7ª MOSTRA ACADÊMICA UNIMEP
Tema:
CIÊNCIA, TECNOLOGIA E INOVAÇÃO:
A universidade e a construção do futuro

17º Congresso de Iniciação Científica

ESTRATÉGIAS DE TESTE DE APLICAÇÕES WEB

Autor(es)

HARLEI MIGUEL DE ARRUDA LEITE

Orientador(es)

PLÍNIO ROBERTO SOUZA VILELA

Apoio Financeiro

PIBIC/CNPQ

1. Introdução

Aplicações web vêm crescendo em tamanho e complexidade; com isso é inevitável que surjam questões referentes à qualidade das aplicações web. Engenharia de Software tem a finalidade de construir software de alta qualidade através de métodos, ferramentas e procedimentos. Uma das formas de garantir que uma aplicação possui alta qualidade é aplicando estratégias e critérios de teste de software. . Uma estratégia de teste de software integra métodos de projeto de casos de teste em uma série bem planejada de passos, que resultam na construção bem-sucedida de software (Pressman, 2006).
As técnicas de teste mais utilizadas são as estruturais e funcionais. Técnicas de teste estruturais derivam seus casos de teste levando em consideração a estrutura lógica da aplicação, enquanto que as técnicas de teste funcionais derivam seus casos de teste levando em consideração a função da aplicação, do ponto de vista do usuário, sem considerar a parte lógica da aplicação. Estas técnicas também são conhecidas como técnicas caixa branca (Estrutural) e caixa preta (Funcional) (Pressman, 2006).
Projetos de aplicações web muitas vezes não se beneficiam de uma abordagem rígida dos métodos, ferramentas e procedimentos da Engenharia de Software, seja por causa de cronogramas curtos de desenvolvimento ou até mesmo pelo fato da Engenharia de Software não oferecer abordagens eficientes para aplicações web. Mesmo existindo uma quantidade razoável de pesquisas na área de Engenharia de Software direcionada para aplicações web, levando-se em conta a quantidade de material publicado para sistemas convencionais (as que não são baseadas na web), vemos que aplicações web possuem carência de um tratamento mais rígido da Engenharia de Software.
Uma das formas observadas para contribuir com aplicações de alta qualidade foi propondo duas novas estratégias de aplicações webLevando-se em consideração o estudo das estruturas de uma aplicação web (a forma de relacionamento entre as páginas) foram propostas as estratégias Bottom-Up e Top-Down, que levam em consideração a melhor ordem para testar uma aplicação web.

2. Objetivos

Uma aplicação web pode ser representada através de uma árvore de chamada. Tendo-se uma árvore de chamada, foi possível observar

que os testes podem ser executados começando tanto de baixo (paginas filhos) em direção para o topo (página pai), quanto de cima (página pai) para baixo (páginas filho). Chamamos estas duas estratégias de Bottom-Up e Top-Down.
Para validar estas duas estratégias as mesmas foram aplicadas em uma aplicação web real. A aplicação escolhida foi o Google Calendar da empresa Google, por ter uma versão gratuita que não impõe muitas restrições em relação ao produto pago, e também por não precisarmos analisar o código fonte da aplicação, visto que os critérios de teste aplicados foram de técnicas funcionais e que a aplicação testada possui código fechado.
Vale ressaltar que a aplicação Google Calendar não foi testada por inteiro, pois este não era o objetivo principal. Foram isoladas para teste somente as páginas que pertenciam a função de criar eventos.
O objetivo principal deste artigo é mostrar os pontos fortes e fracos das duas estratégias, levando-se em conta os resultados obtidos com o teste na aplicação Google Calendar.

3. Desenvolvimento

ÁRVORE DE CHAMADA

Uma aplicação web não segue uma regra definida de ligações entre as páginas. Uma página filho localizada na 3° geração por exemplo pode se ligar diretamente com o pai, sem se comunicar com a 2° geração. Esta característica faz com que uma aplicação web possa ser representada em um grafo direcionado.
Como estabelecer uma estratégia de teste que leve em consideração abordagens do tipo (pai – filho e filho-pai) e aplicá-lo em um grafo direcionado se torna bastante complexo, uma solução seria facilitar a aplicação das estratégias utilizando uma árvore de chamada.
Uma árvore de chamada é uma estrutura do tipo árvore, onde a única ligação de um pai e de um filho é com o próximo filho. Como numa aplicação web este fenômeno não acontece, abordamos a seguinte estratégia de construção de uma árvore de chamada:
Toda vez que um filho precise se ligar com seu pai, seu pai se tornará seu filho. Uma forma de exemplificar isso é com a Figura 1. Observe que a situação de exemplo, de uma aplicação fictícia, possui uma página cadastro, que é filho da página índex. Porém a página cadastro também possui uma ligação com a índex. Neste caso, a página índex se tornou pai e filho a página cadastro.

