

Pitfalls and Strategies in Automated Testing.

Fernando Campos, Nery Signorini Neto

Instituto de Pesquisas Tecnológicas do Estado de São Paulo
Avenida Prof. Almeida Prado, 532. Prédio 56 – São Paulo – SP – Brasil

ferpcampos@gmail.com, nery.signorini@mac.com

***Abstract.** This paper analyses the article “Pitfalls and Strategies in Automated Testing” [Kaner, Cem. April 1997]. This article resumes common problems and possible approaches to conduct software automated tests as well defined and successful process. The article show usual problems during automation tests and reinforces the importance of using a consistent development methodology.*

***Resumo.** Este trabalho apresenta uma resenha do artigo “Pitfalls and Strategies in Automated Testing” [Kaner, Cem. April 1997]. Este artigo procura identificar as armadilhas e estratégias na condução de testes automatizados de software. Aborda principais problemas nos projetos de automação e mostra a importância da utilização de metodologias e de processos consistentes no desenvolvimento dos programas.*

Introdução

Nos últimos anos, diversas ferramentas vêm ajudando os programadores a rapidamente aumentarem a produtividade no desenvolvimento de software. Em contrapartida, aumentou também a pressão para a execução de testes que validem o produto desenvolvido, fazendo mais, testando mais, em menor tempo.

Baseado nesse cenário, a automatização de testes de software é a principal ferramenta para atender essa demanda. Uma ferramenta de automação deve ter como principais atributos: velocidade, eficiência, acuidade, precisão, redução de recursos, poder de simulação e longevidade [Patton 2006].

Armadilhas e Estratégias

A automação de testes pode consumir até 10 vezes mais tempo em etapas como planejar, criar, documentar e executar, em comparação a um teste executado manualmente. Portanto, não automatize testes que você raramente executa (*expensive*¹).

As falhas, por sua vez, tendem a ser descobertas durante a execução dos testes, principalmente nas fases finais do projeto, o que aumentam os custos diretos da

¹ Expensive, delayed bugs, low power tests, automating only easy-run test, Referência ao artigo: Pitfalls and Strategies in automated Testing.

manutenção e correção dos códigos (*delayed bugs*). As estimativas sobre a margem de detecção de erros na automação de testes demonstram que as falhas detectadas variam entre 6% e 30%, como afirma Kaner, Cem, sob o total existente de erros (*low power tests*) [Kaner, Cem 1997]. Em contraste, Patton [Patton 2006] afirma que 60% de falhas são originadas na fase de especificação dos requisitos. Em alguns cenários, como por exemplo, testes complexos e de difícil automação, poderá ser mais eficiente a utilização de um testador habilidoso ao invés da utilização de scripts e códigos (*automating only easy-to-run tests*).

A manutenção dos scripts e casos de testes, baseados em UI (*User Interface*) podem sofrer constantes alterações, sejam nos programas ou telas. Há necessidade de se ter uma estratégia consistente para lidar com essas alterações. Simplesmente re-gravar teclas, cliques de mouses ou telas não pode ser considerada uma estratégia consistente. As ferramentas de captura e gravação (*capture-based and test-case creation*) podem ajudar você a criar os testes, scripts e devem ser utilizadas com precaução. É fundamental ter atenção redobrada a códigos e funções duplicados que dificultarão a manutenção dos casos de teste.

As vantagens da automação não poderão ser obtidas a curto prazo, ou seja, não são imediatas. Há necessidade de que os códigos criados estejam maduros e previamente testados (*bug free*) para serem utilizados, o que normalmente acontece apenas no segundo “release” dos códigos, ex. release N + 1, [Kaner, Cem. 1997].

É uma boa estratégia separar dados e códigos que serão utilizados nos testes, por trazerem independência, facilidade de manutenção e repetição do ciclo de teste. Também deve ser levado em consideração, o grau de maturidade no processo de desenvolvimento de software que inclui: procedimentos, estratégias, requisitos, objetivos, pessoal capacitado, treinado e disciplina na execução. As regras colocadas devem ser seguidas a risca.

Problemas Comuns Encontrados na Automação

Abaixo estão relacionados alguns dos principais problemas encontrados durante o processo de automação de testes.

Falta de Foco na Automação (*Spare time test automation*)

Pessoas envolvidas nas atividades de automação dos testes raramente estão devidamente dedicadas ou possuem um cronograma específico para a atividade de planejamento e execução. Essa situação priva o projeto do tempo, recursos e foco necessários para realização das atividades com sucesso.

