8ª MOSTRA ACADÊMICA UNIMEP
26 a 28/10 de 2010
Tema: Desafios da Educação Superior na Agenda do Novo Milênio

18º Congresso de Iniciação Científica

IMPLEMENTAÇÃO DE UM MODELO DE TESTE DE APLICAÇÕES WEB

Autor(es)

HARLEI MIGUEL DE ARRUDA LEITE

Orientador(es)

PLÍNIO ROBERTO SOUZA VILELA

Apoio Financeiro

PIBIC/CNPQ

1. Introdução

Durante as três primeiras décadas da era da computação, a principal preocupação era desenvolver hardware de computador que reduzisse os custos de processamento e armazenamento de dados (PRESSMAN, 2006). Com o avanço das tecnologias de desenvolvimento de hardware, os preços passaram a ser reduzidos e as capacidades, aumentadas. Com isso, o fator restritivo passaria a ser o software.

Uma das metas da Engenharia de Software é produzir software de alta qualidade (PRESSMAN, 2006). Teste de software é uma das atividades de garantia de qualidade de software e também a chave para aprimorar a produtividade e a confiabilidade do software (CHUSHO, 1987) (CHAIM, 1991), da parte de seus usuários.

Segundo Pressman (2006) *"Qualidade de software pode ser definida como adequação a: i) requisitos funcionais e de desempenho explicitamente estabelecidos; ii) padrões de desenvolvimento explicitamente documentados; e iii) características implícitas que são esperadas de todo software desenvolvido profissionalmente."*

O processo de desenvolvimento de software pode ser dividido em três fases (PRESSMAN, 2006): definição, desenvolvimento e manutenção. Na fase de definição, os requisitos do sistema e do software são identificados. Na fase de desenvolvimento definem-se como os requisitos serão satisfeitos; nela ocorrem três passos distintos: projeto de software, codificação e teste do software. E a fase de manutenção concentra-se nas modificações feitas sobre o software; envolve a reaplicação das fases de definição e desenvolvimento, com o objetivo de acomodar os pedidos de modificação.

Segundo Myers (1979), teste é o processo de executar um programa ou sistema com a finalidade de encontrar defeitos. Vilela (1998) apresenta uma definição alternativa: *"Teste de software é o processo de se experimentar ou avaliar um sistema por meios manuais ou automáticos, de modo a verificar se ele atende às necessidades especificadas, ou de modo a identificar as diferenças entre os resultados esperados e os reais"*.

Assim como todas as etapas do ciclo de vida do software, a atividade de teste requer planejamento e sistematização em sua execução. Uma estratégia de teste estabelece uma sequência de passos a serem seguidos, nos quais técnicas específicas de teste são definidas e aplicadas. A atividade de teste pode ser dividida em quatro fases distintas (PRESSMAN, 2006) (VILELA, 1998) (CHAIM, 1991): i) Teste de Unidade; ii) Teste de Integração; iii) Teste de Validação; e iv) Teste de Sistema. Teste de Unidade visa identificar defeitos de lógica e implementação na menor unidade do projeto de software: o módulo. Teste de Integração visa identificar defeitos na interação entre os módulos e constitui uma técnica para integrar os módulos. Teste de Validação encerra a etapa de integração dos módulos e visa testar o programa como um todo. E o Teste de Sistema visa identificar defeitos de funcionalidade e/ou desempenho do programa

