TESTE DE SOFTWARE USANDO A METODOLOGIA SCRUM

SOFTWARE TEST USING SCRUM METHODOLOGY

RONALDO DE LIMA PORTUGAL[1]
KAMILA RIOS H. RODRIGUES[2]
FABIANA FLORIAN[3]

RESUMO: Este artigo tem a finalidade de discutir sobre a importância da etapa de teste de software nas empresas de Tecnologia da Informação cujo foco está na qualidade do produto. Atividades de teste têm sido indispensáveis, uma vez que reduzem ao máximo a quantidade de erros até o produto chegar ao usuário final. Neste trabalho são apresentados os tipos de testes da literatura e as técnicas, bem como o ciclo de vida do processo usando a metodologia ágil *Scrum,* com seus papéis e práticas. Além disso, é apresentada uma aplicação exemplo, que contém duas funcionalidades para um Sistema de Gerenciamento de Notas Escolares, e que passou pelas etapas de teste usando o método *Scrum*. Resultados parciais apontam uma experiência positiva para o time de desenvolvimento e para o cliente.

Palavras-chave: Teste de Software*,* Metodologia Ágil, *Scrum*.

ABSTRACT: This paper has the purpose of discussing the importance of the software testing stage in IT companies whose focus is on product quality. Testing activities have been indispensable since they reduce to the maximum the amount of errors until the product reaches the end user. In this paper, we present the types of tests and techniques, as well as the process life cycle using the agile Scrum methodology, with its roles and practices. In addition, an example application, which contains two features for a School Note Management System, is presented and

1 Graduando em Sistemas de Informação da Universidade de Araraquara-UNIARA. E-mail: ronaldoportugal60@yahoo.com.br

[2] Orientadora. Doutora da Universidade de Araraquara-UNIARA. E-mail: kemila.uniara@gmail.com
[3] Coorientadora. Doutora da Universidade de Araraquara-UNIARA. E-mail: fflorian@uniara.com.br

passed through the test steps using the Scrum method. Partial results point to a positive experience for both the development team and the customer.

Keywords: Software Testing, Agile Methodology, Scrum.

1 INTRODUÇÃO

Um dos principais objetivos de uma empresa no segmento de Tecnologia da Informação (TI) envolve concorrer com as exigências do mercado para atingir bons resultados, além da busca pela qualidade dos seus produtos. Neste contexto, realizar testes no software pode garantir um produto mais eficiente e eficaz para empresas neste segmento.

Com o crescimento da área de TI, a utilização de softwares se tornou indispensável às inúmeras atividades dentro das empresas. A subárea de teste de software busca revelar os problemas dos produtos digitais e descobrir os defeitos desses antes da entrega e do uso pelos usuários finais.

O processo de desenvolvimento de um software envolve diversas atividades, as quais existem técnicas, métodos e ferramentas a serem empregados. No entanto, mesmo com a aplicação desses recursos, ainda podem ocorrer erros no software que causem impactos no uso.

A atividade de teste consiste, portanto, em uma análise no software para a identificação e a prevenção de erros na entrega do produto final, de modo que não haja problemas críticos para os usuários.

Visando atender à flexibilidade no desenvolvimento de softwares, as metodologias ágeis surgem com o objetivo de reduzir tempo e aproximar os clientes do processo de desenvolvimento.

Uma das metodologias mais empregadas atualmente no mercado é o *Scrum.* Tal metodologia possui um conjunto de valores, princípios e práticas que fornecem a base para que uma determinada organização obtenha resultados satisfatórios, de modo mais sistemático.

Este trabalho pretende ilustrar como os testes de software podem ser utilizados com a metodologia *Scrum,* visando desenvolver soluções de mais qualidade. Para tanto, serão apresentados os conceitos básicos sobre a atividade de teste de softwares, bem como uma explanação sobre a atual importância dos

mesmos e uma avaliação de como deve ser o comportamento dos softwares quando se utilizada a metodologia *Scrum.* O trabalho tem ainda o objetivo de ilustrar como o teste de software influencia na qualidade da solução e como tem sido visto pelo mercado de TI.

É importante ressaltar que para se obter um produto de qualidade, que satisfaça às necessidades dos usuários em suas atividades e que seja entregue em prazos mais curtos, faz-se necessária a utilização dos testes para assegurar que o produto seja entregue o mais correto possível. Quanto mais cedo os defeitos forem encontrados, menor será o custo em relação aos defeitos encontrados em fases posteriores.

Este artigo descreve com fazer uso da metodologia ágil *Scrum* para agilizar os testes durante o processo de desenvolvimento. Os riscos e vantagens de utilizar *Scrum* com o propósito de realizar melhorias na qualidade do software também são discutidos.

O trabalho descreve ainda a aplicação de teste de software utilizando o *Scrum* e uma aplicação prática de testes no processo de desenvolvimento de um sistema. Foi utilizado um software desenvolvido na plataforma *C# e* que gerencia notas escolares. Demonstra-se assim, as etapas necessárias para a aplicação do teste com a metodologia ágil *Scrum.*

2 TESTE DE SOFTWARE

Os seres humanos estão sujeitos a falhas, sendo assim, elas também podem ser encontradas nos softwares e causar transtornos a diferentes partes interessadas no uso da solução computacional. Além disso, muitas equipes trabalham sob pressão, com prazos de entrega curtos, códigos complexos, mão de obra não especializada e mudanças de tecnologia.

