# Teste de Software nas Metodologias Ágeis com ênfase no Scrum

**Bruno Regonha[1], Nivaldo Tadeu Marcusso[2]**

[1]Pós Graduação em Gestão de Projetos em Tecnologia da Informação – Faculdade de Tecnologia de Americana (FATEC-AM)
Americana – SP – Brasil
brunoregonha@gmail.com

[2]Professor do curso de Pós Graduação em Gestão de Projetos em Tecnologia da Informação – Faculdade de Tecnologia de Americana (FATEC-AM)
tadmar@uol.com.br

***Resumo.*** *Este artigo descreve o teste de software utilizando um processo de desenvolvimento baseado no método ágil Scrum. É apresentada a aplicabilidade do Scrum no teste de software, o papel do analista, as atividades que podem ser automatizadas dentro dessa realidade, bem como as ferramentas que dão suporte a estas atividades. Quando se adota um processo de teste ocorre melhora, estabilidade e o amadurecimento do software. A viabilidade e as vantagens de se adequar e automatizar atividades de teste utilizando essas ferramentas vai desde o ganho em tempo de execução dos testes e detecção de defeitos, até a confiabilidade do cliente pelo produto final, sem o aumento de custos.*

***Abstract.*** *This paper describes the software testing using a development process based on the agile method Scrum. Is presented the Scrum applicability in software testing, the analyst function, the activities that can be automated within an agile testing process and tools that support these activities. When adopting a testing process, there is improvement, stability and maturity of the software. The viability and benefits of adapting and automate testing activities with this tools ranging from the gain in time execution of the tests and defects detection, to the reliability of the customer for final product, without an increase in costs.*

## 1. Introdução

Com base no fracasso no desenvolvimento de grandes projetos de software no início da década de 70, surgiu a necessidade da criação de uma nova metodologia, como uma alternativa às metodologias tradicionais.

Uma nova abordagem para desenvolvimento de software tem despertado grande interesse entre as organizações de todo o mundo. Visto que apontamos para uma tendência de desenvolvimento ágil de aplicações devido ao ritmo acelerado de mudanças na tecnologia da informação, pressões por constantes inovações, concorrência

acirrada e grande dinamismo no ambiente de negócios (Kniberg, 2007), as "Metodologias Ágeis" vem se tornando mais popular no Brasil por usar uma abordagem simplificada. No entanto, "ser simples" geralmente é confundido com falta de controle e completa anarquia. Na verdade, ser simples, ter agilidade, é fazer a diferença e, ao contrário do que parece, exige muita disciplina e organização.

O teste de software ágil precisa estar agregado ao processo de desenvolvimento desde o início, acompanhando todo o ciclo de vida de um software. A série de processos executados pelo teste de software valida se o software realmente faz aquilo que foi projetado para fazer. O teste de software se constitui de atividades de extrema importância em projetos de desenvolvimento de sistemas que contribuem para a criação de produtos de melhor qualidade gerando assim uma maior satisfação do cliente que serão vistos adiante neste artigo com mais detalhes.

A atividade de teste deve ser bem administrada para que a fase de execução de testes atenda ao que se é esperado e possa cumprir os prazos definidos. Existem diversas ferramentas de apoio à atividade de testes, as quais ajudam a automatizar as atividades do processo de teste e trazem ganhos para o time.

A automação dos testes que será apresentada poderá ser utilizada tanto no modelo tradicional quanto no modelo ágil de desenvolvimento, porém neste artigo serão mostradas as etapas de teste dentro de um processo ágil e como a automação pode contribuir neste processo.

Pelo fato do *Scrum* (e outros métodos ágeis num geral) ser relativamente recente, o processo de teste inserido nesta metodologia mostra-se em fase de amadurecimento. O enfoque em automação permite que algumas atividades de teste sejam executadas com mais eficiência e possam acompanhar o ritmo mais dinâmico da agilidade.

O objetivo deste artigo é mostrar as atividades de teste dentro dos processos ágeis com ênfase no *Scrum*, juntamente com os papéis do analista de teste, sua automatização, e fazer um levantamento de ferramentas que podem ser utilizadas.

