produto, por parte de qualquer um dos *Stakeholders*. A pilha é ordenada do mais importante no topo da pilha para o de menos importância, tudo isso embasado em seu valor para o negócio da empresa. As atualizações são periódicas, devendo os itens de *ProductBacklog* estarem tão mais detalhados a medida que são mais prioritários na pilha. (SCHWABER, 2004).

Selected Backlog – Não consiste em uma nova pilha, mas sim de apenas um nome distinto para uma seleção de itens de *ProductBacklog* feita para um *Sprint* (SCHWABER, 2004).

Sprint Backlog – É referenciada por uma pilha de tarefas (ou atividades), definida por parte do time e para o time. A pilha irá conter um detalhamento, em dois ou três itens de *Sprint Backlog*, sobre o que será feito em um *Sprint* para cada item de *SelectedBacklog*. Também é utilizada para o *Sprint Burndown* Chart e não deve ser utilizada para um gerenciamento externo, ou seja, realizado por resultados (SCHWABER, 2004).

Impediment Backlog – Consiste em uma pilha de impedimentos, sendo que qualquer evento que impeça o avanço do *Scrum Team*. A ordem é definida por criticidade, com os itens que param o trabalho sendo colocadas no topo da pilha. É mantido pelo *Scrum Master*, que tem por responsabilidade por agir e remover cada impedimento dentro de 24 horas (SCHWABER, 2004).

***Retrospective* Itens** – Refere-se a lista de WWW (em português – o que foi bem) e WCBI (em português – o que pode ser melhorado), separadas por responsabilidade, pode ser “organizacional” ou “time". A reponsabilidade de agir para a lista organizacional é do *ProductOwner* e a lista "time" de responsabilidade do *Scrum Team* (SCHWABER, 2004).

Sprint Burndown – Representa o gráfico contendo o “restante do trabalho” dentro de um determinado *Sprint*, baseando-se na pilha de *Sprint Backlog*. A atualização é feita diariamente, mediante atualização do Daily*Scrum*, pelo *Scrum Master*. Tudo isso visa refletir o avanço do *ScrumTeam* naquele dia (SCHWABER, 2004).

Documento de Gerência de Escopo – O objetivo deste artefato é auxiliar a gerência dos requisitos, bem como a definição e a rastreabilidade dos mesmos, a adaptação das mudanças ocorridas nos requisitos e o gerenciamento de tais mudanças. Engloba ainda a definição de diretrizes, com intuito de validar o Escopo junto com os clientes, além de haver um controle das entregas definidas. O resultado são elementos importantes para a criação, validação e também o controle do Escopo do projeto (LIMA, 2013). O gerenciamento do escopo do projeto será realizado embasado em dois documentos específicos: Especificação do escopo para o escopo funcional do projeto e a Estrutura Analítica do Projeto - EAP para o escopo das atividades que serão realizadas pelo projeto, juntamente com suas devidas entregas. Todas as mudanças no escopo previstas inicialmente para o projeto serão

avaliadas e classificadas dentro do sistema de controle de mudanças de escopo e, assim, vão ser consideradas mudanças de escopo apenas as medidas de correção. As inovações e novas características do projeto não serão consideradas pelo gerenciamento do escopo. Todas as solicitações de mudança no escopo devem ser feitas por escrito ou também via *e-mail*, conforme descrito no plano de gerenciamento das comunicações do projeto presentes no PMBOK (2000).

Justificativa – Com o volume de alterações que o sistema passa durante seu ciclo de vida, a gerência de Escopo irá garantir que tais alterações não modifiquem radicalmente a estruturas do sistema e seu Escopo. O Controle do escopo permite a flexibilidade possível ao projeto e também a rigidez que o mesmo necessita, para que no fim o produto que será entregue possa estar de acordo com o esperado e o projeto que o fornece tenha tido andamento adequado (LIMA, 2013).

