ISSN: 2175-1897

Um estudo de caso do processo de testes automáticos e manuais de software no desenvolvimento ágil

RAQUEL BORTOLUCI
Centro Paula Souza – SP – Brasil
raquelborto@gmail.com

MARCELO DUDUCHI
Centro Paula Souza – SP – Brasil
mduduchi@gmail.com

Resumo - A fim de responder às mudanças do mercado e melhorar a qualidade de software, muitas empresas estão adotando métodos ágeis para desenvolvimento de software. O objetivo deste estudo é identificar quais as técnicas que podem ser utilizadas no processo de garantia de qualidade, a fim de acelerar o processo para que estes se encaixam nos princípios ágeis e mantenham a qualidade do software. No artigo serão analisadas quais técnicas utilizadas na área de teste de software durante o desenvolvimento ágil e como elas podem ser aplicadas para seguir os princípios ágeis e também considerar os fatores de qualidade para criar uma boa estratégia de teste com foco nas técnicas manuais e automáticas neste contexto.

Palavras-chave: ágil; teste de software; teste manual; teste automatizado.

Abstract - In order to respond to the market changes and improve the quality of software, many companies are adopting agile methods to develop software. The goal of this study is to identify what techniques can be used in the quality assurance process in order to accelerate the process, fit the agile principles on it and keep the software quality. In the paper will be analyzed what techniques are utilized in the software testing area during the agile development and how they can be applied to follow the agile principles and also consider the quality factors to create a good test strategy focusing in the manual and automatics techniques in this context.

Keywords: agile; software testing; manual testing; automated testing

1. Introdução

ISSN: 2175-1897

O modelo tradicional de desenvolvimento de software segue uma sequência de etapas de desenvolvimento que, em sua maioria, consideram: a análise e definição de requisitos; o projeto de sistema e software; a implementação e o teste de unidade; a integração de teste de sistema; e a operação e manutenção. O teste de software neste modelo utiliza-se de requistos para derivar casos de testes e a partir destes se implementa a execução de testes manuais e automatizados (ISO/IEC/IEEE 29119-1).

Já os processos no desenvolvimento ágil consideram uma diversidade grande de abordagens. Uma delas é a de trabalhar com pequenas iterações com entregas ao final de cada iteração (Beck, K. *et. Al*, 2001). Por causa da natureza do ciclo de vida deste desenvolvimento, os testes devem ser executados de forma continua e integrada desde o início do projeto, tendo o teste de software como parte integrante de todo o processo de desenvolvimento (CRISPIN e GREGORY, 2009).

Apesar da diferença que existe no conceito do teste de software ágil, muitas vezes, as técnicas e os tipos de testes são os mesmo que os utilizados nos métodos tradicionais (VEENENDAAL, 2010). Distribuir as atividades de teste dentro um *sprint* a fim de garantir que estas sejam executadas de forma continua, requer bastante esforço e deixar os testes para o final do ciclo, leva ao risco de apenas seguimentar o desenvolvimento em pequenos ciclos mantendo em cada um deles a mesma sequência de etapas de desenvolvimento do modelo tradicional (COHN, 2013). Para que isso não aconteça é necessário adaptar as técnicas e práticas de teste ao contexto ágil de forma que possam estar de acordo com os princípios ágeis que valorizam mais os indivíduos e iterações do que processos e ferramentas, o software funcional mais do que a documentação abrangente, a colaboração do cliente mais do que a negociação de contrato e as respostas às mudanças mais do que a orientação a um planejamento (BECK, K. *et. Al*, 2001).

Neste sentido, é imprescindível adotar uma boa estratégia de teste para o sucesso no uso de métodos ágeis para o desenvolvimento de software, empreendendo esforço para tornar as atividades mais rápidas e adequadas aos princípios ágeis. Este artigo tem por objetivo analisar técnicas de teste de software quanto a utilização de práticas manuais e automáticas para acelerar a validação e verificação do software baseando-se na experiência adquirida durante a implementação de métodos ágeis em uma grande empresa de desenvolvimento de software.