Este artigo está estruturado da seguinte forma: Na seção 2, é apresentada a necessidade de teste de software. Na seção 3, é apresentado o ciclo de vida de teste. Na seção 4 são apresentados os papéis no analista de teste. Na seção 5 é apresentação a automação de testes juntamente com suas técnicas e na seção 6 são as considerações finais deste artigo.

## 2. Certificação Scrum

A certificação dos profissionais em *Scrum* é um limitante na adoção da tecnologia. O curso mostra os conceitos básicos, práticas e princípios que todos precisam saber pra cumprir seus papéis no time. No atual mercado de trabalho competitivo, certificação é uma importante forma de identificar os profissionais da área com um completo entendimento do *Scrum* e como aplicar seus princípios no local de trabalho.

## 3. A necessidade de teste de software

Uma vez que objetivo primário de um projeto de software é o próprio software e não um grande conjunto de documentação sobre ele, onde o *Scrum* apresenta forte

presença de liderança e gerenciamento dos requisitos, um ponto essencial para garantia de que exista fidelidade das implementações em relação aos requisitos está nos testes executados sobre o código produzido.

## 4. Ciclo de vida de testes

Para o desenvolvimento de software ser realmente ágil ele deve ter conceitos ágeis aplicados na gestão do projeto, na engenharia do software que contempla o levantamento de requisitos e arquitetura e, também, no desenvolvimento que engloba as atividades de codificação, testes e refatoração. Neste artigo serão considerados conceitos do framework *Scrum* para gestão do projeto e como são aplicado os testes (Santos, 2010).

O *Scrum* possui um ciclo incremental e iterativo para desenvolvimento de software orientados a objeto (Schwaber, 2004). Nesta abordagem os requisitos do sistema são chamados de itens do *backlog*. O *Scrum* utiliza o conceito de entrega contínua de valor, ou seja, versões do software que possuam um conjunto de itens de *backlog* implementados e utilizáveis. Outro conceito importante é o de *Sprint* que é uma iteração que deve ser realizada num período de tempo curto (2 a 4 semanas em média), no qual a equipe do projeto deverá produzir uma entrega de valor para o cliente. São realizadas reuniões diárias para acompanhar e verificar o andamento do projeto e reorganizar o planejamento caso algo não esteja saindo conforme planejamento.

## 5. Os papéis do Analista de Teste no Scrum

O analista de teste dentro do time no *Scrum* assume vários papéis como líder, arquiteto e automatizador. Em cada fase do projeto ele “dança conforme a música”, ou seja, executa o que for necessário.

Um dos papéis que o analista de teste pode exercer é o de *Product Owner* (PO). O PO é responsável criar e manter a visão do produto, garantir o ROI (Retorno do Investimento) do projeto, pela priorização dos itens no *Product Backlog*, aceitar ou rejeitar entregas entre outras coisas.

Para a definição dessa visão, o PO, colhe informações com o time, *stakeholders*, executivos, clientes, gerentes e usuários finais.

O analista de teste participa da fase de definição da visão do produto contribuindo com as seguintes tarefas:

• Colaboração nas reuniões com o cliente para levantamento dos requisitos;

• Participação das reuniões de *brainstorm*;

• Foco na visão da qualidade do produto;

• Avalia a abordagem de teste;

• Identifica as habilidades de teste necessárias para o projeto.

O PO cria uma lista inicial de necessidades (*Product Backlog*) para que a visão do produto seja bem sucedida. O analista de teste participa desta etapa realizando as seguintes tarefas:

• Participa de discussões de arquitetura;

• Participa das definições de tecnologias utilizadas;

• Identifica as necessidades do ambiente de teste;

• Identifica restrições tecnológicas;

• Identifica ferramentas de teste necessárias para auxiliar na execução do projeto;

• Utiliza mapa mental para auxiliar no entendimento das funcionalidades/requisitos;

• Identifica os “melhores” analistas de teste para o projeto em questão;

• Identifica a necessidade de suporte do time de suporte/operações.

Estimar é uma atividade do time no *Scrum*. Como o analista de teste faz parte do time, ele também participa desse processo. Além dele, o *Scrum Master* deve mediar o processo; o PO deve estar disponível para esclarecer eventuais dúvidas.

