Análise de ferramentas de teste automatizado de software

Thyago Peres Carvalho[1], Joslaine C. Jeske de Freitas[1]

[1] Universidade Federal de Goiás (UFG) *Campus* Jataí (CAJ)
Caixa Postal 03 – 75.801-615 – Jataí – GO – Brasil

{thyagopcarvalho, joslaine}@gmail.com

Abstract. *There are many software test tool, which can be used to increase the quality of programs produced. This work will make a comparison of some tools chosen, which will be analyzed. In Brazil have a large amount of micro and small companies producing software, the work has a focus on a evaluation of these tools to help with these companies to make more feasible the inclusion of software testing process in software production. The end of the study contribute these companies making if feasible to include software testing in the production process of programs.*

Resumo. *Existem muitas ferramentas de teste de software que podem ser utilizadas para aumentar a qualidade dos programas produzidos. O presente trabalho, ainda em andamento, fará uma comparação de algumas ferramentas de teste automatizado. Devido ao grande quantidade de micro e pequenas empresas produtoras de software existentes no Brasil, o trabalho tem um enfoque em fazer uma avaliação de algumas destas ferramentas automatizadas. Espera-se ao término da pesquisa contribuir com essas empresas tornando mais viável a inclusão do teste de software no processo de produção de programas.*

1. Introdução

O mundo hoje é dependente de sistemas de computadores, sendo eles complexos ou não. Manter a qualidade desses sistemas é um ponto crucial. Ao codificar um *software* por mais preciosismo que se use para que não ocorram defeitos, eles são difíceis de serem evitados. Pfleeger (2004) afirma que todos os programadores gostariam de ser perfeitos, e não cometer nenhum erro ao produzir um programa, mas esta máxima não acontece.

Cheque e Kon (2008) alegam que não é uma tarefa fácil alcançar uma boa qualidade em um *software*, pois os mesmos são complexos, e envolvem problemas no processo de desenvolvimento, como questões humanas, técnicas, burocráticas, de negócio e políticas, Pressman (2006) garante que o teste de *software* é um fator decisivo para garantir a qualidade de um programa.

Como pode ser observada, a fase de teste é uma atividade complexa, demorada, cansativa e cara. As micro e pequenas empresas não dispõem de nenhum desses recursos para que possam executar os testes nos seus projetos desenvolvimento.

Segundo relatório da ABES (2010) o Brasil movimentou 5,51 bilhões de dólares em *software*, o que representou perto de 2,2% do mercado mundial. Neste mesmo ano o Brasil tinha o seu mercado explorado por 8520 empresas empenhadas no desenvolvimento, produção e distribuição de programas de computador, as quais 94% dedicadas em desenvolvimento, são classificadas como micro e pequenas empresas.

Fazer o teste de forma manual se torna uma tarefa inviável de ser realizada, principalmente pela grande quantidade de tempo gasto para que um testador execute cada caminho possível de um programa.

A necessidade de se automatizar o processo de teste torna-se evidente, fazendo então que os testes sejam feitos por ferramentas e não mais por mão de obra humana. O teste automatizado possibilita que a cada mudança no código do programa os testes possam ser novamente executados, testes exaustivos para que sejam encontrados problemas de codificação do objeto do teste.

Para confirmar a necessidade levantada acima é possível encontrar muitas ferramentas para cada um dos tipos, tornando enorme o universo de mecanismos de auxílio ao teste. Mas mesmo com a vasta gama de ferramentas que auxiliam no teste, Veloso (2010) afirma que as necessidades não são supridas por completo. Santos (2010) também afirma com seu trabalho a partir de avaliações realizadas pelo referido, que existem muitas características que podem ser aprimoradas nos instrumentos.

São muitos os tipos de ferramenta para o teste automatizado. Pressman (2006) cita alguns tipos de ferramentas: Analisadores estáticos, auditores de código, processadores de asserção, geradores de arquivos de teste, geradores de dados de teste, verificadores de teste, bancadas de teste, comparadores de saída, sistemas de execução simbólica, simuladores de ambiente, analisadores de fluxo de dados.