Falta de Objetivos (*Lack of clear goals*)

Certamente existem várias razões para se automatizar testes. Automatização pode reduzir tempo, facilitar a execução, aumentar a cobertura e motivar o pessoal envolvido. Porém, não é necessário fazer tudo isso ao mesmo tempo. Como há diversas expectativas envolvidas que devem ser satisfeitas, é importante que o teste tenha um foco específico, que deverá ser atingido. O teste deve ter seus objetivos claros e acordados entre todos os envolvidos, entre eles *sponsors*, *stakeholders*, usuários, gerentes de projetos e testadores.

Falta de Experiência (*Lack of experience*)

A experiência dos programadores é fator fundamental para a qualidade nos códigos gerados e afeta diretamente a precisão na detecção de erros ou falhas. Programadores inexperientes testam seus próprios limites, indo além de suas reais habilidades, comprometendo o desempenho da atividade confiada.

Alta Rotatividade (*High turnover*)

Os recursos demoram para adquirir maturidade e experiência necessários ao processo de automação de testes, logo, reter recursos capacitados é fator fundamental para trazer qualidade ao produto que será desenvolvido e executado.

Reação frente ao Desespero (*Reaction to desperation*)

Os problemas já existem no software que será testado e os testes apenas os evidenciam [Pettichord, Bret 2001]. O processo de teste, correção e re-teste pode gerar uma situação de desespero entre a equipe de teste e de desenvolvimento, especialmente se houver um cronograma com datas agressivas e que não podem ser renegociadas. Nessa situação, a automação pode ser o caminho correto, mas certamente não será a melhor opção, podendo ser apenas um desejo do que uma proposta realística.

Pensando nos Testes (*Reluctance to think about testing*)

Automatizar testes pode ser mais excitante do que o realizar o teste propriamente dito. Pessoas se envolvem mais no processo de construção do teste *test-case* do que na sua execução, análise, comparação e documentação dos resultados.

Foco na Tecnologia (*Technology focus*)

A escolha da tecnologia utilizada para automação é um interessante problema que cativa técnicos e gera grandes discussões e argumentações. O problema aqui é a perda do foco original do teste e de seus reais objetivos.

Siga as Regras para Desenvolvimento de Software

O desenvolvimento de software para automação de testes deve ser encarado como um desenvolvimento formal, como em qualquer outro projeto, com começo, meio e fim. Os desenvolvedores devem estar dedicados à atividade de codificação. O programador que não possuir experiência no desenvolvimento de software tão específico e detalhado, deverá, necessariamente, consultar os programadores mais experientes. Como todo projeto de desenvolvimento, os códigos devem ser guardados, documentados e deve ser instituído um controle de versão. Certamente os códigos gerados conterão falhas (*bugs*) que obviamente deverão ser eliminados antes da execução dos testes.

Abaixo estão relacionados 7 passos para se obter sucesso na automação de teste de software.

Passo 1: Melhore o Processo de Testes (*Improve the testing process*)

Melhore seu processo de teste. Um processo bem definido e bem implementado facilita a utilização, construção de códigos e aumenta a probabilidade de sucesso. Treine todas as pessoas para que entendam o processo, torne a abordagem do teste claro, documente bem os requisitos, objetivos e detalhes do teste, como características do *design* e resultados esperados.

Garantindo uma documentação básica, você garantirá também que qualquer pessoa, com o mínimo de conhecimento, poderá entender, executar e até alterar os códigos criados para o teste. Outro importante aspecto é o desempenho do teste. Se há lentidão durante a execução, registre a situação e indique o erro para correção. De maneira geral, você aumentará a eficiência do processo de teste aumentando a qualidade do produto gerado.

Passo 2: Defina os Requisitos (*Define requirements*)

Há diferentes objetivos entre testadores e patrocinadores (*automators and sponsors*). Para evitar essa situação é necessário que todos os requisitos definidos para a automação do teste estejam aprovados por todos os envolvidos e responsáveis. Ex. como a arquitetura a ser utilizada, os detalhes do processo de automação e quais e quando os objetivos deverão ser atingidos. Não é necessário automatizar cada parte do teste, procure por oportunidades onde o retorno (*payback*²) é maior em relação ao custo de automação.

Razões para automatizar os testes:

- Aumento da velocidade de construção de testes, acelerando novas versões.
- Permitir que os testes sejam executados mais freqüentes.
- Reduzir custos de construção de testes, reduzindo o custo do trabalho manual (relação de custo homem/hora).
- Aumento da cobertura do teste.
- Garantir consistência no teste.
- Aumento da eficiência do teste.
- Permitir que o teste seja executado por pessoas mais inexperientes.
- Definir o processo de teste, diminuindo assim a dependências de pessoas e do conhecimento entre elas.
- Tornar o teste mais interessante.
- Desenvolver as habilidades dos programadores

Passo 3: Prova de Conceito (*Prove the concept*)

Teste a eficiência de suas ferramentas de automação *test-suite* o quanto antes em modelos ou protótipos. A validação prévia das ferramentas aumentará as chances de sucesso do teste.