Na primeira parte do *Planning Meeting* o PO define a meta da *Sprint* e comunica para o time os itens prioritários do *Product Backlog*. O time deve estimar os itens em tamanho e selecionar o que vai ser feito. O analista de teste participa dessa fase realizando as seguintes tarefas:

• Ajuda a garantir que os itens estão de acordo com a meta do projeto e teste;

• Analisa indicadores de desempenho;

• Revisa a estimativa, caso necessário;

• Gerencia os riscos encontrados;

• Tira dúvidas com o PO;

• Define quais serão os tipos de teste (funcional, aceitação, regressão, etc) necessários.

Na segunda parte da *Planning Meeting* o time colhe mais detalhes dos itens do *Product Backlog* e os divide em tarefas gerando o *Sprint Backlog*. Após a análise, cada membro do time seleciona as tarefas que deseja executar durante a *Sprint* e estimá-las. O analista de teste participa desta fase com a seguinte contribuição:

• Define o nível de regressão de teste de acordo com a priorização dos itens/tarefas do *Sprint Backlog*;

• Atualiza a matriz de teste por funcionalidade;

• Atualiza o mapa mental.

Após a definição da *Sprint*, o quadro de acompanhamento (Kanban) é preparado. Ele pode ser alterado de acordo com a sua realidade e necessidade.

Durante a execução da *Sprint*, o *Scrum Master*, desenvolvendo seu papel de liderança e colaboração, facilita o trabalho do time removendo os impedimentos

encontrados e protege de possíveis interferências externas, garantindo assim a boa aplicação do *Scrum*.

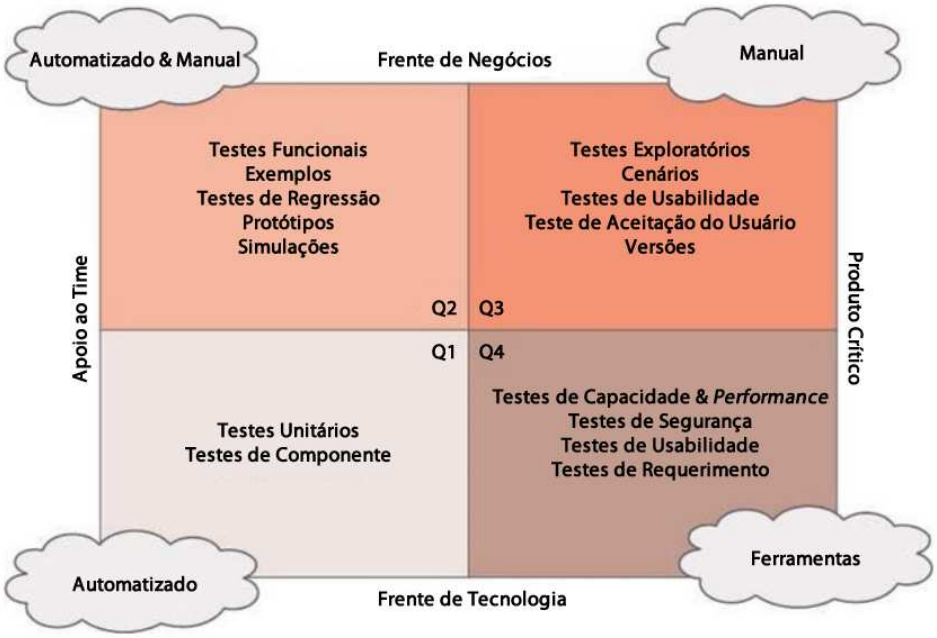


Figura 1 - Tipos de Teste – Adaptado de (Duong, 2009)

• Q1 – São os testes que focam na arquitetura. A responsabilidade é dos desenvolvedores e os analistas de teste auxiliam na elaboração dos testes unitários automáticos;

• Q2 – São os testes que focam nas regras de negócio. A responsabilidade é dos analistas de teste em conjunto com outros envolvidos no projeto (clientes, usuários, etc). Ajuda no entendimento das funcionalidades;

• Q3 – São os testes que focam em encontrar defeitos e nas regras de negócios. A responsabilidade é dos analistas de teste;

• Q4 – São os testes que focam na arquitetura, estrutura do software. A responsabilidade é dos analistas de teste.

