AUTOMAÇÃO EM TESTES ÁGEIS

AUTOMATION IN AGILE TESTING

Monique F. da Silva[1], Autran G. Moreno[2]

[1] Instituto Recôncavo de Tecnologia
[2] GESA – Grupo de Engenharia de Software e Aplicações
Universidade Salvador (UNIFACS)
monique@reconcavo.org.br, autrangm@gmail.com

Resumo
Este artigo apresenta as atividades que podem ser automatizadas dentro de um processo de teste ágil e as ferramentas que dão suporte a estas atividades. Apresenta também uma avaliação sobre o uso das ferramentas de automação de teste mencionadas.

Palavras-chave: teste ágil; ferrramentas de teste; engenharia de software.

Abstract
This paper describes the activities that can be automated within an agile testing process and tools that support these activities. It also presents an evaluation of the use of test automation tools mentioned.

Keywords: Agile testing; testing tools; software engineering.

1 INTRODUÇÃO

Teste de software é um processo, ou uma série de processos, executado para validar se o código de computador faz o que foi projetado para fazer e não faz algo fora do intencionado. Os testes podem ter diferentes objetivos como encontrar defeitos, aumentar o nível de confiança provendo informações do nível de qualidade do software e prevenir defeitos (GRAHAM et al, 2008, p. 18). O teste de software se constitui de atividades de grande importância em projetos de desenvolvimento de sistemas que contribuem para a criação de produtos de melhor qualidade e, consequentemente, geram uma maior satisfação do cliente.

Existem diferentes abordagens para o desenvolvimento de software e uma delas tem ganhado bastante destaque, a abordagem ágil. Tal abordagem apóia-se em dois objetivos fundamentais: oferecer produtos inovadores aos clientes (particularmente em situações altamente incertas) e criação de ambientes de trabalho em que as pessoas desejam ansiosamente ir ao trabalho todos os dias. (HIGHSMITH, 2004). Os métodos ágeis aplicam algumas práticas que são guiadas por seus princípios e valores fundamentais: a valorização dos participantes no processo e suas interações, priorizar a funcionalidade do software em detrimento da documentação abrangente, maior participação do cliente e principalmente capacidade adaptativa que serão vistos adiante neste artigo com mais detalhes.

Com a popularização das metodologias ágeis, muitos são os processos de desenvolvimento de software que surgiram para trazer mudanças ao molde tradicional de desenvolvimento. O teste de software ágil precisa estar agregado ao processo de desenvolvimento desde o início, acompanhando todo o ciclo de vida de um software.

Métodos ágeis enfatizam a colaboração entre as pessoas e as iterações curtas em um projeto de desenvolvimento de software. Por isso, a atividade de teste deve ser bem planejada e gerenciada 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.

As ferramentas quando utilizadas no momento correto trazem ganhos não só para a equipe de testes, mas para toda a equipe de um projeto de software. A utilização de ferramentas permite uma boa sinergia entre toda a equipe, a atualização de todos envolvidos no projeto, redução no tempo entre a identificação e a correção de erros e maior segurança ao se efetuar refatorações no código.

A automação das atividades de teste que será apresentada poderá ser utilizada tanto no modelo convencional 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 da metodologia ágil ser relativamente recente, o processo de teste inserido nesta metodologia mostra-se em fase de amadurecimento. O enfoque em

automação permite que, em longo prazo, algumas atividades de teste sejam executadas com mais eficiência e possam acompanhar o ritmo mais dinâmico da agilidade.

O objetivo deste artigo é identificar atividades dentro do processo de testes ágeis que podem ser automatizadas, e fazer um levantamento de ferramentas que podem ser utilizadas, mostrando como cada uma destas ferramentas pode contribuir em cada etapa do processo. Será apresentada uma avaliação baseada em relatos de experiência referente ao uso de ferramentas de automação de teste. Esta avaliação irá fornecer informações que podem nortear a escolha de ferramentas em um projeto de desenvolvimento de software ágil.

Este artigo está estruturado da seguinte forma: Na seção 2, são apresentados os conceitos relacionados a testes e metodologia ágil. Na seção 3, são apresentadas algumas ferramentas de automação de testes. Na seção 4 é apresentada a avaliação das ferramentas analisadas neste trabalho e na seção 5 é a conclusão deste artigo

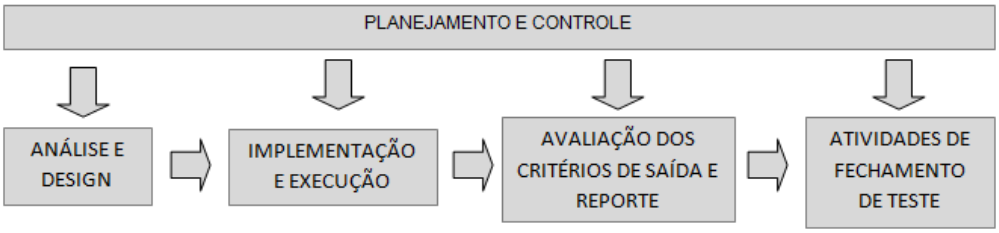
2 FUNDAMENTAÇÃO TEÓRICA

Nesta seção serão mostrados conceitos importantes dentro do universo de testes como: processo de teste tradicional, valores fundamentais dos métodos ágeis aplicados aos testes e automação de testes ágeis.

2.1. Processo de Teste Tradicional

O teste de software é um processo composto de uma série de atividades. Existem várias propostas de processo de teste para a abordagem tradicional na literatura, mas a maioria contém as seguintes atividades: planejamento e controle, análise e design, implementação e execução, avaliação dos critérios de saída e reporte e atividades de fechamento de teste (GRAHAM et al, 2008, p. 20), conforme mostra a Figura 1.