4.3 – Considerações

Em linhas gerais, as equipes de desenvolvimento *Scrum* são compostas por cinco a nove membros, representando um número relativamente pequeno. Sendo assim, uma alternativa viável a ser destacada para equipes de maior porte é o chamado "*Scrum* de *Scrum*s", que consiste em diversas equipes trabalhando em prol de um mesmo projeto. Cada equipe identifica um membro que será responsável por participar da reunião de "*Scrum* de *Scrum*", que consiste em coordenar e direcionar o trabalho das várias equipes envolvidas no projeto. Essas reuniões são análogas às reuniões diárias do *Scrum*, mas não necessariamente acontecem todos os dias. Em suma, seriam formadas "equipes de equipes" (MONTAIN, 2015).

Abaixo a figura mostra uma analogia do "*Scrum* de *Scrum*s" e também como tal abordagem permite ampliar o tamanho da equipe *Scrum*. Cada célula representa um membro da equipe *Scrum*. Uma pessoa de cada equipe (a célula de cor diferente) será responsável por coordenar o trabalho dessa equipe e participar da equipe de reunião de "*Scrum* de *Scrum*s". Em seguida, outra pessoa é selecionada dentro dessas equipes (a célula que possui cor diferente e também é marcada por um ponto) para formar outra equipe para coordenar o trabalho das demais. Tais equipes vão se formando de acordo com a necessidade envolvida no

projeto e o tamanho do time de desenvolvimento, a fim de serem aderentes ao *Scrum* (MONTAIN, 2015).

O progresso em um projeto *Scrum* pode ser monitorado por meio de um gráfico de release *Burndown*. O *Scrum Master* deve atualizar o gráfico *Burndown* ao fim de cada *sprint*. É importante ressaltar que o release *Burndown* é para uma execução completa do *Scrum*. (MONTAIN, 2015).

O eixo horizontal do gráfico de *sprint Burndown* representa os *Sprint*s; o eixo vertical mostra a quantidade de trabalho a serem feitos no início de cada *Sprint*, como representa a figura 11 (MONTAIN, 2015).

Este tipo de representação *Burndown* se adere muito bem em muitas situações e para muitas equipes. O gráfico *Burndown* é uma parte preponderante de qualquer projeto ágil e também é uma forma de a equipe ter clareza sobre o que está acontecendo e como o progresso está sendo feito durante cada *sprint* (MONTAIN, 2015). No nicho das empresas, foco deste trabalho, o gráfico seria uma alternativa para o controle de versões as quais serão desenvolvidas durante o ciclo de vida do produto.

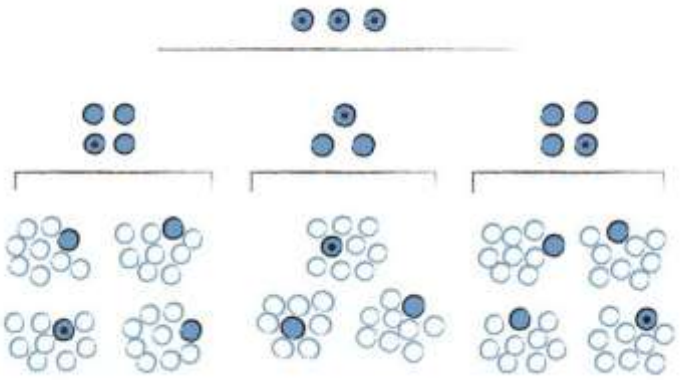

Figura 10: Analogia do processo *Scrum* de *Scrum*s. Fonte: Montain (2015)