Durante a execução da Sprint o analista de teste possui o seguinte papel:

• Monta e configura o ambiente e infraestrutura de teste;

• Executa testes;

• Automatiza casos e tarefas de teste;

• Auxilia os desenvolvedores na elaboração dos testes unitários automáticos;

• Evidencia os resultados;

• Acompanha os defeitos encontrados;

• Desenvolve novas habilidades;

• Aprovação. Nada é considerado finalizado em um *Sprint* até que ele diga que está;

• É responsável em ficar focado na meta da *Sprint*.

Na reunião diária de 15 minutos, onde o *Scrum Master* é o facilitador, o time ganha visibilidade de como está o caminho para a meta e planeja o dia de trabalho. O analista de teste também participa dessas reuniões contribuindo com as seguintes ações:

• Relatório de teste atualizado;

• Evidência de teste atualizada;

• Lista de defeitos atualizada;

• Lista de impedimentos atualizada;

• Quadro de acompanhamento atualizado.

O analista de teste participa de outras duas reuniões: reunião de revisão, que possui o propósito de apresentar o que foi feito para o PO e convidados e reunião de retrospectiva que possui o objetivo de discutir as lições aprendidas.

Na reunião de revisão, é realizada uma apresentação por todo o time e o PO avalia se a meta foi alcançada e faz anotações que podem ser transformar em novos itens para o *Product Backlog*.

Na reunião de retrospectiva o time avalia o que foi bom na última *Sprint*, o que deve ser melhorado e quem está no controle.

O analista de teste deve participar de todas as fases do *Scrum*, pois ele faz parte de um time. Não existe time de teste ou desenvolvimento, mas sim, time do Scrum.

## 6. Automação de testes

Há diversas medidas que podem ser tomadas para se obter testes melhores, mais ágeis, efetivos e baratos. Dentre estas podemos citar: sistematizar as rotinas de testes; definir os papéis e responsabilidades atreladas ao teste; antecipar a preparação dos testes já nas fases de construção; conhecer as técnicas relacionadas ao assunto; testar diferentes características do software nas diferentes fases de testes; estimar adequadamente os esforços para os testes; ter um controle efetivo das falhas encontradas; e automatizar os processos de testes, tanto planejamento quanto execução (Costa, 2006).

Automatizar testes significa utilizar softwares que controlem a execução dos casos de teste. O uso desta prática pode reduzir o esforço desprendido para os testes em um projeto de software, ou seja, executar maiores quantidades de testes em menor tempo. Testes manuais que levariam horas para serem totalmente executados poderiam levar minutos caso fossem automatizados (Tuschling, 2008).

Apesar dos testes manuais também encontrarem erros em uma aplicação, é um trabalho desgastante que consome muito tempo. Automatizar seus testes significa executar mais testes, com mais critérios, em menos tempo e sem muito esforço na execução (Costa, 2006). Deve-se também levar em consideração as inúmeras

possibilidades que um script automático traz, como: cobertura de requisito e profundidade nos testes (extensibilidade), além de ser efetivo quanto à quantidade de dados de entrada que podem ser processadas (confiabilidade). Outras vantagens, que podem ser destacadas ao transformar rotinas de testes em scripts automáticos, são: segurança, reusabilidade e manutenibilidade (Bernardo e Kon, 2008), (Costa, 2006).

Testes melhores e mais elaborados refletem diretamente no aumento na qualidade do produto final. Entretanto, elaborar um conjunto de scripts de teste automáticos requer uma padronização de atividades e conhecimento em codificação e análise por parte do testador. Para fazer testes automáticos eficientes, caso o teste seja de unidade, é necessário entender a arquitetura do software e a base do código. Caso o teste seja de sistema, é necessário conhecer as regras de negócio, as funcionalidades, e os valores e situações esperadas como resultado.

Em todos os casos de automação o esforço inicial requerido é maior. Isso ocorre porque é necessário fazer um planejamento para levantar O Quê, Como e Por Que será automatizado. Depois de completado o planejamento, deve-se então dar início a confecção dos scripts automáticos.

