CONGRESSO NACIONAL DE EXCELÊNCIA EM GESTÃO
ISSN 1984-9354
INOVARSE
XII CONGRESSO NACIONAL DE EXCELÊNCIA EM GESTÃO
& III INOVARSE – RESPONSABILIDADE SOCIAL APLICADA.
29 e 30 de setembro de 2016.

PROPOSTA DE UM NOVO MODELO DE APLICAÇÃO DA METODOLOGIA ÁGIL SCRUM PARA DOCUMENTAÇÃO

Área temática: Gestão do Conhecimento Organizacional

Nilton Freitas Junior
niltonfjunior@gmail.com

Davi Rocha
davicandidorocha92@gmail.com

Ariadne Gomes
ariadnegms@gmail.com

Resumo: *Em um processo de desenvolvimento de software, a documentação é essencial para garantir a qualidade e manutenção de um sistema. Um dos problemas das metodologias ágeis de desenvolvimento é a precária atenção oferecida a documentação de software. Ao adotar o framework Scrum, essa falta de documentação pode ser uma barreira à inserção de novos membros na equipe de desenvolvimento e também na manutenção e atualização de sistemas. A proposta apresentada nesse trabalho é a criação de um novo personagem ao framework, o Scrum Writer. O objetivo é o desenvolvimento de documentos que não firam os princípios defendidos pelo manifesto ágil e ao mesmo tempo, garantam a qualidade e melhoria contínua dos projetos de sistemas computacionais.*

Palavras-chaves: *SCRUM, Metodologias Ágeis, Engenharia de Software, Documentação de Software, Tecnologia da Informação.*

1 INTRODUÇÃO

Desenvolvimento de sistemas é um conjunto organizado de atividades, que utilizam metodologias para planejar, construir, implantar, avaliar e prestar manutenção. O objetivo dessas metodologias é estabelecer diretrizes para o levantamento de requisitos, concepção e construção de *softwares*, com a especificação de atividades e técnicas que serão utilizadas durante o ciclo de vida do projeto.

No desenvolvimento, a documentação pode ser considerada a parte visível do *software*, isto porque ela descreve as funcionalidades do sistema, desde o código fonte até o manual de instruções para o público alvo. Desta forma, torna-se um importante artefato de *software*, que permite a redução de erros e facilita a manutenção, mesmo quando há trocas na equipe de desenvolvimento.

Metodologias tradicionais se caracterizam pelo excesso de documentação, onde geralmente, o sistema é planejado completamente e só então, passa a ser construído. É notório que esse processo causa a desmotivação da equipe, afetando diretamente na produtividade e refletindo em uma entrega de produtos que nem sempre atendem aos requisitos.

Metodologias ágeis surgem como alternativa, dando ênfase ao desenvolvimento de *softwares* operáveis, onde se prioriza a entrega adiantada, com o objetivo de satisfação do cliente. Mudanças de requisitos passam a ser aceitas, mesmo no fim da implementação; equipe de desenvolvimento e clientes passam a trabalhar juntos; ênfase se dá a equipes motivadas, simplicidade na construção do projeto, contínua atenção a excelência técnica e bom *design*. Todos esses princípios que permeiam o desenvolvimento ágil acabam deixando a documentação de *software* de lado.

Este trabalho apresenta os princípios norteadores da Metodologia Ágil *Scrum* e faz a proposta do desenvolvimento de um novo personagem, ao qual será atribuída da denominação de *SCRUM Writer*, com o objetivo de aprimorar o processo de documentação de *softwares* em *Scrum*. Com a implementação deste, aloca-se uma ou mais pessoas, responsáveis por documentar todo o trabalho desenvolvido pela Equipe (*Scrum Master* e *Scrum Team*) e o Cliente (*Product Owner*).

1.1 Metodologia

Este trabalho é amparado por uma pesquisa bibliográfica, narrativa e descritiva, em livros e artigos científicos sobre a Metodologia Ágil *Scrum*. Através de imagens obtidas em obras referenciadas, será apresentado o ciclo de vida do *Scrum* e, por proposta do trabalho, adaptações a estas imagens serão feitas para compor a presença do novo papel, *Scrum Writer*.

Discute-se sobre o perfil profissional esperado para a papel do *Scrum Writer*, além de ressaltar a importância da documentação na qualidade do desenvolvimento de *software*, apontando os pontos de inserção ideais dentro do ciclo de vida *Scrum*, para o trabalho do *Scrum Writer*. São apresentadas as propostas de novos artefatos para o trabalho desenvolvido pelo *Scrum Writer*, na documentação de um *software.*

Os artefatos sugeridos para o trabalho do *Scrum Writer* serão criados por autoria própria, e classificados como figuras. Sua composição será superficial por não haver a aplicação prática do trabalho, apenas uma concepção de uso destes modelos.

2. ANÁLISE DE SISTEMAS E A IMPORTÂNCIA DA DOCUMENTAÇÃO

O avanço tecnológico provocou a evolução da antiga sociedade industrial para uma sociedade baseada na informação e no conhecimento. Consequentemente, a informação passa a ser um bem primordial, e por isso precisa de uma forma adequada e eficiente para ser gerenciada. Laudon relata que:

> As empresas estão sempre tentando melhorar a eficiência de suas operações a fim de conseguir maior lucratividade. Das ferramentas de que os administradores dispõem, as tecnologias e os sistemas de informação estão entre as mais importantes para atingir altos níveis de eficiência e produtividade nas operações, especialmente quando combinadas com mudanças no comportamento da administração e nas práticas de negócio. (LAUDON, 2007, pág. 6).