ESTRATÉGIA BOTTOM-UP

A estratégia Bottom-Up ou (de filhos para pai) é eficiente em aplicações de tamanho pequeno a moderado, pois esta estratégia não permite uma boa integração entre páginas. Uma das vantagens desta estratégia é o fácil isolamento de erro por um motivo bem simples: as páginas filhos muitas vezes são menos complexas que a página pai. Mas essa não é uma regra geral para todas as aplicações web; muitas vezes a aplicação pai atua somente como uma conexão com a página filho; neste caso, a estratégia Bottom-Up não é a mais adequada.
Podemos utilizar duas abordagens diferentes na estratégia Bottom-Up. A primeira é a em largura – onde todas as páginas do mesmo nível são testadas antes de subir um nível adiante; e por profundidade – onde se começa o teste pela página filho e logo em seguida se testa a página pai; chegado ao último nível, a índex, se volta para a última geração e começa o mesmo processo novamente.
A abordagem por largura é a mais adequada quando as aplicações não possuem uma alta complexidade, visto que nessas aplicações, muitas vezes para se testar suficientemente uma página, se deve testar também a página pai em conjunto. A partir do momento que se testa todas as páginas de um mesmo nível, e passa para a geração anterior, muitas vezes a correção dos erros que geraram os defeitos incompatibiliza as páginas filhos com as páginas pai. Este erro é muito comum quando se integra as páginas.
Utilizando a abordagem por profundidade, esses defeitos podem ser descobertas com maior antecedência e consertados mais rapidamente. Visto que, a partir do momento que se testou suficientemente uma página, se vai em seguida para a página pai da página testada anteriormente. Esta abordagem foi mais natural quando aplicada no Google Calendar, conforme pode-se ver no estudo de caso mais adiante. A abordagem em profundidade é a mais adequada dentro da estratégia Bottom-Up.
Utilizando a estratégia Bottom-Up com a abordagem por largura na aplicação Google Calendar, os testes no Google Calendar ocorreram de maneira satisfatória, visto que as páginas filhos nesta aplicação possuíam sempre uma complexidade menor que as páginas pai. Porém, um defeito poderia ser descoberto com maior antecedência se a abordagem utilizada fosse por profundidade. Não era possível marcar através da index um compromisso que ocorresse na virada de um dia para outro, por exemplo, das 23 horas a 1 da manhã. Porém, através da página criarEvento que é filho da índex, se podia marcar compromisso em qualquer horário, até mesmo na virada de um dia para outro. Este atraso na identificação do defeito poderia ser evitado utilizando a abordagem por profundidade, visto que logo após ser testada a página criarEvento, a índex seria a próxima a receber os critérios de teste.
Durante a estratégia Bottom-Up com a abordagem por profundidade, os testes fluíram de maneira mais satisfatória, pois é muito mais confortável testar todas as ligações de página possíveis entre a última geração em direção à primeira geração, até porque conseguimos descobrir com maior facilidade erros de integração em relação à abordagem por largura.

ESTRATÉGIA TOP-DOWN

A estratégia Top-Down, (de pai para filhos) é a mais adequada em aplicações de alta complexidade, pois os testes de integração podem ser realizados de forma mais natural. Uma das vantagens desta estratégia é que possíveis defeitos encontrados nas páginas filhos muitas vezes não afetam as páginas pais depois de consertados; ou seja, as páginas pai depois de suficientemente testado dificilmente irão sofrer com uma futura correção das páginas filho – porém esta não é uma regra geral.
Assim como a estratégia Bottom-Up, a estratégia Top-Down pode ser abordada por largura e por profundidade. O processo é semelhante ao Bottom-Up, em ambas as abordagens, porém, a diferença principal é que começa com os pais para depois migrar para os filhos: por largura, todas as páginas de um mesmo nível são testadas antes de se passar para os filhos, e por profundidade se começa pela index e parte em direção ao filhos.
A estratégia Top-Down com a abordagem por largura foi menos eficiente nos testes do Google Calendar. O fato de a aplicação ter as páginas de gerações superiores mais complexas que as páginas de gerações inferiores fez com que se perdesse muito tempo até descobrir o primeiro defeito do programa – marcar compromisso na virada de um dia para outro.
A abordagem por profundidade foi a que mais se adequou à aplicação Google Calendar. O erro da virada de um dia para outro foi descoberto logo no inicio dos testes, além de que testar as integrações entre as páginas se tornou muito mais natural.

4. Resultado e Discussão

Ambas as estratégias atingiram seu objetivo: Encontrar defeitos. Porém, a estratégia Top-Down com a abordagem por profundidade foi a que mais se adequou à aplicação, por encontrar num período menor de tempo o defeito da virada de um dia para outro e também por facilitar a integração entre as páginas.
Foi observado que podem ser criadas novas regras para encontrar a melhor ordem com que as páginas devem ser testadas, que não necessariamente precisam ser de pai para filhos ou de filhos para pai. Podem ocorrer casos em que nenhuma das duas estratégias seja adequada de forma eficaz na aplicação, visto que as estratégias Bottom-Up e Top-Down são modelos que devem ser seguidos sempre com a mesma metodologia para todo tipo de aplicação. Um conjunto de regras que visem o número de conexões que cada página contém, a complexidade de cada página ou até mesmo um grau de importância de cada página dentro de uma aplicação web pode ser eficiente para identificar uma nova ordem, que seja adequada da melhor forma possível ao cenário da aplicação.

5. Considerações Finais

As aplicações reais vêm tomando proporções cada vez maiores, e o processo de criar as árvores de chamada de forma manual se torna inconveniente e ineficaz. Uma ferramenta que automatize a geração de árvores de chamada e consiga mostrar ao Engenheiro de Software qual a melhor estratégia a ser aplicada se torna necessária.
Também se torna necessária para trabalhos futuros a criação de novas regras que identifiquem estratégias inéditas para cada tipo de aplicação web, com base em suas características, como foi dito anteriormente.

Referências Bibliográficas

Pressman, R. S., Engenhaira de Software. McGRAW-HILL, 2006.

Anexos

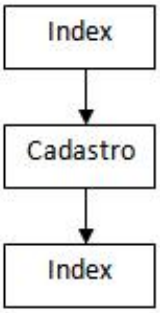

Figura 1 – Árvore de Chamada