Figura 1 - Atividades fundamentais de teste



As atividades de planejamento dos testes englobam a determinação do escopo do teste, determinação dos riscos, dos objetivos dos testes, a abordagem do teste, definição de recursos necessários e planejamento as atividades posteriores. O controle das atividades tem como o objetivo medir e documentar o progresso das demais atividades e avaliar se o plano traçado está sendo cumprido.

Na fase de análise e design, os objetivos são transformados em condições de testes. Neste momento, identifica-se o que será testado e é avaliado se o sistema e os seus requisitos são testáveis.

Nas atividades de implementação e execução são desenvolvidos os casos, as suítes de teste e os scripts de teste. Além disso, a execução dos testes é iniciada, os resultados são documentados e os *bugs* são reportados.

Após a execução dos testes, são realizadas as atividades de avaliação de critérios de saída e reporte. Nesta etapa, os resultados dos testes são verificados para avaliar se a quantidades de testes foi suficiente ou se mais testes serão necessários. Ainda nesta etapa é verificado se os critérios de saída estão de acordo com o especificado e um relatório provendo informação a respeito dos testes é gerado.

O processo é finalizado com as atividades de fechamento. Neste momento é verificado se o que foi planejado está sendo entregue. Os testes são avaliados e as lições aprendidas são levantadas. Com os resultados dos testes é possível tomar medidas para melhorar o processo.

O processo de teste tem lugar em todo o ciclo de vida de desenvolvimento de software e não apenas no final. É importante ter isto em mente, pois segundo a regra 10 de Meyers, o custo da correção dos defeitos tende a subir quanto mais tarde eles

forem corrigidos (BASTOS et al., 2007). Se for possível encontrar e corrigir defeitos durante a etapa de requisitos a correção desses defeitos será feita com menor custo global. Assim, projetar os testes no início do ciclo de vida pode ajudar a prevenir que defeitos sejam introduzidos no código.

2.2. Valores Fundamentais dos Métodos Ágeis Aplicados aos Testes

Testes Ágeis são atividades da etapa de testes num processo de desenvolvimento de software que adotam conceitos de metodologias ágeis. Para entender melhor o que muda ao se utilizar testes ágeis em vez de métodos tradicionais de teste, é preciso entender primeiro quais os conceitos que a metodologia ágil trouxe para o universo de desenvolvimento de software. Ideias e métodos ágeis já existiam há algum tempo, porém ficaram consolidados com a divulgação do manifesto ágil em fevereiro de 2001 (FOWLER; HIGHSMITH, 2001, p.2) que tem como seus quatro valores fundamentais:

1. Os indivíduos e suas interações acima de procedimentos e ferramentas;
2. O funcionamento do software acima de documentação abrangente;
3. A colaboração dos clientes acima da negociação de contratos;
4. A capacidade de resposta a mudanças acima de um plano pré-estabelecido.

Diante destes valores identifica-se que o processo de testes inserido em um ciclo de desenvolvimento ágil necessita de adaptações importantes (CRISPIN; GREGORY, 2009) como:

- Ter a equipe de testes envolvida no projeto;
- A equipe de teste deve trabalhar mais próxima dos desenvolvedores;
- Evitar o tradicional: “Desculpe, os requisitos estão *congelados*, podemos adicionar esta funcionalidade na próxima versão”;
- Executar testes em iterações curtas;
- O cliente, não a equipe de teste, decide os critérios de qualidade para o produto.

Os métodos ágeis defendem uma equipe de testes mais dinâmica e participativa, logo, é importante que ela esteja envolvida no projeto desde o início e participe de reuniões com o cliente para conseguir absorver qual a sua real necessidade. Estimula-se também uma interação maior com os desenvolvedores que resultará em agir colaborativamente para alcançar os critérios de qualidade esperados pelo cliente. Iterações curtas são comumente utilizadas em métodos ágeis, pois assim se permite maior flexibilidade para mudanças. Considerando este contexto ágil, contar com ferramentas que possam auxiliar a automatizar certas atividades do processo, certamente irá contribuir no dinamismo exigido nesta filosofia ágil.

2.2.1 Ciclo de Vida de Testes Ágeis

Rildo Santos (2010) afirma que para que o desenvolvimento de software seja ágil realmente 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 XP para engenharia e desenvolvimento do software.

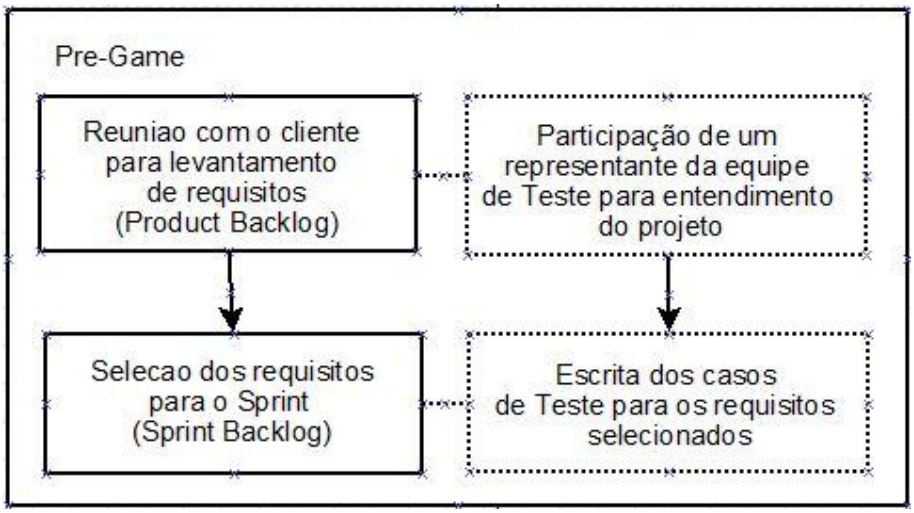
SCRUM é um framework que possui um ciclo iterativo e incremental para desenvolvimento de software orientados a objeto (SCHWABER, 1997). Nesta abordagem os requisitos do sistema são chamados de itens do *backlog*. O SCRUM utiliza o conceito de entrega contínua de valor, ou seja, versões do software que 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 um entregável de valor para o cliente. É defendido que se façam reuniões frequentes para acompanhar o andamento do projeto e reorganizar o planejamento caso algo não esteja saindo conforme deveria.