implantado no ambiente de operação.
Além de uma estratégia de teste bem definida, também se torna necessária a escolha de critérios de teste de software a serem aplicados em um determinado software, a fim de procurar defeitos. Atualmente, esses critérios são divididos em dois grupos: Teste Funcional e Teste Estrutural.
Critérios de teste pertencente ao grupo de teste funcional derivam seus casos de teste baseados na especificação do programa, e comumente são aplicados no final do ciclo de vida do software (o que pode levar a alguns problemas, como se verá mais adiante). Os critérios de teste funcional permitem que o testador se aproxime do uso com que o sistema irá ser submetido ao cliente. Exemplos de critérios de teste funcional: Particionamento de Equivalência e Análise de Valor Limite.
Critérios de teste pertencente ao grupo de teste estrutural derivam seus casos de teste baseado no código do programa, e comumente são aplicados durante o processo de desenvolvimento do código, na sua grande maioria, em forma de teste de unidade. Os primeiros critérios de teste estrutural para o teste de unidade utilizados eram baseados unicamente no fluxo de controle de programas, sendo os mais conhecidos: O Critério Todos-Nós, o Critério Todos-Arcos e o Critério Todos-Caminhos. O Critério Todos-Nós requer que todos os comandos sejam exercitados pelo menos uma vez; o Critério Todos-Arcos requer que toda transferência de controle entre blocos de comandos seja exercitada pelo menos uma vez; e, por sua vez, o Critério Todos-Caminhos requer que todos os caminhos possíveis do programa sejam exercitados (VILELA, 1998) (HOWDEN, 1975) (KING, 1976).
Também dentro do grupo de teste estrutural, surgiram os critérios baseados em análise de fluxo de dados do programa. Esses critérios usam informações de fluxo de dados do programa para derivar os requisitos de teste; em geral, classificam as ocorrências de variáveis em um programa como sendo uma definição ou como um uso (VILELA, 1998) (CHAIM, 1991) (MALDONADO, 1991).
À medida que os sistemas de software cresceram em tamanho e complexidade, o esforço requerido para testar tais sistemas tem crescido muito além das expectativas, implicando altos custos e produtos com baixa confiabilidade (CHAIM, 1991). Essa situação fez com que fosse necessária a criação de ferramentas automáticas para auxiliar na produção de testes efetivos e na análise dos resultados dos testes. Segundo Chaim (1991) *"O uso de uma ferramenta para auxílio ao teste de programas pode ser vinculado a um critério de teste de duas maneiras: i) A ferramenta de teste pode utilizar o critério de teste como um guia para a geração de casos de teste que satisfaçam o critério; e ii) Utilizar o critério para a análise de cobertura de um conjunto de casos de teste, isto é, verificar se os casos de teste aplicados preencheram os requisitos de teste do critério".*
O objetivo deste trabalho é descrever o projeto e a implementação de uma ferramenta de suporte a testes de Aplicações Web, denominada WTT – Web Testing Tool. Essa ferramenta está associada a uma estratégia de teste estrutural, conforme discutido anteriormente. A principal motivação para o desenvolvimento da WTT foi a falta de ferramentas livres e comerciais para o auxílio a testes de aplicações web.

2. Objetivos

A complexidade para se testar uma aplicação web é igual ou até mesmo maior em relação a uma aplicação desktop. Um dos motivos dessa grande complexidade é que uma aplicação web irá interagir com diferentes tipos de pessoas, e que deverá ser projetada e construída levando em consideração as características da Internet.
Atualmente, poucos estudos concentram-se diretamente no desenvolvimento e no processo de teste de aplicações web, o que representa um grande risco, principalmente considerando-se o grande número de aplicações Web desenvolvidas atualmente.
Frente a este cenário, uma ferramenta de apoio ao processo de teste de aplicações web foi criada. O objetivo da WTT é proporcionar um ambiente que favorecesse o processo de teste mostrando a cobertura de um determinado caso de teste.
Neste artigo, a WTT será apresentada e suas principais características de projeto discutido.

3. Desenvolvimento

Segundo Chaim (1991) e Maldonado (1991), ferramentas acadêmicas para o processo de teste tendem a ser de difícil uso e são, geralmente, ineficientes em cenários reais, como em empresas de desenvolvimento de software. Como forma de mudar este cenário, a WTT foi desenvolvida levando em consideração características propícias a cenários reais.
Para começar, podemos citar a sua interface simples e limpa, onde encontramos o mínimo de componentes de interação com o usuário, facilitando assim o seu uso, já que grande parte das operações são feitas de forma automática. Além disso, a ferramenta oferece opções para visualizar dados da aplicação e um analisador de aplicações web, que tem a finalidade de encontrar possíveis inconsistências.
O principal problema das rotinas de teste é saber se o programa foi suficientemente testado (PRESSMAN, 2006). A WTT procura