A relevância desta pesquisa é justificada por alguns dados bibliográficos: Pressman (2006) garante que o teste no processo de desenvolvimento de *software* gasta 40% de esforço, já Harrold (2000 apud VELOSO, 2010) assegura que o custo do teste na fase de desenvolvimento pode chegar próximo a 50% do custo total do desenvolvimento.

2. Conceitos Gerais

2.1 O teste de *software*

Segundo Delamaro, Maldonado e Jino (2007) existem uma sucessão de atividades, que tem o intuito de não deixar que os erros perdurem. Essas atividades são chamadas de "validação, verificação e teste" ou "VV&T". As atividades de VV&T podem ser divididas em estáticas, que não tem a necessidade de um programa executável para serem conduzidas, e dinâmicas, que tem a necessidade de um modelo ou de um executável. Presmman (2006) alega que a atividade de teste de *software* é um componente da verificação e validação ou “V&V”, o teste exerce um papel muito importante na V&V, mas muitas outras atividades são necessárias para completar essa fase.

De acordo com Presmman (2006) a verificação, é o aglomerado de atividades que asseguram a implementação correta de uma função específica, e validação, é um conjunto de diferentes atividades que garantem que o *software* segue as especificações do cliente.

Conforme Delamaro, Maldonado e Jino (2007), a atividade de teste de *software* é uma atividade dinâmica, e tem como objetivo demonstrar que o *software* não tem seu funcionamento conforme o previsto. Esta fase do processo de produção de *software* tem como meta executar um programa ou um modelo com algumas entradas previamente conhecidas, verificando se o comportamento é o esperado. Caso esta ação traga algum resultado diferente do esperado, pode-se dizer que um erro ou defeito foi encontrado.

Delamaro, Maldonado e Jino (2007) e Pressman (2006) afirmam que a fase de teste no processo de construção de um *software* é uma fase complexa. Pois existem muitos fatores que contribuem para a ocorrência de erros. Pfleeger (2004) aponta alguns problemas que podem causar a ocorrência de erros: os *software*s geralmente trabalham com um amontoado de estados, funções, atividades e algoritmos complexos; são utilizadas as ferramentas que estão à disposição, para demonstrar a construção do *software* para um cliente, que na maioria das vezes, estará indeciso sobre a sua real necessidade; o tamanho do projeto de um *software* e a quantidade de pessoas envolvidas.

É muito importante para um *software* garantir sua confiabilidade, definida por Delamaro, Maldonado e Jino (2007) como a execução de um *software* de modo esperado. Eles ainda afirmam que a confiabilidade é uma das características mais importantes de um *software*, se não for a mais importante.

2.2 Técnicas ou estratégia para o teste de *software*

Para se testar um *software* são utilizadas técnicas, com o objetivo principal de encontrar falhas no *software*. Abaixo são descritas as técnicas mais conhecidas.

A técnica de teste de caixa-branca é definida por Pressman (2006) como uma técnica que se baseia num detalhado exame dos procedimentos. Casos de teste são derivados dos caminhos lógicos do *software* que está sendo testado. Os casos de teste servem para demonstrar se condições e/ou laços estão procedendo de forma correta.

Enquanto que a técnica de teste de caixa preta também definida pelo autor citado acima, sendo uma técnica realizada nas interfaces dos programas testados. Apesar de ser uma estratégia com objetivo de descobrir erros. Ela é utilizada também para demonstrar que as funções são operacionais; que a entrada e saída são feitas de maneira correta; que a integridade das informações exteriores ao *software* é mantida.

2.3 Tipos de teste de *software*

São encontradas na literatura diferentes definição para os tipos de teste de *software*.

Este trabalho segue a definição de Delamaro, Maldonado e Jino (2007) que resume o conjunto de testes, declarando que a atividade de teste pode ser divida em fases, cada uma com um objetivo distinto. A atividade é dividida em três fases, sendo elas: teste de unidade, teste de integração e o teste de sistema.

