Princípios e Aplicações da Computação no Brasil

Ernane Rosa Martins
(Organizador)

Atena
Editora
Ano 2019

Ernane Rosa Martins
(Organizador)

Princípios e Aplicações da Computação no Brasil

Atena Editora
2019

2019 *by Atena Editora*
Copyright © da Atena Editora
Editora Chefe: Profª Drª Antonella Carvalho de Oliveira
Diagramação e Edição de Arte: Geraldo Alves e Natália Sandrini
Revisão: Os autores

Dados Internacionais de Catalogação na Publicação (CIP)
(eDOC BRASIL, Belo Horizonte/MG)

P957 Princípios e aplicações da computação no brasil [recurso eletrônico] / Organizador Ernane Rosa Martins. – Ponta Grossa (PR): Atena Editora, 2019. – (Princípios e aplicações da computação no Brasil; v. 1)

Formato: PDF
Requisito de sistema: Adobe Acrobat Reader
Modo de acesso: World Wide Web
Inclui bibliografia
ISBN 978-85-7247-046-9
DOI 10.22533/at.ed.469191601

1. Computação. 2. Informática. 3. Redes sociais. I. Martins, Ernane Rosa. II. Título. III. Série.

CDD 004

Elaborado por Maurício Amormino Júnior – CRB6/2422

APRESENTAÇÃO

Esta obra se propõe a permitir conhecer melhor o panorama atual da computação no Brasil por meio dos textos dos 15 capítulos que a constituem. Assim, estes trazem a reflexão temas importantes da área, tais como: performance web de e-commerce, análise de redes sociais, teoria de redes complexas, automação de teste em sistemas legados, ambiente virtual, arquitetura e organização de computadores, sistema integrado de gestão, sistema de apoio à avaliação de atividades de programação, rastreamento de objetos em vídeo, segurança da informação, ensino de programação, ensino de teoria da computação, sistemas de informação, fábrica de software, interdisciplinaridade, estilos de aprendizagem em computação, plataformas multiprocessadoras baseadas em barramentos.

Deste modo, esta obra reúne debates e análises acerca de questões relevantes, tais como: Qual o tamanho médio das páginas das lojas virtuais brasileiras e como estão em comparação com a média mundial? Quais informações estratégicas, para a segurança pública, podem ser obtidas com o uso da análise das redes sociais e complexas provenientes de uma base de dados de Tatuagens em Criminosos? A proposta de um novo ambiente virtual de simulação pode apoiar a aprendizagem? A proposta de um sistema de reconhecimento automático de possíveis soluções com mapeamento destas em escores atribuídos por professores, pode auxiliar professores na avaliação de exercícios de programação? A proposta de uma metodologia para rastreamento de múltiplos objetos em vídeos usando subtração de plano de fundo via mistura de gaussianas, morfologia matemática e o filtro de Kalman é mais precisa do que quando feita usando somente a subtração de plano de fundo? Como mensurar e priorizar a segurança da informação corporativa com base nos atuais arcabouços existentes na área? Quais páginas mais se preocupam com o usuário? Algumas ferramentas que foram propostas em trabalhos anteriores e que são utilizadas no ensino de programação atendem a nova realidade do ensino inicial de programação para crianças e jovens? Um projeto de extensão de uma Fábrica de Software, pode propiciar aos alunos capacitação nas principais tecnologias de mercado e vivência no mundo do trabalho?

Nesse sentido, este material ganha importância por constituir-se numa coletânea de trabalhos, experimentos e vivências de seus autores, tendo por objetivo reunir e socializar os estudos desenvolvidos em grandes universidades brasileiras. Certamente os trabalhos apresentados nesta obra são de grande relevância para o meio acadêmico, proporcionando ao leitor textos científicos que permitem análises e discussões sobre assuntos pertinentes à computação, por meio de linguagem clara e concisa, propiciando a aproximação e o entendimento sobre temas desta área do conhecimento. A cada autor, nossos agradecimentos a submissão de seus estudos na Editora Atena. Aos leitores, desejo proveitosa reflexão sobre as temáticas abordadas.