O SCRUM possui 3 grupos de fases, para cada uma destas fases são recomendadas atividades de teste em paralelo. O ciclo de vida de testes apresentado

com as atividades exibidas a seguir foram baseados em estudos realizados (KARLSON, 2009),(CRISPIN, 2009) e serão demonstradas nos diagramas como caixas de linha pontilhada:

1) O **pre-game** (**Figura 2**) onde é feito o planejamento, o levantamento de itens do *backlog*, a priorização destes itens pelo cliente e a definição de quais itens farão parte de quais *sprints*.

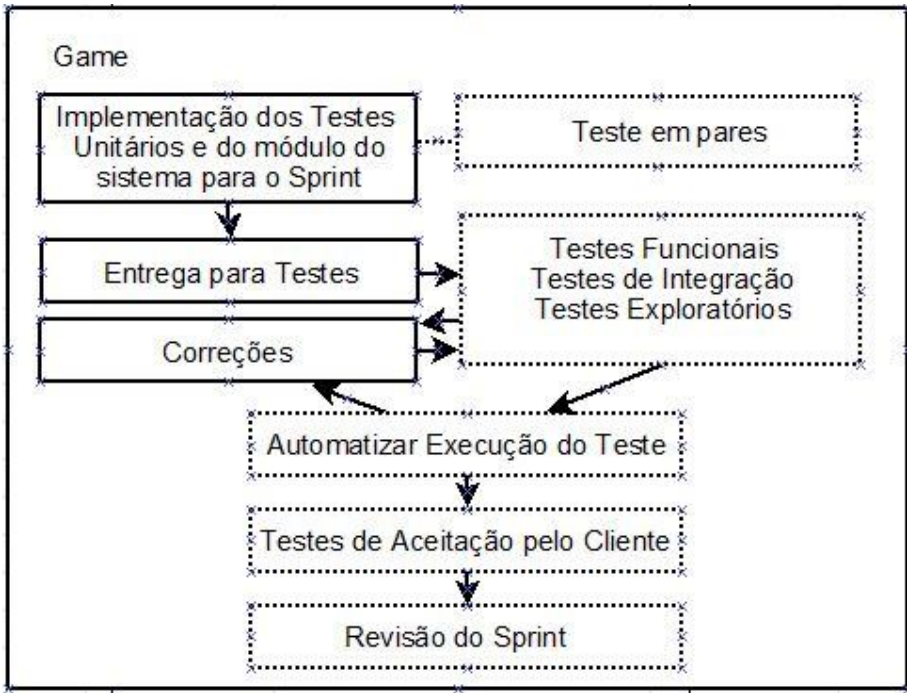
Figura 2 - Pre-Game



Em estudos de caso apresentador por Yugank Bhatnagar (2008) e Erik Karlsson (2009), observou-se que escrita de casos de testes detalhados demandam muito tempo para serem realizados. Em vez de ser descrito o passo-a-passo destes casos, eles mostram que apenas a identificação de todos os cenários de teste possíveis é suficiente para conduzir os testes de maneira eficiente.

2) **Game (Figura 3)** que é o desenvolvimento de funcionalidades que serão liberadas em um *sprint*, respeitando constantemente as variáveis de tempo, requisitos, qualidade e custo.

Figura 3 - Game



Segundo Aderson Bastos (2007), testes unitários são aqueles aplicados aos menores componentes de código criados. Na grande maioria dos casos são realizados por desenvolvedores. Os testes funcionais são aqueles realizados para verificar se os requisitos e as especificações do sistema foram atendidos. Os testes de regressão são executados em segmentos já testados após a implementação de uma mudança em outra parte do software. Os testes exploratórios são executados quando existe pouca documentação ou quando o prazo é curto e são baseados na experiência e intuição do testador.

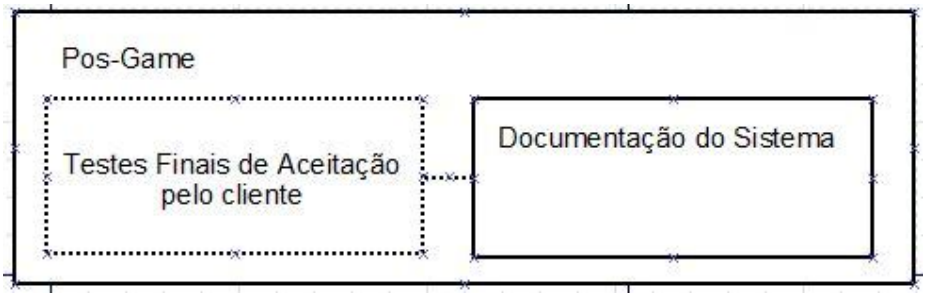
Os testes de regressão devem executados ao final de cada *sprint* (BHATNAGAR, 2008) (KARLSSON, 2009). Como os testes de regressão são executados com bastante frequência, teste manual, neste caso, iria demandar muito tempo. Por isso, recomenda-se a automação destes testes.

