Ministério da Educação – Brasil
Universidade Federal dos Vales do Jequitinhonha e Mucuri – UFVJM
Minas Gerais – Brasil
Revista Vozes dos Vales: Publicações Acadêmicas
Reg.: 120.2.095 – 2011 – UFVJM
ISSN: 2238-6424
QUALIS/CAPES – LATINDEX
Nº. 16 – Ano VIII – 10/2019
http://www.ufvjm.edu.br/vozes

USO E ANÁLISE DO *SCRUM* NA CAPACITAÇÃO DE ALUNOS EM DESENVOLVIMENTO DE *SOFTWARE*: UM ESTUDO DE CASO REALIZADO NA UFVJM

Prof. Yvssa Carneiro Desmots Eliote
Mestre em Tecnologia Ambiente e Sociedade pela Universidade Federal dos Vales do Jequitinhonha e Mucuri (UFVJM-Campus Mucuri) - Teófilo Otoni, MG, Brasil.
http://lattes.cnpq.br/9223085400618408
E-mail: yvssa@hotmail.com

Prof. Dr. Wederson Marcos Alves
Doutor em Engenharia Agrícola pela Universidade Federal de Viçosa (UFV)
Docente da Universidade Federal dos Vales do Jequitinhonha e Mucuri - UFVJM
http://lattes.cnpq.br/8599448364867450
E-mail: wederson.alves@ufvjm.edu.br

Prof. Luiz Cláudio Mesquita de Aquino
Doutorando em Modelagem Computacional pela
Universidade Federal de Juiz de Fora (UJFJ) - MG, Brasil
Docente da Universidade Federal dos Vales do Jequitinhonha e Mucuri - UFVJM
http://lattes.cnpq.br/9081918642785880
E-mail: luiz.aquino@ufvjm.edu.br

Prof. Dr. Mauro Lúcio Franco
Doutor em Química pela Universidade Federal de Minas Gerais (UFMG)
Docente da Universidade Federal dos Vales do Jequitinhonha e Mucuri - UFVJM
http://lattes.cnpq.br/5529582752535382
E-mail: ml.franco@ufvjm.edu.br

Prof. Luiz Fernando Alves Souza
Mestrando em Tecnologia, Ambiente e Sociedade da UFVJM
E-mail: luizfernandoalvesti@gmail.com

Resumo: O presente trabalho expõe um estudo de caso sobre o uso do *framework Scrum* para gerenciar o treinamento de um grupo de alunos do PET (Programa de Educação Tutorial) no aprendizado do próprio *Scrum,* no contexto do projeto do *website* Nosso Exercício, realizado na Universidade Federal dos Vales do Jequitinhonha e Mucuri (UFVJM). Considerando a pouca experiência destes alunos no desenvolvimento de *software*, o objetivo desta pesquisa consistiu em verificar se o uso do *framework Scrum* auxiliaria a equipe no cumprimento das metas e prazos estabelecidos no projeto, no fortalecimento do trabalho em equipe e no entendimento das tarefas a serem realizadas. Além disso, analisou-se o quão flexível seria este método ágil de forma a se adaptar às características de tempo e espaço disponíveis para a execução do projeto. Os resultados apontaram que, mesmo uma equipe inexperiente, pode obter benefícios consideráveis com o uso deste *framework*, principalmente no que se diz respeito ao fortalecimento do trabalho em equipe, foco nas metas e melhoria contínua no aprendizado do método, das tecnologias envolvidas e do *software* proposto.

Palavras Chave: Engenharia de *Software*. Metodologia Ágil. *Framework Scrum*.

Introdução

O desenvolvimento de *software* é uma atividade complexa e representa um desafio para a comunidade de desenvolvedores que se esforça continuamente para entregar produtos fáceis, rápidos e de qualidade, dentro do tempo e custos previamente definidos (PRESSMAN, 2011).

O estudo mais conhecido da empresa americana *Standish Group*, o relatório *Chaos Report*, divulgado a cada dois anos desde 1994, tem evidenciado ao longo do tempo os altos índices de falhas nos projetos de desenvolvimento de *software* em todo o mundo. A pesquisa feita em 2014 por esta instituição em empresas de pequeno, médio e grande porte concluiu que a taxa de insucesso dos projetos analisados foi de 72%. Estes projetos foram cancelados ou atrasaram, estouraram o orçamento e/ou entregaram menos funcionalidades para o cliente do que o planejado. Apenas 28% obtiveram sucesso, isto é, foram entregues no prazo, dentro do orçamento e atendendo ao escopo completo (STANDISH GROUP, 2014).

Apesar de não refletirem o cenário ideal, esses números representam uma evolução desde os primórdios do desenvolvimento de *software*. Um dos fatores atribuídos a este progresso consiste no uso de métodos, processos e ferramentas provenientes dos princípios da Engenharia de *Software*, “uma disciplina de

engenharia relacionada com todos os aspectos da produção de *software* [...]" (SOMMERVILLE, 2007, p. 5).

Duas importantes abordagens comumente utilizadas em projetos de *software,* provenientes da Engenharia de *Software*, consistem no desenvolvimento tradicional e ágil. A primeira reflete um processo pesado, que se caracteriza pelo foco em planos detalhados, definidos no princípio do projeto, documentação extensa e uso de processos cada vez mais complexos (SABBAGH, 2016). A segunda, considerada leve, prevê uma entrega mais rápida do *software* para o cliente, envolve pequenas equipes auto-organizadas que utilizam métodos informais e focam na entrega operacional de cada parte do *software* a ser desenvolvido (PRESSMAN, 2006).