Há relatos na literatura de softwares que causaram problemas na fase de produção e trouxeram custos altos, bem como riscos a vida de seres humanos em tratamento de câncer, por exemplo (BASE 2 TECNOLOGIA, 2014), neste caso os hospitais dos EUA utilizavam um aparelho chamado Therac-25 para o tratamento e nele havia um erro de programação no software que aplicava a radiação 100 vezes maior que a recomendada nos pacientes.

Entre as falhas mais graves que a literatura reporta destaca-se a de 1979, quando os sistemas de defesa dos Estados Unidos identificaram que a União Soviética estava preparada para um ataque com mísseis contra o país. Antes de acontecer uma terceira guerra mundial, foi identificado que era apenas um programa de simulação iniciado acidentalmente (BASE 2 TECNOLOGIA, 2014).

Controlar a qualidade de sistemas de software é um grande desafio devido à alta complexidade dos produtos e as inúmeras dificuldades relacionadas ao processo de desenvolvimento, que envolve questões humanas, técnicas burocráticas de negócio e política. (BERNARDO: KON, 2008, pg 54-57).

Segundo Pressman (2011), teste de software é o processo de execução de um produto para determinar se ele atingiu suas especificações e se funciona corretamente no ambiente para o qual foi projetado. O objetivo dessa atividade é resolver falhas em um produto, para que as causas dessas falhas sejam identificadas e possam ser corrigidas pela equipe de desenvolvimento antes da entrega final.

De forma simplificada, testar um software significa verificar por meio de uma execução, se o seu comportamento ocorre de acordo com o especificado. O objetivo principal dessa tarefa é revelar o número máximo de falhas, dispondo do mínimo de esforço.

O teste de software *busca* mostrar o que um programa faz e descobrir os defeitos do programa antes do uso. Quando existe a etapa de teste, dados fictícios podem ser utilizados e os resultados são verificados com o propósito de encontrar erros ou informações sobre os requisitos não funcionais do programa (SOMMERVILLE, 2011). O teste é um tópico mais amplo, muitas vezes conhecido na literatura como verificação e validação (V & V) (PRESSMAN, 2011).

A Verificação diz respeito ao conjunto de tarefas que garantem que o software implementa corretamente uma função específica. A Validação, por sua vez, representa um conjunto de tarefas que asseguram que o software foi criado e pode ser rastreado segundo os requisitos do cliente (PRESSMAN, 2011).

> *O teste proporciona o último elemento a partir da qual a qualidade do software pode ser estimada e, mais pragmaticamente, os erros podem ser descobertos. Mas o teste não deve ser visto como uma rede de segurança. A qualidade é incorporada ao software através do processo de engenharia de software. A aplicação correta de métodos e ferramentas, de revisões, técnicas eficazes, e de um sólido gerenciamento e avaliação conduzem todos à qualidade que é confirmada durante o teste* (*PRESMAN, 2011, pg 468*).

Atualmente existem técnicas para testar um software. De acordo com a norma IEEE 610.12 – 1990 (DEVMEDIA, 2011) essas técnicas são procedimentos técnicos e gerenciais que ajudam a avaliação e a melhoria do processo. As técnicas mais utilizadas são: teste de caixa branca e teste de caixa preta.

O *teste de caixa branca*, também chamada de caixa de vidro, é uma filosofia de projeto de casos de teste que usa a estrutura de controle - descrita como parte do projeto no nível de componentes - para derivar casos de teste. Quando se usa esse método, o engenheiro de software pode criar casos de teste que garantam que todos os caminhos, independentes de um módulo, foram exercitados pelo menos uma vez, e exercitam todas as decisões lógicas nos seus estados verdadeiro e falso (PRESSMAN, 2011).

O *teste de caixa preta* tem o foco nos requisitos funcionais do software. As técnicas do teste de caixa preta permitem derivar séries de condições de entrada que utilizarão completamente todos os requisitos funcionais de um programa. Esse teste tenta encontrar erros nas seguintes categorias: funções incorretas ou faltantes, erros de interface, erros em estruturas de dados ou acesso à base de dados externos, erros de comportamento ou de desempenho, erros de localização e término (PRESSMAN, 2011).