Diferente de um processo de teste tradicional, num processo de teste ágil defende-se uma aproximação entre testadores e desenvolvedores, eles devem se

apoiar para garantir a qualidade do software através de testes. O testador pode sugerir ao desenvolvedor ideias para criação de testes e vice-versa (TALBY, 2006).

3) **Pos-game (Figura 4)**: Preparação para liberar a versão final, incluindo sua documentação e testes.

Figura 4 - Pos-Game



O XP é um framework ágil para desenvolvimento de software. Dele serão usados os conceitos :

- Refatorar quando e onde for possível. Refatorar é mudar o código já implementado de maneira a escolher dentre as soluções de um problema qual a mais flexível, eficiente e eficaz.
- Codificar o teste unitário primeiro.
- Toda produção de código é programada em pares.
- Quando um *bug* é encontrado, testes são criados.

2.3. Automação de Testes Ágeis

"O propósito da automação de testes pode ser resumidamente descrito como a aplicação de estratégias e ferramentas tendo em vista a redução do envolvimento humano em atividades manuais repetitivas." (DIZ, 2009 apud KANER, 2001)

Fazer o uso de algumas ferramentas que automatizam etapas do processo de teste pode fornecer aos envolvidos ganho de produtividade e evitar erros humanos, caso esta automação seja usada de maneira apropriada. Alguns dos princípios da agilidade podem ser facilitados com o uso de ferramentas e tecnologias. Ferramentas *web online* de reporte de defeitos, por exemplo, promovem uma integração maior dos envolvidos no processo de desenvolvimento como analistas, testadores e

desenvolvedores. A praticidade que é dada por muitas destas ferramentas ajuda a ter uma documentação mais fácil de ser atualizada, o que naturalmente reduz o tempo gasto com editores tradicionais de texto que oferecem pouco apoio à atividade de documentação. Segundo Bach (2003), automação de teste é muito mais do que fazer com que o computador execute os testes. Existem mais atividades que podem ser automatizadas para facilitar o trabalho da equipe como, por exemplo, Geração de dados e scripts de teste, configuração de sistema, simuladores, gravação de atividades, análise de cobertura de testes, Gerenciamento de Testes etc. Automação está no coração do desenvolvimento ágil de software e é a chave para testes ágeis (KARLSSON, 2009).

As ferramentas existentes de apoio ao teste atualmente podem ajudar não apenas na etapa de execução, mas também no planejamento, na criação dos casos de teste, no reporte de defeitos etc.

2.3.1 Situações candidatas à automação

Quando se deseja aplicar a automação na etapa de execução dos testes das funcionalidades do sistema que, em geral, é feita manualmente por um testador, vários fatores devem ser avaliados para se identificar se é vantagem ou não se automatizar. Segundo Jorge Diz (2009), testes automatizados têm algumas características que diferem de testes manuais conforme podemos ver no quadro abaixo:

Tabela 1 - Testes Manuais X Testes Automatizados

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	Exigem que cada ação seja programada
Execução	São demorados para executar	São 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 teste e desenvolvimento

De modo geral, existem testes que são mais aconselháveis a serem automatizados do que outros como, por exemplo:

- 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 às principais funcionalidades do sistema. Este teste visa identificar defeitos nestas funcionalidades com o intuito

de avaliar se a versão liberada pode continuar sendo testada pela equipe de testes.

- Cálculos matemáticos - evitar erros humanos.
- Funcionalidades críticas - como seres humanos são influenciados pelo humor para execução de uma tarefa, funcionalidades críticas não deveriam depender de um testador que pode esquecer-se de testar algo importante.

Figura 5 - Testes candidatos à automação (DIZ, 2009)

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

- Funcionalidades novas - pois estas têm uma grande probabilidade de mudanças, o ideal seria criar o teste quando se alcança certa estabilidade.
- Funcionalidades pouco usadas - haverá um custo para automatizar e um esforço desnecessário para automatizá-los, pois estes testes serão pouco aplicados já que a funcionalidade não terá muita utilização.
- Funcionalidades que exigem inspeção visual - Não existe uma maneira de 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 no início de um projeto de software, para ser exibido para clientes ou para a própria organização. Por isso, não há a necessidade de automatizar testes em protótipos já que este ainda não é o software propriamente dito.

Figura 2 -Testes que não se recomenda a automação (DIZ, 2009)

Vale lembrar que a aquisição de ferramentas, compartilhadas pelas diferentes equipes no processo de desenvolvimento de software, deve ser discutida com todos os envolvidos, visando selecionar aquela que melhor se aplica aos processos que serão adotados e evitar gastos desnecessários.

3 FERRAMENTAS DE TESTES

A seguir serão descritas algumas ferramentas não pagas que apoiam as atividades de teste mencionadas na seção 2.1.1 Ciclo de Vida de Testes Ágeis.

Testlink – O testlink é uma ferramenta gratuita e *Open Source* de gerenciamento de casos de teste utilizada em plataforma Web. Esta ferramenta pode

ser utilizada para registro dos requisitos do sistema e criação dos seus casos de teste. Estes casos de teste poderão ser vinculados aos requisitos permitindo sua rastreabilidade. Após o desenvolvimento, esta ferramenta pode ser utilizada para apoio na execução destes casos, registrando seus status (passou, falhou ou bloqueado) durante a fase de testes. Após a finalização dos testes, é possível gerar relatórios com toda a informação registrada durante a execução dos testes. O testlink permite criar vários projetos de teste e organizar os casos de teste de cada projeto em suítes, de acordo com a necessidade. Dessa maneira, caso existam vários projetos de software em andamento, é possível ter em uma mesma ferramenta todos os casos de teste disponíveis para cada projeto.