Neste contexto destaca-se o *Scrum*, método ágil empírico amplamente utilizado em todo o mundo (VERSIONONE, 2016), que, conforme define seus criadores, *Ken Schwaber* e *Jeff Sutherland*, consiste num "*framework*[1] dentro do qual pessoas podem tratar e resolver problemas complexos e adaptativos, enquanto produtiva e criativamente entregam produtos com o mais alto valor possível" (SCHWABER; SUTHERLAND, 2016, p. 3).

Scrum é indicado para o desenvolvimento de produtos imersos em ambientes complexos, envolvidos em constantes mudanças. Usa uma abordagem iterativa e incremental para entregar valor com frequência para o cliente, reduzindo assim, os riscos do projeto (SABBAGH, 2016). Fundamenta-se em três pilares: transparência, inspeção e adaptação. A transparência se baseia na visibilidade, ou seja, todos os envolvidos no projeto devem ter uma visão comum e clara do processo. As inspeções são realizadas a fim de se avaliar se o que está sendo feito pela equipe está direcionado às metas. Já as adaptações são essenciais para que os ajustes necessários sejam feitos (SCHWABER; SUTHERLAND, 2016).

O presente trabalho expõe um estudo de caso sobre o uso do *Scrum* para gerenciar o treinamento de um grupo de alunos em formação acadêmica no aprendizado do próprio *Scrum,* no contexto do projeto do *website* Nosso Exercício realizado na Universidade Federal dos Vales do Jequitinhonha e Mucuri (UFVJM), Campus do Mucuri, situada na cidade de Teófilo Otoni-MG. Considerando a pouca experiência destes alunos no desenvolvimento de *software*, o objetivo desta

[1] [...] Estrutura básica que pretende servir de suporte e guia para a construção [...] de algo com uso prático. (SABBAGH, 2016, p. 33).

pesquisa consistiu em verificar se o uso do *framework Scrum* auxiliaria a equipe no cumprimento das metas e prazos estabelecidos no projeto, no fortalecimento do trabalho em equipe e no entendimento das tarefas a serem realizadas. Além disso, analisou-se o quão flexível seria este método ágil de forma a se adaptar às características de tempo e espaço disponíveis para a execução do projeto.

Framework *Scrum*

Apesar de comumente ser utilizado em projetos de *software*, o termo *Scrum* não surgiu nessa área do conhecimento. Foi citado inicialmente no artigo "O novo jogo para o desenvolvimento de novos produtos" de *Hirotaka Takeuchi* e *Ikujiro Nonaka* em 1986. No artigo, os autores associaram o comportamento de equipes que atuavam nas empresas mais produtivas e inovadoras da época com uma jogada de *Rugby* denominada *Scrum*. Estas equipes apresentavam algumas características em comum: tinham autonomia, eram multifuncionais, os líderes eram participativos e se comprometiam a retirar os obstáculos encontrados durante a realização do trabalho, conforme espera-se de um Time *Scrum* (SUTHERLAND, 2014).

Três papéis são assumidos em um Time *Scrum*. O *Product Owner*, ou dono do projeto, que representa o cliente de uma empresa ou instituição e é o responsável pela entrega do produto. O *Scrum Master* garante o bom andamento do projeto, assegurando o uso correto do *framework*, removendo os obstáculos encontrados pela equipe. E o Time é quem realiza as ações de construção do projeto (BROD, 2015).

Praticar o *Scrum* é fazer com que estes papéis executem alguns eventos e construam alguns artefatos dentro de um espaço de tempo bem definido. *Schwaber* e *Sutherland* (2016) no Guia do *Scrum* definem os quatro artefatos para este *framework*:

- O *Backlog do Produto* é uma lista ordenada de tudo que deve ser necessário no produto. O *Product Owner* é responsável pela inclusão do seu conteúdo, disponibilidade e ordenação. Esta lista nunca está completa, devendo acompanhar a dinâmica de evolução do produto e do ambiente no qual ele será utilizado;

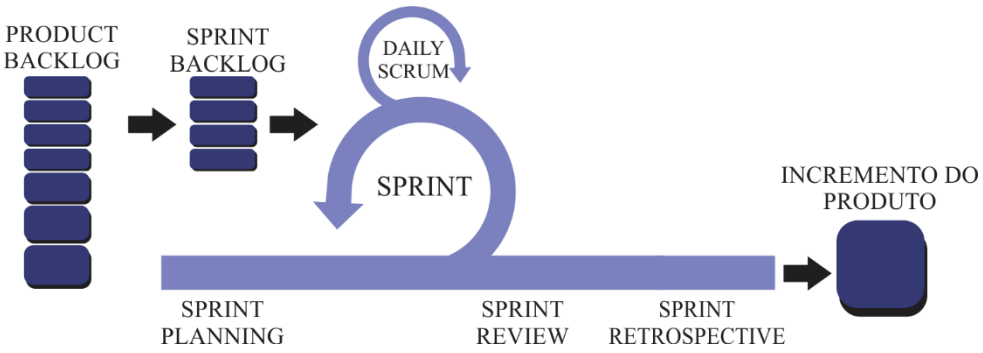
- O *Backlog do Sprint* consiste num conjunto de itens do *Backlog do Produto,* selecionados para serem trabalhados dentro de um *Sprint;*

- A Definição de “Pronto” indica para o Time quando um item do *Backlog do Produto* ou um incremento está “Pronto”. Em um Time *Scrum* todos devem ter o mesmo entendimento de quando o trabalho está completo, ou seja, o significado de “Pronto”.