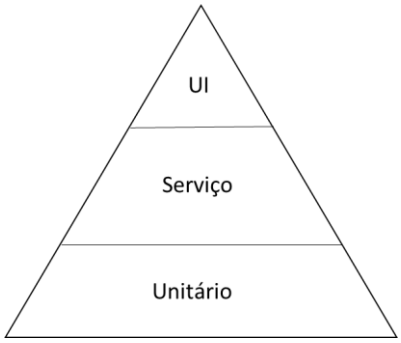
2. Referencial Teórico

Define-se como teste de software o processo de execução de um programa com a finalidade de encontrar erros, aumentando, assim, a confiabilidade do software (MYERS, BADGETT e SANDLER, 2012). Observa-se que a atividade de teste é a principal técnica de verificação e validação de software (SOMMERVILLE, 2007). Enquanto nas equipes tradicionais o teste de software é realizado em fases e por equipes diferentes do desenvolvimento, o teste ágil é um processo colaborativo, no qual todos estão envolvidos desde a definição até a execução do plano de testes (MYERS, BADGETT e SANDLER, 2012). Já que o desenvolvimento em ágil prevê um processo iterativo e incremental, o teste deve ocorrer desde o início até o fim de um *sprint*, no qual são utilizados tanto testes automatizados quanto manuais para atingir este objetivo (CRISPIN e GREGORY, 2009).

Os testes automáticos são programas ou *scripts* que exercitam as funcionalidades do sistema e desta forma verificam se o software está funcionado de acordo com o esperado (BERNARDO e KON, 2008). Estes testes são mais caros no início do projeto, mas quando corretamente implementados, diminuem o custo do processo de teste de software, pois podem reduzir o tempo de execução de tarefas repetitivas (ALSMADI, 2012). Muitos fatores contribuem para o sucesso da automação de testes. Entre os elementos mais importantes para esta contribuição estão o suporte da gerência para automação, definindo metas realísticas e provendo recursos apropriados para alcançar os objetos; e uma boa estrutura técnica, utilizando-se níveis certos de abstração que possam dar flexibilidade, adaptabilidade e redução de custos (GRAHAM e FEWSTER, 2011).

Mesmo antes da ascendência dos métodos ágeis, já se dava importância à automação de testes, mas por ser considerada uma atividade cara e demorada, não era realizada constantemente pelas equipes. Um dos motivos para este problema é que muitas vezes se aplicava uma estratégia de automação no nível errado. Uma estratégia de automação efetiva pode ser realizada nos seguintes níveis: unitário, de serviços e de interface de usuário (*User Interface* - UI) (COHN, 2013) conforme observado na figura 1.

Figura 1 - Pirâmide do teste de automação



Fonte: Adaptado de Cohn (2013) – *Succeeding with agile*

Nesta proposta de estratégia de automação, verifica-se que a maioria dos testes automatizados devem ser executados no nível de teste unitário. Estes testes, por serem escritos com a mesma linguagem do sistema, são mais confortáveis para os desenvolvedores escrevê-los. Os testes de serviço devem ser executados amplamente. Eles servem para verificar as respostas da aplicação em relação à diferentes entradas ao sistema, portanto ao invés de se aplicar vários testes ao nível de interface estes testes são aplicados ao nível de serviço. Por último, os testes de interface de usuário que devem ser executados em uma escala menor, pois são mais frágeis (qualquer alteração do sistema pode quebrar a automação). A criação destes testes tem custo mais elevado e estes levam mais tempo para serem executados (COHN, 2013).

Observa-se que a automação de testes garante certo nível de confiança ao produto que é desenvolvido de forma incremental e iterativa, permitindo a entrega em tempo hábil e o atingimento dos objetivos dos métodos ágeis no contexto do teste de software (OLUWOLE, 2013). Crispin e Gregory (2013) destacam que nem sempre é possivel manter um conjunto grande de testes automatizados sendo executados durante todo o processo de desenvolvimento, pois o custo para mantê-

ISSN: 2175-1897

los pode ser maior que o benefício que este irá trazer, portanto deve-se sempre verificar o risco e o custos destes testes e fazer ajustes para se ter uma execução adequada ao contexto de desenvolvimento.