Além disso, por ser uma ferramenta web, é possível ser acessada de qualquer local através de um browser. Qualquer alteração realizada por um membro da equipe, em um caso de teste, por exemplo, será visualizada imediatamente pelos demais, evitando a utilização de casos de teste obsoletos. Esta interação imediata favorece o controle e gerenciamento dos casos de testes nas equipes ágeis.

O testlink é uma ferramenta simples e intuitiva que dispensa muito tempo de treinamento.

Salomé TMF – O Salomé TMF é uma ferramenta que apoia todo o processo de teste. Ou seja, permite criar casos de teste, executar testes manuais ou automatizados (através do plug-in Junit), registrar os resultados, gerenciar os requisitos e os defeitos do sistema e produzir relatórios de resultados em html. Na criação dos casos de teste, é possível criar casos detalhados, anexar arquivos ou associar url´s. Esta ferramenta permite que casos de teste gerados em arquivos xml ou xls possam ser importados para a ferramenta. Além dos casos de teste, os ambientes também são cadastrados e gerenciados, de acordo com o projeto. Este ambiente pode ser um browser, um sistema operacional, uma versão do produto etc.

Para executar os casos de teste, estes precisam estar associados a campanhas. Cada campanha pode ter casos de teste diferentes, dependendo da necessidade do

teste. Estas campanhas, por sua vez, precisam estar associadas a uma execução de campanha.

O Salomé TMF possui uma integração com a ferramenta de gestão de defeitos Mantis e permite que defeitos sejam cadastrados no Mantis, assim que um caso de teste falhe. Basta selecionar a opção Mantis no menu e solicitar a inserção de uma anomalia. Uma janela se abre com os campos necessários ao preenchimento do defeito. A anomalia fica então associada ao resultado do teste que falhou.

Esta ferramenta possui mais funcionalidades que o testlink, por isso, exige maior esforço para o seu aprendizado. É uma ferramenta mais completa e complexa. Porém, de maneira semelhante ao testlink, é uma ferramenta web e, por isso, pode facilitar o processo de teste ágil.

Mantis – é uma ferramenta gratuita e *Open Source* que permite o gerenciamento de defeitos. Com o mantis é possível gerenciar os defeitos de vários projetos ao mesmo tempo. Os projetos cadastrados no mantis podem ser acessados facilmente através de uma lista de projetos. A partir do momento em que um projeto é selecionado, é possível criar, editar ou remover um defeito associado a ele.

Para realizar o controle das operações realizadas sobre um defeito, o mantis possibilita o controle de perfis de usuários. Cada perfil pode ser configurado de acordo com a necessidade dos projetos. Além disso, é possível gerenciar os usuários em relação a visibilidade dos projetos. Ou seja, caso existam dois projetos, um usuário cadastrado pode ter apenas a visão de um destes projetos.

Os defeitos cadastrados podem ter diversas severidades, *status* e prioridades associadas a eles. Estes dados podem ser configurados, para atender aos processos da organização, por ser uma ferramenta de código aberto.

Por ser uma ferramenta *web*, é possível acessá-la de qualquer lugar através de um browser.

Com a utilização do mantis, assim que um defeito é cadastrado pelo testador, este pode ser imediatamente atribuído a um desenvolvedor para ser corrigido. Dessa forma, enquanto os testadores trabalham, os desenvolvedores também podem

executar sua atividade, permitindo, assim, a geração de uma versão corretiva do software em teste de maneira mais rápida.

Redmine – é uma ferramenta Web gratuita e *Open Source* de gerenciamento de projetos e gerenciamento de defeitos. É possível realizar o gerenciamento de mais de um projeto através do Redmine. De maneira semelhante ao Mantis, é possível definir perfis de usuários e realizar o controle de acesso aos projetos. Além disso, é possível definir status de tickets (defeitos ou novas atividades) e campos personalizáveis de acordo com a necessidade da organização. Esta ferramenta permite criar tickets, definir tempo de trabalho para eles e atribuir à atividade para uma determinada pessoa. Dessa forma, quem deverá realizar a atividade receberá uma notificação por e-mail informando sobre a atividade que ele deverá realizar. À medida que a atividade está sendo executada, o usuário atualiza o ticket, indicando a porcentagem ~~da tarefa~~ que já foi realizada.

O Redmine permite a integração com sistema de controle de versão (svn, git, cvs) e contém calendário e gráficos de *Gantt* para representar os projetos e seus deadlines (prazos de entrega).

Esta ferramenta, como o Mantis, permite que durante os testes, os desenvolvedores tomem conhecimento dos defeitos na ferramenta e possam executar as correções em paralelo. As versões de teste são geradas mais rapidamente, o que favorece as equipes ágeis.

xUnit – são em geral frameworks para criação de testes unitários. O teste unitário é um teste que faz verificações na unidade mais básica do software: suas funções ou métodos. Ao contrário de outras atividades de teste, a criação dos testes unitários é de responsabilidade do programador e é feita durante a implementação do sistema.

Testes Unitários podem ser criados antes, durante ou após o desenvolvimento, porém recomendável é que se use as práticas de TDD (Desenvolvimento Orientado a Testes) que recomendam que o teste seja criado antes da funcionalidade, pois desta

maneira quando os métodos forem desenvolvidos irão forçar a seguir um padrão e já poderão ser validados quanto à corretude.