Os testes de software são divididos de acordo com suas funcionalidades. Os principais tipos são:

- Teste de configuração, que verifica a operação do sistema em diferentes configurações de software e hardware;
- Teste de instalação, que proporciona que o software possa ser instalado sob diferentes condições (nova instalação, atualização ou instalação personalizada);

- Teste de integridade, que executa os métodos de acesso e processos, utilizando cada dado válido e inválido, ou solicitações de dados. Deve-se examinar o banco de dados para garantir que os dados tenham sido povoados conforme o planejado. Esse teste também revisa os dados retornados para garantir que estão corretos;
- Teste de segurança, que verifica se os mecanismos de projeção presentes em um sistema vão protegê-lo de evasão imprópria e indevida;
- Teste funcional, em que os softwares são testados como um todo e verifica-se as funcionalidades implementadas, se essas estão de acordo com o que foi especificado nos casos de uso e nas regras de negócios;
- Teste de unidade, cujo foco está no esforço de verificação na menor unidade de projeto do software, buscando avaliar e encontrar possíveis problemas nos módulos que compõem o mesmo, analisando-os isoladamente;
- Teste de instalação, uma técnica sistemática para construir a arquitetura de software ao mesmo tempo em que se conduz testes para descobrir erros associados à interface;
- Teste de usabilidade, usado para determinar o grau com o qual a interface é fácil de usar para o usuário, detectando problemas na integração com o mesmo e;
- Teste de regressão, realizado quando um componente novo ou modificado pode apresentar defeito.

O ciclo de vida da tarefa de teste é composto por cinco etapas: planejamento, preparação, especificação, execução e entrega. O foco principal do processo de teste está nos requisitos que darão origem ao sistema de informação, sendo esses analisados para evitar ambiguidade (BASTOS, 2007).

Na etapa de planejamento são elaboradas as estratégias que serão adotadas para andamento das atividades. A etapa de preparação consiste em preparar o ambiente para que o teste possa ser executado corretamente. Ocorre ainda a preparação dos equipamentos, das pessoas, das ferramentas de automação e da configuração. A etapa de especificação consiste na elaboração de documentos, os casos de testes que serão utilizados durante os procedimentos. Na etapa de execução a avaliação é feita de acordo com os requisitos aprovados pelo cliente. Os

resultados obtidos são registrados. A etapa de entrega é a ultima fase do ciclo de vida dos testes, e esses devem estar finalizados e documentados em arquivo. A entrega do produto deve estar de acordo com o que foi planejado com o cliente (BASTOS, 2007).

3 DESENVOLVIMENTO DE SOFTWARE

O desenvolvimento de softwares é composto por um conjunto de atividades relacionadas, que levam à produção de um produto de software. Essas atividades envolvem o desenvolvimento desde as etapas iniciais de uma determinada linguagem padrão de programação (PRESSMAN, 2011).

Atualmente, novos softwares são desenvolvidos por meio da extensão e modificação de sistemas existentes, ou por meio de configuração e integração de prateleira ou componentes do sistema. Existem quatro atividades fundamentais para a área da Engenharia de Software (SOMMERVILLE, 2011):

1. Especificação de software: A funcionalidade do software, suas restrições e seu funcionamento devem ser definidos;
2. Projeto e implementação de software: O software deve ser produzido para atender às especificações;
3. Validação de software: O software deve ser validado para garantir que atenda à demanda do cliente;
4. Evolução de software: O software deve evoluir para atender às necessidades de mudança dos clientes.

Os modelos de processos de software são uma representação simplificada de um processo de software. Cada modelo representa uma perspectiva particular de um processo e, portanto, fornece informações parciais sobre ele. Modelos de atividades podem mostrar as atividades e as suas sequências, mas não mostra os papéis das pessoas envolvidas (PRESSMAN, 2011).

Os principais modelos de processos abordados na literatura são: cascata, prototipação, espiral, incremental, e modelo unificado (PRESSMAN, 2011).

No Modelo Cascata (Figura1), também chamado de *ciclo de vida clássica,* é sugerida uma abordagem sequencial e sistemática para o desenvolvimento do

software. Inicialmente é feito o levantamento das necessidades do cliente e, na sequência, avança pelas fases de planejamento, modelagem, construção, emprego, culminando no suporte contínuo do software já concluído. Esse modelo é usado em sistemas críticos, quando os requisitos são bem compreendidos e quando há pouca probabilidade desses requisitos mudarem (PRESSMAN, 2011). O modelo Cascata tem como vantagem a documentação rígida em cada atividade, o que reflete abordagens adotadas em outras engenharias. Como desvantagens, ocorre que projetos reais raramente seguem um fluxo sequencial, sendo assim, é difícil de adaptar mudanças inevitáveis de requisitos e uma versão executável somente ficará pronta na fase final do projeto (PRESSMAN, 2011).