Sistemas de Informações podem ser definidos como uma combinação de pessoas, dados, processos e tecnologias que se relacionam com o propósito de melhorar o processo de negócio empresarial. Entretanto, projetar sistemas computadorizados é uma atividade complexa, isto porque pode originar uma série de implicações, que vão desde a mudança da rotina empresarial até a reestruturação das práticas de negócio (LAUDON, 2007).

O analista de sistemas possui a responsabilidade de traduzir as necessidades dos usuários, em características de um *software*. A partir de uma análise dos requisitos levantados, deve construir modelos de desenvolvimento para uma melhor representação do sistema que será construído. Para a produção desses, a documentação torna-se uma peça primordial, como pode ser observado nos dizeres de Falbo:

> Uma documentação de qualidade propicia uma maior organização durante o desenvolvimento de um sistema, facilitando modificações e futuras manutenções no mesmo. Além disso, reduz o impacto da perda de membros da equipe, reduz o tempo de desenvolvimento de seus posteriores, reduz o tempo de manutenção e contribui para a redução de erros, aumentando assim, a qualidade do processo e do produto gerado. Dessa forma, a criação da documentação é tão importante quanto a criação do *software* em si (FALBO, 2005, pág. 21).

A documentação é importante para todas as pessoas envolvidas no projeto. Portanto, o planejamento de uma boa documentação é um fator primordial no desenvolvimento bem sucedido do *software* (SCHACH, 2010).

Embora a documentação seja parte importante do processo de desenvolvimento de *software*, é preciso que o trabalho e tempo dispensados a essa tarefa não se transforme em algo oneroso, que comprometa o próprio desenvolvimento. Mudanças nos requisitos são fatores que podem fazer com que uma documentação necessite de constantes revisões e, caso o desenvolvimento só ocorra a partir de sua completude, condições como prazo de entrega e custo do projeto podem ser comprometidas.

2.1 Metodologias Ágeis e a documentação do *software*

Com a Crise do *Software* em 1970, várias metodologias de desenvolvimento surgiram, com o objetivo de entregar *software* de qualidade, dentro do prazo e custo estimados. Através de documentações em que os clientes poderiam ter uma visão holística do projeto, como a UML, o desenvolvimento passa por problemas como, por exemplo, mudanças de requisitos, atrasos na entrega e falta de atualização e revisão de documentos gerados.

O processo de desenvolvimento de *software* ágil é adaptável a mudanças, priorizando feedbacks constantes dos clientes, respeito e confiança entre os membros da equipe. Acerca da importância da metodologia ágil e da documentação de *software*, LOBO (2008) relata que:

> Desenvolver um *software* sem nenhuma sistematização é um verdadeiro caos, mas utilizar processos pesados também se torna uma carga insustentável para as

> empresas, em custo e tempo. O meio termo está na metodologia ágil de desenvolvimento de *software*, pois ela define regras e rotinas, que não "pesam" no desenvolvimento do *software*, mas torna o ambiente mais controlado e rápido. [...] A documentação da metodologia ágil é leve, permitindo assim dinamismo e flexibilidade às mudanças no projeto. Para que a documentação de um *software* seja útil, ela deve ser fácil de ser criada e entendida pela equipe de *software*. (LOBO, 2008, pág. 42).

Como observado, a documentação não deve ser "abandonada" em metodologias ágeis. A diferença entre documentos ágeis e tradicionais, é que o primeiro deve ser dinâmico, não deve substituir a comunicação entre os membros da equipe, deve ser leve, fácil de ser criada e entendida e o mais importante, só se deve documentar o necessário para o projeto de desenvolvimento (LOBO, 2008).

2.2 Conceitos da Metodologia *SCRUM*

Existem vários tipos de *frameworks* que são utilizados em Metodologias Ágeis, como por exemplo, o *SCRUM*, o qual pode ser utilizado para planejar, gerenciar e desenvolver qualquer tipo de projeto. A respeito de sua importância no mercado, MARTINS (2007) relata:

> *Scrum* é um processo bastante leve para gerenciar e controlar projetos de desenvolvimento de *software* e para criação de produtos. O *Scrum* é uma metodologia ágil que segue as filosofias iterativa e incremental. Ele se concentra no que é realmente importante: gerenciar o projeto e criar um produto que acrescente valor para o negócio. O valor decorre da funcionalidade propriamente dita, do prazo em que ela é necessária, do custo e da qualidade (MARTINS, 2007, pág. 270).

No *Scrum*, os projetos são divididos em ciclos, iterativos e incrementais os quais são denominados de *Sprints*. Cada *Sprint* contém uma lista de requisitos (*Product Backlog*), os quais serão implementados pela Equipe de Desenvolvimento (*Scrum Team*). Em *Scrum*, existem três personagens que são responsáveis pelo projeto, como relatado por Gomes, Willi e Rehem (2014):

Product Owner (Dono do Produto): Responsável por definir, priorizar e gerenciar os requisitos (*Product Backlog*) do projeto. É ele que define a visão do produto, que tem autoridade para cancelar *Sprints*, que define a importância dos itens, que gerencia o orçamento e aceita ou rejeita os itens entregues ao final de cada *Sprint*.