Ernane Rosa Martins

SUMÁRIO

CAPÍTULO 4

UM ESTUDO DE CASO DE AUTOMAÇÃO DE TESTE EM SISTEMAS LEGADOS SOBRE PLATAFORMA FLEX

Augusto Boehme Tepedino Martins
Universidade Federal de Santa Catarina
Florianpópolis - SC

Jean Carlo Rossa Hauck
Universidade Federal de Santa Catarina
Florianpópolis - SC

RESUMO: A realização de testes demonstra cada vez mais sua importância nas organizações que desenvolvem software. Ferramentas empregadas servem para trazer uma melhor confiabilidade e agilidade ao que está sendo produzido. Realizar testes, no entanto, é caro e demorado, e automatizar testes é uma alternativa interessante e tecnicamente viável. Sistemas legados comumente apresentam limitações para a automação de testes e isso ocorre também em sistemas Web desenvolvidos utilizando a plataforma Flex. Assim, este trabalho apresenta o desenvolvimento da automação de testes para plataforma Flex. Requisitos foram desenvolvidos para a escolha de uma ferramenta de automação de testes, a fim de escolher a que melhor atendesse as necessidades das aplicações, e que evitasse retrabalho. Um estudo de caso é então planejado e realizado objetivando a automação de testes de um sistema desenvolvido em plataforma Flex. Casos de teste são elaborados, uma ferramenta é selecionada, os scripts de teste são desenvolvidos e executados em um estudo de caso em uma empresa de desenvolvimento de software. Os resultados observados apontam indícios de que a automação de que testes pode aumentar inicialmente os esforços de elaboração dos casos de teste mas reduzem o tempo de execução, tendendo a gerar benefícios futuros.

PALAVRAS-CHAVE: Testes. Integração. Automação. Flex.

ABSTRACT: Software testing is increasing its importance in software development organizations. Used tools serve to bring better reliability and agility to what is being produced. Performing tests, however, is costly and time consuming, and automate testing is an interesting and technically viable alternative. Legacy systems often have limitations for test automation and it occurs on Web systems developed using the Flex platform. Thus, this work presents the development of test automation for Flex platform, test cases are elaborated, an automation tool is selected, test scripts are developed and executed in a case study over a software developer company. Test cases are developed, a tool is selected, test scripts are defined and implemented in a case study on a software development company. The observed results indicate initial evidence that test automation may initially increase the

development efforts of the test cases but reduce the runtime, tending to generate future benefits.
KEYWORDS: Tests. Integration. Automation. Flex.

1 I INTRODUÇÃO

Com a tecnologia cada vez mais avançada, muitas pessoas acabam não observando o quanto os softwares estão envolvidos em suas vidas, possivelmente porque suas ações são de tal forma transparentes aos seus usuários que se tornam algo trivial no seu dia-a-dia.

Com essa dependência atual de softwares em grande parte das atividades humanas, os produtos de software e o processo de criação de um software passaram a ser objeto de estudo, mesmo que um software gratuito venha a ser produzido, não será bem aceito se não for de fácil manuseio ou se possuir defeitos nas suas funcionalidades.

Especificamente falando de aplicativos que rodam na Web, algumas características esperadas de qualidade, como: disponibilidade de acesso, desempenho, evolução contínua e segurança [1]. Entretanto, a qualidade esperada nem sempre é alcançada. Algumas vezes, produtos de software têm sido entregues aos clientes cobrindo apenas os requisitos funcionais especificados, sem eliminar possíveis erros. Uma das maneiras de garantir que a qualidade de um produto de software seja atendida, e que as funcionalidades que o cliente necessita estejam presentes, é a aplicação de testes de software.

Entretanto, alguns testes são feitos por pessoas e, portanto, não são perfeitos, são suscetíveis a falhas. Muitas vezes as pessoas acabam deixando alguns pequenos problemas passarem despercebidos ou, também, possuem visões diferentes de como fazer os testes.

Nesse sentido, as padronizações e a automação de testes destacam-se na forma de garantir a qualidade de software. A padronização dos testes, como por exemplo, a documentação sobre casos de testes, com o passo-a-passo de como o teste deve ser feito, diminui a chance de variações na forma como os testes são realizados interfiram nos seus resultados. Já a automação de testes tende a diminuir a falha humana nos processos de testes. Assim, um grupo menor de pessoas poderia automatizar alguns tipos de testes que seriam repetidos toda vez que fosse necessário.

Ao se produzir um novo software, é necessária uma documentação dos testes para possuir um controle do que foi feito e do que será feito em relação ao software. Existem algumas ferramentas específicas para isso. Mas, infelizmente, nem todas as ferramentas de automatização de testes possuem uma integração com tais ferramentas. Assim, existe uma chance que esta documentação dos testes automatizados seja perdida, ou que saia de controle.