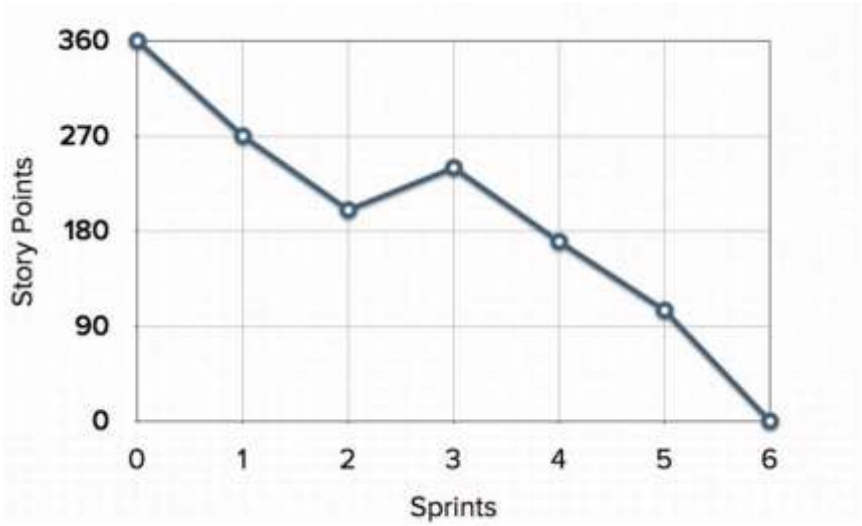

Figura 11: O gráfico release borndown. Fonte: Montain (2015)

05 – CONCLUSÃO

O desenvolvimento do trabalho indica que o *Scrum* pode ser adaptado para ser utilizado por empresas que desenvolvem sistemas de gestão empresarial desde que algumas adaptações sejam feitas nos elementos do processo, e algumas premissas sejam respeitas.

Em uma na análise realizada durante o desenvolvimento do trabalho, percebe-se que o *Scrum* adaptado seria aderente a empresas que possuam sistemas de gestão e equipes de tamanho médio ou inferior. Além disso, é necessário definir o conjunto de alterações solicitadas pelos clientes que cumprirá o papel do projeto a ser cumprido durante uma execução do *Scrum*, a qual dará origem a uma nova versão do produto a ser liberada para os clientes.

06 – REFERÊNCIAS

AKVG. 2011. *Abordagens para agilizar o desenvolvimento. Disponível em: <http://akvg.com.br/project-view/project-no-7/>. Acesso em 30 out. 2017.*

CATUNDA, E. et al. *Implementação do nível f do mr-mps com práticas ágeis do scrum em uma fábrica de software*. SBQS2011, 2011.

CORDEIRO, J. C. *Técnicas para Gerenciamento de Projetos de Software*. Rio de Janeiro: Brasport, 2007.

DESENVOLVIMENTO ágil de Software: Scrum. 2015. Disponível em: <http://www.desenvolvimentoagil.com.br/scrum>. Acesso em: 30 out. 2017.

JOCHAM, R.; HUIZER, H. J. *Gone are release planning and the release burndown*. 2011. Disponível em: <https://www.scrum.org/About/All-Articles/articleType/ArticleView/articleId/17/Gone-are-Release-Planning-and-the-Release-Burndown>. Acesso em: 30 out. 2017.

KALINOWSKI, M. et al. *Gerência de configuração*. Engenharia de Software Magazine, 2014. Disponível em: <http://www.researchgate.net/publication/256091444_Gerncia_de_Configurao_Definies_Iniciais_Ferramentas_e_Processo>.

KROLL, Per; KRUCHTEN, Philippe. T*he rational unified process made easy: a practitioner's guide to the RUP*. Addison-Wesley Professional, 2003.

KRUCHTEN, P. *The rational unified process: an introduction*. [S.l.]: Addison-Wesley Professional, 2004.

LIMA, F. de. *Pmbok: Gerenciamento de escopo de projetos*. Engenharia de Software Magazine, 2014.

LOH, T. C.; KOH*, S. *Critical elements for a successful enterprise resource planning implementation in small-and medium-sized enterprises*. International journal of production research, Taylor & Francis, v. 42, n. 17, p. 3433–3455, 2004.

MINDMASTER: Scrum: *A metodologia Ágil explicada de forma definitiva*. 2015. Disponível em: <http://www.mindmaster.com.br/scrum/>. Acesso em: 30 out. 2017.