Scrum Team (Time): O time de desenvolvimento é responsável pela transformação do *Product Backlog* em funcionalidades do produto, que serão entregues ao final de cada iteração. É o time que define a estimativa em relação ao tamanho dos itens a serem

desenvolvidos. Recomenda-se que os membros da equipe sejam interdisciplinares, ou seja, pessoas com conhecimento especializado em várias áreas, como por exemplo, banco de dados, controle de qualidade e programação.

Scrum Master: Considerada a pessoa que mais entende do *Scrum*, ele é responsável por orientar o *Scrum Team*, através de treinamentos que o levem a uma maior produtividade no desenvolvimento de produtos com maior qualidade, isto é, ajuda o time a seguir valores e princípios ágeis de *Scrum*. Orienta o *Product Owner* na definição e priorização dos requisitos, além de assumir papel importante na visualização e solução de problemas relativos ao desenvolvimento.

É importante dizer que em *Scrum* todos os atores possuem suas responsabilidades e importância dentro do projeto. No time, todos os membros possuem uma visão holística da equipe, mas cada um deve ser responsável por uma etapa ou parte do processo, fato esse, que acaba evitando muita burocracia no desenvolvimento (CRUZ, 2015).

Com o intuito de incentivar uma documentação de qualidade, este trabalho busca propor um novo ator dentro do *framework Scrum*, o qual receberá a denominação *Scrum Writer*. Esse ator será responsável por estabelecer e elaborar níveis onde a documentação se torna desejável ao projeto, com o intuito de gerar documentos que não ferem a agilidade defendida pelo Manifesto Ágil e que auxilie tanto a equipe de desenvolvimento e os clientes.

3. PERFIL PROFISSIONAL DO *SCRUM WRITER* E SUA ATUAÇÃO

O objetivo de introduzir um novo profissional na metodologia *Scrum* é criar e manter atualizada as informações geradas durante o processo de desenvolvimento, permitindo assim o armazenamento de um repositório de documentos, que poderão facilitar o entendimento dos produtos gerados e até mesmo servir como alicerce de projetos futuros. Com o intuito de tirar a responsabilidade em se criar e manter uma documentação atualizada dos desenvolvedores, o *Scrum Writer* (documentador), ficaria responsável pela sua realização.

Entretanto, esse novo profissional precisa ser eficiente e com habilidades de manter informações valiosas ao projeto, de maneira organizada, ordenada e acessível. Ter criatividade para elaborar e apresentar documentos, que gerem valor. Ter domínio de gramática para escrever documentos sucintos e claros, a quaisquer pessoas. Ser multidisciplinar, isto é, conhecer um pouco de todas as áreas envolvidas no projeto, desde a

utilização de linguagens de programação e banco de dados, até a gerência de projetos. Além disso, o *Scrum Writer* deve possuir conhecimentos nas áreas de especificação de requisitos e elaboração de casos de uso.

Essas características ajudarão o *Scrum Writer* na elaboração de documentos, sem ferir os princípios da agilidade, como menciona CRUZ (2015).

> O fundamental, no Manifesto Ágil, é que a documentação de um *software* é importante sim e deve ser realizada, porém sempre considerando o que é importante para o produto e o que é minimamente necessário e imprescindível. Por isso o próprio valor utiliza a palavra "abrangente" ao descrever a documentação (CRUZ, 2015, pág. 16).

O objetivo do *Scrum Writer* não é criar documentações abrangentes, mas sim uma documentação de qualidade e que ajude tanto a equipe de desenvolvimento, quanto os clientes.

3.1 Momentos de atuação do *Scrum Writer* no ciclo de vida *Scrum*

Os projetos que utilizam o *framework Scrum*, possuem seu sequenciamento, ritmo e estrutura bem definidos através do Ciclo de Vida, que são marcados por *Sprints* (CRUZ, 2015). Para uma melhor visão de todo o processo de desenvolvimento, faz-se necessário o entendimento do Ciclo de Vida *Scrum*, como pode ser visto na Figura 01.