² Relação custo benefício em automatizar o teste daquela circunstância em particular.

Passo 4: Sucesso do Produto (*Champion product testability*)

Busque uma abordagem correta de automação para cada tipo de particularidade encontrada durante o processo de entendimento do domínio do problema. Cada problema identificado tem uma estratégia específica e poderá ser necessária a utilização de diversas estratégias e abordagens para se criar um caso de teste eficiente.

Passo 5: Longevidade do Teste (*Design for sustainability*)

Sucesso em automação de software requer uma visão a longo prazo que inclui:

Performance: Aumento do desempenho, que geralmente inclui o aumento da complexidade do código utilizado na automação.

Ease of analysis: Instrumente seu código para registrar todas as passagens, mensagens ou códigos de erros, que serão eventualmente necessários para a análise dos resultados e da comprovação da existência de uma falha (*bug*).

Reviewability: Tenha uma boa documentação que sustente seu caso de teste e programas utilizados. Faça uma análise detalhada da cobertura dos testes.

Maintainability: Facilite a manutenção dos códigos utilizados no processo de automação, instrumente os códigos, utilize bibliotecas comuns para mensagens e comandos por mais de um programa.

Integrity: Integridade do teste traz credibilidade para o processo de automação do teste e eleva a moral dos envolvidos. Os relatórios gerados durante a execução do caso de teste devem ser objetivos e conclusivos e falhas devem ser evidenciadas.

Independence: É imprescindível que os testes tenham independência de execução, podendo ser executados de forma sequencial ou em paralelo, conforme a necessidade.

Repeatability: Capacidade de re-execução dos testes.

Libraries: Utilização de bibliotecas de funções e comandos facilita a manutenção dos códigos.

Data-Driven Tests: Semelhantes às bibliotecas, são programas que interpretam comandos e dados que serão utilizados no teste.

Heuristic Verification: Verificação manual do registro de execução do teste gerado pelos programas. É a análise do resultado e da efetivação da falha.

Passo 6: Plano de Implementação (*Plan for Deployment*)

Pense que seu *test-suite* é um produto verdadeiro (e realmente é). Você tem que testá-lo. Garanta que ele não depende de bibliotecas ou serviços especiais instalados apenas em seu computador. Treinamentos devem ser realizados e documentações devem estar atualizadas e disponíveis para todos os integrantes da equipe. Finalize o *test-suite* e utilize-o. Encerre o ciclo.

Passo 7: Enfrente os Desafios do Sucesso (*Face the Challenges of Success*)

Seu *test-suite* está pronto, testado e documentado e todas as pessoas envolvidas entendem os testes, como eles funcionam, como são executados e seus objetivos. Os erros e falhas ficarão mais resistentes a medida que você avançar em seus testes. A partir de então, surgirá um novo cenário, ainda mais desafiador.

Conclusão

Testes apenas mostram os erros nos programas, mas não garantem que todos os erros sejam encontrados [Sommerville, 2007]. É fator crítico de sucesso saber discernir qual tipo de estratégia de automação será adotada, para cada situação encontrada. A utilização de um método formal de levantamento de requisitos e de desenvolvimento de software ajudará a aumentar a eficácia do produto final, melhorando as chances de realização dos objetivos estipulados.

Referências Bibliográficas

Pittichord, Bret (2001), “Seven Steps to Test Automation Success” Presented at Quality Week (May) <http://www.io.com/~wazmo/succpap.htm>.

Pittichord, Bret (2001), “Success with Test Automation” Presented at Quality Week (May) <http://www.io.com/%7Ewazmo/succpap.htm>.

Kaner, Cem. (1997), “Improving the Maintainability of Automated Test Suites” Presented at Quality Week (May) <http://www.kaner.com/lawst1.htm>.

Kaner, Cem. (1997) "Pitfalls and strategies in automated testing." *IEEE Computer*, April, 1997, p. 114.

Patton, Ron (2006), “Software Testing 2nd Edition”, Editora Sams Publishing.

Meyers, Glenford J. (2004) "The Art of Software Testing, 2nd Edition", Editora John Wiley & Sons, Inc..

Sommerville, I. (2007) "Software Engineering 8", Editora Addison Wesley.