Estudo sobre a Utilização de Testes Automatizados em Projeto de Software de Grande Porte

Wisney Cardeal [1]; Walteno Martins Parreira Júnior [2]

[1] Aluno do curso de Pós-Graduação em Análise e Desenvolvimento de Sistemas Aplicados à Gestão Empresarial; IFTM - Instituto Federal de Educação, Ciência e Tecnologia do Triangulo Mineiro – Campus Uberlândia Centro; Uberlândia, MG; wisneycardeal@gmail.com; [2] Mestre. Professor; IFTM – Campus Uberlândia Centro; waltenomartins@iftm.edu.br.

RESUMO: Garantia de qualidade, aderência aos requisitos e diminuição do retrabalho em correções do código são desafios vividos na maior parte dos projetos de *software*. A implantação de processos de automação de testes é cada vez maior na busca de eficiência e redução do custo do projeto e melhoria de produtividade. Contudo, o sucesso na utilização de ferramentas de automação de testes, depende das empresas possuírem um grau mínimo de maturidade no processo de testes, responsabilidades bem definidas e profissionais qualificados e com perfil adequado para planejamento e execução de testes. Este estudo apresenta os principais impactos e resultados obtidos com a utilização de testes automatizados em uma empresa de mercado em um projeto de *software* comercial de grande porte, com base em um estudo de caso realizado durante o andamento do projeto.

Palavras-chave: Automação de Testes. Testes Unitários. Testes Processo. *PHPUnit*. *CasperJS*.

INTRODUÇÃO

No processo de desenvolvimento de *software* existem diversas dificuldades, a garantia da aderência e atendimento a todos requisitos dos sistemas são algumas das principais, devido, principalmente, à alta complexidade dos produtos, pois além dos requisitos também é necessário que o *software* possua bom desempenho, segurança, eficiência, flexibilidade, escalabilidade e ainda atenda as necessidades e expectativas do cliente, além de todas estas características é preciso ainda que o *software* possua fácil entendimento e manutenção (PRESSMAN, 2010).

Para assegurar estas características é bastante comum a utilização de testes manuais dos sistemas. Antes de colocar o sistema em produção, é natural que este seja submetido a um processo de avaliação de qualidade, geralmente realizado através de testes manuais executados por outros desenvolvedores, por equipes especializadas em testes e também por usuários do sistema, com o objetivo de verificar possíveis falhas em relação aos requisitos iniciais (CRISPIN; HOUSE, 2003), contudo, a cada mudança significativa do código, nem sempre são executados todos os casos de testes, devido ao tempo necessário para execução dos mesmos e à falsa confiança de que, o que já estava funcionando não foi afetado. Assim, surgem alguns dos principais erros de *software*.

Conforme Beck & Andres (2004) muitos métodos ágeis recomendam que o controle de qualidade e aderência do produto seja realizado a todo momento durante todo o processo de desenvolvimento. A programação extrema (*XP*) recomenda explicitamente a utilização de testes automatizados para ajudar a garantir a qualidade dos sistemas e a reexecução do maior número de conjuntos e cenários de testes em menor tempo e menor custo.

Testes automatizados é uma técnica para programar a execução dos testes utilizando ferramentas ou *frameworks* específicos para testes. A automação dos testes é realizada através de programas ou *scripts* simples que simulam as tarefas manuais executando as funcionalidades do *software* e verificando se os resultados obtidos são os resultados esperados, tendo como principal benefício, a redução de tempo do ciclo de execução de testes e também o aumento da abrangência dos mesmos, dispondo ainda da possibilidade de todos os casos de teste serem fácil e rapidamente repetidos a qualquer momento e com menor esforço (BECK; ANDRES; 2004).

Com a automatização dos testes também é possível criar cenários bem mais elaborados e complexos do que os cenários de testes executados manualmente, permitindo também expansão da magnitude dos casos de testes, como por exemplo, simular milhares de usuários simultaneamente utilizando o sistema ou manipulação de milhares de registros na base de dados. Assim, mudanças significativas no sistema são suportadas com maior segurança, o que contribui diretamente para uma maior vida útil do produto (BERNARDO; KON, 2008).