Verifica-se que os testes automatizados são bastante úteis para substituição de tarefas repetitivas, no entanto novos defeitos são em geral descobertos com a utilização de testes manuais (BERNER, WEBER e KELLER, 2005). Os testes manuais são processos de execução do software feitas pelo testador sem o apoio de software que automatiza este processo com o objetivo de encontrar defeitos. Em geral, o testador executa a aplicação como o usuário final para garantir o funcionamento correto do software (KUMAR, 2013).

Definir um design efetivo de casos de testes é importante, pois criar um plano de teste que consiga testar todo o programa é, geralmente, impossível e se deve priorizar casos de testes com mais vantagens de encontrar defeitos. Vários métodos existem para criação de casos de testes. Dentro das estratégias de caixa preta, no qual não se tem acesso ao código durante o teste, destaca-se a equivalência de partição, análises de valores limites, gráfico de causa e efeito, erros por adivinhação. Nas estratégias de caixa branca, com acesso ao código, destaca-se cobertura de declarações, cobertura de decisão, cobertura de condição, cobertura de condição/decisão e cobertura de múltiplas decisões (MYERS, BADGETT e SANDLER, 2012). Observa-se, assim, que planejamento de casos de testes traz eficiência ao processo, no qual se faz a execução dos testes necessários.

Já os testes exploratórios permitem ao testador utilizar as habilidades de ouvir, ler e pensar e reportar rigorosamente e efetivamente sem ter que utilizar instruções de um pré-roteiro. Define-se teste exploratório qualquer teste que o testador controla ativamente o *design* dos testes enquanto estes são executados e utiliza estas novas informações para projetar novos e melhores testes. Níveis de controle para execução de testes exploratórios resultam em uma série de notas que podem atualizar o material de teste, assim como a massa de dados para teste. Esses testes são realizados em um ciclo contínuo, concebendo perguntas para explorar o software até que se encontre resultados. A fim de gerenciar os testes exploratórios muitas equipes utilizam guias de cobertura para organizar o esforço de teste (BACH, 2015).

Muito se discute sobre a eficiência dos testes quando se tem um *design* previsto e documentado, ou quando utilizando técnicas exploratórias. Pesquisas cientificas destacam que ambos são efetivos na descoberta de defeitos, mas que os testes exploratórios são mais eficientes, pois requerem menos tempo de *design* e por isso são mais indicados em ambientes com restrição de tempo (ITKONEN e MÄNTYLÄ, 2014). Destaca-se, ainda, que todas as técnicas de *design* são relevantes no contexto de teste exploratório, pois estas também podem ser utilizadas em execução de tempo real (HENDRISCKSON, 2013)

No contexto ágil, em geral, os testes manuais são vistos, primariamente, como testes exploratórios. Neste tipo de teste executam-se testes através de um ciclo rápido de passos para o planejamento de testes, *design* e execução. Além de encontrar defeitos, estes testes podem identificar casos de testes que podem ser adicionados aos níveis de testes automatizados e descobrir ideias que poderiam estar faltando nas histórias de usuários (VEENENDAAL, 2010).

Discute-se, também, a falta de documentação na execução de testes no ambiente ágil. Esta, no entanto, pode ser suprida pela própria dinâmica do

desenvolvimento ágil que busca entregar valor frequentemente em um ritmo sustentável. O próprio teste automatizado pode ser um exemplo de documentação que é sempre atualizado para estar compatível com as mudanças feitas ao decorrer do desenvolvimento. Em ambientes que requerem auditoria, técnicas de guardar evidências dos testes, tais como fotos das telas, podem ser utilizadas. Utilizar da organização do método ágil para que os *stakeholders* participem da solução e ajam como auditores é uma alternativa para a falta de documentação e eficiência da auditoria (CRISPIN e GREGORY, 2015).

Os testes manuais são importantes, principalmente, em níveis mais elevados, tais como testes de aceitação. Nos times ágeis observa-se que as práticas de testes exploratórios devem seguir os princípios ágeis e para isso priorizar os *backlogs* de testes. Devem ter a presença de testadores com conhecimento e domínio do sistema a ser testado e se adaptar ao princípio da simplicidade, utilizando planilhas de trabalho e lista de objetivos de teste para priorizar os esforços de testes, deixando públicos todos os *backlogs* de teste para que exista transparência com o time (GERAS, 2010).