### 6.1. Situações candidatas à automação

Quando se deseja aplicar a automação na execução dos testes do sistema que geralmente é feita manualmente por um testador, vários fatores devem ser avaliados para se identificar se é vantajoso ou não a automatização. Testes automatizados possuem algumas características que diferem de testes manuais conforme podemos ver no quadro abaixo:

Tabela 1 - Testes Manuais X Testes Automatizados (Diz, 2009)

| Características | Manual | Automatizado |
|---|---|---|
| **Esforço** | Requer um grande esforço de criação e manutenção | Requer um grande esforço de criação e manutenção |
| **Reuso** | Baixa reutilização | Alta reutilização |
| **Interpretação das regras de negócio** | Dependente da linguagem natural que é ambígua | Exige que cada ação seja programada |
| **Execução** | Demorados para executar | Rápidos |
| **Produtividade** | Susceptível ao humor do testador | Não susceptível ao humor do testador |
| **Criatividade/Mudanças** | Permite a exploração de situações diferentes | Não são capazes de explorar situações diferentes, se limitando ao propósito que foram criados |
| **Requisitos do testador** | Exige profissionais com experiência em testes | Exige profissionais com experiência em testes e desenvolvimento |

De uma maneira geral, há testes que são mais indicados a serem automatizados do que outros como, por exemplo:

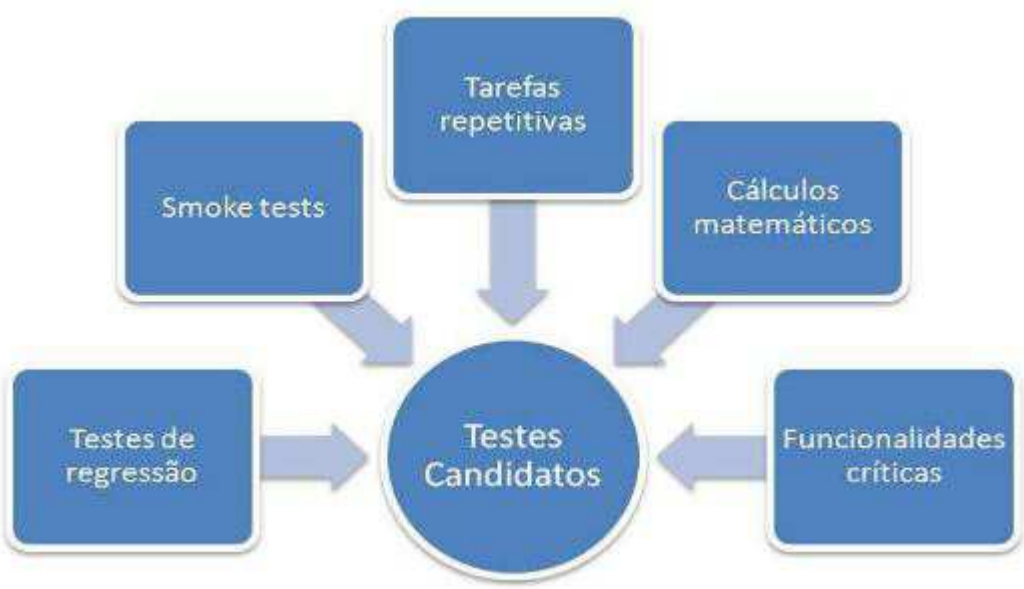


Figura 2 - Testes candidatos à automação (Diz, 2009)

• Testes de regressão - teste de regressão é a realização de testes das ocorrências que foram abertas no teste anterior e testes nos módulos que foram afetados pelas mudanças realizadas.

• Tarefas repetitivas - a automação é ideal para este caso, pois permite que o testador foque sua atenção em situações inesperadas.

• *Smoke tests* - são testes básicos aplicados as principais funcionalidades do sistema. Visa identificar defeitos nessas funcionalidades com o objetivo de avaliar se a versão liberada pode continuar sendo testada pela equipe de testes.

• Cálculos matemáticos - evitar erros humanos que podem ocorrer.