Delamaro, Maldonado e Jino (2007) definem o alvo do teste de unidade sendo as menores partes de um programa, sendo funções, procedimentos, métodos ou classes.

De acordo com Delamaro, Maldonado e Jino (2007) no teste de unidade é esperado que os erros encontrados sejam relacionados a algoritmos incorretos ou mal construídos, estruturas de dados incorretas, ou simples erros de programação. Como o teste de unidade é executado com cada unidade em separado, ele pode ser aplicado ao longo da implementação do *software* e não somente quando o sistema estiver completo.

Pfleeger (2004) elucida o teste de integração, sendo uma verificação se os componentes (comerciais, reusáveis, adaptados a determinado sistema ou novos) tem seu funcionamento em conjunto, se são chamados de modo correto e se transferem dados corretamente no tempo correto através de suas interfaces. Para se executar o teste de integração e necessário, que o teste de unidade tenha sido finalizado.

Já o teste de sistema tem a necessidade dos outros testes anteriores terem sido concluídos. Delamaro, Maldonado e Jino (2007) declaram que ele tem o objetivo de

verificar se as funcionalidades dos aplicativos estão implementadas corretamente, levando em consideração o documento de requisitos.

2.4 Ferramentas de teste automatizado

Segundo Cheque e Kon (2008) o teste automatizado de *software* são ferramentas ou *scripts* que tem como meta auxiliar os testadores a exercer sua tarefa, essas ferramentas são altamente úteis, possibilitando que o profissional possa repetir o procedimento de teste de forma fácil e ágil.

De acordo com Delamaro, Maldonado e Jino (2007) as ferramentas que apoiam o teste estrutural, possibilitam que seja feita uma análise de cobertura de um conjunto de casos de teste segundo algum critério selecionado.

O framework JUnit é uma ferramenta automatizada de teste de *software* para teste estrutural, de acordo com Sommerville (2007) ele é um grupo de classes na linguagem Java, que podem ser estendidas e então criar um ambiente de testes automatizados. Esta ferramenta é um gerenciador de testes. Delamaro, Maldonado e Jino (2007) ratificam que o JUnit tem aumentado muito a sua quantidade de usuários. Os autores ainda afirmam que o *framework* tem a possibilidade de escrever e executar automaticamente um conjunto de teste, fornecendo ainda um relatório da forma que cada caso de teste, que não teve seu comportamento de acordo com o especificado, se comportou.

A JaBUTi é outra ferramenta de apoio ao teste estrutural. Segundo Delamaro, Maldonado e Jino (2007) a diferença dela com as outras ferramentas de teste é que esta não necessita do código fonte do programa para que sejam realizados os testes. Basicamente, se uma linguagem de programação produz um código objeto compatível com as especificações da máquina virtual java, pode ser testado por esta ferramenta.

A JeasyTest é um *plugin* criado para simplificar o teste de unidade, difícil de se testar usando a técnica de mutação, como por exemplo usando o código legado.

3. Metodologia

A pesquisa tem seu fundamento teórico baseado em autores reconhecidos como Sommerville, Pfleeger e Pressman. Segundo os autores, o teste de *software* é uma fase muito importante do processo de desenvolvimento de *software*.

A pesquisa será empírica, explicativa, com estudo de campo, laboratorial e quantitativa, baseada em dados de natureza secundária, pois não foi encontrado nenhum trabalho com os dados iguais aos que serão encontrados com essa pesquisa. O intuito da pesquisa é fazer uma avaliação das ferramentas escolhidas de teste de *software*.

Como maneira de analisar o objeto de estudo, a pesquisa insere-se sob um enfoque multidisciplinar, ou seja, é um composto unido de áreas diferentes que se juntam para um propósito único. Estando na área da engenharia de *software* e na subárea de teste de *software*, mas utilizando de lógica de programação, linguagem de programação e estatística para alcançar os objetivos.