Na grande maioria das vezes, os testes automatizados são escritos programaticamente utilizando ferramentas específicas para automatização de testes, por isso é necessário conhecimento de programação. Contudo, a implantação da automação de testes depende de testes manuais maduros e consistentes (CAETANO, 2008).

De acordo com Bernardo & Kon (2008) no passado para criar testes automáticos era comum a criação de uma função de teste em cada módulo ou classe do sistema que continha algumas simulações de uso da unidade. O problema desta prática está na falta de organização, já que o código adicional era misturado ao próprio sistema afetando a legibilidade do código. Por isso, surgiram os *frameworks* que auxiliam e padronizam a escrita de testes automatizados, facilitando o isolamento do código de teste do código da aplicação. Estes *frameworks* de testes de unidade são conhecidos como parte da família de *frameworks xUnit*[1]. Por exemplo, existe o *framework* pioneiro *SUnit*[2] para *SmallTalk*[3] criado por Kent Beck, que foi seguido por implementações para outras linguagens, tais como *JUnit*[4] e *TestNG*[5] para *Java*[6], *JSUnit*[7] para *Javascript*[8], *CppTest*[9] e *CppUnit*[10] para *C++*[11], *csUnit*[12] para *.NET*[13], *PHPUnit*[14] para *PHP*[15] entre outras.

Segundo Carvalho *et. al.* (2010) as primeiras ferramentas de testes de *software* surgiram por volta de 1980 e após essas ferramentas ganharem popularidade, surgiram as ferramentas de gestão de testes que tinham como objetivo organizar e guardar dados dos testes, organizarem o resultado da execução dos testes e apresentar relatórios de testes. Na década de 1990 surgiram as primeiras ferramentas para automatização de casos de testes. Estas eram inicialmente focadas em medições básicas de algumas plataformas.

Ainda de acordo com Carvalho *et. al.* (2010) as ferramentas de automatização atuais são de fácil utilização e permitem realizar um maior número de funções em relação às ferramentas desenvolvidas inicialmente. Este fato, deve-se à automatização de testes de *software* ser vista como uma das principais medidas para melhorar a eficiência da atividade de teste, agregar qualidade ao produto e antecipar a descoberta de falhas e incompatibilidades, reduzindo assim o custo do projeto e melhorando a produtividade a vários níveis.

Assim, várias soluções para automatização de testes têm surgido no mercado, como: *Selenium IDE*[16], *iMacros*[17], *Watir*[18], *TestMaker*[19], *Quick Test Pro*[20], *LoadRunner*[21], *PhatomJS*[22], *CasperJS*[23] entre outras.

É importante frisar que o presente estudo não tem por objetivo analisar ou descrever ferramentas de testes ou técnicas de automação de testes. Busca apenas apresentar os principais impactos, esforços, benefícios e resultados obtidos através da aplicação de testes automatizados em uma empresa ativa no mercado em um projeto de *software* comercial de grande porte, com base em um estudo de caso realizado na empresa através de entrevistas e análise de processos e artefatos produzidos, além de pesquisa em materiais especializados, principalmente livros, artigos científicos e sites que abordam a temática a fim de obter uma melhor descrição da atividade além de informações sobre teorias e outras pesquisas já realizadas.

---

[1] http://www.martinfowler.com/bliki/Xunit.html
[2] http://sunit.sourceforge.net/
[3] http://www.smalltalk.org/main/
[4] http://junit.org/
[5] http://testng.org/doc/index.html
[6] http://www.oracle.com/technetwork/java/index.html
[7] http://www.jsunit.net/
[8] https://www.javascript.com/
[9] http://cpptest.sourceforge.net/
[10] http://cppunit.sourceforge.net/doc/cvs/cppunit_cookbook.html
[11] http://www.cplusplus.com/
[12] http://www.csunit.org/
[13] http://www.microsoft.com/net
[14] https://phpunit.de/
[15] https://secure.php.net/
[16] http://www.seleniumhq.org/
[17] http://imacros.net/overview
[18] http://watir.com/
[19] http://www.epsteineducation.com/home/testmaker/
[20] http://www.automation-consultants.com/index.php/products/hp-products/hp-unified-functional-testing-quick-test-professional
[21] http://www8.hp.com/us/en/software-solutions/loadrunner-load-testing/
[22] http://phantomjs.org/
[23] http://casperjs.org/