MODEL, B. P. *Notation (bpmn) version 2.0.* OMG Specification, Object Management Group, 2011.

MONTAIN Goat Software: *Scrum overview for agile software development*. 2015. Disponível em: <https://www.mountaingoatsoftware.com/agile/scrum/overview>. Acesso em: 30 out. 2017.

MORAIS, MARCOS. 2012. *Engenharia de Software.* Disponível em: <http://marcosmoraisdesousa.blogspot.com/2012/04/engenharia-de-software.html>. Acesso em: 15 out. 2017.

NASCIMENTO, F. *Processo de mudança de requisitos na prática. Engenharia de Software Magazine*, 2014.

NETO, A. C. D. *Introdução a teste de software*. Engenharia de Software Magazine, 2014. Disponível em: <http://www.devmedia.com.br/artigo-engenharia-de-software-introducao-a-teste-de-software/8035>.

NOTATION, B.-B. P. M. *Bizagi Process Modeler*. [S.l.]: Copyright, 2011.

OLIVEIRA, H. V. F. de. *A importância da engenharia de requisitos*. Engenharia de Software Magazine, 2014.

PADUELLI, M. M. *Dificuldades e problemas encontrados na manutenção de softwares*. Engenharia de Software Magazine, 2014.

PMBOK, A. *Guide to the project management body of knowledge*. Project Management Institute, Pennsylvania USA, 2000.

PRESSMAN, R. S. *Engenharia de Software*. São Paulo: AMGH, 2010.

SANTOS, D. V. dos. *Abordando a engenharia de requisitos no mps.br*. Engenharia de Software Magazine, 2014.

SCHWABER, K. *Agile project management with Scrum*. [S.l.]: Microsoft Press, 2004.

SILVA, F.; HOENTSCH, S. C.; SILVA, L. *Uma análise das metodologias ágeis fdd e scrum sob a perspectiva do modelo de qualidade mps. br*. Scientia Plena, v. 5, n. 12, 2009.

SOFTEX: *Mps.br - melhoria de processo do software brasileiro - guia geral mps de software.* 2013. Disponível em: <http://www.softex.br/wp-content/uploads/2013/07/MPS.BR_Guia_Geral_Software_2012-c-ISBN-1.pdf>. Acesso em: 04 nov. 2017.

SOFTEX: *Guia de implementação – parte 1: Fundamentação para implementação do nível g do mr-mps-sw. 2013. 2013.* Disponível em: <http://www.softex.br/wp-

content/ uploads/2013/07/MPS.BR_Guia_de_Implementacao_Parte_1_2013.pdf>. Acesso em: 04 nov. 2017.

SOFTEX: *Guia de implementação – parte 2: Fundamentação para implementação do nível f do mr-mps-sw. 2013. 2013.* Disponível em: <http://www.softex.br/wp-content/uploads/2013/07/MPS.BR_Guia_de_Implementacao_Parte_2_20131.pdf>. Acesso em: 10 nov. 2017.

SOMMERVILLE, I. *Engenharia de Software*. São Paulo: Pearson Educarion do Brasil, 2007.

SUTHERLAND, J.; SCHWABER, K. *The scrum guide. the definitive guide to scrum: The rules of the game*. Scrum. OrgOctober, 2013.

SZIMANSKI, F.; ALBUQUERQUE, J.; FURTADO, F. *Implementando maturidade e agilidade em uma fábrica de software através de scrum e mps. br nível g*. XI Encontro de Estudantes de Informática do Tocantins. Centro Universitário Luterano de Palmas, p. 161–170, 2009.

WAZLAWICK, R. S. *Antálise e Projeto de Sistemas de Informação Orientados a Objetos*. Rio de Janeiro: Campus, 2004.

WAZLAWICK, R. S. *Engenharia de software: conceitos e práticas*. Elsevier Brasil, 2013.

ZULIANI, EMERSON. 2015. *O que é Scrum* disponível em: <http://emersonzuliani.com.br/o-que-e-scrum/>. Acesso em: 15 out. 2017.