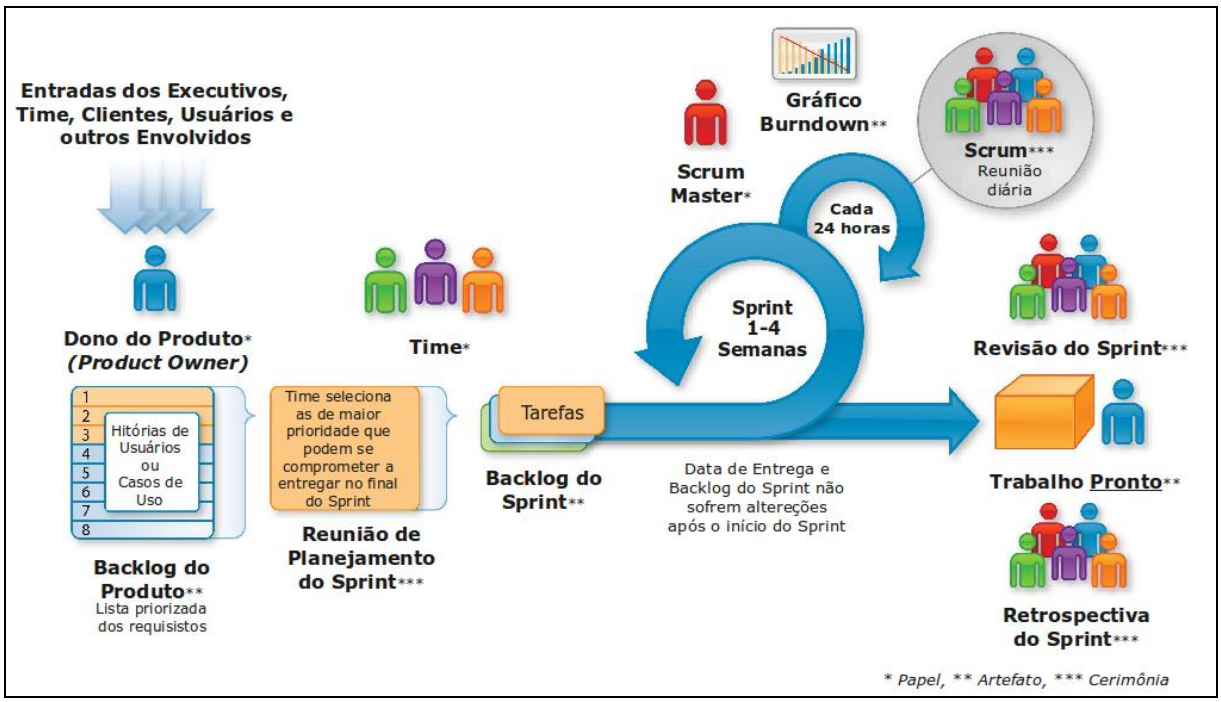

Figura 01 – Ciclo de Vida *SCRUM*. Fonte: BRAS, 2011.

O Ciclo de Vida *Scrum* é marcado por características e papéis bem definidos. As funcionalidades fazem parte do *Product Backlog*, o time sequencia suas atividades através de *Sprints*, o *Scrum Master* gerencia todo o processo de desenvolvimento e o *Product Owner* é considerado o dono do produto. Entretanto, esse ciclo não possui nada definido a respeito de documentação, por isso, o *Scrum Writer* poderá se tornar um importante ator dentro desse processo (CRUZ, 2015).

Recomenda-se que o *Scrum Writer* inicie suas atividades a partir da reunião de planejamento do *Sprint*, fazendo anotações que o auxiliem na elaboração da documentação final do produto de *software*. É importante destacar que a documentação não deverá ser utilizada como um insumo do projeto e sim, como resultado de tudo o que foi feito (BORGES, 2013).

A documentação deverá ser elaborada a partir do final de cada *Sprint*, isso porque evita o retrabalho quando há mudanças provocadas durante o seu desenvolvimento, uma vez que, a proposta de documentação não pode ferir os princípios e valores do Manifesto Ágil.

Com a documentação sendo feita ao final de cada *Sprint*, ela se tornará a alma da Retrospectiva do *Sprint*, onde o *Scrum Writer*, poderá apresentar a todos os envolvidos no projeto, tudo aquilo que foi proposto na Reunião de Planejamento e tudo aquilo que foi

desenvolvido pela equipe, evitando assim, desentendimentos entre o Time e o *Product Owner*. Com estas mudanças, consequentemente o Ciclo de Vida *Scrum* se alteraria, como pode ser observado na Figura 02:

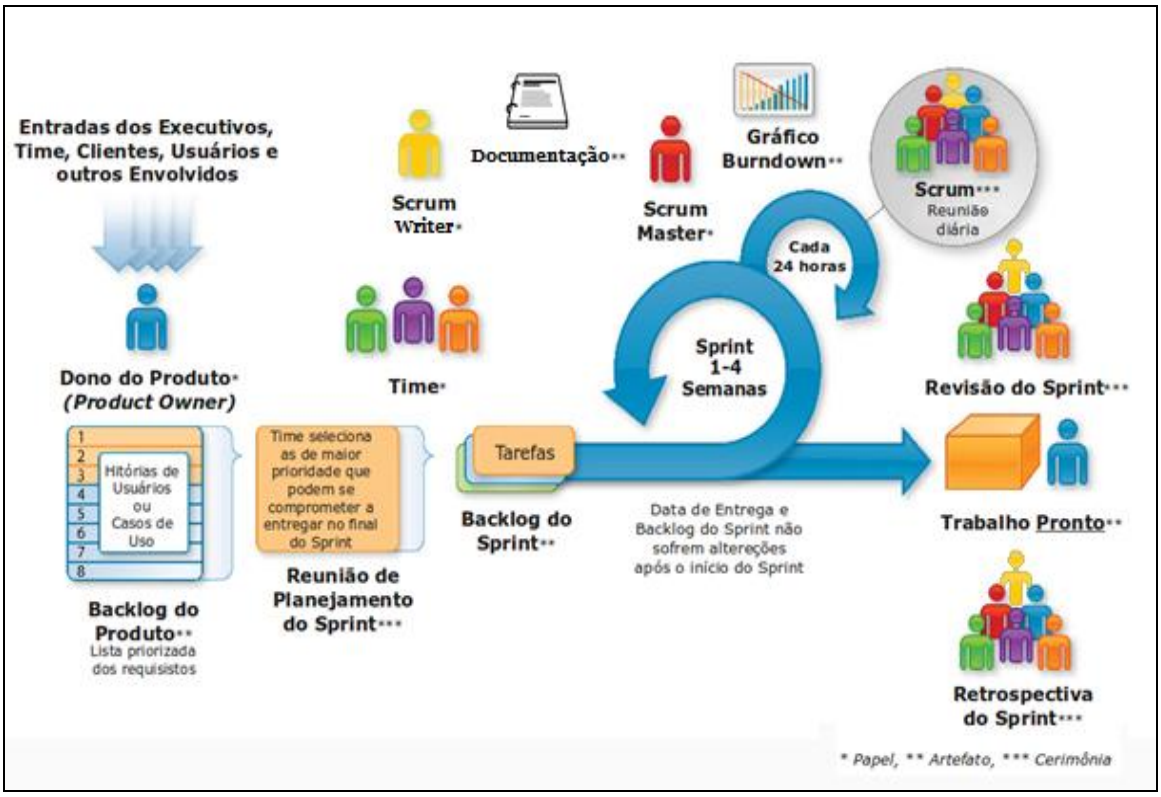

Figura 02 – Ciclo de Vida *SCRUM* proposto. Fonte: BRAS, 2011; adaptado.

De acordo com a Figura 02, o *Scrum Writer* participará de praticamente todas as fases do Ciclo de Vida *Scrum* e deverá elaborar a documentação, após o final de cada *Sprint*. Nas cerimônias de entrega da *Sprint*, o *Scrum Writer* receberá a responsabilidade de preparar e apresentar tudo o que foi proposto e contemplado, para todos os membros da equipe, desde o *Scrum Master* e o *Scrum Team*, até o *Product Owner*.

3.2 Artefatos produzidos pelo *Scrum Writer*

No desenvolvimento de sistemas, as documentações feitas com qualidade e que seguem um padrão, se tornam artefatos de *software* fundamentais, uma vez que agilizam o desenvolvimento e permitem que alterações possam ser realizadas com facilidade.

Durante o ciclo de vida de desenvolvimento de um *software*, vários documentos podem ser realizados, desde a especificação das histórias de usuários (*Product Backlog*) até uma documentação mais abrangente. Entretanto, para que esse tipo de documentação, não interfira nos princípios defendidos pelo Manifesto Ágil, é preciso que se crie uma padronização de como esses documentos devem ser feitos.

O *Scrum Writer* inicia seu trabalho dentro do *Scrum*, a partir da Reunião de Planejamento dos *Sprints*. Essas reuniões funcionam como um tipo de acordo firmado entre o *Product Owner* e o *Scrum Team*, de modo que, se define quais funcionalidades deverão ser desenvolvidas durante a *Sprint* que se inicia.

Durante essa reunião, o *Scrum Writer*, deverá elaborar uma espécie de figura, que indica qual é o número de ordem, a missão, as funcionalidades, a justificativa, a data de início e término daquela *Sprint*. Esse documento, é chamado de *Product Planning* (Produto de Planejamento da *Sprint)*, como já consta no *Scrum*. A Figura 03, define o esboço desse documento:

Nº de ordem	**Missão**	**Funcionalidades**	**Justificativa**	**Data de Início**	**Data de Término**

Figura 03 – Esboço do Product Planning. Fonte: FILHO, 2000; adaptado.

A Reunião de Planejamento é considerada uma importante fase do ciclo de vida, onde torna-se essencial que todos os membros da equipe compreendam melhor o projeto que será desenvolvido, e por isso, justifica-se o documento da Figura 06. Em *Scrum*, os projetos são divididos em ciclos, denominados *Sprints*, onde serão desenvolvidas as tarefas de acordo com sua prioridade e importância, que são definidas pelo *Product Owner*.

Os *Sprints*, possuem uma duração entre 1 a 4 semanas, então é importante que a equipe não ultrapasse esse limite, uma vez que, depois de iniciada, a data de entrega e a *Backlog* dos *Sprints*, não podem sofrer alterações. O *Scrum Writer* deve ficar atento à essas informações e registra-las corretamente no documento *Product Planning*.

Uma característica em *Scrum*, são as Reuniões Diárias (*Daily Meeting)*, as quais possuem uma duração de 15 minutos e os membros da equipe devem responder a três perguntas, sendo elas: 1) O que eu fiz ontem? 2) O que será feito nesse dia que se inicia? 3) Existe algum impedimento?. Essas perguntas auxiliam a equipe a entender o progresso, além

de levantar questões que possam impedir e/ou retardar o desenvolvimento do projeto (PHAN, 2011).

Nas reuniões diárias, o *Scrum Writer* deve elaborar um documento, chamado de Controle de Impedimentos, como já consta em *Scrum*. Por intermédio dele, devem ser registrados o número de ordem da *Sprint*, a funcionalidade com problema, uma breve justificativa a respeito do impedimento e um outro campo que relate a solução encontrada para o problema em questão. O esboço desse documento encontra-se na Figura 04:

Número de Ordem :	Funcionalidade:
Impedimento:	
Solução:	

Figura 04 – Esboço do Controle de Impedimentos. Fonte: PEREIRA, 2013; adaptado.

Como os projetos são divididos em ciclos (*Sprints*), o final de cada *Sprint* determina que um pedaço do projeto foi construído. Portanto, todas as funcionalidades implementadas deverão ser revisadas, através da Reunião de Revisão do *Sprint* (*Sprint Review Meeting*).