• Funcionalidades críticas - como seres humanos sofrem influência do humor para execução de uma tarefa, funcionalidades críticas não devem depender de um testador que pode esquecer-se de algo importante.

Por outro lado a automação não é recomendada para os seguintes casos:


Figura 3 - Testes que não se recomenda a automação (Diz, 2009)

• Funcionalidades novas - tem uma grande probabilidade de sofrer mudanças. O ideal seria testar quando se alcança certa estabilidade.

• Funcionalidades pouco usadas - haverá esforço e custo desnecessário para automatizá-los, pois estes testes serão pouco aplicados uma vez que a funcionalidade não terá muita utilização.

• Funcionalidades que exigem inspeção visual - não existe uma ferramenta para avaliar a interface de um sistema e a sua usabilidade. Para isso, é necessária a visão de um testador.

• Protótipos - o protótipo é algo que será utilizado momentaneamente somente no início de um projeto de software, voltada sua exibição para clientes ou para a própria organização. Devido a isso, não há a necessidade de automatizar testes em protótipos já que este ainda não é o software propriamente dito.

Cabe ressaltar que a aquisição de ferramentas, compartilhadas pelas diferentes equipes no processo de desenvolvimento de software, deve ser alinhada com todos os envolvidos, tendo em vista selecionar aquela que melhor se encaixa aos processos que serão adotados e evitar gastos desnecessários.

### 6.2. Técnicas de automação

Há várias abordagens para se automatizar os testes, dentre elas se destaca a utilização de ferramentas baseadas em interface gráfica que gravam e executam os casos de teste. São as ferramentas conhecidas como *Rec-and-Play* (*Record and Playback*). A ferramenta simula um usuário real interagindo diretamente com a aplicação. Na medida em que o sistema está sendo executado manualmente, a ferramenta captura as ações e as transforma em *scripts*, que posteriormente podem ser executados (Fantinato, 2009).

Para testes utilizando esta abordagem, não há necessidade de modificação, no sentido de padronizar o código, para que a aplicação se torne fácil de testar (testabilidade). Os testes nessa são baseados na mesma interface gráfica que o usuário irá utilizar. A situação aqui é encontrar uma ferramenta que ofereça o recurso necessário para testar sua aplicação específica.

Existem ferramentas *Rec-and-Play* para aplicações Desktop (Java Swing, interface gráfica como o KDE) e Web. A maior desvantagem dessa técnica de automação é que o script se torna "escravo" da interface da aplicação. Para que *scripts*, utilizando esta abordagem, sejam criados, as ferramentas fazem uso dos nomes, posições e das propriedades dos componentes, e objetos que estão dispostos na interface da aplicação. Se alguma mudança de posição, nome, tipo do componente, ocorrer, o script "quebra", ou seja, não funciona mais (Fantinato, 2009).

Outra maneira muito utilizada de automatizar casos de teste é o uso de Lógica de Negócio (*Script Programming*). Neste caso são observadas as funcionalidades da aplicação, sem interagir com a interface gráfica. Com esta abordagem serão testadas das maiores até as menores porções do código: funções, métodos, classes, componentes, entre outros (Fantinato, 2009).

Nesse caso, geralmente é pedido que se faça uma alteração no código para que o trabalho de automação fique mais simples e produtivo. Muitas vezes o código é melhorado (refatoração) e padronizado para que se consiga testar mais facilmente e sem muitos problemas. São necessários profissionais com conhecimento em código e programação para criar os *scripts* automatizados. O uso deste método traz muitos benefícios, como por exemplo, testes que necessitam de centenas de repetições, cálculos complexos e até mesmo integração entre sistemas são feitos facilmente utilizando esta metodologia.

Existem bibliotecas, ferramentas e *frameworks* que suportam este método. Bons exemplos são: JUnit, para testar unidade de código Java; FitNesse, que é um *framework* para testes de aceitação; MbUnit, para testar unidade em códigos .NET.

## 7. Considerações Finais

Neste artigo foi apresentado o processo de teste adequado a realidade das metodologias ágeis com ênfase no *Scrum*.