ajudar na resposta dessa pergunta, mostrando qual foi a abrangência dos casos de teste em uma determinada execução dos casos de teste.
A ferramenta WTT de forma alguma participa do processo de execução dos casos de testes, apenas gera uma aplicação Web instrumentada com algumas características que irão, por elas mesmas, gerar arquivos com informações sobre a abrangência dos casos de teste, isto é: através da instrumentação será criada uma versão instrumentada do programa original que mostram quais foram os caminhos percorridos durante a execução de um caso de teste.
Podemos dizer que os principais objetivos da WTT são: i) identificar quais os caminhos de execução da aplicação Web; ii) gerar uma aplicação instrumentada da aplicação Web; e iii) calcular qual foi a abrangência dos casos de teste.
Pelo fato da aplicação ser Web, e existirem muitas linguagens para a sua construção, a WTT foi arquitetada a fim de dar suporte ao processo de teste de aplicações Web escritas somente com HTML e PHP, num primeiro momento. Obviamente, existem diversas linguagens para se escreverem aplicações dessa categoria; para facilitar a adaptação da WTT para trabalhar com linguagens de programação diferentes, partes de seu código foram escritas levando em consideração uma determinada linguagem (no atual momento a linguagem PHP) e partes de seu código foram escritas para funcionarem independentemente de linguagem. Isto significa que para adaptar a WTT a uma nova linguagem devemos somente alterar o código da classe que reconhece as ligações entre as páginas.
A forma com que a WTT carrega as páginas é fundamental para a geração dos arquivos e da aplicação instrumentada, como veremos mais adiante. A teoria que dá sustentação a tais realizações é bastante simples: dado um diretório qualquer, todos os arquivos nele contidos deverão ser carregados e, quando possível, relacionados uns com os outros.
A forma de relacionamento entre uma página da aplicação com outras é representada através de uma lista de adjacências – da mesma forma com que representamos a estrutura de dados de uma árvore não binária. A escolha de tal estrutura de dados foi necessária porque uma aplicação Web pode ser representada por um grafo direcionado – Figura 1. Porém, uma forma mais elegante e de fácil implementação seria a conversão desse grafo direcionado em uma árvore dirigida. Este processo foi possível com a implementação da lista de adjacências.
Com base em tal lista de adjacências, podemos identificar a ligação entre as páginas. Sendo assim, podemos também identificar quais páginas estão isoladas das demais, ou seja, daquelas páginas que não contêm conexão para nenhuma outra e nenhuma outra possui conexão para elas.
Com base nas informações da estrutura de dados discutida anteriormente e com o processo de instrumentação da WTT, podemos gerar um Trace de um caso de teste em particular, isto é, podemos identificar qual foi a cobertura de um dado caso de teste. A Figura 2 mostra algumas telas da WTT com resultados obtidos frente a um caso de teste executado em uma aplicação web real.

4. Resultado e Discussão

A internet vem crescendo cada dia mais, e, com isso, pessoas e empresas têm se tornado mais dependentes dela. Nos dias de hoje, grandes empresas já confiam à internet o armazenamento de arquivos confidenciais, por exemplo, ou até mesmo hospeda seus sistemas críticos dentro dela. Apesar da internet oferecer vários benefícios com tais tipos de serviços, devemos lembrar que ela é uma rede pública e mundial.
Com foco nesse cenário, não podemos esquecer que, da mesma forma que sistemas desktop passam por métricas da engenharia de software, aplicações Web também devem passar. Em contrapartida, é visível que aplicações Web sejam desenvolvidas de forma pouco segura, seja por falta de conhecimentos ou mesmo por falta de métricas e ferramentas da engenharia de software direcionadas para esse tipo de aplicação.
Com base nisso, um estudo sobre a área de engenharia de software, mais especificamente sobre a área teste de software, foi desenvolvido. Através desse estudo, foi identificada uma necessidade de uma ferramenta para o auxílio ao processo de teste de software de aplicações Web, já que atualmente não existem ferramentas eficientes para tal processo. Essa ferramenta, denominada WTT – Web Testing Tool – foi especificada e desenvolvida com a finalidade de ser de fácil utilização e eficiente para o auxílio ao processo de teste de software.
O fato de aplicações Web serem escritas em diversas linguagens nos levou a adotar como padrão da ferramenta WTT o auxílio ao processo de teste de aplicações Web escritas em PHP. Porém, a WTT foi arquitetada de forma a possibilitar futuras modificações para o suporte a novas linguagens Web.