O objetivo das Reuniões de Revisão do *Sprint*, é examinar a *Sprint* desenvolvida e também avaliar e adaptar a *Backlog* do Produto, se for necesssário. É uma reunião onde todos os membros devem participar, desde o *Scrum Master*, até o *Product Owner* e os *Stakeholders* (partes interessadas no projeto). Como é considerada uma reunião informal, onde pode haver mudanças no *Backlog* do Produto, torna-se fundamental uma documentação que registre todos os acontecimentos importantes.

Dessa forma, o *Scrum Writer* deverá elaborar uma ata, esse documento será chamado de Ata de Revisão da *Sprint* (em inglês, *Sprint Review Act*). A Figura 05 define o esboço desse documento:

Nome da Empresa:	
Nome do Projeto	
Documento Elaborado por : (Nome do *Scrum Writer*)	**Data da Reunião**: __/__/__
Facilitador: (informar o nome do membro que conduziu a reunião, ou seja, o *Scrum Master*)	**Participantes** ***Product Owner*:** ***Scrum Team*:** ***Stakeholders*:**

Assunto / Tópico	**Pontos discutidos**	**Problemas Encontrados**	**Decisão / Solução**
Descrição de uma funcionalidade implementada	Descrição de pontos discutidos relativos ao assunto/tópico	Descrição de possíveis problemas encontrados em relação ao assunto/tópico	Descrição da solução ou decisão encontrada para o problema.
Aprovação da *Sprint*:_____ (Assinatura do *Product Owner*)			

Figura 05 – Esboço do *Sprint Review Act*. Fonte: autoria própria.

Ao término da aprovação do *Product Owner* o trabalho referente aquela *Sprint* encontra-se na fase de pronto. Para dar sequência ao ciclo *Scrum* é necessária uma Reunião de Retrospectiva da *Sprint* que acaba de ser entregue ao cliente. A Reunião de Retrospectiva é considerada essencial ao projeto, porque através dela o *Scrum Team*, pode examinar todo o seu trabalho e promover melhorias ao desenvolvimento da próxima *Sprint*.

Na retrospectiva do *Sprint*, o *Scrum Writer* assume um papel fundamental e essencial ao processo de desenvolvimento *Scrum*. Como a documentação em projetos ágeis fora deixado um pouco de lado, os times de desenvolvimento muitas vezes, não registram as informações pertinentes aos projetos. Um exemplo, é o os problemas encontrados durante a implementação de um *Sprint*.

É inquestionável dizer que com os documentos criados pelo *Scrum Writer* (*Product Planning*, Controle de Impedimentos e a *Sprint Review Act*), o *Scrum Team*, poderá visualizar todos os problemas levantados durante o desenvolvimento. Como o *Scrum Writer* já identificou os principais problemas que foram abordados, principalmente durante as reuniões diárias, o Time *Scrum*, deverá analisar esses documentos e identificar melhorias e adaptações para os próximos *Sprints*.

O *Scrum Writer* deverá utilizar como base o Ciclo do PDCA[1], para junto com o *Scrum Team*, *Scrum Master* e *Product Owner*, definir melhorias contínuas ao processo de desenvolvimento do ciclo de vida *Scrum*. Abaixo, encontra-se, a Figura 09, que contém o esboço do modelo que o *Scrum Writer* deverá seguir para elaborar o PDCA voltado ao *Scrum*:

[1] Ciclo PDCA: Muito utilizado por empresas de todo o mundo, para a melhoria contínua de processos de gestão. Seu objetivo principal é transformar os processos mais ágeis, claros e objetivos, através das seguintes fases: Planejamento (P – Plan), Execução (D – Do), Verificação (C – Check) e Ação (A – Action) (PERIARD, 2011).

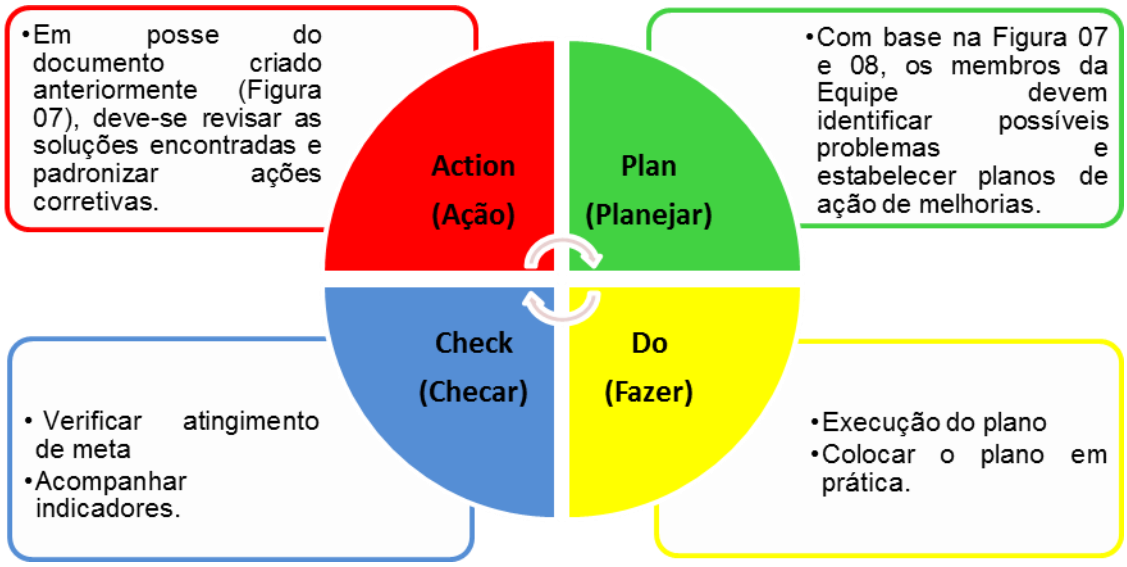

Figura 06 – Esboço do Ciclo PDCA aplicado à SCRUM. Fonte: PERIARD (2011); adaptado.