Assim, o presente trabalho apresenta um estudo de caso de desenvolvimento de automação de testes para a plataforma Flex com objetivo de reduzir a falha humana e a necessidade de repetir testes manuais no sistema legado.

As próximas seções são divididas da seguinte forma: na seção 2 é apresentado o estado da arte, seguindo da seção 3 onde a abordagem metodológica deste trabalho é brevemente apresentada; na seção 4 o estudo de caso é descrito e na seção 5 as conclusões são apresentadas.

2 | ESTADO DA ARTE

Como o presente trabalho trata da automação de testes de uma aplicação legada sobre a plataforma Flex, este tópico busca por soluções de automação já experimentadas para essa plataforma.

Assim, a busca por trabalhos similares foi realizada em consulta as seguintes ferramentas de pesquisa: Google (http://www.google.com), Google Scholar (http://scholar.google.com), Portal CAPES (http://www-periodicos-capes-gov-br.Ez46.periodicos.capes.gov.br/index.php?option=com_phome), pela facilidade de acesso e relevância como fontes de pesquisa, procurando envolver tanto referências da literatura quanto experiências da indústria.

Nessas ferramentas de busca, foram utilizadas as seguintes palavras de busca: “Software”, “Teste automatizado”, “Flex” e algumas variações/complementos: “Automação de testes de software”, “benefícios de testes automatizados”, “automatização de testes com Flex”, “Diferenças entre Flex e HTML”. Também foram utilizadas nas pesquisas as traduções na língua inglesa dos mesmos termos.

Sobre os resultados encontrados, para selecionar aqueles mais pertinentes a esta pesquisa, os seguintes critérios de inclusão foram utilizados:

- Ferramentas de automação de Testes com suporte à plataforma Flex: pois uma ferramenta para automatizar testes que não suporta a plataforma Flex, não é relevante ao trabalho;
- Plug-ins de automação de teste para Flex: tendo mesmo motivo de procura como foi relatado no critério acima;
- Relatos de experiência de automação de teste com Flex: para saber como ferramentas trabalham nesta plataforma e ter uma melhor decisão sobre o software escolhido

Para apoiar a busca, alguns requisitos de automação de testes para uma ferramenta legada sobre plataforma Flex foram identificados, com base: (i) nas características de ferramentas de automação de testes já previamente identificadas na literatura; (ii) nas necessidades para automação de testes coletadas por meio de entrevistas com analistas de testes da empresa desenvolvedora do software (vide

seção 4); e (iii) em entrevista com o gerente de projetos da empresa desenvolvedora do software; e, especialmente, (iv) nas limitações técnicas de automação de testes na plataforma Flex, que impossibilita a utilização de webdrivers por não manipular html diretamente:

- R1: Possuir alguma versão gratuita;
- R2: Suportar testes automatizados;
- R3: Estar atualizada e possuir comunidade ativa;
- R4: Suportar testes automatizados de aplicações que utilizam a plataforma Flex;
- R5: Suportar as versões mais recentes dos navegadores: Firefox, Chrome, Internet Explorer, e Opera;
- R6: Suportar testes automatizados em Aplicações Web;
- R7: Possuir documentação abrangente;
- R8: Ser capaz de utilizar um repositório central para que múltiplos usuários possam utilizar os scripts gerados

Após a execução das buscas nas ferramentas de pesquisa indicadas, foram encontradas as seguintes ferramentas que satisfazem, ao menos em parte, os requisitos propostos (vide Tabela 1).

Ferramenta	Descrição	R1	R2	R3	R4	R5	R6	R7	R8
Flex-ui-Selenium	Flex-ui-Selenium é uma extensão da ferramenta Selenium para a plataforma Flex, estendendo as possibilidades de uso do Selenium Webdriver, e pode gerar scripts nas linguagens Java, C#, Perl, Ruby, e PHP.	+	+	+	+	+	+	-	+
FlexMonkey	FlexMonkey é um aplicativo Adobe AIR gratuito, desenvolvida pela GorillaLogic que permite a criação de testes automatizados em plataformas AIR e Flex	+	+	-	+	+	+	+	+