Em geral é bastante custoso ou até impossível em alguns casos se criar testes unitários para projetos que não foram planejados para isto, por exemplo, quando funções são grandes demais, ou quando se retorna *void* em um método ou função.

Testes unitários apoiam a flexibilidade para mudanças em métodos ágeis, pois permitem que se faça refatoração ou melhorias na aplicação de maneira mais confiável, porque uma vez alterados pode-se tentar identificar impactos rodando-se os testes criados. Além disto, alguns desenvolvedores ágeis defendem que testes unitários bem criados podem ser considerados como parte da documentação, pois mostram regras de negócios do sistema.

 **Jester** – é uma ferramenta de apoio ao uso de testes unitários criados com JUnit (xUnit para Java). Seu objetivo é ajudar a identificar se os testes unitários criados estão de fato sendo úteis e cobrindo o código fonte apontando erros caso uma manutenção corretiva ou evolutiva seja feita. Para iniciar suas atividades é necessário se certificar primeiro de que nenhum teste unitário falhe. Ele realiza as seguintes atividades:

1. Encontra o código que não está coberto por testes;
2. Faz algumas alterações no código fonte;
3. Executa testes;
4. Exibe um relatório dizendo o que mudou;
5. Mostra as alterações feitas que levaram os testes a falhar.

Como o Jester faz alterações no código fonte é importante ter um backup antes de executá-lo. Um código é considerado bem coberto por testes unitários caso alterações no código levem os testes a falhar, caso contrário significa que o número de testes unitários criados ainda não é suficiente para dar certa margem de segurança na refatoração.

Selenium – ferramenta para automatizar a execução de testes em sistemas *web*. O selenium IDE é um Add-on para o navegador Firefox que permite a gravação de um roteiro durante a execução de um teste manualmente. Este roteiro pode ser usado para repetição automática dos testes executados manualmente e também convertido em código java ou outras linguagens.

A princípio o selenium é uma ferramenta de teste de caixa preta, ou seja, um teste de software para verificar a saída dos dados usando entradas de vários tipos ignorando detalhes da perspectiva interna do sistema (código fonte). No entanto, utilizando o selenium RC o desenvolvedor poderá integrar seus testes com particularidades da implementação do sistema tornando-o assim num teste de caixa cinzenta. Para uso do selenium RC é requerido o conhecimento de pelo menos uma das linguagens suportadas o que pode impedir que alguns testadores o utilizem.

O selenium atua na automação de testes funcionais e assim como xUnit contribuem para refatoração na metodologia ágil e documentação das regras de negócio.

Sahi – é uma ferramenta de teste *Open Source* semelhante ao Selenium, ou seja, que automatiza os testes de aplicações web. Diferentemente do Selenium, que é um Add-on de um navegador, o Sahi utiliza um Proxy para injetar Javascript em páginas web e, dessa maneira, executar os testes. O Sahi pode ser executado a partir de qualquer browser. Quando o Sahi é executado, após ser configurado corretamente, ele automaticamente modifica o Proxy do browser e, a partir de então, poderá ser iniciada a gravação dos testes executados manualmente gerando, assim, os scripts automatizados.

Nos scripts de teste gerados poderão ser introduzidas funções lógicas, o que pode facilitar o seu entendimento e deixar melhor organizado.

Após a execução de um script, é possível visualizar o *log* da execução. Caso o teste falhe, à presença de defeitos é identificada através do realce das linhas executadas na cor vermelha. Em caso de sucesso, o realce das linhas será em verde.

A vantagem desta ferramenta é que apenas utilizando-a, devidamente configurada, é possível executar um script a partir de qualquer browser. Além disso, não existe a necessidade de conhecimento de alguma linguagem de programação para desenvolver os testes automatizados.

Os testes criados a partir do Sahi são executados um de cada vez. Não é possível executar mais de um script criado de uma só vez.

Assim como o Selenium, o Sahi possibilita que as tarefas repetitivas do teste sejam automatizadas. Tais tarefas, caso sejam executadas manualmente, são demoradas e comprometem as entregas curtas que caracterizam os processos ágeis.

4 AVALIAÇÃO DAS FERRAMENTAS DE AUTOMAÇÃO

Esta seção apresenta uma avaliação feita sobre as ferramentas de automação de testes. A avaliação de algumas das ferramentas só foi possível através de informações obtidas por pesquisas realizadas na internet nos sites oficiais, fóruns e comunidades das ferramentas. O resultado alcançado foi baseado nestas informações obtidas, bem como na experiência pessoal de uso das ferramentas pelos autores deste artigo. O objetivo é que este levantamento sirva de base para consulta permitindo a escolha das ferramentas que mais se adequam a cada situação. Os critérios utilizados para a avaliação das ferramentas foram os mesmos usados por Elton Sixpence (2011). Porém, foram incluídos alguns adicionais, os quais foram acrescentados com base nos valores fundamentais da abordagem ágil já apresentados.

Os critérios serão os seguintes:

4.1 Compatibilidade e Navegadores Suportados

Este critério é importante, pois informa se a ferramenta é compatível com os sistemas operacionais e navegadores, a depender do caso, mais utilizados atualmente.

4.2 Linguagem de Programação dos Scripts

Muitas das ferramentas que automatizam a execução de testes utilizam scripts que descrevem os passos desta execução. Estes scripts também podem utilizar linguagens diversas. A avaliação irá pontuar com valor maior, ferramentas que suportam uma maior variedade de linguagens.