O objetivo em se criar essas documentações de *software* no ciclo de vida *Scrum*, é principalmente a reutilização de códigos e projetos existentes, diminuindo o tempo e os esforços exigidos na criação de um novo projeto.

A integração de todas essas documentações aos demais artefatos de *software*, possibilita um desenvolvimento mais planejado, eficiente e consequentemente de maior qualidade. Isto porque, durante o ciclo de vida, qualquer membro que faça parte do projeto, pode obter informações sobre o andamento, os problemas relatados, além de acompanhar tudo o que já foi desenvolvido.

A documentação de um *software* é tão importante quanto o próprio sistema. Por isso, ela deve ocorrer em todo o desenvolvimento, de forma paralela ao ciclo de vida *Scrum*. Dessa forma, faz se necessária a criação de repositórios digitais, onde todos esses documentos devem ser guardados.

Os repositórios digitais serão importantes "peças" para garantir a confidencialidade, disponibilidade e autenticidade dos documentos de *software*, a curto, médio e longo prazo.

A confidencialidade consiste em não permitir que as informações contidas nesses documentos, sejam divulgadas ou disponibilizadas, sem consentimento e autorização. A disponibilidade garante que somente pessoas autorizadas terão esses documentos acessíveis. E por último, a autenticidade, que provê a credibilidade desse documento, ou seja, impede a alteração e/ou modificação de informações.

O *Scrum Writer* é o membro mais indicado para criar e manter esses documentos atualizados nos repositórios digitais, e por isso, evidencia sua importância no ciclo de vida *Scrum*.

4 CONSIDERAÇÕES FINAIS

Desde que surgiu o Manifesto Ágil não é incomum o encontro de relatos ou entendimentos que posicionam a documentação tradicional de *software* fora do cenário produtivo, como se esta fosse apenas um empecilho para o bom resultado da produção. Por pressupostos, uma comunicação entre a equipe de desenvolvimento e o cliente tornar-se-ia suficiente forma de negociação. Entretanto, a comunicação, por si só, pode não ser o bastante para registrar todas as etapas do ciclo de vida, além de dificultar a manutenção do projeto.

Em se tratando do *framework* ágil, *Scrum*, o projeto de desenvolvimento conta com três atores, sendo eles: *Product Owner*, *Scrum Master* e *Scrum Team*. Dentro do ciclo de vida, todas as negociações e estimativas de desenvolvimento, são realizadas apenas através de comunicações e reuniões entre esses atores. A falta de uma documentação que garanta que o planejamento e a qualidade sejam garantidos dificulta em sobremaneira esse projeto.

Em Scrum, os projetos são divididos em ciclos, chamados *Sprints*. Esses *Sprints* possui uma duração de 1 a 4 semanas, dependendo da complexidade do projeto. Durante os *Sprints*, são realizadas reuniões diárias, que tem como característica demonstrar tudo o que já foi implementado, o que será feito no dia seguinte e também relatar os problemas que possam impedir o desenvolvimento. Contudo, não há a previsão de nenhum documento formal, que registre todas essas informações.

Como o objetivo de criar uma documentação que gere valor ao projeto e que ajude tanto os desenvolvedores, quanto os clientes, foi proposto a inserção de um novo personagem, denominado *Scrum Writer*, responsável por criar e manter documentos de *software* atualizados e disponíveis em um repositório digital, a todos os membros do projeto, desde o *Product Owner* até o *Scrum Master*. No total, foram propostos quatro documentos, que ajudarão no acompanhamento do projeto e também servirão de base para novos projetos.

Os documentos foram chamados de: *Product Planning*, *Sprint Review Act*, Controle de Impedimentos e Ciclo do PDCA aplicado ao *Scrum*. O *Product Planning* será responsável pelo registro de todas as informações negociadas durante o planejamento do *Sprint*. O *Sprint*

Review Act será desenvolvido durante as Reuniões de Revisão e tem como objetivo, registrar possíveis mudanças no *Backlog* do Produto e também os pontos discutidos entre os membros. O controle de impedimentos será realizado de acordo com os problemas levantados durante as Reuniões Diárias e, além disso, registrará todas as soluções que foram encontradas para os respectivos problemas. Por último, o Ciclo do PDCA auxiliará na melhoria contínua dos projetos de desenvolvimento.

Apesar dos documentos serem criados em diferentes etapas do ciclo de vida *Scrum*, deve-se considerar que eles serão importantes e seguirão como guia, que auxiliará na compreensão e execução do projeto. Além disso, se houver troca da equipe de desenvolvimento, esses documentos ajudarão a novos membros na compreensão do projeto que está sendo desenvolvido.