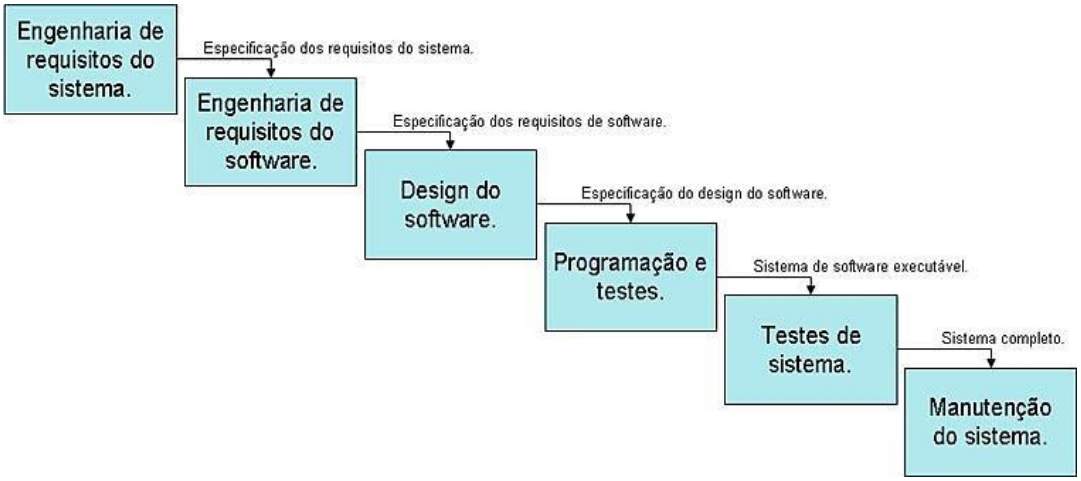

Figura 1- Modelo de processo em cascata.

Fonte: PRESSMAN (2011).

O Modelo de Prototipação (Figura 2), é uma versão inicial de um sistema de software usada para demostrar conceitos, experimentar opções de projeto e descobrir mais problemas e suas possíveis soluções. Um protótipo de software pode ser usado em um processo de desenvolvimento para ajudar a antecipar as mudanças que podem ocorrer. É utilizado para produzir as incertezas do produto, pois são criadas interfaces para que o cliente possa compreender melhor os requisitos do sistema, auxiliando na identificação de uma solução final mais assertiva. É utilizada para criação rápida de software (SOMMERVILLE, 2011). As vantagens desse modelo incluem auxiliar no entendimento dos requisitos que estão mal explicados, bem como auxiliar no envolvimento do usuário com os desenvolvedores/analistas. Já as desvantagens incluem que o cliente tende a

confundir o protótipo com a versão final, o que causa baixa qualidade no produto e alto custo de desenvolvimento.

> *O protótipo pode servir como "o primeiro sistema". Aquele que Brooks recomenda que se jogue fora. Porém, essa pode ser uma visão idealizada. Embora alguns protótipos sejam construídos com "descartáveis", outros são evolucionários, no sentido de que evoluem lentamente até se transformar no sistema real* (PRESSMAN, 2016, 46).

Figura 2 – Modelo de Prototipação.

Fonte: PRESSMAN (2011, p. 36).

O Modelo Espiral (Figura 3), é um modelo de processo de software evolucionário que agrega a natureza interativa da prototipação, com aspectos sistemáticos e controlados do Modelo Cascata, e fornece potencial para o rápido desenvolvimento de versões cada vez mais completas do software. Usando-se o modelo espiral o software será desenvolvido em uma série de versões evolucionárias. Nas primeiras iterações a versão pode consistir em um modelo ou em um projeto. Ele é utilizado no desenvolvimento de jogos e projetos em que os objetivos são instáveis.

A iterações nesse modelo torna-se mais realísticas com progresso do trabalho, uma vez que os problemas importantes são descobertos mais cedo, e se torna mais fácil lidar com mudanças que o desenvolvimento exige. É ainda, fácil de decidir o que testar.

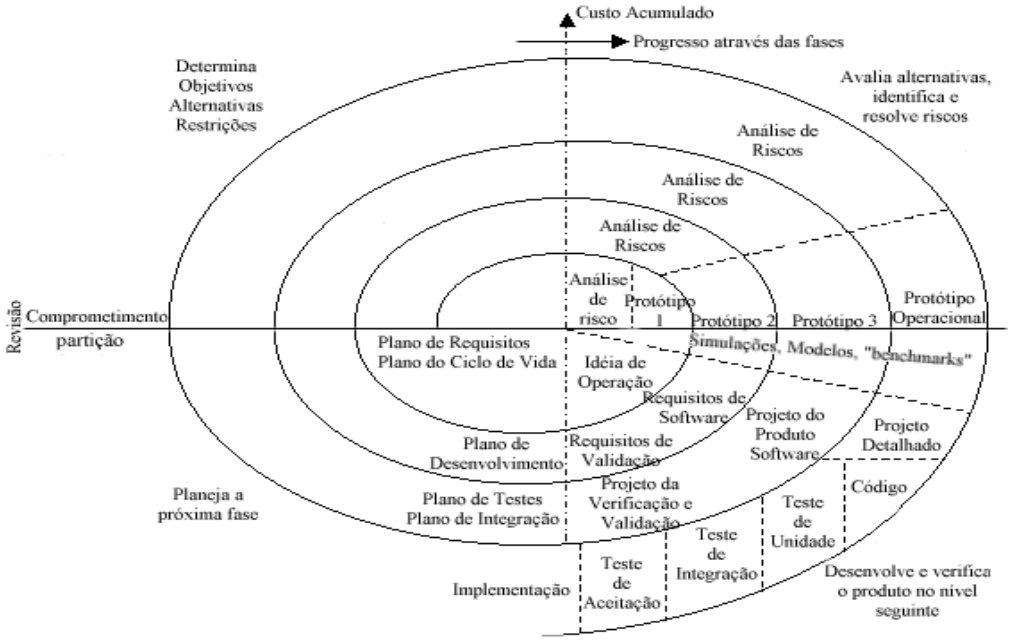

Figura 3 – Modelo de processo espiral.