MATERIAIS E MÉTODOS

Caracterização da pesquisa

Este trabalho de pesquisa foi realizado dentro do ambiente corporativo da empresa Softbox Sistemas de Informação, situada na cidade de Uberlândia/MG, sendo uma *software house* especializada em consultoria, realização de projetos de TI e desenvolvimento de soluções e estratégias para mobilidade e ambientes corporativos (SOFTBOX, 2015). Por se tratar de um contexto específico que está sendo analisado em profundidade, e também por consistir na análise da utilização e aplicação da automação de testes em projetos da empresa, a pesquisa caracteriza-se como um estudo de caso descritivo (YIN, 2005).

Procedimentos adotados

Este estudo foi desenvolvido através de entrevistas com a equipe de qualidade e gestores, além da análise dos processos e aplicações de testes automatizados, dentro do seu contexto de vida real, além dos artefatos utilizados na especificação, concepção, elaboração, desenvolvimento e principalmente na garantia de qualidade do projeto de um sistema ERP Web para um grande cliente varejista de eletrônicos e eletrodomésticos, presente no mercado nacional, através de lojas físicas e portais de e-commerce que oferecem mais de 50.000 produtos e entregas para todo o Brasil.

Neste estudo foi possível analisar uma ampla variedade de evidências como documentos, artefatos, opiniões e observações dentro do ambiente corporativo.

O desenvolvimento deste trabalho tomou como base os resultados obtidos pela equipe de qualidade de *software* da Softbox na aplicação da automação de testes durante todo o desenvolvimento do sistema até sua implantação e sustentação contínua até o início do mês de Outubro de 2015.

O estudo de caso

O objeto de estudo foi a aplicação de testes automatizados em um projeto de *software* de um novo sistema ERP em plataforma WEB. O projeto contemplou desde a modelagem de negócio até a implantação, manutenção e sustentação do produto. O projeto foi estimado inicialmente em 56.000 horas de trabalho. Do início do projeto *(KickOff)* até a implantação *(GoLive)* do sistema, foram pouco mais de 2 anos, o projeto continua em andamento com a sustentação e melhoria contínua do produto.

Conforme especificação do cliente, para desenvolvimento do *software* deveria se utilizar a linguagem de programação *PHP*[24] e o gerenciador de banco de dados *MySQL*[25] e a qualidade do produto deveria ser mantida especificando e desenvolvendo cenários de teste que contemplassem a maior parte possível das funcionalidades do sistema, não sendo explicitado o tipo de testes a serem executados, se manuais, ou se automatizados.

Devido à experiência de mercado e também experiências de projetos anteriores a Softbox incorporou ao seu processo de desenvolvimento a implementação de testes automatizados.

Para a automação dos testes a empresa buscou ferramentas *open source* que dispusessem de linguagem de fácil manutenção, baixa curva de aprendizagem, alta curva de produtividade, execução de testes mais rápidos com resposta de sanidade do código confiável. Devido à linguagem de programação utilizada no desenvolvimento do *software* ser o *PHP* a empresa optou por utilizar como ferramenta de automação de testes o *PHPUnit* para automatização dos testes unitários.

O *PHPUnit* é um *framework* de testes direcionado a programadores, para a linguagem *PHP*. É uma instância da arquitetura *xUnit* para *frameworks* de testes unitários. Por ser baseado no *xUnit*, segue seu formato, tornando mais simples a criação de códigos automatizados para testes de unidade (PHPUNIT, 2015; ROCHA, 2013).

O teste unitário ou teste de unidade foca nas unidades do projeto de *software*, procurando identificar possíveis falhas de lógica ou de implementação em cada um dos módulos do *software*, em separado, unitariamente (PRESSMAN, 2010).