- O Incremento do Produto é a soma de todos os itens do *Backlog do Produto* já completados durante os *Sprints*. Ao final de cada *Sprint* um novo incremento deve estar “Pronto”, o que significa que deve estar na condição utilizável pelo *Product Owner*.

A Figura 1 apresenta uma visão geral do *Scrum* com seus artefatos e eventos. Eventos no Scrum correspondem ao seu próprio ciclo de desenvolvimento. Os ciclos de desenvolvimento no *Scrum* são divididos em tempos menores denominados *Sprints,* “um *time-box* de um mês ou menos, durante o qual um 'Pronto', versão incremental potencialmente utilizável do produto, é criado” (SCHWABER; SUTHERLAND, 2016, p.8). No decorrer de um projeto com *Scrum* os *Sprints* são executados, um após o outro, sem intervalos, tornando-se cada vez mais completo, até ser finalizado (SABBAGH, 2016).

Figura 1- Visão geral do *Scrum*



Fonte: Sabbagh, 2016, p.42. Adaptado.

Os *Sprints* são compostos por uma reunião de planejamento do *Sprint* (*Sprint Planning*), reuniões diárias (*Daily Scrum*), o trabalho de desenvolvimento, uma revisão do *Sprint* (*Sprint Review*) e a retrospectiva do *Sprint* (*Sprint Retrospective*) (SCHWABER; SUTHERLAND, 2016).

É na reunião de planejamento, primeiro dia do *Sprint,* que a equipe planeja o ciclo de desenvolvimento do *Sprin*t atual, define a sua meta e cria um plano de trabalho a ser realizado. O conjunto de itens do *Sprint* e o seu planejamento é denominado *Sprint Backlog* (SABBAGH, 2016). Ainda segundo o autor, o *Daily Scrum* é uma reunião curta (sugere-se um tempo máximo de 15 minutos), realizada diariamente pelo Time de Desenvolvimento com intuito de responder a três perguntas: O que eu fiz desde a última reunião? O que pretendo fazer até a próxima reunião? Que impedimentos estão em meu caminho? As respostas destas perguntas promovem a comunicação do Time, dão visibilidade ao trabalho, auxiliam na remoção de obstáculos e no cumprimento da meta do *Sprint*.

No encerramento do *Sprint* ocorrem duas reuniões, a *Sprint Review* e a *Retrospective Review*. A *Sprint Review* objetiva obter o *feedback* do cliente sobre o incremento do produto desenvolvido no *Sprint*. Através desta reunião é possível diminuir os riscos do projeto, pois o Time, através do *feedback* do cliente poderá fazer os ajustes necessários, corrigindo os erros, atendendo melhor às necessidades do mesmo (SABBAGH, 2016). Por fim, segundo o autor, o último evento do *Scrum* denominado *Retrospective Sprint* permite ao Time aprender com seus acertos e erros durante o último *Sprint.* Tudo é inspecionado, os processos de trabalho utilizados, o comportamento das pessoas, os relacionamentos, as ferramentas, o ambiente. Assim, busca-se manter o que deu certo e melhorar o que não deu para os próximos *Sprints*. A partir daí um novo ciclo se inicia, considerando novos itens do *Product Backlog* ou ainda contemplando as mudanças solicitadas pelo *Product Owner* no *Sprint* anterior. O ciclo permanece até que o *software* seja concluído (SABBAGH, 2016).

Algumas técnicas complementares ao *Scrum*, oriundas de outros métodos ágeis, foram utilizadas no desenvolvimento deste trabalho como as *User Story,* ou história de usuário, e o quadro *Kanban*.

Uma história de usuário (*User Story*) representa uma funcionalidade do sistema, ou seja, contém a descrição de uma necessidade do usuário do produto sob o seu ponto de vista. No *Scrum,* o uso das histórias de usuário é opcional. Cada item do *Product Backlog* que será desenvolvido pode ser representado no formato de uma história de usuário. Recomenda-se que sejam escritas pelo próprio cliente (*Product Owner*) e refinadas para o trabalho em conjunto com o Time de

Desenvolvimento. Devem ser seguidas por uma série de conversas entre o *Product Owner* e os membros do Time (SABBAGH, 2016).

Segundo Mariotti (2012), o *Kanban* consiste num quadro de tarefas, geralmente dividido em três colunas: a fazer; em execução; feito. Em cada coluna cartões são afixados e vão avançando no quadro à medida que o trabalho da equipe progride (BROD, 2015). Cada cartão contém uma história de usuário (ou tarefa), sua descrição e a identificação do membro do Time de Desenvolvimento que está atuando sobre aquele cartão. Quando as atividades envolvidas com o cartão na coluna 'em execução' são finalizadas, o mesmo é movido para a coluna seguinte, liberando espaço para entrada de um novo cartão. Ainda segundo este autor "O *Kanban* atua fornecendo visibilidade nos processos, deixando explícitos os problemas e prendendo o foco da equipe em qualidade" (MARIOTTI, 2012, p.7).

Metodologia

A estratégia metodológica escolhida para a realização desta pesquisa foi o estudo de caso, que conforme Gil (2010, p.37) "consiste no estudo profundo e exaustivo de um ou mais objetos de maneira que permita seu amplo e detalhado conhecimento". Assim, um estudo aprofundado da plataforma aberta Nosso Exercício foi o ponto de partida para a proposta deste trabalho.