Badboy	Badboy é uma ferramenta de automação de testes baseada na interface gráfica do usuário (GUI – Graphics User Interface), trabalhando com a movimentação do mouse e com um pequeno script, usando de modo Record/Playback para tornar ações do usuário em Scripts	+	+	+	-	+	+	+	+
Testmaker	Testmaker é uma ferramenta gratuita que consegue tornar um script de teste em um teste funcional, um teste de carga, ou um teste de performance, que cria scripts em Java, Ruby, Python, e PHP	+	+	+	?	-	+	-	+
Sikuli	Sikuli é uma ferramenta de automações de teste hibrida, na qual trabalha tanto com GUI, como com scripts de usuário. Interage com a tela a partir da captura de imagens, ou regiões, definidas no script	+	+	+	+	+	+	+	+
Bugbuster Test Record	Bugbuster Test Record é uma extensão feita para o Google Chrome, permitindo a criação de testes a partir de cliques feitos na janela do browser	+	+	+	?	-	+	+	-

Tabela 1. Ferramentas selecionadas

- +: atende completamente
- -: atende parcialmente ou não atende
- ?: não foi possível determinar

As ferramentas foram então instaladas e avaliadas uma a uma em relação aos requisitos propostos.

Conforme observado na Tabela 1, o Flex-ui-Selenium não cumpre o R7, pois não foi possível encontrar uma documentação abrangente, o que dificultou pesquisas e tentativas de utilização da ferramenta. O FlexMonkey não cumpre o R3, pois a ferramenta foi descontinuada, e não receberá mais atualizações.

A ferramenta Badboy falha no R4, pois não oferece suporte à plataforma Flex:

A ferramenta espera um retorno do comando para dar continuidade às ações, mas o Flex não responde este comando. Não foi possível descobrir se o Testmaker suporta a Plataforma Flex, pois não foi possível encontrar documentação para poder até mesmo instalar a ferramenta com facilidade, e o TestMaker funciona apenas sobre Firefox. Já a ferramenta Bugbuster foi possível de ser instalada, mas não foi possível determinar se a ferramenta suporta testes de aplicações na linguagem Flex, e a ferramenta apenas funciona na plataforma Google Chrome.

Após essa avaliação, a ferramenta que melhor atendeu os requisitos foi a ferramenta Sikuli, que cobriu todos os requisitos definidos.

3 I ABORDAGEM METODOLÓGICA

Nesta seção, são apresentadas as etapas da abordagem metodológica do presente trabalho.

Segundo Runeson [2], pode-se definir um estudo de caso como um estudo focado de um fenômeno em seu contexto, especialmente quanto quando os limites entre o fenômeno e o contexto não podem ser bem definidos. Runeson [2] indica também que é compreensível aplicar estudos de caso às áreas relacionadas à engenharia de software tais como desenvolvimento, operação e manutenção de software. É necessário planejar o estudo de caso, com uma base bem definida, para se analisar as unidades, os métodos e técnicas para coletas de dados e para se definir o controle e os resultados da pesquisa.

- Cada uma das etapas é brevemente apresentada na sequência:
- Planejamento: objetivos são definidos e o estudo de caso é planejado;
- Preparação: procedimentos e protocolos para coleta de dados são definidos;
- Coleta: execução e coleta de dados do estudo de caso;
- Análise: onde os dados coletados são analisados e as conclusões são extraída;

Relatório: onde os dados analisados são reportados.

A seção 4 a seguir detalha as principais etapas desta pesquisa no contexto do estudo de caso.

4 I ESTUDO DE CASO

De acordo com [2], é essencial para o sucesso de um estudo de caso que ele possua um bom planejamento. Assim, no contexto deste trabalho, o estudo de caso foi planejado onde foram definidos: a unidade de análise, os métodos utilizados para a coleta de dados, cronograma, participantes, etc. De forma resumida, o planejamento

do estudo de caso é apresentado na sequência, iniciando pela definição da unidade de análise.

4.1 Contextualização do Estudo de Caso

A empresa SPECTO Tecnologia é uma empresa que iniciou suas atividades no início da década de 90, e vem desenvolvendo produtos atualmente por meio de três linhas de produtos: CXM (gestão de atendimento), DCIM (gestão de datacenters), e BMS (gestão de prédios inteligentes).