Fonte: BOEHM (1986).

O modelo Espiral tende a ser difícil de convencer os clientes, a abordagem evolucionaria é controlável e não é usado na mesma extensão que o linear, e o de prototipação é bem aplicado somente a sistemas em larga escala (PRESSMAN, 2011).

O Modelo Incremental, (Figura 4) é baseado na ideia de desenvolver uma implementação inicial, expô-la aos comentários dos usuários e continuar por meio da criação de várias versões até que um sistema adequado seja desenvolvido. Esse modelo permite trabalhar com o cliente o entendimento dos requisitos, e é possível começar o sistema pelas partes melhor entendidas. O processo pode não ser muito claro nesse modelo e o produto final é frequentemente mal estruturado, pois mudanças contínuas tendem a corromper a modularidade do sistema (SOMMERVILLE, 2011).

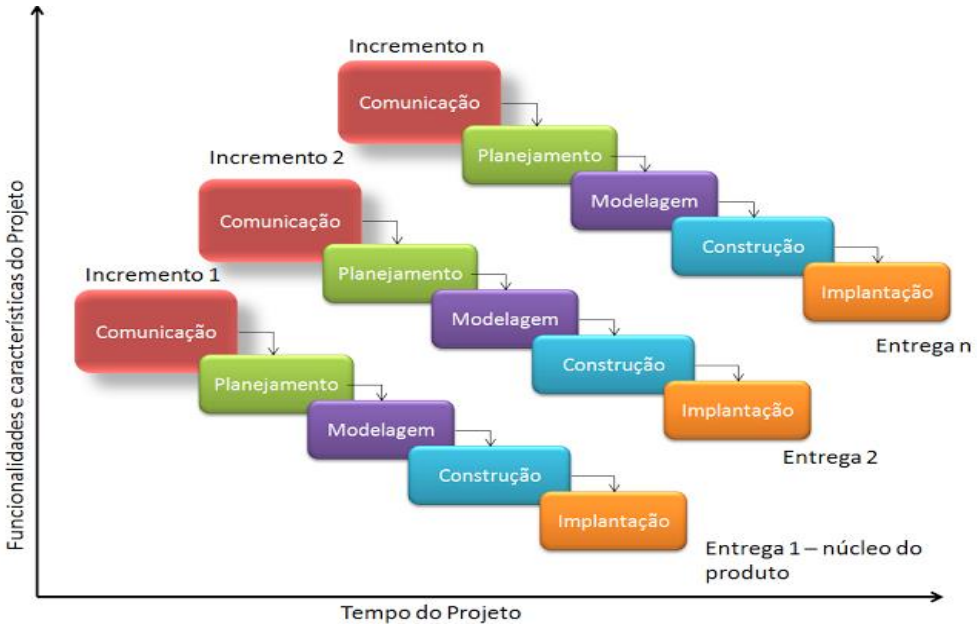

Figura 4 – Modelo incremental.

Fonte: PRESSMAN (2010).

O Modelo Unificado é uma tentativa de aproveitar os melhores recursos e características dos modelos tradicionais de processos de software, caracterizando-os de modo a implementar os melhores princípios do desenvolvimento ágil de software. O processo unificado reconhece a importância da comunicação com o cliente e de métodos racionalizados para descrever a visão do cliente sobre um sistema. Esse modelo tem a base sólida para construção do software, bem como a identificação de riscos e o aprendizado facilitado, mas, é utilizado em um processo para software grande e usa o UML (*Unified Modeling Language*) como linguagem de modelagem (PRESSMAN, 2011).

O Desenvolvimento Ágil é um conjunto de metodologias de desenvolvimento de software que providencia uma estrutura conceitual para reger projetos de Engenharia de Software (PRESSMAN, 2011).

Em 2001 desenvolvedores, autores e consultores renomados na área de software assinaram um “Manifesto” para o desenvolvimento ágil. Normalmente, um manifesto é associado a um movimento político emergente, sugerindo uma mudança revolucionária (PRESSMAN, 2011). Para esse manifesto disseminar e ajudar outros a desenvolverem software, passou-se a valorizar passos como:

- Indivíduos e iteração acima, de processos e ferramentas;

- Software operacional acima, de documentação completa;
- Colaboração dos clientes, acima da negociação contratual;
- Respostas às mudanças, acima de seguir um plano.

Os métodos ágeis constituem um conjunto de práticas criadas com o objetivo de tornar o desenvolvimento do software mais rápido e com custo acessível. Outra caraterística dos métodos ágeis é a flexibilidade oferecida ao desenvolvedor. Sendo assim, é possível se adaptar facilmente a mudanças no decorrer do desenvolvimento.

Os processos ágeis de desenvolvimento de software são caracterizados por relacionar uma série de preceitos acerca da maioria dos projetos de software, entre eles (PRESSMAN, 2011):

I. É difícil afirmar antecipadamente quais requisitos de software irão persistir e quais sofrerão alterações. É difícil prever de que maneira as prioridades do cliente sofrerão alterações conforme o projeto avança;
II. Projetos “Inter conduzidos”, ou seja, em sequência, para que sejam aprovados conforme sejam criados;
III. Análise, projetos, construções (desenvolvimento) e testes não são tão previsíveis.