O Nosso Exercício consiste num projeto do Programa de Educação Tutorial (PET), da UFVJM - Campus do Mucuri, Teófilo Otoni - MG e tem como objetivo o compartilhamento de exercícios didáticos de diversas áreas do conhecimento e níveis de ensino. Através deste *website* alunos e professores podem pesquisar por exercícios didáticos, criar suas próprias listas de exercícios e compartilhá-las, bem como alimentar a plataforma com novos conteúdos. Atualmente existem mais de 2.370 exercícios cadastrados na plataforma.

A versão inicial do Nosso Exercício foi lançada no primeiro semestre de 2016. Todas as funcionalidades disponíveis até então foram de responsabilidade de apenas um programador, professor do curso de Matemática, da instituição mencionada. Buscando auxílio para acrescentar novas funcionalidades à plataforma, este professor iniciou uma parceria com membros do PET e do programa do

mestrado em Tecnologia, Ambiente e Sociedade da UFVJM - Campus do Mucuri, dando origem assim ao Time *Scrum*.

Aplicação do *Scrum*: Estudo de Caso no Projeto Nosso Exercício

A distribuição dos papéis, conforme foram definidos para o Time *Scrum,* pode ser conferida no Quadro 1. Esta distribuição levou em consideração o conhecimento prévio de cada membro nas áreas do saber demandadas na execução do projeto. O Time foi formado por um colaborador externo com alta experiência em desenvolvimento de *software* e quatro alunos do PET que, apesar de não terem formação na área da Computação, já haviam cursado duas disciplinas de programação básica e um curso de Introdução ao Desenvolvimento *Web*. O papel de *Scrum Master* foi assumido por uma aluna do mestrado, formada em Ciência da Computação, com experiência docente em projetos didáticos sobre metodologias ágeis. Por fim, o professor idealizador do projeto atuou no papel de *Product Owner*.

Quadro 1: Distribuição dos papéis definidos para o Time *Scrum*

MEMBROS DA EQUIPE	PAPÉIS
Alunos do PET	Time
Coordenador do PET	Cliente
Professor Programador	Product Owner
Aluna do Mestrado	Scrum Master
Colaborador Externo	Time

Fonte: Própria autora.

A escolha do *Scrum* para o gerenciamento do projeto deu-se por vários fatores. A princípio, a flexibilidade do *framework* permitiu uma adequação à realidade da equipe disponível que só podia se reunir presencialmente uma vez por semana. Nos outros dias, o trabalho era realizado remotamente e a comunicação do grupo foi mantida através do uso da ferramenta Trello, uma espécie de *Kanban* digital. Como os alunos do PET desconheciam a dinâmica do método escolhido, foi realizado um minicurso pela *Scrum Master* que apresentou os seus principais conceitos. A simplicidade na compreensão do método permitiu que os membros do grupo focassem na sua principal deficiência, a falta de domínio das linguagens e ferramentas tecnológicas utilizadas para o desenvolvimento do *website*.

Com intuito de se obter um entendimento mais aprofundado do *software* a ser desenvolvido, foram realizadas reuniões com o *Product Owner* para que as funcionalidades (requisitos[2]) existentes e desejáveis para o *software web* fossem levantadas, compreendidas e registradas (Quadro 2). As funcionalidades desejáveis transformaram-se, posteriormente, no *Product Backlog* do projeto.

Quadro 2: Funcionalidades existentes e desejáveis para o Nosso Exercício

FUNCIONALIDADES EXISTENTES	FUNCIONALIDADES DESEJÁVEIS
Pesquisar exercícios	Permitir o download das listas de exercícios.
Gerenciar lista de exercícios (criar, editar, excluir)	Permitir que o usuário crie uma lista com uma certa quantidade 'n' de exercícios escolhidos aleatoriamente de certo conteúdo.
Gerenciar exercícios (adicionar, editar, excluir)	Permitir criação de questões do tipo múltipla escolha.
Inserir comentários nos exercícios	Permitir que o usuário crie grupos, de modo que apenas aquele grupo veja uma certa lista.
Avaliar nível de dificuldade dos exercícios	Gravar os exercícios sendo digitados como "rascunho", de modo que se algum erro de conexão ocorrer o usuário não perca o seu trabalho.
Denunciar erro (nos casos de erros nas soluções dos exercícios)	Em listas públicas, permitir para o usuário anônimo, exibir link para visualização completa do exercício (atualmente isto só é possível com usuários autenticados).
Denunciar infração (nos casos em que haja infração dos direitos autorais)	Ter um conjunto de links ao lado da barra de busca que exiba todos os exercícios por uma certa disciplina.
	Melhorar o layout/usabilidade da plataforma.
	Realizar buscas apenas por exercícios com comentários e/ou respostas.
	Ao realizar uma pesquisa, retornar a quantidade de itens resultantes da busca.
	Ao compartilhar a lista com alguém, permitir a opção 'com' ou 'sem resposta'.

Fonte: Própria autora.

[2] São descrições dos serviços fornecidos pelo sistema e as suas restrições operacionais. [...] Refletem as necessidades dos clientes de um sistema que ajuda a resolver algum problema [...] (SOMMERVILLE, 2007, p.79).

Alguns riscos já eram conhecidos antes mesmo do início do trabalho, como por exemplo, o nível heterogêneo de conhecimento e experiência nas áreas do saber necessárias para a condução do projeto. Ao discutir e analisar os itens do *Product Backlog* com o Time, mesmo com os treinamentos realizados, foi identificada uma insegurança elevada de alguns membros da equipe (alunos do PET) com as tecnologias envolvidas no desenvolvimento do *website* e até mesmo com a prática do *Scrum.* Em face desta realidade, ficou decidido que mais uma capacitação seria necessária a fim de dirimir estas deficiências. Este estudo de caso é focado nesta capacitação que visou utilizar o *Scrum* para treinamento do próprio *Scrum*, incluindo também as tecnologias envolvidas, a saber: linguagem de programação PHP, linguagem de marcação HTML, linguagem de estilo CSS e o banco de dados MYSQL.