O presente estudo de caso é realizado na divisão de gestão de datacenters (DCIM) da empresa. A DCIM tem como objetivo o desenvolvimento de produtos de software para monitorar e controlar o ambiente de datacenters, permitindo, por exemplo, alertar o usuário que um ambiente está com muita fumaça, ou que um dispositivo de monitoramento foi desconectado, por exemplo.

O sistema utilizado neste estudo de caso é o DataFaz Unity, um gerenciador de infraestrutura de datacenters, monitorando parâmetros físicos de ambiente, tais como: umidade, temperatura, energia, controle de acesso, etc. O sistema utiliza Flex para o desenvolvimento de sua interface com o usuário e o backend é desenvolvido em Java, com banco de dados tipicamente utilizado sendo PostgreSQL.

Neste estudo de caso são envolvidos os membros das equipes responsáveis pelos testes do sistema DataFaz Unity (membros da equipe de garantia de qualidade de produto), sendo três analistas de testes (dois formados em ciências da computação e um em automação) e dois testadores (graduandos de Ciências da Computação).

4.2 Preparação

- A coleta de dados corresponde ao conjunto das operações, através das quais o modelo de análise é submetido ao teste dos fatos e confrontado com dados observáveis[3]. O princípio da coleta de dados vem da utilização de várias fontes de evidência, criação de um banco de dados, de forma a encadear evidências de forma a buscar respostas para a pergunta de pesquisa deste trabalho: “Seria possível automatizar testes para um software Flex de forma a reduzir o esforço, tempo e retrabalho, tomando por base uma ferramenta já existente para modelagem e automatização de testes?”, os seguintes dados foram planejados para serem coletados:
- Quantidade de pontos de caso de uso;
- Quantidade de erros encontrados na execução dos testes automatizados;
- Avaliação subjetiva dos resultados observados pelos envolvidos na área de testes;

Quantidade de esforço realizado para a elaboração dos casos de testes automatizados.

4.3 Execução e coleta de Dados

O estudo de caso foi executado durante o período de quatro meses, após a análise de qual ferramenta de automação seria utilizada.

A ferramenta Sikuli, conforme já apresentada, é uma ferramenta de automação de testes de GUI que mescla Scripts com imagens, utilizadas como parâmetros, similares a constantes declaradas em qualquer linguagem de programação, que são procuradas quando o programa está rodando como. O Sikuli possui uma IDE própria que suporta linguagens Python e Ruby em uma plataforma Java: Jython e JRuby respectivamente, mas pode ser utilizado como um script de outras linguagens, como por exemplo Java, para utilização de outras IDEs, como o Eclipse e NetBeans. A IDE facilita o trabalho para capturar e adaptar imagens às necessidades de cada automação. Entretanto, sua utilização em outra IDE é mais complicada de se utilizar, porém pode ser integrada com outras linguagens e ferramentas.

Segundo Yeh et al. [4], a ideia de utilizar imagens para auxiliar na automação de testes vem da própria forma como ocorre interação na comunicação humana. Assim, na ferramenta Sikuli, as imagens são utilizadas como parâmetros, de maneira similar a constantes declaradas em qualquer linguagem de programação, onde o usuário pode definir o nome da imagem, o grau de similaridade que procura na tela, e aonde irá selecionar a tecla quando necessário.

Para utilizar a ferramenta, foi necessário instalar o Java JDK. Entretanto, infelizmente durante os testes não foi possível integrá-la ao Eclipse e utilizar a linguagem Java, pois não foi encontrada documentação que facilitasse a utilização do script com a ferramenta. Assim, os scripts de teste foram implementados em Jython.

Com a ferramenta instalada e pronta para executar os testes de forma automatizada, foi preparado o ambiente com as demais ferramentas necessárias e realizada a criação dos scripts de automação de testes. Para efetuar os testes manuais, configurou-se um computador para possuir o sistema DataFaz Unity instalado localmente, e também o software Enterprise Architect utilizado pela empresa para documentar a análise e modelagem do software (para organizar e documentar os casos de uso e os casos de teste com seus cenários), o PgAdmin para administração do banco de dados e recuperação do backup do banco de dados somente contendo a configuração inicial.

Com esse ambiente instalado, foram elaborados os casos de teste que seriam realizados tanto manualmente quanto de forma automatizada, de forma a possibilitar a comparação mais objetiva dos resultados da automação. Os casos de testes foram documentados na ferramenta Enterprise Architect, conforme já citado.