Um processo ágil de software deve ser adaptado de maneira incremental. Para isso, a equipe ágil precisa de *feedbacks* do cliente para fazer as adaptações apropriadas. Um efeito catalisador para o *feedback* do cliente é um protótipo operacional, ou parte de um sistema operacional. Dessa forma, deve se instruir uma estratégia de desenvolvimento incremental (PRESSMAN, 2011).

Scrum é um exemplo de método de desenvolvimento ágil de software, criado por Jeff Sutherland e sua equipe no início dos anos 1990 (PRESSMAN, 2011). Os princípios do *Scrum* são consistentes com o manifesto ágil e são usados para orientar as atividades de desenvolvimento dentro de um processo que incorpora as seguintes atividades estruturais: requisitos, análise, projeto, evolução e entrega. Em cada atividade metodológica há tarefas a serem realizadas dentro de um padrão de processo chamado *Sprint.* O trabalho realizado dentro de um *Sprint* é adaptado ao

problema em questão definido e, muitas vezes, modificado em tempo real pela equipe *Scrum* (PRESSMAN, 2011).

3.1 PAPÉIS PRESENTES NO *SCRUM* E A SUA IMPORTÂNCIA

3.1.1 *Product Owner* - PO

O PO é o conhecedor do negócio e da necessidade do usuário final. Tendo esse conhecimento, ele pode priorizar as necessidades do usuário e decidir o que mais agrega valor ao negócio, para que seja assim, entregar primeiro. O PO pode ainda decidir o que foi desenvolvido, se é suficiente para atender as necessidades do cliente e finalizar o projeto. Ele é o ator que separa uma lista de desejos, chamada de *Product Backlog,* e que contém todos os requisitos que inicialmente são considerados necessários para atender ao negócio do usuário final (MINDMASTER, 2018).

3.1.2 *Scrum Master*

Scrum Master provém de uma atividade que ocorre durante a partida de *rugby*. Esse ator atua como líder de equipe e *coach* tanto do *Product Owner,* quanto do time de desenvolvimento. Ele conhece o processo e assim tem condições de instruir o *PO* em relação às suas atribuições. Ele também guia o PO e o time durante o projeto (MINDMASTER, 2018).

3.1.3 *Time Scrum*

O *Time Scrum* é composto por aqueles que criam os incrementos no produto. Todos os membros podem desempenhar qualquer função que seja necessária para o projeto (em alguns projetos, os membros têm funções específicas). O tamanho do time varia de acordo com o projeto. As equipes devem ter entre 5 e 9 membros. O time deve ser auto organizável, ou seja, o próprio quem decide quem realiza quais tarefas e o que cabe ou não na *Sprint* (MINDMASTER, 2018).

3.2 PRÁTICAS DO *SCRUM*

3.2.1 *Sprint*

No *Scrum* o trabalho é realizado em iterações ou ciclos de até um mês de calendário chamado de *Sprint.* O trabalho realizado em cada *Sprint* deve criar algo

de valor tangível para o cliente ou usuário. *Sprints* têm sempre um início e fim, com uma data fixa.

3.2.2 *Sprin*t de Planejamento

Para determinar quais subconjuntos de itens são mais importantes e para construir a próxima *Sprint*, o *Product Owner,* em conjunto com o time de desenvolvimento e *Scrum Master,* realizam o planejamento da *Sprint,* chamada de *Sprint Planning*. As equipes *Scrum* que realizam *Sprints* com duração de duas semanas a um mês tentam completar o planejamento em cerca de 4 a 8 horas.

3.2.3 Reunião Diária - *Daily*

No mesmo horário, todos os dias, todos os membros da equipe de desenvolvimento realizam uma reunião de no máximo 15 minutos chamada de *Daily*. Nessa reunião rápida, em que todos os membros estão alinhados com informações do projeto, o *Scrum Master* faz três perguntas para cada membro. São elas (MINDMASTER, 2018).

1) O que fiz ontem que ajudou o time a atingir a meta do *Sprint*?
2) O que eu vou fazer hoje para ajudar o time a atingir a meta do *Sprint*?
3) Existe algum impedimento que não permita a mim ou ao time atingir a meta do *Sprint*?

3.3.4 Revisão do *Sprint*

Ao final de cada *Sprint* é realizada uma reunião, em que o projeto é avaliado e o Time *Scrum* apresenta os resultados obtidos com os itens 100% prontos. Essa reunião tem duração de no máximo 2 horas e nela são dispensados vídeos e *slides*, pois é uma reunião informal. Se houver necessidade de mudanças com a incorporação de novas tarefas no projeto, essas são agendadas e revisadas novamente.

3.3.5 Retrospectiva do *Sprint*

Depois da revisão do *Sprint* faz-se a retrospectiva do *Sprint.* Essa reunião tem duração de 3 horas, para um *Sprint* de um mês, e verifica a adaptação no processo de trabalho (MINDMASTER, 2018).

A seção a seguir descreve a aplicabilidade do *Scrum* nas tarefas de teste de software.