Verifica-se, na literatura, que tanto os testes automatizados, quanto os testes manuais são importantes no contexto ágil. O teste automatizado permite que o software seja desenvolvido de forma iterativa e incremental com certo nível de confiança, enquanto o manual permite encontrar defeitos mais complexos que não são descobertos com a execução de testes automáticos.

3. Método

Este estudo é baseado na experiência de implantação de métodos ágeis em uma grande empresa de desenvolvimento de software. A empresa já vem utilizando métodos ágeis para desenvolvimento por cerca de 5 anos e muito pode ser observado sobre as práticas de testes utilizadas neste ambiente.

A equipe de desenvolvimento é composta por desenvolvedores, testadores e escritores técnicos. Existe, também uma equipe dedicada apenas a execução e suporte à automação de testes. Todos os membros da equipe são qualificados em engenharia de software e muitos dos membros possuem certificações tanto na área de desenvolvimento, quanto na área de teste de software. A grande maioria do time está na empresa desde a implantação dos métodos ágeis e são engenheiros seniores. A equipe, também, conta com participantes juniores que tem se adaptado aos métodos ágeis e aos processos da empresa.

O software desenvolvido pela empresa é um produto de mercado para gestão de ativos de TI com suporte ao ITIL. Novas versões do produto e *fixpacks* são lançados durante o decorrer do ano de acordo com as necessidades de mercado. Existe um *feedback* do software ao final de todos os sprints, mas o software é entregue, de uma vez com todas as funcionalidades novas selecionadas para aquela versão ou correções do *fixpack*.

Este artigo baseia-se na experiência vivida na implantação de métodos ágeis no desenvolvimento e manutenção do software descrito. Observa-se que nem sempre pesquisas analíticas são suficientes para investigar problemas complexos. Nesses casos a utilização de estudos de casos pode ajudar a investigar problemas contemporâneos dentro do seu contexto (RUNESON e HÖST, 2009). Sendo assim, a análise é feita a partir do estudo de caso sobre uso dos testes automáticos

ISSN: 2175-1897

e manuais executados com intuito de alcançar os objetivos ágeis e mostrar os principais desafios e benefícios que se pode obter na execução de testes ao adotar métodos ágeis, assim como a verificação do que é mencionado na literatura e que é realizado na prática da empresa analisada. O estudo pode servir para novas pesquisas em outras empresas, a fim de verificar o impacto das práticas utilizadas na qualidade do software desenvolvido.

4. Resultados e Discussão

A fim de discutir o processo de teste em ambientes ágeis, inicialmente, será feita um descrição do processo dentro do ambiente analisado de forma verificar as estratégias utilizadas tanto no ambito de testes automatizados, quanto no de testes manuais. A automação de testes se deu nos nível de testes unitários e no de UI.Os testes manuais foram feitos utilizando *design* pré-definido e de forma exploratória.

No nível de teste unitario, a automação foi feita pelos desenvolvedores. Estes foram criados na mesma linguagem de desenvolvimento do software. Já no nível de UI, a automação de testes foi realizada pelos engenheitos de testes.Um *framework* foi criado para que todos seguissem o mesmo padrão e desenvolvessem testes utilizando técnicas de programação que ajudaram da reutilização de código, tais como definidas na programação orientada a objeto.

A fim de validar o software a cada novo *build*, foram criados testes de BVT (*Build Verification Test*). Estes testes executavam a instalação de *build* automaticamente e também um conjunto de casos de testes automatizados que asseguravam que as funcionalidades básicas estavam funcionando corretamente. Os testes de BVT foram escalados para serem executados ao final do lançamento de cada novo *build*. Com está prática o time sempre garantiu que as funcionalidades básicas para execução do sistema não estavam comprometidas.

Uma equipe de automação de testes foi criada para dar suporte à todos da equipe e executar os testes BVT e de regressão. Todos os testadores deveriam criar testes automatizados das funcionalidades por eles testadas em um *sprint* e a equipe de automação deveria dar apenas o suporte necessário para a criação. Em alguns momentos, no entanto, por falta de tempo dos testadores ou conhecimento técnico, alguns testes foram automatizados pela própria equipe de automação.