A equipe de qualidade visualizou que havia necessidade de aumentar a cobertura de testes e automatizar, além dos testes unitários, também os testes de processos (união dos testes de integração e testes de sistema), testes mais completos que contemplassem desde a utilização das telas da aplicação (simulando a utilização do usuário no *Front-End*) até o resultado final dos processos.

Os testes de integração, são atividades executadas durante a integração dos módulos do produto, focando a avaliação e verificação de falhas associados às interfaces entre os módulos; com

[24] https://secure.php.net/
[25] https://www.mysql.com/

objetivo de, a partir dos módulos testados unitariamente, testar toda a estrutura de integração do programa especificada no projeto. Os testes de sistemas, são os procedimentos de teste realizados após a integração do sistema, buscando identificar falhas e erros de características e comportamento que não estejam de acordo com as especificações (PRESSMAN, 2010).

Então de acordo com um estudo realizado pelo time de arquitetura presando os mesmos princípios que levaram a escolha do *PHPUnit* e ainda buscando por ferramenta de testes funcionais que simulasse o comportamento do usuário e que suportassem a automação de testes em sistema web, a empresa optou por utilizar o *framework* de testes *CasperJS*, que possibilita a geração de *scripts* de navegação e automação de testes utilizando *JavaScript* e que abstrai o *PhatomJS* que é executado de forma *headless*, ou seja, sem usar a interface gráfica, utilizando em segundo plano o *WebKit*[26], um motor de navegador de código aberto (*Engine* do *Google Chrome*), o que resulta numa diminuição real do tempo de execução dos testes (CASPERJS, 2015).

RESULTADOS E DISCUSSÃO

A Softbox decidiu utilizar testes automatizados em seus projetos, por estratégia de mercado, devido ao histórico de outros grandes projetos, que exigiram um grande esforço com testes manuais a cada mudança crítica no sistema. E pelo fato de que, os analistas de testes, por limitações humanas naturais, nem sempre validavam a execução e resultados obtidos com a mesma eficácia todas as vezes que executava o mesmo teste, principalmente nos testes de regressão.

Com base neste passado, a empresa decidiu investir em testes automatizados para aumentar a aderência das entregas aos requisitos iniciais e garantir os processos de negócios modelados a cada projeto, pois os testes automatizados executam e validam o mesmo teste quantas vezes forem necessárias com a mesma eficiência e não exige esforço para testes de regressão no sistema.

Devido à sua experiência, a Softbox compartilha da opinião que testar as partes separadamente, não garante o todo. Com isso, e também, buscando otimizar o tempo de programação dos testes unitários, criou um *framework* proprietário para programação de testes utilizando o *PHPUnit*. Utilizando planilhas, os analistas de testes escrevem determinada sequência de procedimentos/métodos a serem executados, que representam o cenário de testes como um todo. O *framework*, através da interpretação destas planilhas, realiza a geração automática do código de programação dos casos de testes em conformidade com o *PHPUnit*. Estes casos de testes, ao serem executados, rodam um conjunto de testes que podem representar a funcionalidade de determinada tela ou processo de negócio, realizando os testes unitários, porém, não testando apenas métodos e procedimentos individualmente. Contudo, os testes unitários puros podem ser vistos nos testes do *framework* e *API's* de desenvolvimento utilizados pela empresa. Esta estratégia foi tomada, pois a empresa constatou que a utilização de testes de unidade, única e exclusivamente, demandaria por volta de 2 vezes e meia a mais o tempo de desenvolvimento.

No projeto do ERP, tomado como objeto deste estudo, a Softbox utiliza como insumo do planejamento de testes os documentos de processo de negócio modelados junto ao cliente. E de acordo com a criticidade destes processos e os processos classificados com maior risco ao negócio, são priorizados os desenvolvimentos de casos de testes com roteiros de execução e comportamento que são utilizados tanto para execução de testes manuais quanto para a automatização dos casos de testes.