4 APLICABILIDADE DO TESTE DE SOFTWARE COM O SCRUM

Um estudo foi realizado no contexto deste trabalho para verificar a aplicação do teste de software em conjunto com o método ágil *Scrum,* bem como para destacar e documentar os resultados obtidos para que outros pesquisadores possam usar como referências.

Para realizar os testes, um software de gerenciamento de notas escolares foi desenvolvido usando a plataforma C# *desktop* (AUTORIA PRÓPRIA). Foram utilizadas duas funcionalidades do sistema: 1) Cadastrar Usuário e 2) Inserir Notas.

A separação dos papéis do *Scrum* neste pequeno projeto é a seguinte: a) **PO** - representa o conhecedor do produto e das necessidades do usuário final. Neste projeto teve a função de conhecer e defender os interesses do cliente para que o produto fosse entregue tudo no prazo acordado; b) ***Scrum Master*** - foi o responsável pelo andamento do projeto, resolvendo as dificuldades que surgiram; c) **Time *Scrum*** - responsáveis por planejar e aplicar os testes no sistema. Quando os mesmos encontraram ocorrências durante a análise, eles registraram e apresentaram os resultados no final das atividades; d) **Analista de Teste** - responsável pela visão geral do produto, no decorrer da *Sprint.* Participou da fase de definição, na colaboração nas reuniões com clientes, e com o foco na qualidade do produto; e) ***Testadores*** - responsáveis pela execução dos testes para encontrar possíveis defeitos e assim, usar as técnicas e métodos que foram planejados.

4.1. Planejamento do Teste

As principais etapas na execução do teste no sistema, de acordo com as ferramentas e ao *Scrum* foram:

- 1º etapa: Elaboração dos documentos – casos de uso foram utilizados na descrição e controle dos requisitos. O caso de uso também descreve os requisitos implementados e validados;

- 2º etapa: Definição da duração da *Sprint:* Teve duração de cinco dias, sendo dois dias úteis para os testes, para cumprir as tarefas e entregar funcionalidades, testadas e validadas de acordo com os requisitos;
- 3º etapa: Aplicação dos testes: As técnicas de teste utilizadas para a definição foram: Teste Funcional; Teste de Integridade; Teste de Segurança; Teste de Integração e Teste de regressão.

4.2. Elaboração dos Documentos

Foram desenvolvidos dois tipos de teste, o caso de uso e o caso de teste. A Figura 5 ilustra o caso de Uso do Sistema Gerenciamento de Notas Escolar. A Figura 6 ilustra o caso de teste criado.

Figura 5 – Caso de uso: Projeto Sistemas Gerenciamento de Notas Escolar.

EC001: Cadastrar Usuário
O Sistema contém uma tela ADM, em que podem ser cadastrados três tipos de usuários: Aluno, Professor e Administrador. Para cada dos tipos foi atribuído um ID, que será usado na validação do programa. Ao acessar o botão usuário na tela do Administrador, o sistema abre uma janela para cadastrar dados do usuário, como login e senha. Esse procedimento serve para os três tipos de usuários.
EC002: Inserir Notas
O sistema contém um botão para Inserir Notas que é acessado pelo Professor em sua página, ele abre a tela de inserção de Notas, seleciona a disciplina e perfil dos alunos cadastrados na disciplina.

Fonte: Autoria Própria.

Figura 6 – Caso de teste: Projeto Sistemas Gerenciamento de Notas Escolar.

Caso de Teste	**Ação**	**Resultado Esperado**	**Resultado Obtido**
CT001	Acessar a área do ADM no sistema para o cadastro de usuário.	O sistema identifica e exibe o comando do administrador na tela de cadastro.	Aprovado
CT002	Acessar a área do Professor para Inserir as Notas.	O Sistema identifica e exibe o comando do administrador na tela do professor para Inserir as Notas.	Aprovado

Fonte: Autoria Própria.

4.3. Execuções dos Testes

A execuções dos testes nos dois casos de uso EC001 e EC002 foram feitos em dois ciclos, sendo realizados por um profissional distinto dentro do time. O testador seguiu os passos acessando a tela inicial do sistema de Gerenciamento de Notas Escolar, na função do cadastro de usuários, tendo como base as informações para validar sua funcionalidade.

Após a execução do teste na função cadastro de usuário, os resultados observados foram positivos, puderam ser atualizados com sucesso, sendo, portanto, aprovado.

Na execução do teste, na função de inserir notas, o testador seguiu os mesmos passos, obtendo um resultado também positivo para essa função que foi atualizada como aprovada.

4.4. *Daily Meeting* – Encontro Diário

Nas reuniões diárias foi discutido sobre as atividades de teste para o time. Nesse encontro são questionados todos os problemas encontrados pelo testador no seu primeiro dia, as ocorrências feitas e discute-se rapidamente como foram solucionadas. Também nessa reunião foram colocadas em pauta as atividades do dia seguinte, referente ao caso de uso para ser testado e validado, e finalizadas as atividades de teste na *Sprint.*

4.5. Defeitos Encontrados

Para o gerenciamento dos defeitos encontrados nos testes, recomenda-se utilizar uma ferramenta que controle desde a sua fase inicial, até seu encerramento.