O primeiro dia de execução do projeto foi dedicado ao planejamento. A *Scrum Master* apresentou ao Time o *Product Backlog* e os papéis que cada membro deveria assumir. Foi reforçado pela *Scrum Master* as ações esperadas de cada papel e a importância do comprometimento no trabalho realizado. Devido à mudança repentina de cidade do *Product Owner*, a *Scrum Master* ficou responsável por assumir também o seu papel e ser a porta-voz do andamento das atividades para este membro, que manteve a sua participação no projeto através de videoconferência.

O planejamento traçado para o projeto e os novos requisitos adaptados dentro do contexto do Nosso Exercício e do nível de conhecimento apresentado pelos alunos até então, podem ser conferidos no Quadro 3.

Quadro 3: Planejamento realizado

DATA	*SPRINT*	EVENTOS	REQUISITOS
15/03/17		Reunião de Planejamento do Projeto (2h)	
22/03/17	1	Reunião de Planejamento do *Sprint*1 (1h). Execução do *Sprint*1 (1h_Desenvolvimento/Suporte presencial). (Mais 4 horas remotamente)	Permitir o cadastro de exercícios didáticos.
29/03/17	1 e 2	Reuniões de Revisão e Retrospectiva do *Sprint*1	Permitir a consulta dos

		(1h). Reunião de Planejamento do *Sprint* 2 (1h). (Mais 4 horas remotamente)	exercícios didáticos cadastrados.
05/04/17	2	Execução da *Sprint*2 (1h_ Desenvolvimento/Suporte presencial). Reuniões de Revisão e Retrospectiva da *Sprint*2 (1h).	
12/04/17	3	Reunião de planejamento da *Sprint*3 (1h). Execução da *Sprint*3 (1h_ Desenvolvimento/Suporte presencial). (Mais 4 horas remotamente)	Permitir a criação de listas de exercícios pelos usuários.
19/04/17	3 e 4	Reuniões de Revisão e Retrospectiva da Sprint 3 (1h). Reunião de Planejamento da *Sprint*4 (1h). (Mais 4 horas remotamente)	Definir o layout e realizar a integração do software.
26/04/17	4	Execução da *Sprint*4 (1h_ Desenvolvimento/Suporte presencial). Reuniões de Revisão e Retrospectiva da *Sprint*4 (1h).	
28/04/17		Apresentação do software.	

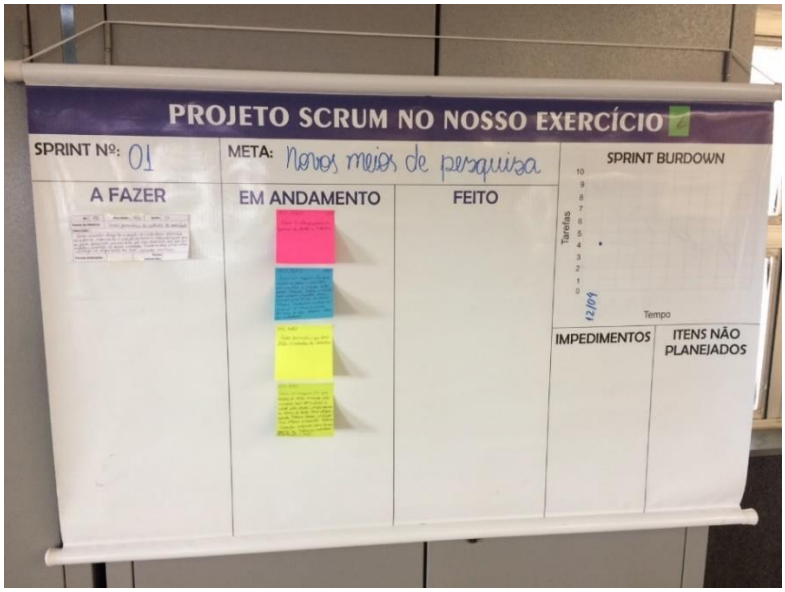
Fonte: Própria autora.

O projeto foi dividido em quatro *Sprints,* cada um com duração de uma semana. Pelo fato da equipe estar envolvida em outras atividades, o tempo total dedicado ao projeto foram de 6 horas semanais, 2 horas presenciais e 4 horas remotamente. Assim, durante os encontros presencias o tempo era em sua maior parte, utilizado para as reuniões de Revisão e Retrospectiva do *Sprint* realizado e para as reuniões de planejamento do novo *Sprint* a ser considerado.

Durante as Reuniões de Planejamento as principais dúvidas sobre o entendimento das funcionalidades a serem executadas em cada *Sprint* eram sanadas pela *Scrum Master* e pelo membro colaborador externo. Devido a sua experiência, este último exerceu um papel significativo no esclarecimento das questões tecnológicas envolvidas no projeto. O conceito de “Pronto” também foi apresentado ao Time. Nesse projeto uma tarefa pronta significava que a funcionalidade havia sido desenvolvida, testada pelo membro que a fez, testada e aprovada pela *Scrum Master*.