Foram elaborados, ao todo, dez casos de testes completos, cada um possuindo de três a cinco diferentes cenários, tomando-se por base as principais funcionalidades do sistema DataFaz Unity. Os casos de teste foram documentados seguindo as melhores práticas definidas na norma ISO/IEC/IEEE 29119 [5]. Para cada caso de teste foi realizada a implementação do seu correspondente em script de teste automatizado.

A Figura 1 mostra o extrato de um caso de teste para execução manual e a figura 2 mostra o mesmo caso de teste para execução automatiza e parte seu código de automação.

Identificador	TST-001.01
Item de teste	Login Padrão
Especificações de Entrada	1. Usuário: fazion; 2. Senha de acesso: 22; 3. Clicar no botão "Ok".
Procedimentos	1. Inserir o dado "Usuário" na área "Nome de Usuário"; 2. Inserir o dado "Senha de acesso" na área "Senha"; 3. Selecionar o botão "Enviar".
Especificações de Saída	Após o procedimento ser realizado, a tela deve ser alterada, apresentando a tela principal da aplicação.
Ambiente necessário	DataFaz Unity, banco de dados.
Exigências especiais	Não aplicável.
Interdependências	Fazer uso do caso de teste específico e PST - Proposta Técnica.

Figura 1 – Extrato de Caso de Teste para execução manual

TST-001.01	Código Sikuli IDE
Login Padrão	

1. Usuário: fazion;
2. Senha de acesso: 22;
3. Clicar no botão "Ok".
	# Especificações de entrada:
	login = "fazion" #Login do sistema
	password = "22" #Senha do sistema

1. Inserir o dado "Usuário" na área "Nome de Usuário";
2. Inserir o dado "Senha de acesso" na área "Senha";
3. Selecionar o botão "Enviar".
	# Definição para abrir uma nova Janela do Chrome
	defopenff():
	ffLoc = r'C:\Program Files (x86)\Mozilla Firefox\firefox.Exe'
	App(ffLoc).focus() # opens FF
	if exists("HttpBar.png",10):
	keyDown(Key.CTRL + "l")
	paste("https://localhost:8443/datafaz/")
	type(Key.ENTER)
	
	# Definição para realizar Login
	defsystemlogin(lgin, pssw):
	
	# Procura o padrão da imagem e seleciona com o mouse um campo 47 pixels deslocados para baixo do eixo X.
	# Em código puro:
	# t = find(Pattern("LoginTab.png").targetOffset(0,47))
	t = find(

Figura 2 – Extrato de Caso de Teste e código de automação

Para garantia da qualidade dos testes, um computador possuindo o sistema DataFaz Unity instalado localmente, o Enterprise Architect com a modelagem do DataFaz Unity para organizar e documentar os casos de uso e casos de teste com seus cenários, o PgAdmin como banco de dados, e um arquivo de backup, com dados limpos a ser utilizado como cenário inicial dos casos de teste.

Cada um dos casos de teste foi então executado manualmente por dois testadores diferentes e cada um dos scripts de teste automatizado foi executado dez vezes em um mesmo computador. Ao final dos testes, os analistas de teste e testadores foram submetidos a um questionário de avaliação, composto de quatro perguntas utilizando escala Likert de respostas, procurando identificar as percepções sobre o resultado da aplicação dos testes automatizados.

4.4 Análise dos Resultados

Durante toda a execução dos testes, os dados de duração foram coletados por meio do registro das tarefas no sistema de gerência de projetos utilizado pela empresa[1]. Como resultado, conforme esperado, foi percebido que o tempo necessário para elaboração dos testes automatizados e manuais apresentou grande diferença, devido ao tempo utilizado na elaboração dos Scripts de testes automatizados, conforme mostra a Tabela 2.

Métricas	Teste Manual	Teste Automatizado
Elaboração dos casos de teste	05h00min	05h00min
Elaboração Scripts de Testes Automatizados	-	07h30min
Tempo médio de execução	00h08min	00h02min30seg
Total Duração	**05h08min**	**12h32min30seg**

Tabela 2 - Comparações entre teste manual e automatizado

Entretanto, conforme também mostra a Tabela 2, o tempo necessário para execução dos testes automatizados foi consideravelmente inferior. Espera-se que as futuras execuções dos testes automatizados venham a compensar o maior esforço aplicado na elaboração dos scripts de teste.