4.3 Necessidade de Formação

Ferramentas complexas podem trazer a necessidade de treinamento para facilitar o entendimento e uso por parte da equipe de envolvidos no projeto.

4.4 Estabilidade e Versões da Ferramenta

Uma medida de qualidade para um software pode ser a quanto tempo ele existe e com que freqüência é atualizado, isto porque, a cada versão liberada, novas funcionalidades são adicionadas e erros são corrigidos. Além disto, o risco de se obter uma ferramenta que não será continuada é menor.

4.5 Interface Gráfica e Usabilidade

Uma ferramenta que tenha boa usabilidade proporciona uma interface gráfica simples e fácil de usar. Atualmente esta é uma das características importantes para a seleção de uma ferramenta, pois este fato reduz o esforço e tempo gastos no seu aprendizado.

4.6 Documentação, Suporte e Comunidade de Utilizadores

Para adotar uma ferramenta, é necessário fazer uma avaliação das suas características, instalar e utilizar. Por isso, é de grande importância que a documentação e o suporte à ferramenta sejam suficientes para guiar estas atividades.

A quantidade de utilizadores pode ser utilizada como um indicador de maturidade da ferramenta.

4.7 Instalação e Configuração do Ambiente de Testes

Este quesito deve ser analisado, pois a instalação e configuração de um ambiente de testes automatizado são mais custosas do que um ambiente de testes manuais devido à maior complexidade. Por isso, é importante estimar os custos antes de partir para a prática.

4.8 Limitações

Para avaliar se a ferramenta irá atender às necessidades, é importante analisar as suas limitações. Por ser também um software, é normal serem encontradas limitações diversas na ferramenta. Mas para uma determinada organização uma limitação pode não ser motivo para a não utilização dela. Enquanto que para outra pode ser um grande empecilho.

4.9 Executa Testes de Caixa Branca

O teste de caixa branca é o teste baseado em programa que requer a especificação do código fonte e seleção dos casos de teste que exercitem partes do código e não de sua especificação. (MALDONADO et al. , 2007, p.3).

4.10 Executa Testes de Caixa Preta

O teste de caixa preta é o teste baseado em especificação cujo objetivo é determinar se o programa satisfaz aos requisitos funcionais e não-funcionais que foram especificados. (MALDONADO et al. , 2007, p. 4).

4.11 Viabiliza melhor comunicação entre a equipe

A comunicação é uma das principais características de uma equipe ágil. Em um projeto de software, a equipe pode estar geograficamente espalhada e, por isso, é extremamente importante a existência de ferramentas que facilitem e contribuam neste ponto.

4.12 Documentação mais eficiente (reduz necessidade de papel, facilita criação e atualização)

Conforme citado anteriormente, um dos valores da metodologia ágil é o funcionamento do software acima de documentação abrangente (FOWLER; HIGHSMITH, 2001, p.2). Apesar da metodologia ágil não despender muito tempo em documentação, é importante que todas as informações necessárias ao projeto estejam acessíveis de alguma forma.

As ferramentas podem fornecer formulários para criação de documentação dos testes de maneira simples, prática e eficiente. Dessa forma, proporciona uma redução da necessidade de criar grandes *templates* e documentar em papel.

4.13 Ouvir o cliente

Para o cliente é importante existir um canal de comunicação além das reuniões presenciais ou por telefone. É importante que as requisições de mudanças que ele solicite fiquem registradas e que possam ser visualizadas por todos. O apoio de uma ferramenta neste caso facilita a comunicação do cliente e dissemina as solicitações do cliente para toda a equipe de maneira prática.

4.14 Apoia Adaptatividade e Flexibilidade do Software

Para ser realizada a refatoração de um sistema de forma segura, é necessário garantir que não serão introduzidos erros no sistema e que as funcionalidades do

sistema não serão alteradas do ponto de vista do usuário. A existência de uma ferramenta que apoie esta atividade é de grande importância para o desenvolvimento de software, principalmente para aqueles mais complexos.

A **Tabela 2** apresentada a seguir utiliza a escala de numeração de 1 a 5 para pontuar os critérios acima citados sendo:

- 1 – ruim
- 2 – regular
- 3 – médio
- 4 – bom
- 5 – ótimo
- X – não se aplica

Tabela 2 - Avaliação das Ferramentas de Automação de Testes

	Testlink	Salomé TMF	Mantis	Redmine	XUnit	Jester	Selenium	Sahi
Compatibilidade e Navegadores Suportados	5	5	5	5	5	5	5	5
Linguagem de Programação dos Scripts	X	1	X	X	4	1	5	5
Necessidade de Formação	5	3	5	5	2	2	3	5
Estabilidade e Versões da Ferramenta	5	3	5	5	3	2	4	5
Interface Gráfica e Usabilidade	5	4	5	5	1	1	4	5
Documentação, Suporte e Comunidade de Utilizadores	5	3	5	5	5	2	4	4
Instalação e Configuração do Ambiente de Testes	4	3	3	3	2	2	3	5

Limitações	3	3	3	4	3	3	4	4
Executa Testes de Caixa Branca	X	2	X	X	5	X	3	X
Executa Testes de Caixa Preta	X	4	X	X	1	X	4	4
Viabiliza melhor comunicação entre a equipe	3	3	4	5	X	X	X	X
Documentação mais eficiente	4	4	4	4	4	X	3	3
Ouvir o cliente	X	X	4	4	X	X	X	X
Adaptatividade e Flexibilidade	X	2	X	X	5	4	5	5

5 CONCLUSÃO

Existem várias atividades no processo de testes ágeis que podem ser beneficiadas com o uso de ferramentas de automação. 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. O número de software gratuito com a finalidade de apoiar o teste já é considerável, e muitos deles já estão ganhando certo nível de maturidade. O uso destas ferramentas 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. Uma análise de ferramentas pagas poderia ser feita em trabalhos futuros criando-se critérios de comparação para avaliar as diferenças entre estas e as ferramentas gratuitas para o usuário medir se vale à pena ou não um investimento financeiro nesta área. Percebe-se que os instrumentos de automação existentes atualmente não englobam todas as etapas de um processo de testes sendo que cada uma é específica para uma ou algumas das etapas em que pode contribuir. Caso ocorra um amadurecimento maior unindo estas diversas contribuições em uma ferramenta única seria de grande valia.