É fundamental ressaltar que o teste de *software* é uma fase crítica no desenvolvimento de *software*, portanto, é de vital importância para as pequenas e médias empresas aperfeiçoar essa fase. O trabalho tem o intuito de mostrar qual das ferramentas escolhidas teve o melhor comportamento em comparação as outras, ou seja, definir de forma geral qual teve o melhor desempenho.

Alguns procedimentos específicos deverão ser adotados: escolha das ferramentas a serem testadas, avaliação das mesmas através de um modelo de componente bem definido e comparação dos dados coletados por meio dos testes.

O universo para obtenção de dados são as muitas ferramentas de teste de *software* disponíveis no mercado. A população são as ferramentas de teste de *software* com enfoque em teste estrutural. Já a amostragem são as ferramentas Junit, JaBUTi e JEasyTest.

Um *checklist* será proposta para executar os testes das ferramentas, e assim de forma padronizada os dados para a avaliação serão coletados. O checklist ou lista de checagem será construído para padronizar os testes executados nos aplicativos de testes, fazendo assim, com que todos os passos sejam cumpridos.

A análise e interpretação dos dados será feita utilizando a estatística descritiva para a comparação entre os dados obtidos de cada ferramenta pertencente ao estudo.

4. Conclusão

O atual trabalho está em andamento e ao ser findado, o mesmo auxiliará em uma solução para a seguinte problemática: o alto custo do teste de *software*. A solução será apresentada especificando dentre as ferramentas avaliadas a ou as mais eficazes para a realização do processo, possibilitando assim, que micro e pequenas empresas produtoras de *software* executem esta etapa nos seus processos de desenvolvimento de forma satisfatória e lucrativa.

Perante a avaliação de cada ferramenta em separado, então será realizado um comparativo entre elas para determinar qual teve maior sucesso em relação as outras levando em consideração os testes realizados.

De forma a atingir essa meta, há que se cumprir, especificamente, as seguintes etapas: pesquisar por diferentes ferramentas que auxiliem no teste estrutural de *software*, para que então possa ser feita a análise dos pontos positivos e negativos das ferramentas pesquisadas e então por fim fazer um comparativo entre elas.

5. Referências

ASSOCIAÇÃO BRASILEIRA DAS EMPRESAS DE SOFTWARE. (2010) "Mercado Brasileiro de software: panorama e tendências". São Paulo.

BERNARDO, P. C.; KON, F. (2008) "A Importância dos testes automatizados – Controle ágil, rápido e confiável de qualidade". Artigo publicado na Engenharia de Software Magazine. Rio de Janeiro, p. 54-57, 03 jul.

DELAMARO, M. E.; MALDONADO, J. C; JINO, M (org.). (2007) "Introdução ao teste de software". Rio de Janeiro: Elsevier.

PFLEEGER, S. L. (2004) "Engenharia de software". 2. ed. Tradução de Dino Franklin. São Paulo: Prentice Hall.

PRESSMAN, R. S. (2006) "Engenharia de software". São Paulo: McGraw-Hill.

SANTOS, I. S.; NETO, Pedro A. S.; RESENDE, Rodolfo S. Ferreira de; PÁDUA, Clarindo I. P. da Silva. (2010) "Requisitos e aspectos técnicos desejados em ferramentas de testes de software: um estudo a partir do uso do SQFD. Revista Eletrônica de Sistemas de Informação", v. 9, n. 2, artigo 10.

SOMMERVILLE, I. (2007) “Engenharia de software”. 8. ed. Tradução de Selma Shin Shimizu Melnikoff, Reginaldo Arakaki, Edilson de Andrade Barbosa. São Paulo:Pearson Addison-Wesley.

VELOSO, J. de S.; NETO, Pedro de A. dos S.; SANTOS, Ismayle de S.; BRITTO, Ricardo de S. (2010) “Avaliação de ferramentas de apoio ao teste de sistemas de informação”. Revista Brasileira de Sistemas de Informação, Vol. 3.