Foi providenciado para acompanhamento das tarefas dos *Sprints* um quadro *Kanban* que ficou exposto no ambiente em que as reuniões presenciais aconteceram, visível a todos os integrantes. Neste quadro, as histórias de usuário de cada *Sprint* foram anexadas na coluna “A Fazer” e divididas em tarefas dispostas em *post-is* (notas adesivas) que mudavam de posição no quadro toda vez que um membro prosseguia nas suas atividades. Os *post-is* foram separados por cor, na qual, cada uma representava um membro do Time de Desenvolvimento (Figura 2). Assim, era facilmente possível identificar a tarefa de cada membro e o *status* em que se encontrava.

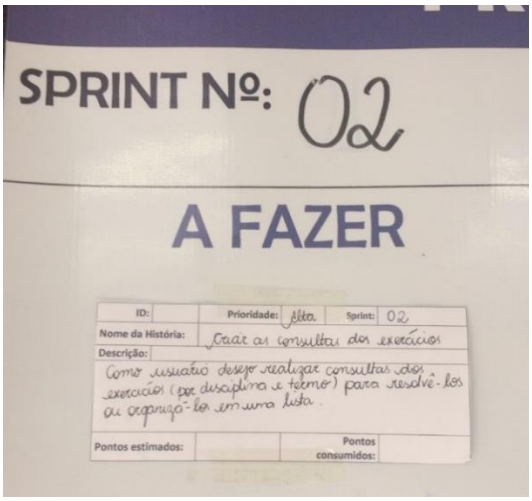
Figura 2- Quadro Kanban utilizado no projeto Nosso Exercício



Fonte: Própria autora.

Teles (2006, p.70) afirma que “não devem existir regras de formato” para a escrita de uma história de usuário. A Figura 3 demonstra um exemplo de história do usuário usada neste trabalho para descrever uma funcionalidade.

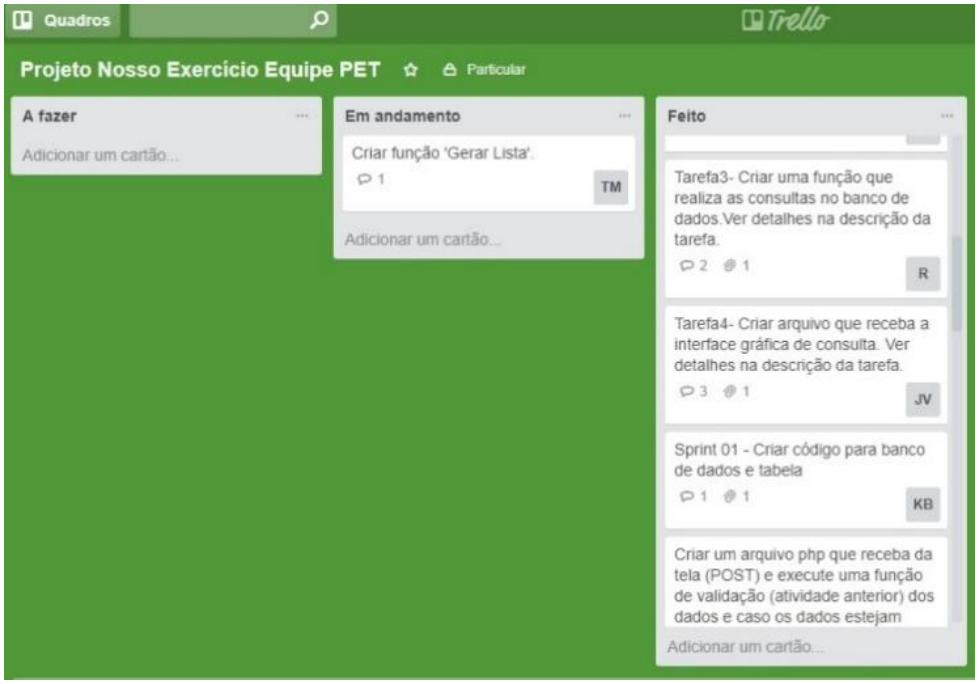
Figura 3- Exemplo de história de usuário utilizada no projeto



Fonte: Própria autora.

Ficou decidido que as Reuniões Diárias não aconteceriam durante os encontros, pois, conforme já mencionado, estes não eram utilizados para a realização da prática. Assim, as perguntas comumente feitas nesta reunião não fariam sentido. Uma alternativa encontrada para isso foi usar a ferramenta Trello e um grupo no *whatsapp* para viabilizar a comunicação da equipe remotamente. Através do Trello, todos os integrantes da equipe, incluindo o *Product Owner*, a qualquer momento podiam ter a visualização do andamento do projeto. Da mesma forma que no quadro físico, cada cartão adicionado no Trello correspondia a uma tarefa que deveria ser executada por um dos membros do Time. Cada membro movia o seu cartão à medida que o trabalho avançava. A Figura 4 demonstra uma tela do Trello no final de um *Sprint*.

Figura 4- Quadro *Kanban* digital



Fonte: própria autora.

Os encontros presenciais iniciavam com o Time atualizando o quadro *Kanban,* sincronizando-o com o Trello, permitindo a todos manter uma única visão sobre o projeto. Durante as Reuniões de Revisão cada membro apresentava o resultado do seu trabalho e o raciocínio utilizado para o mesmo, compartilhando assim o conhecimento entre o grupo. Neste momento também, cada funcionalidade era testada pela *Scrum Master* que fornecia o seu *feedback*. Assim, o Time sabia onde havia acertado e quais correções deveriam fazer para o próximo *Sprint*.