REFERÊNCIAS

FOWLER, Martin; HIGHSMITH, Jim. The Agile Manifesto. **Software Development Magazine**. 2001. Disponível em: <http://www.scribd.com/doc/55710137/The-Agile-Manifesto-SDMagazine>. Acesso em: 01 ago. 2011.

HIGHSMITH, Jim. **Agile project management**: creating innovative products. Boston: Pearson Education, Inc. 2004. p. 19.

BACH, James. Agile Test Automation. **British Computer Society SIGST**. 2003. White Paper. Disponível em: <http://www.satisfice.com/presentations.shtml> Acesso em 03 ago. 2011.

MYERS, J. Glenford. **The Art of Software Testing**. Segunda Edição. New Jersey: John Wiley & Sons Inc. 2004. p. 8-11.

GRAHAM, Dorothy; VEENENDAAL, Erik van; EVANS, Isabel; BLACK, Rex. **Foundations of Software Testing: ISTQB Certification**. Cengage Learning Business Press., 2008.

CRISPIN, Lisa; GREGORY, Janet. **Agile Testing: A practical guide for testers.** Addison-Wesley Professional, 2009.

KARLSSON, Erik; MÅRTENSSON, Fredrik. **Test processes for a Scrum team**. 2009. 107 f. Tese (Mestrado) - Lund University, Sweden. 2009.

MULLER, Matthiaas M.; TICHY, Walter F. Tichy. **Case Study: Extreme Programming in a University Environment.** IEEE. 2001. Disponível em: <http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=919128> Acesso em 24 set. 2011.

TALBY, David; KEREN, Arie; HAZZAN, Orit; DUBINSKY, Yael. **Agile Software Testing in a Large-Scale Project**. 2006. Publicado pela IEEE Computer Society.

BHATNAGAR, Yugank; BHAMBRI, Varun. **Our Approach in Overcoming Testing Challenges in Agile Environment**. In: Test 2008: Agility in Testing. 2008. Nova Deli, India.

DIZ, Jorge; NOGUEIRA, Elias. **Agilidade com Ferramentas de Automação**. TDC In: The Developers Conference. 2010. São Paulo. Disponível em: <http://www.slideshare.net/elias.nogueira/agilidade-com-ferramentas-de-automao-como-e-por-qu>. Acesso em 01 ago. 2011.

SIXPENCE, Elton; ADÃO, Pedro; SMITH, Camaron. **Automatização De Casos De Teste Como Processo De Melhoria Da Qualidade Do Software: O Caso Da Aplicação E-Learning Isupac3 No Isutc.** 2011. Disponível em: <http://www.math.ist.utl.pt/~padao/publications/11-SAS-CLME/11-SAS-ISUPAC3.pdf> Acesso em: 16 set 2011.

SANTOS, Rildo. **Engenharia de Software 100% Ágil (SCRUM, FDD e XP)**. 2010. Disponível em: <http://pt.scribd.com/doc/71207440/Scrum> Acesso em: 17 set 2011.

MALDONADO, José Carlos; BARBOSA, Ellen Francine; VINCENZI, Auri Marcelo Rizzo; DELAMARO, Márcio Eduardo; SOUZA, Simone do Rocio Senger de; JINO, Mario. Introdução ao Teste de Software. In: SIMPÓSIO BRASILEIRO DE ENGENHARIA DE SOFTWARE, 19., 2007. João Pessoa **Anais...** PB. Minicurso. 2007.

ALVES, Angela Maria; JUNIOR, Claudio André da Silva. **Avaliação da Maturidade de um Software de Automação de Testes Livres**. In: CONFERÊNCIA IADIS IBERO-AMERICANA WWW/INTERNET. 2008. Lisboa, Portugal

BASTOS, Aderson; RIOS, Emerson; CRISTALLI, Ricardo; Moreira, Trayahú. **Base de conhecimento em teste de software**. São Paulo: Martins, 2007.

FEITOSA, Daniela. **Um estudo sobre o impacto do uso de desenvolvimento orientado por testes na melhoria da qualidade de software.** 2007. 55 f. Monografia (Graduação) - Ciências da Computação, Universidade Federal da Bahia, Salvador, 2007.

RIBEIRO, Eliza Fabiane Martins. **Automação em Testes de Software – Utilização da ferramenta Selenium.** 2010. Disponível em: <http://pt.scribd.com/doc/38644873/AutomacaoEmTestesDeSoftware-UtilizacaoDaFerramentaSelenium> Acesso em 24 set. 2011.

CAETANO, Cristiano. **Automação e Gerenciamento de Testes - Aumentando a Produtividade com as Principais Soluções Open Source e Gratuitas (e-book)**. 2007. Disponível em: <http://www.digitalworks.eti.br/DigitalWorks/Upload/Curso/automa_teste.pdf> Acesso em 26 set. 2011.