Os testes de regressão foram executados de forma automatizada, no entanto a execução de forma integral, mesmo utilizando várias máquinas virtuais e rodando vários testes simultaneamente, não foi viável dentro do *sprint.* Por isso optou-se pela execução em paralelo aos *sprints* com resultados de execução ao final de um conjunto de *sprints*. Esta prática não comprometeu a qualidade final do produto, pois o teste de BVT garantiu que o software estive disponível para testes dentro dos *sprints* e o teste de regressão automatizado validou o sistema em três bancos de dados de forma continua e ampla.

Os testes manuais foram utilizados para verificar as novas funcionalidades desenvolvidas durante o *sprint* e para posterior automação. No início da implantação do desenvolvimento ágil, a documentação e a utilização de ferramentas de controle de execução de casos de testes foram mantidas o que ajudou muito na mudança de paradigma. Posteriormente, a documentação e o controle de execução dos testes passou a ser feita nas histórias.

ISSN: 2175-1897

Foram utilizadas ferramentas de apoio à gestão para a administração dos testes manuais. Com elas foi possível acompanhar a evolução das tarefas de cada história, assim como as atividades de testes que estavam representadas no *sprint backlog*, que podia ser acessado por todos da equipe. Grande parte dos novos defeitos foram encontradas na execução de testes manuais, destacando assim, a sua importância no contexto tanto para validação quanto a verificação do software. A execução manual dos testes permitiu que os testadores obtivessem melhor entendimento do software e pudessem criar testes automatizados a partir delas.

Ao final de todos os *sprints* foram feitas reuniões de revisão da *sprint* com uma demonstração do software. A demostração ajudou na melhoria da qualidade do software. Estas reuniões trouxeram ao time novas formas de avaliar o software com a ajuda de especialistas do mercado e que permitiram ao time a compreender melhor o software e as necessidades dos clientes.

4.1 Resultados

Neste estudo verificou-se que a estratégia de automação empregadas pela equipe estudada cobriu apenas os níveis de unitários e de UI. As automações dos testes unitários foram realizadas pelos desenvolvedores. Fator que ajudou o time a ser mais rápido na criação destes testes. A desvantagem desta abordagem foi a não participação do testador em nenhuma etapa do desenvolvimento de código, o que fez com que ele não estivesse ciente do que era verificado no teste unitário e desta forma não pode otimizar seus próprios testes.

Na automaçao da interface de usuário, observou-se a importância do *framework* utilizado, pois este facilitou a criação dos testes e permitiu que houvesse reutilização de código, tornando o desenvolvimento de testes mais rápido. Isto, também, ajudou na manutenção dos testes, pois ao alterar apenas o método com necessidade de manutenção, eram habilitados todos os *scripts* para a mudança. Nesse sentido foi possível observar que boas práticas de programação são bastante úteis para o desenvolvimento de testes automatizados. A utilização de *scripts* gerados por ferramentas pode ser mais rápida em um primeiro momento, mas a utilização de meios estruturados facilita a manutenção e o reaproveitamento de código.

Observou-se que a forma com que o software é estruturado internamente impacta diretamente nas execuções destes testes. Muitos problemas encontrados nos *logs* de execução eram por causa de pequenas alterações feitas no software, que não alteravam a forma de apresentar o software, mas faziam com que o teste falhasse. Por isso a interação entre os desenvolvedores e os testadores deveria ser constante no decorrer do desenvolvimento do código para a prevenção de falhas no teste ou para criação de padrões no desenvolvimento do código para que estes pudessem estar sempre compatíveis com os *scripts* de automação.

A execução de testes automatizados na camada de UI foi mais vantajosa do que se estivessem sido executados manualmente. Está técnica permitiu que vários testes pudessem ser executados em várias máquinas virtuais, otimizando assim, o tempo de execução. A forma com que os testes foram desenvolvidos se preocupou em estrutura-los de forma que criassem *logs* de fácil compreensão. A análise

ISSN: 2175-1897

destes *logs*, mesmo sendo bem estruturados, foi uma tarefa que precisou ser executada de maneira manual e com bastante cuidado para verificar se as falhas eram problemas nos testes automatizados ou se eram problemas reais na aplicação.