É crucial que esses documentos sejam armazenados em repositórios digitais confiáveis, para não permitir que as informações contidas nele possam ser alteradas e acessadas por pessoas não autorizadas. A inserção desse novo personagem ao framework *Scrum*, não tem por objetivo ferir aos princípios do manifesto ágil, pelo contrário. O *Scrum Writer* trará benefícios inquestionáveis ao processo de desenvolvimento, porque uma documentação de qualidade pode auxiliar no reuso de códigos em projetos futuros e na manutenção e atualização de sistemas já existentes.

Vale ressaltar que, os modelos de documentos propostos servirão como base para os projetos de desenvolvimento *Scrum*. As empresas que adotam esse modelo devem se conscientizar que a documentação é importante para garantir a qualidade do desenvolvimento, todavia, não há um modelo que seja considerado o correto, e sim, aquele que atende as necessidades e aos recursos que cada organização possui. A importância em se acrescentar o *Scrum Writer*, é não sobrecarregar a equipe de desenvolvimento, ao contrário, seu objetivo é ajudar e auxiliar a equipe a atingir os níveis de qualidade e a melhoria contínua dos projetos.

Como trabalhos futuros, sugere-se que a presença do *Scrum Writer* seja adotada por uma empresa que utilize o framework *Scrum* para a gestão e melhoria de seus processos. Apesar da crença de que a documentação de *software* pode causar lentidão ao processo, a inserção deste novo personagem, poderá aumentar os níveis de qualidade das empresas que aderirem.

Deverão ser observadas todas as etapas do ciclo de vida, adequando-o às novas atividades que serão desenvolvidas pelo *Scrum Writer*. Conforme o andamento dos projetos,

as próprias empresas, poderão fazer adaptações aos momentos de atuação desse novo personagem, de forma que, tanto a empresa quanto o *Scrum Writer* avaliem a melhor forma em se criar uma documentação de qualidade e que não ocasione em muita burocracia aos projetos de desenvolvimento.

REFERÊNCIAS

BORGES, Eduardo. METODOLOGIAS ágeis e documentação de software. Direção de Eduardo Borges. 2013. (10 min.), son., P&B. Disponível em: <http://www.youtube.com/watch?v=3Smbhnmue7Y>. Acesso em: 20 de julho de 2013.

BRAS, Alan. Introdução ao Scrum, Alan Bras – Mestre IC Unicamp – Pesquisador em Engenharia Software Ágil (IBM), 2011. Disponível em: < http://www.alanbraz.com.br/ic/scrum.pdf>. Acesso em: 25 de julho de 2015.

CRUZ, Fábio. Scrum e Agile em Projetos: Guia Completo. Rio de Janeiro: Brasport, 2015. Disponível em: <https://books.google.com.br/books?isbn=8574527130>. Acesso em: 25 de julho de 2015.

FALBO, Ricardo de Almeida. Engenharia de Software - UFES - Universidade Federal do Espírito (Notas de Aula) - 2005. Disponível em: <http://www.inf.ufes.br/~falbo/download/aulas/es-g/2005-1/NotasDeAula.pdf>. Acesso: 12 de julho de 2015.

FILHO, Wilson de Pádua Paula. Engenharia de Software: fundamentos, métodos e padrões, 2000. Disponível em: <http://aulasprof.6te.net/Arquivos_Aulas/07-Proces_Desen_Soft/Livro_Eng_Soft_Fund_Met_Padroes.pdf>. Acesso em: 20 de setembro de 2015.

GOMES, Alexandre, WILLI, Renato & REHEM. Manifesto Ágil: História Scrum (2014). Disponível em: < file:///C:/Users/AriiCosta/Downloads/LivroMetodosAgeis-Capitulos1a3-ManifestoAgilHistoriaScrum.pdf>. Acesso em: 21 de julho de 2015.

LAUDON, Kenneth C. Sistemas de informações gerenciais. 7. Ed. São Paulo: Pearson Prentice Hall, 2007.

LOBO, Edson J. R. Curso de Engenharia de Software. São Paulo: Digerati Books, 2008. Disponível em: <https://books.google.com.br/books?isbn=8578730100>. Acesso em 19 de julho de 2015.

MARTINS, José Carlos Cordeiro. Técnicas para gerenciamento de projetos de software. Rio de Janeiro: Brasport, 2007. Disponível em: <https://books.google.com.br/books?isbn=8574523089>. Acesso em: 12 de julho de 2015.

PEREIRA, Bruno. Desenvolvimento ágil com Jira, Greenhopper e Tempo Plugin, 2013. Disponível em: <http://blog.rivendel.com.br/2013/05/12/desenvolvimento-agil-com-jira-greenhopper-e-tempo-plugin/>. Acesso em: 27 de setembro de 2015.

PERIARD, Gustavo. O Ciclo PDCA e a melhoria contínua, 2011. Disponível em: <http://www.sobreadministracao.com/o-ciclo-pdca-deming-e-a-melhoria-continua/>. Acesso em: 27 de setembro de 2015.

PHAN, Andrew. Scrum em Ação: gerenciamento e desenvolvimento ágil de projetos de software. Disponível em: <https://books.google.com.br/books?isbn=8575222856>. Acesso em: 27 de setembro de 2015.

SCHACH, Stephen R. Engenharia de software: os paradigmas clássicos orientados a objetos. Porto Alegre: AMGH, 2010. Disponível em: <https://books.google.com.br/books?isbn=8563308440>. Acesso em: 12 de julho de 2015.