5. Considerações Finais

Desde o início da pesquisa, o objetivo principal foi entender o processo de teste de aplicações web e identificar como construir uma

ferramenta que auxiliasse no processo de teste de aplicações web.
A necessidade de criar uma ferramenta de aplicações web veio do fato de não existirem muitas ferramentas com esta finalidade, e também pelo fato das que existem não possuírem funções que encontrem defeitos automaticamente. Na maior parte dos casos, essas ferramentas somente geram informações sobre a aplicação web.
Outra necessidade verificada foi encontrar uma ferramenta de código aberto que possibilitasse a adaptação para novas linguagens de forma fácil. Até o final deste projeto, nenhuma ferramenta de teste de aplicações web com esta característica foi encontrada.
Esperamos que, com o fruto dessa pesquisa, possamos ajudar desenvolvedores e testadores de aplicações web, e quem sabe, em um futuro próximo, incrementar a ferramenta para a inclusão de novas funcionalidades.

Referências Bibliográficas

M. L. Chaim. **Poke-Tool – Uma ferramenta para suporte ao teste estrutural de programas baseado em análise de fluxo de dados**. Dissertação de Mestrado, DCA/FEE/UNICAMP, Campinas – SP, Brasil, Abril de 1991.

T. Chusho. **Test Data Selection and Quality Estimation based on the Concept of Essential Branches for Path Testing**. IEEE Trans. Software Eng. V. 13, N° 5, Maio 1987, p. 509-517.

W. E. Howden. **Methodology for the generation of program test data**. IEEE Trans, 1975.

J. C. King. **Simbolic execution and program testing**. Comm. Of ACM, 1976.

J. C. Maldonado. **Critérios Potenciais Usos: Uma contribuição ao teste estrutural de software. Tese de doutorado.** DCA-FEE-UNICAMP, Campinas – SP, Brasil, Julho de 1991.
G. Myers. **The Art of Software Testing**. Wiley, New York, 1979.

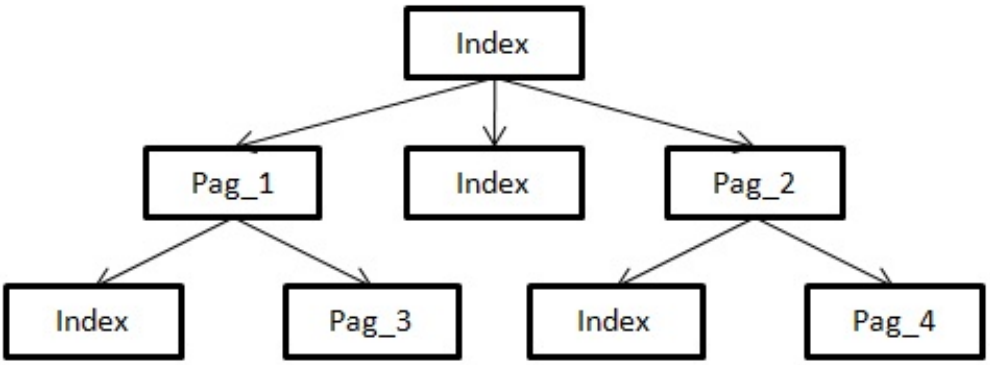
R. S. Pressman. **Software Engineering: A Practitioner's Approach** . Mc GRAW-HILL, 6° Edição, 2006.

P. R. S. Vilela. **Critérios Potenciais Usos de Integração: Definição e Análise**. Tese de Doutorado, DCA-FEE-UNICAMP, Abril 1998.

Anexos


WTT - Web Testing Tool
Arquivo Aplicação Instrumentação Detalhes Consistência Help
Trace - Resultado
Número de Acessos gerais
0- colunistas.php
Recebeu 1 acessos
1- contato.php
Recebeu 0 acessos
2- desenvolvimento.php
Recebeu 1 acessos
3- downloads.php
Recebeu 1 acessos
4- forum.php
Recebeu 0 acessos
5- index.php
Recebeu 2 acessos


WTT - Web Testing Tool
Arquivo Aplicação Instrumentação Detalhes Consistência Help
Verificar Consistência
Iniciar Verificação de Consistência
diversos.php possui 0 páginas de entrada e pode ser uma página inconsistente


Index
Pag_1
Index
Pag_2
Index
Pag_3
Index
Pag_4