Por causa do tempo de execução dos testes automatizados no nível de UI, o teste de regressão não pode ser executado em todos os *sprints* em sua totalidade. Este, no entanto, foi executado várias vezes até o final do *release* utilizando configurações diferentes e cobrindo diversas plataformas. Fator que trouxe bastante confiabilidade quanto as mudanças feitas no software ao decorrer dos *sprints.* A vantangem de se utilizar a execução de testes automatizados foi a alocação de recursos humanos para criação de execução de testes novos e a utilização de máquinas virtuais para executar os testes automatizados, que permitiram executar um conjunto de testes de forma mais rápida e com apenas um recurso humano para configuração das ferramentas e análise do logs.

Utilizar testes de BVT durante todos *sprints* trouxe confiança no software e fez com que não houvesse desperdício de esforços quando este falhava, pois as comunicações sobre os resultados sempre foram constantes e transparentes para todo o time, garantido que o time pudesse testar os *builds* sem erros de instalação e com as funcionalidades básicas sem problemas. Esta prática ajudou o time a ser mais eficiente, pois o testador apenas fazia a sua instalação e testava as funcionalidades desenvolvidas anteriormente quando tinha um *build* sem problema de grande impacto.

Os testes manuais foram importantes para encontrar defeitos novos e definir os testes automatizados. A falta de documentação foi suprida por definição de tarefas nas histórias e todo o time pode acompanhar o andamento dos testes. Observou-se que a falta de documentação de testes manuais, acarretou em percas na transmissão de conhecimento. Com os testes documentados em passo a passo a execução por membros novos na equipe era mais fácil, do que com a definição encontrada nas histórias. Já os testes automatizados são difíceis de serem interpretados para aquisição de conhecimento. A participação de um técnico responsável pela documentação de guias e manuais foi importante e supriu de forma relativa a falta de documentação de testes. No guia, no entanto, apenas o funcionamento normal do sistema é definido, os casos alternativos que servem para encontrar defeitos não são documentados.

Verificou-se, também, que no ambiente ágil as habilidades dos testadores tiveram que ser aprimoradas. O testador passou a ter que conhecimentos de código para automatizar os testes e analisar *logs*. Ao trabalhar em um ambiente com menos processos planejados, a responsabilidade diante da execução de testes passou a ser do testador e este precisou se atualizar conhecer bem as técnicas de testes para execução de manual e criar formas de organizar e otimizar os processos de validação e verificação. Este, também, precisou se comunicar com mais frequência com os desenvolvedores para a criação de testes.

5. Considerações finais

O presente estudo observou que no contexto ágil tanto os testes automatizados, quanto os testes manuais foram importantes para a estratégia de teste da empresa. Os testes automatizados garantiram maior confiabilidade do software durante o processo e desenvolvimento, além de ajudar a cobrir diversas plataformas de forma mais rápida do que em testes manuais. A implantação de testes ágeis sem estratégias de automação não traria à empresa confiança em relação ao desenvolvimento de software. Por este motivo destaca-se a importância de definir qual objetivo se quer alcançar com a automação de testes. No caso analisado, foi importante para dar confiança ao software ao final de um *release*. No entanto, para se alcançar objetivos dentro de um *sprint*, estratégias mais enxutas devem ser utilizadas.

Observa-se que a estratégia de utilizar testes automatizados nas camadas mais baixas e nas camadas altas fez com que não fosse possível executar testes de regressão em todos os *sprints*. O modelo adotado poderia trazer riscos, caso tivesse que entregar no mercado o produto ao final de cada *sprint*. Verifica-se que a estratégia utilizada para automação distancia o processo de desenvolvimento sobre o que é definido em ágil.

Já os testes manuais foram importantes para descoberta de novos defeitos, assim como para o entendimento do software e para definição de testes automatizados. Encontrar meios ou ferramentas para organizar os testes é bastante importante e isso ajudou a equipe acompanhar o desenvolvimento do software como um todo. Capacitar os testadores aos novos paradigmas dos métodos ágeis também foi relevante. Estes devem estar preparados para automatizar testes, se adequar aos princípios de simplicidade e se comunicar de forma efetiva para conseguir testar o software com eficiência.