Com relação ao tempo de esforço comparando a execução de testes manuais e a automatização de testes e posterior utilização dos mesmos, o esforço para automatização é grande no começo, porém, este tempo gasto, vai se diluindo nas inúmeras vezes que estes testes são executados automaticamente ao invés de sempre necessitar de uma pessoa para realização de tal função.

No andamento atual do projeto a equipe de automação de testes utiliza em média 2 horas de esforço para criação de um teste automatizado utilizando o *framework* proprietário sobre o *PHPUnit* e em média 4 horas de trabalho para automatização utilizando o *CasperJS*.

O projeto do ERP possui aproximadamente 60% dos seus processos com casos de teste especificados, e para estes, existem cerca de 250 cenários de testes em *PHPUnit*, considerando que cada cenário possui um mínimo de 10 casos de testes unitários. Existem também cerca de 50 processos completos com testes automatizados em *CasperJS*, o que representa 45% do sistema como um todo.

Os testes automatizados são executados inúmeras vezes pelos desenvolvedores a cada implementação, sempre que necessário algum ajuste ou correção de *bug* ou ainda na disponibilização

[26] https://www.webkit.org/

de novas funcionalidades. São executados, também, pela equipe de qualidade para validação e disponibilização de novas versões do sistema. E ainda assim, conforme possíveis incidentes relatados pelos usuários, até 100% dos testes podem ser executados manualmente, caso seja necessário, para validação de um comportamento ou cenário específico.

Para manutenção e sustentação do projeto a equipe de projetistas e analistas de testes dedicam aproximadamente 4 horas diárias para manutenção das rotinas de testes automatizados e implementação de automatização de novos casos de testes.

A proporção de *bugs* ou ajustes relatados pelos usuários tem queda considerável a medida que é aumentada a cobertura, amplitude e profundidade dos testes automatizados.

Antes das automatizações, eram relatados em média 20 *bugs* diários. Com a implementação dos testes unitários automatizados com o *PHPUnit*, este número caiu para média de 10 *bugs* diários. Com a implementação dos testes de processo automatizados com o *CasperJS* em complemento aos testes unitários em *PHPUnit*, o número médio de *bugs* relatados pelos usuários caiu para cerca de 10 *bugs* por semana.

Contudo, o projeto tem um ganho notório e satisfatório com a aplicação dos testes automatizados, pois tem 0% de esforço para testes de regressão no sistema, tem maior garantia de qualidade nas entregas, garantia na continuidade dos processos de negócios do cliente e tem resposta rápida e assertiva sobre possíveis impactos causados nas mudanças do sistema.

CONSIDERAÇÕES FINAIS

O grande benefício da automação de testes não é apenas a execução dos testes de forma mais rápida e ou a qualquer hora do dia ou da noite, mas sim, o aumento da complexidade dos casos de testes além da maior amplitude e profundidade da cobertura dos testes.

Na prática, a automação de 100% dos testes manuais nem sempre é a melhor estratégia. Testes muito complexos ou que exigem interações de integração entre sistemas ou que tem necessidade de validações subjetivas devem permanecer manuais, para uma análise mais criteriosa e cuidadosa, com o objetivo de garantir a qualidade e aderência aos requisitos, contemplando a corretude do sistema e não procurando falhas no mesmo.

Mesmo testes que já possuem automatização, podem, em algum cenário específico, necessitar da execução manual para devida análise de comportamento do sistema.

O ganho do projeto com a aplicação dos testes automatizados é altamente perceptível principalmente na qualidade das entregas dos desenvolvedores e na diminuição de *bugs* relatados pelos usuários.

A execução de testes manuais ou automatizados não garantem que o sistema não tenha falhas, garante apenas que está funcionando e atendendo ao que foi requisitado, porém em algum momento do desenvolvimento, pode ser que fique algum tipo de falha em alguma linha de código e que possivelmente algum dos requisitos não seja atendido com 100% de aderência. Por isso, facilidade e agilidade na execução dos testes em qualquer momento é imprescindível, e os testes automatizados proporcionam esta viabilidade.