Durante a Reunião de Retrospectiva, o Time tinha a oportunidade de discutir os aspectos positivos e negativos do último *Sprint.* Aqueles considerados positivos eram mantidos e para o que não funcionaram bem, novos cursos de ação eram traçados para que o erro não fosse repetido. Com isso, os riscos iam sendo eliminados à medida que o projeto prosseguia.

Resultados e discussão

Durante a realização deste projeto várias adaptações foram necessárias para que o *framework Scrum* se adequasse à realidade da equipe do Nosso Exercício. O Quadro 4 demonstra um comparativo entre a estrutura original do *framework,* conforme é sugerida comumente na bibliografia, e como este foi efetivamente utilizado neste estudo de caso.

Quadro 4: Adaptações do *Scrum* feitas no desenvolvimento deste trabalho

RECOMENDAÇÕES DO *SCRUM*...	COMO FOI REALIZADO...
Equipes multidisciplinares com habilidades e conhecimentos para desenvolver o software de ponta a ponta (SABBAGH, 2016).	O conhecimento da equipe era heterogêneo, alguns membros possuíam pouca experiência em desenvolvimento de *software*.
Equipes trabalham no mesmo ambiente físico, 8 horas por dia.	O time trabalhou presencialmente e remotamente, 6 horas por semana.
Tamanho ideal para equipes *Scrum*: 6 a 9 componentes.	Tamanho da equipe formada: 7 componentes.
Sprints de tamanho fixo de 1 a 4 semanas.	*Sprints* de tamanho fixo de 1 semana.
Três papéis: *Product Owner*, *Scrum Master* e time de desenvolvimento.	Os três papéis foram definidos, porém, a *Scrum Master* teve que assumir também o papel do *Product Owner*.
O *Product Backlog* é criado e priorizado pelo *Product Owner*.	Apesar de ter sido modificado para se adaptar ao nível de conhecimento do time, o *Product Backlog* foi criado e priorizado pelo *Product Owner/Scrum Master.*
Reuniões de planejamento do *Sprint* devem acontecer no início de cada *Sprint*.	Conforme original.
Reuniões diárias acontecem no início de cada jornada de trabalho.	Optou-se por não realizar as reuniões diárias já que a maior parte do desenvolvimento foi realizado remotamente.
Reuniões de revisão e retrospectiva acontecem no final de cada *Sprint.*	Conforme original.
O time deve estimar o tempo de desenvolvimento para cada história/tarefa.	Não foi cumprido. Alguns membros não registravam o tempo gasto nas tarefas desenvolvidas, além disso, o tempo disponível nas reuniões presenciais nunca eram suficientes para a equipe levantar esta

	discussão.
Os itens do *Product Backlog* são transformados em histórias de usuários.	Atendido. As histórias eram apresentadas para o time e divididas em tarefas entre os membros.

Fonte: Própria autora.

O planejamento traçado não foi fielmente cumprido. Ao todo, foram planejados 4 *Sprints*, porém, apenas 3 foram executados. Inicialmente, alguns encontros presenciais haviam sido programados para que o Time tivesse o tempo de uma hora de desenvolvimento, momento que poderia ser utilizado para a produção em conjunto. A ideia era promover aos membros a experiência de um trabalho em equipe presencial, colaborativo e de fácil comunicação. Já no *Sprint 1* percebeu-se que essa estratégia não poderia ser adotada, pois, mesmo sendo realizada a reunião de planejamento onde as tarefas foram divididas e discutidas entre o Time, o tempo que seria destinado à prática foi tomado por dúvidas em relação às tecnologias utilizadas. Além disso, alguns membros também se perderam na execução dos seus papéis, conforme prevê o *Scrum*. Assim, a meta almejada para este *Sprint* não foi alcançada.

A partir disso, os encontros foram basicamente utilizados para as reuniões. Durante as reuniões de planejamento, as histórias eram apresentadas ao Time, desmembradas em tarefas, discutidas e distribuídas entre os membros. Neste momento também, a *Scrum Master* e o colaborador externo repassavam as orientações técnicas de como desenvolvê-las.

Nas reuniões de revisão e retrospectiva os pontos fortes e fracos do projeto eram desvendados, contribuindo para um aprendizado coletivo contínuo. Durante as reuniões de revisão, cada membro apresentava para a *Scrum Master* o seu trabalho realizado remotamente, explicando-o de forma objetiva para todo o grupo, compartilhando o conhecimento com todos. Nas reuniões de retrospectiva dos *Sprints* os membros reviam as práticas que contribuíram positivamente para a produtividade do grupo, que deveriam ser mantidas e, ainda, aquelas que interferiram negativamente devendo ser eliminadas ou corrigidas.

Os pontos fracos mais citados pelo Time durante as reuniões de revisão foram: ausência ou baixa qualidade dos testes realizados no *software*; necessidade de melhoria da padronização do código, com intuito de deixá-lo mais legível e de fácil compreensão para todos os membros; dificuldades na comunicação entre o

grupo, ocasionando entendimentos ambíguos sobre as tarefas a serem desenvolvidas; dificuldades com a tecnologia adotada. Como a cada *Sprint* estas mesmas deficiências eram relatadas, a equipe foi, aos poucos, criando estratégias para contorná-las. Assim, a comunicação remota tornou-se mais eficiente através do uso do Trello, e-mail e *Whatsapp*, mantendo o grupo em sintonia constante. Ficou estabelecido que, ao final de cada *Sprint*, antes de apresentar o trabalho realizado para a *Scrum Master*, de forma alternada, um membro do Time iria integrar o *software* e testá-lo a fim de melhorar a qualidade do produto final entregue. O próprio Time foi aprimorando seu mecanismo de padronização do código, cobrando uns dos outros a notação definida por eles. A cada *Sprint,* as dificuldades com a tecnologia eram amenizadas através da prática e melhoria na comunicação realizada.