Uma vez que os times ágeis tendem a utilizar menos documentação, deve se encontrar meios de sistematizar o teste de software e organizá-lo de forma a encontrar defeitos com mais eficiência. Alguns problemas tendem a ser descobertos apenas com a utilização do software e a visão crítica do testador é capaz de antecipar vários destes problemas. A utilização de métodos para criação de testes, tanto na utilização de abordagens com design pré-definido, tanto em abordagens exploratórias se fazem necessárias para melhor aproveitamento do tempo de teste.

Observa-se, também, que o *feedback* obtido nas reuniões de apresentação do software ao final das *sprints* foram importantes para o time e ajudaram na definição de novos testes com maior preocupação das necessidades dos clientes. Estas reuniões servem, também, para avaliar o software e conscientizar todos do time sobre a qualidade do software que é apresentada, servindo, não apenas, como reunião de *feedback,* mas também de auditoria por todos membros da equipe em relação a qualidade do software.

Referências

ALSMADI, Izzat. *Advanced Automated Software Testing: Frameworks for Refined Practice*, IGI Global, 2012.

BACH, James. Exploratory Testing Explained. 2003. Disponível em <https://www.satisfice.com/articles/et-article.pdf> Acesso em: 01/05/2015.

BECK, Kent.; *et. Al*. Manifesto for Agile Software Development, 2001. Disponível em: <http://Agilemanifesto.org>. Acesso em: 14/04/2015.

BERNARDO, Paulo, Cheque; KON, Fabio. A Importância dos Testes Automatizados. *Engenharia de Software Magazine*, 2008.

BERNER, Stefan; WEBER, Roland; Keller, Rudolf; Observations and Lessons Learned from Automated Testing, *Proceedings of International Conference on Software Engineering*, 2005.

COHN, Mike. *Succeeding with agile. Software development using scrum*. Boston: Addison Wesly, 2013.

CRISPIN, Lisa; GREGORY, Janet. *Agile Testing: A Practical Guide for Testers and Agile Teams*. 1. ed. Boston: Addison Wesley, 2009.

CRISPIN, Lisa; GREGORY, Janet. *More Agile Testing: Learning Journeys for the whole team*. 1. ed. Boston: Addison Wesley, 2015.

GERAS, Adam. Leading Manual Test Efforts with Agile Methods. *Agile 2008 Conference*, 2008.

GRAHAM, Dorothy; FEWSTER, Mark. *Experiences of test automation: case of studies of software test automation*. Pearson Addison-Wesley, 2011.

HENDRISCKSON, Elisabeth. *Explore it: Reduce Risk and Increase Confidence with Exploratory Testing*. Raleigh: The pragmatic Bookshelf, 2013.

ISO/IEC/IEEE 29119-1. *Concepts and definitions* http://www.iso.org/iso/catalogue_detail.htm?csnumber=45142. 2013

ITKONEN, Juha; MÄNTYLÄ, Mika. V. Are Test Cases Needed? Replicated Comparison between Exploratory and Test-Case-Based Software Testing. *Empirical Software Engineering*. 2014, Volume 19, Issue 2, pp 303-342

KUMAR, Vivek. Comparison of Manual and Automation Testing. *International Journal of Research in Science And Technology*, 2012

MYERS, Glenford, J.; BADGETT, Tom; SANDLER, Corey. *The Art of Software Testing*, 3. ed. New Jersey: Wiley Publishing, 2012.

OLUWOLE, Dele. Agile Methodology Is Not All About Exploratory Testing. *Scrum Alliance*, 2013. Disponível em: <https://www.scrumalliance.org/community/articles/2013/march/agile-methodology-is-not-all-about-exploratory-tes>. Acesso em: 01/07/2015.

RUNESON, Per.;HÖST, Martin. Guidelines for conducting and reporting case study research in software engineering. *Empirical Software Engineering*. 2009 Volume 14, Issue 2, pp 131-164.

SOMMERVILLE, Ian. *Engenharia de Software*. 8. ed. São Paulo: Pearson Addison-Wesley, 2007.

VEENENDAAL, Erik, Van. Scrum & Testing: Assessing The Risks. *Agile Record: The Magazine For Agile Developers and Agile*. 2010.