Métodos ágeis defendem a automação, portanto a busca por ferramentas de auxílio ao teste se torna uma tarefa importante em um projeto norteado pela agilidade. Existem várias atividades no processo de testes ágeis que podem ser beneficiadas com o uso de ferramentas de automação. O uso destas ferramentas com finalidade de apoiar o teste pode trazer um ganho de produtividade e aumento da qualidade do software, bem como um maior controle sobre o processo de teste como um todo.

Um fator que contribui para o sucesso do *Scrum* é a maneira como se trabalha em equipe, dividindo todos os recursos, em pequenos grupos, isso faz com que todos sejam mais participativos e se empenhem melhor em suas atividades. Com a participação ativa de todos, o andamento do processo ocorre de forma clara, gerando também, aumento de produtividade.

Corporações usam o *Scrum* pela facilidade em agendar as interações, controlar as alterações do projeto e a possibilidade de o cliente acompanhar a evolução do desenvolvimento. É cada vez mais comum e empresas de TI preferirem especialistas com certificação nesta metodologia. O documento mostra que a pessoa adquiriu o conhecimento teórico e prático da tecnologia, uma vez que o interessado não consegue obter a certificação sem estudar sobre seu conteúdo. A capacitação técnica e experiência dos desenvolvedores são de extrema importância para o sucesso na aplicação dos processos.

Devido a flexibilidade, a organização pode ter que realizar alterações na sua estrutura organizacional, para melhor adequar ao funcionamento do *Scrum* já que, a metodologia exige certo nível de formação de todos os envolvidos.

As empresas que aderem à utilização de um processo ágil como o *Scrum* têm boas experiências com ótimos resultados. A equipe trabalha mais unida, o retrabalho reduz e a qualidade, a qual o teste está diretamente relacionado, aumenta consideravelmente.

## 8. Referências


Bernardo, P. e Kon, F. A Importância dos Testes Automatizados. 2008. Disponível em: <http://www.ime.usp.br/~kon/papers/EngSoftMagazine-IntroducaoTestes.pdf>.

Costa, M. Estratégia de Automação em Testes: Requisitos, Arquitetura e Acompanhamento de sua implantação. 2006. Disponível em: <http://libdigi.unicamp.br/document/?down=vtls000389004>.

Crispin, Lisa; Gregory, Janet. Agile Testing: A practical guide for testers. Addison-Wesley Professional. 2009.

CSP: Become a Professional. 2010. Disponível em: <http://www.scrumalliance.org/pages/certified_scrum_professional/>.

Diz, Jorge; Nogueira, Elias. Agilidade com Ferramentas de Automação. TDC In: The Developers Conference. 2010. Disponível em: <http://www.slideshare.net/elias.nogueira/agilidade-com-ferramentas-de-automaocomo-e-por-qu>.

Duong, Luu. Functional Testing Tools. Disponível em: <http://luuduong.com/blog/>.

Fantinato, M. AutoTest – Um Framework Reutilizável para a Automação de Teste Funcional de Software. 2009. Disponível em: <http://www.sbc.org.br/bibliotecadigital/download.php?paper=255>.

Gomes, Handerson. Porque usar Scrum, Nov, 2008. Disponível em: <http://webinsider.uol.com.br/index.php/2008/11/06/porque-usar-Scrum/>.

Karlsson, Erik; Martensson, Fredrik. Test processes for a Scrum team. Lund University, Sweden. 2009. Disponível em: <http://fileadmin.cs.lth.se/cs/Education/Examensarbete/Rapporter/2009/2009-18_Rapport.pdf>.

Kniberg, Henrik. Scrum e XP direto das trincheiras. Ed. C4Media. 2007.

OpenSource Functional Testing Tools. OpenSource Functional Testing Tools – News and Discussion. 2009. Disponível em: <http://www.opensourcetesting.org/functional.php>.

Schwaber, Ken. Agile Project Management with Scrum. Microsoft Press. 2004.

Santos, Rildo. Engenharia de Software 100% Ágil (SCRUM, FDD e XP). 2010. Disponível em: <http://pt.scribd.com/doc/71207440/Scrum>.

Tuschling, Oliver. Software Test Automation. 2008. Disponível em: <http://www.stickyminds.com/getfile.asp?ot=XML&id=14908&fn=XDD14908filelistfilename1%2Epdf>.