Além disto, o aumento da senioridade da equipe, leva à uma melhoria na escrita do código do sistema e também na diminuição da complexidade ciclomática dos métodos, contribuindo para uma melhor qualidade do produto e menor probabilidade de falhas.

A automação de testes dá uma segurança maior à toda equipe principalmente no que tange às correções no código, implementação de novas funcionalidades ou até mesmo refatoração.

A automação de testes deve ser introduzida como uma técnica adicional ao processo de desenvolvimento e de qualidade, na qual o principal objetivo é agregar valor ao serviço prestado e às entregas ao cliente.

AGRADECIMENTOS

À Deus, acima de tudo, agradeço, pela saúde, serenidade e perseverança sobre mim concedidas, para superar mais este desafio.

À toda minha família, em especial minha esposa Kelly e minha filha Sarah Manuella, pelo apoio incondicional e paciência nas minhas ausências por dedicação aos estudos. Ao meu falecido pai Walner a quem terei sempre como exemplo de vida e profissional, à minha mãe Maria José e meu sobrinho Walney, exemplos de superação e dedicação.

Ao IFTM e todos os professores do curso de Pós-Graduação que contribuíram para o meu crescimento profissional, em especial ao meu orientador Professor Mestre Walteno Martins Parreira Junior, pela disponibilidade, parceria e contribuição no desenvolvimento e conclusão da pesquisa. E

também um agradecimento especial à Softbox por autorizar o uso de um projeto real de grande porte como objeto de estudo e aos gestores e companheiros de trabalho pelo apoio e contribuição na elaboração e levantamento dos resultados.

REFERÊNCIAS

BECK, K; ANDRES, C. **Extreme Programming Explained**: Embrace Change. 2. ed. Reading: Addison-Wesley Professional, 2004, ISBN 9780321278654

BERNARDO, P. C.; KON, F. **A importância dos testes automatizados.** Engenharia de Software Magazine, 3. ed., 2008, p.54-57.

CAETANO, C. **Melhores Práticas na Automação de Testes.** Engenharia de Software Magazine, 5. ed., 2008, p.42-47.

CARVALHO, M. F. A. *et al.* **Automatização de testes de software**: Dashboard QMSanalyser. Coimbra: Instituto Politécnico de Coimbra, 2010. 87 p. Dissertação (Mestrado em Instalações e Equipamentos em Edifícios) - Departamento de Engenharia Electrotécnica do Instituto Superior de Engenharia de Coimbra, Coimbra, 2010.

CASPERJS. **Casper JS Navigation scripting & testing utility for PhantomJS and SlimerJS written in Javascript.** 2015. Disponível em: <http://casperjs.org/> Acesso em: 25 set. 2015.

CRISPIN, L.; HOUSE, T. **Testing Extreme Programming**. Reading: Addison-Wesley Professional, 2003, ISBN 9780321113559

PHPUNIT. **PHP Unit.** 2015. Disponível em: <https://phpunit.de>, Acesso em 20 set. 2015.

PRESSMAN, R. S. **Software Engineering**: A Practitioner's Approach. 7. ed. New York: McGraw-Hill Education, 2010, ISBN 9780073375977.

ROCHA, F. G. **Qualidade em desenvolvimento Web**: PHP com teste unitário. DevMedia, 2013. Disponível em <http://www.devmedia.com.br/qualidade-em-desenvolvimento-web-php-com-teste-unitario/27550>, Acesso em: 01 out. 2015.

SOFTBOX. **Softbox Sistemas de Informação Ltda**. 2015. Disponível em: <http://www.softbox.com.br/empresa.php?l=br&id=1>, Acesso em: 15 set. 2015.

YIN, R.K. **Estudo de caso**: planejamento e métodos. 5. ed. Porto Alegre: Bookman, 2015, ISBN 9781452242569

Referencias:

CARDEAL, Wisney; PARREIRA JÚNIOR, Walteno Martins. Estudo sobre a Utilização de Testes Automatizados em Projeto de Software de Grande Porte. In: Simpósio de Pós-Graduação do IFTM (Simpós), 2., 2015. **Anais...** Uberaba: IFTM, 2015. ISSN 2359-0130.