Além dos dados de tempo de execução, a partir da aplicação do questionário, foi possível coletar a impressão subjetiva dos participantes em relação aos resultados da automação de testes no sistema legado utilizando plataforma Flex.

Questões	Concordam
Facilidade na compreensão dos testes	80%
Cenários cobriram a funcionalidade proposta	80%
A infraestrutura disponibilizada para o ambiente de testes foi satisfatória para a execução dos testes	80%
O tempo gasto com teste automatizado diminui esforços de custo e tempo com execuções repetitivas	100%

Tabela 3 - Resultados do questionário

Conforme pode ser percebido pelas respostas dos participantes, a automação de testes utilizando a ferramenta Sikuli, atendeu às expectativas da maioria. Entretanto, foram percebidas oportunidades de melhoria pela equipe, como por exemplo, a possibilidade de integração direta com o Eclipse, possibilitando a implementação dos scripts em Java.

1 http://www.jexperts.com.br/category/pse/pse-pch/

4.4.1 Ameaças à validade

Como tipicamente ocorre em estudos de caso [6][7], o presente trabalho somente foi aplicado em um único software de uma única empresa, não sendo possível desta forma, generalizar seus resultados a outras empresas e cenários, mesmo que positivos.

Da mesma forma, a empresa possui poucos analistas de testes na unidade organizacional do estudo de caso, fazendo com que os resultados obtidos fossem escassos, mais analistas trariam mais respostas, e o resultado da aplicação questionário seria mais abrangente. Além disso, o autor do trabalho teve participação ativa na execução do trabalho, e este envolvimento direto pode gerar viés de interpretação de dados coletados.

5 I CONCLUSÕES

Este trabalho apresenta uma experiência de automação de testes em um sistema legado desenvolvido sobre a plataforma Flex. Tendo em mente as limitações de automação de testes em Flex, foram analisadas diferentes abordagens de automação. Foi possível notar que nenhuma ferramenta iria poder automatizar a aplicação em sua totalidade, então foram criados requisitos para escolher qual ferramenta seria utilizada no trabalho. A ferramenta Sikuli, que cobriu a maior parte dos requisitos propostos, foi escolhida.

No contexto de um estudo de caso, casos de teste foram elaborados, sendo implementados scripts de teste na linguagem aceita pelo Sikuli. Os testes foram então executados de forma manual e automatizada, em um ambiente controlado.

Os dados coletados de esforço e tempo durante a elaboração e realização dos testes no estudo de caso levantam indícios de que o ganho de tempo na automação de testes é pequeno inicialmente. Espera-se, entretanto, que a organização envolvida no estudo de caso obtenha benefícios diretos com a redução de custos de retrabalho e aumento de confiabilidade da aplicação com a execução dos testes automatizados em futuros releases do sistema.

Como trabalhos futuros planeja-se a integração dos testes automatizados com a ferramenta de documentação e modelagem dos testes. Além disso, seria importante expandir os casos de teste automatizados para o restante das funcionalidades do sistema.

6 I AGRADECIMENTOS

Os autores agradecem aos membros da equipe da empresa Specto que participaram do estudo de caso. Também agradecem a cooperação e interesse da gerência da empresa no estudo de caso.

REFERÊNCIAS

Pressman, R. S. **Engenharia de software: uma abordagem profissional**; Tradução Ariovaldo Griesi e Mario Moro Feccio. 7ª ed. São Paulo: AMGH, 2011.

Runeson, P., and M. Höst. **Guidelines for conducting and reporting case study research in software engineering**. Empirical Software Engineering 14.2, 2009.

Quivy, R. et al. (1992). **Manual de Investigação em Ciências Sociais**. Lisboa: Gradiva.

Yeh, Tom, Chang, Tsung-Hsiang, Miller, Robert C. (2009): **Sikuli: using GUI screenshots for search and automation**. In: Proceedings of the ACM Symposium on User Interface Software and Technology, 2009, pp. 183-192. http://doi.acm.org/10.1145/1622176.1622213. International Organization for Standardization. ISO/IEC/IEEE 29119. Software Testing Standard. 2010.

Zelkowitz, M. V. **An update to experimental models for validating computer technology**, J. Syst. Softw., vol. 82, pp. 373–376, 2009.

Yin, R. K. **Applications of case study research**, ed. 3, 2011.

Agência Brasileira do ISBN
ISBN 978-85-7247-046-9