4.6. Verificação e Validação para testes usando o *Scrum*

Quando é aberta uma ocorrência de defeito, o desenvolvedor realiza uma validação para certificar se de fato há um defeito a ser corrigido. Se na verificação o desenvolvedor encontrar essa falha ele realiza a correção. Caso não encontre o erro no sistema, ele mesmo pode rejeitar a ocorrência.

Os testadores refazem os testes seguindo todos os passos. Se o problema estiver corrigido, eles atualizam o *status* para aprovado e encerra a ocorrência. Se o

defeito persistir, o testador retorna a ocorrência para que o desenvolvedor realize a correção de maneira apropriada. Esse processo é recorrente até que o defeito seja corrigido e validado.

4.7. Encerramento da Ocorrência

Quando a ocorrência é corrigida pelo desenvolvedor e validada pelo testador, pode-se encerrá-la. O testador aprova a ocorrência através do sistema de gerenciamento e finaliza o trabalho.

4.8. Levantamento dos Resultados

A Metodologia Ágil *Scrum* foi utilizada neste trabalho para ilustrar que é possível manter um gerenciamento eficaz nas atividades de teste. A *Sprint* seguiu as atividades que foram planejadas passo a passo e assim, pôde ser entregue dentro dos prazos que foram estipulados pelo cliente, com qualidade e satisfazendo as necessidades desse cliente. A equipe obteve resultados agradáveis e positivos para todos os envolvidos.

5 CONCLUSÃO

Neste artigo foram apresentados os conceitos básicos e fundamentais da atividade de teste de software, juntamente com o processo ágil *Scrum,* que tem sido bastante utilizado pelas empresas de TI.

Após as análises e as informações observadas com o estudo, apresentou-se os principais passos do ciclo de vida do teste e a separação dos papéis das equipes usando o *Scrum*. Discutiu-se ainda ações para serrem feitas na solução quando se encontra defeitos, bem como sugestões sobre a validação, a verificação e os levantamentos dos resultados obtidos.

Com esse estudo foi possível demonstrar os benefícios da aplicação do *Scrum* no teste de software. A equipe adquire boas experiências, trabalha mais unida e com maior integração. É possível obter ainda um aumento de produtividade com previsões mais acertadas nas entregas. Essas ações podem garantir a qualidade do produto desenvolvido, proporcionando ao cliente o resultado esperado, dentro do prazo planejado e atendendo aos requisitos solicitados.

REFERÊNCIAS

BASTOS, E. R. C. **Base de conhecimentos em teste de Software**. Segunda edição, São Paulo: Martins, 2007.

BERNARDO: KON, 2008, pg 54-57 A importância de Testes Automatizados.

Ciclo de vida do Teste Disponível em: **<**http://analistadetestes.blogspot.com.br/2013/05/ciclo-de-vida-dos-testes-e-do.html> Acesso em: 21 Fev. 2018.

ESPIRAL Disponível em: <https://engenhariasoftware.wordpress.com/2013/02/01/os-modelos-evolutivos-de-processo-de-software-espiral/ > Acesso em: 04 Mar. 2018.

FALHA DE TESTE Disponível em <http://www.base2.com.br/2014/01/29/10-falhas-software-marcaram-historia/> Acesso em: 04 Mar. 2018.

PAPÉIS PRESENTES NO SCRUM Disponível em: <https://agilemomentum.com.br/2014/04/30/quem-e-voce-os-3-papeis-do-scrum-e-sua-importancia/ > Acesso em: 25 Abr. 2018.

PRÁTICAS DO SCRUM Disponível em: <http://www.mindmaster.com.br/scrum/> Acesso em: 25 Abr. 2018.

PRESSMAN, R.S. **Engenharia de Software Uma abordagem Profissional**. Sétima edição Porto Alegre: AMGH, 2011.

PROCESSO ÁGIL. Disponível em: <https://pt.wikipedia.org/wiki/Desenvolvimento_%C3%A1gildesoftware> Acesso em: 17 Mar. 2018.

SCRUM. Disponível em <http://www.mindmaster.com.br/scrum/>, <https://www.significados.com.br/scrum/> Acesso em: 20 Mar. 2018.

SOMMERVILLE, I. **Engenharia de Software**. Nona edição São Paulo: Pearson Prentice Hall, 2011.

SOFTWARE. Disponível em:< http://www.devmedia.com.br/artigo-engenharia-de-software-introducao-a-teste-de-software/8035> Acesso em: 15 fev. 2018.

SOFTWARE. **Configuração e instalação**. Disponível em: <http://pts.datasus.gov.br/PTS/default.php?area=0204L17825> Acesso em: 16 Fev. 2018.

Técnica de Teste Disponível em < https://www.devmedia.com.br/testes-de-software-tecnicas/22283 > Acesso 17 Fev. 2018

Teste de Regressão. Disponível em: <http://www.ic.unicamp.br/~ranido/mc626/Regressao.pdf > Acesso em: 21 Fev.2018.

TESTE DE INTEGRIDADE. Disponível em

<http://pts.datasus.gov.br/PTS/default.php?area=0204L17818> Acesso em: 04 Mar. 2018.