Considerações Finais

Após análise do estudo de caso constatou-se que o uso do *framework Scrum* por estudantes inexperientes em desenvolvimento de *software*, colaborou significativamente para que o Time atingisse seus objetivos no projeto proposto. As metas definidas para cada *Sprint* foram cumpridas, mesmo com atrasos ou mudanças no planejamento. Ao final, o time conseguiu entregar todas as funcionalidades solicitadas, obtendo o aceite da *Scrum Master* e do *Product Owner*. Percebeu-se que, quanto mais o projeto avançava, melhores eram os resultados.

O ciclo de vida iterativo e incremental proposto pelo *framework* favoreceu entregas frequentes do *software* e um aprendizado contínuo da equipe envolvida. A flexibilidade para se adaptar à realidade da equipe permitiu que um método de gerenciamento fosse adotado, orientando o Time em suas atividades, papéis, controle de tempo e metas. As cerimônias realizadas (reuniões), em cada *Sprint*, promoveram ao Time a oportunidade de nivelar o conhecimento sobre o entendimento das tarefas a serem realizadas, de fazer revisões e análises constantes do trabalho executado. Além disso, a transparência do método permitiu ao *Product Owner*, mesmo de longe, ter uma visão realista do andamento do projeto durante todo o tempo.

Relatos feitos pelos alunos envolvidos neste trabalho apontaram como fatores positivos do *Scrum*, em ordem de importância:

- Um alto grau de aprendizado do trabalho em equipe;

- O ritmo imposto, contribuindo para a manutenção do foco em direção às metas;

- A possibilidade de melhoria contínua no aprendizado do método, das tecnologias envolvidas e do *software* proposto; e

- A flexibilidade do *framework*, ajustando à realidade do grupo e propiciando ao Time manter o projeto sob controle até mesmo remotamente.

Já as dificuldades relatadas se referiram ao pouco tempo disponível para a realização do projeto e, principalmente, à falta de domínio das tecnologias utilizadas.

Importante reforçar que, mesmo sendo aplicado em um escopo enxuto, o uso do *framework Scrum* neste projeto permitiu à equipe ter uma visão completa das etapas de desenvolvimento de um *software*, enriquecendo ainda mais o seu aprendizado. Caso o desenvolvimento tivesse sido baseado em um processo tradicional, nas mesmas condições de tempo e espaço, este ganho não poderia ser obtido devido à burocracia, detalhamento e análise aprofundada exigida por este método.

Referências

BROD, Cesar. ***Scrum*_ Guia prático para projetos ágeis**. 2ª Edição. São Paulo: Novatec, 2015. 198p.

GIL, Antônio Carlos. Como elaborar projetos de pesquisa. 5ª edição. São Paulo: Atlas, 2010.

MARIOTTI, Flávio S. **Kanban o ágil adaptativo**. 45 ed. São Paulo. 2012. Disponível em: <http://www.devmedia.com.br/kanban-o-agil-adaptativo-revista-engenharia-de-software-magazine-45/23560 >. Acesso em 05 de junho de 2017.

PRESSMAN, Roger S. **Engenharia de *Software***. 6ª Edição. São Paulo: McGraw-Hill. 2006. 720p.

PRESSMAN, Roger S. **Engenharia de *Software*: uma abordagem profissional.** 7ª Edição. São Paulo: McGraw Hill, 2011. 779p.

SABBAGH, Rafael. ***Scrum*: Gestão ágil para projetos de sucesso**. Editora Casa do Código, 2ª Edição, 2016.

SCHWABER, Ken. SUTHERLAND, Jeff. **"O Guia do *Scrum*. 2016."** Tradução: CRUZ (2016). Disponível em: < http://www.fabiocruz.com.br/wp-content/uploads/2016/09/2016-Guia-doScrum-PtBR-v1FC.pdf>. Acessado em: 12/12/2016.

SOMMERVILLE, Ian. **Engenharia de *Software***. 8ª Ed. Tradução Selma Shin Melnikoff; Reginaldo Arakaki; Edilson de Andrade Barbosa. São Paulo: Pearson, 2007.

STANDISH, Group. ***The Standish Group Caos Report*, 2014**. Disponível em: < http://blog.standishgroup.com/post/18 >. Acessado em:15 de junho de 2017.
SUTHERLAND, Jeff. ***Scrum*: a arte de fazer o dobro do trabalho na metade do tempo.** Leya, 2014.

TELES, Vinícius M. Extreme Programming – Aprenda Como Encantar Seus Usuários Desenvolvendo Software Com Agilidade E Alta Qualidade. Rio De Janeiro. Novatec Editora, 2006. 316 p.
VERSIONONE. **11º Relatório Anual do Estado de Agile, 2016**. Disponível em: < https://explore.versionone.com/state-of-agile/versionone-11th-annual-state-of-agile-report-2> Acessado em: 06/07/2017.

Processo de Avaliação por Pares: (*Blind Review* - Análise do Texto Anônimo)

Publicado na Revista Vozes dos Vales - www.ufvjm.edu.br/vozes em: 10/2019

Revista Científica Vozes Dos Vales - UFVJM - Minas Gerais - Brasil

Www.Ufvjm.Edu.Br/Vozes

Www.Facebook.Com/Revistavozesdosvales

UFVJM: 120.2.095-2011 - QUALIS/CAPES - LATINDEX: 22524 - ISSN: 2238-6424