Uma Comparação entre Testes Manuais e Testes Automatizados para Garantia da Qualidade em Softwares para Dispositivos Móveis

Lindomar P. Reitz, Anita Maria da Rocha Fernandes

Pós Graduação em Qualidade e Engenharia de Software – Universidade do Vale do Itajaí (UNIVALI)
Rodovia SC 401, 5025 – 88.032-005 – Florianópolis – SC – Brasil

lindomar.reitz@gmail.com, anita.fernandes@univali.br

Abstract. *Mobile applications are being used more and more. With this growth, the demand about quality is increasing. Operation systems, like Android, have a big fragmentation of devices and versions, which is a challenge to ensure the quality of the software produced. This paper presents a study scenario with five test scenarios and five devices. For it, manual and automated tests were performed in order to compare their efficiency. At the end, it was observed that the execution of automated testes in parallel presented a 45.86% efficiency at execution time.*

Resumo. *Aplicativos móveis estão sendo cada vez mais utilizados. Com esse crescimento, a exigência pela qualidade é cada vez maior. Sistemas operacionais, como o Android, possuem uma grande fragmentação de dispositivos e versões, sendo este um desafio em garantir a qualidade do software produzido. Este trabalho apresenta um cenário de estudo contendo cinco cenários de teste e cinco dispositivos. Para isso foram executados testes manuais e automatizados, com o objetivo de comparar a sua eficiência. Ao final, foi observado que a execução dos testes automatizados em paralelo apresentaram uma eficiência de 45,86% no tempo de execução.*

1. Introdução

Segundo Starov *et al* (2015), nos últimos anos houve um aumento exponencial no uso de aplicativos móveis, assim como cresceu também o número de plataformas e fabricantes de celulares. Com esse crescimento, também veio a necessidade de entregar software de qualidade em ciclos curtos, pois a exigência dos usuários está cada vez maior, seja no aspecto funcional, usabilidade, performance e segurança [Starov *et al* 2015; Gao *et al* 2014; Dantas 2009].

Com um mercado tão concorrido, a falta de qualidade pode acarretar em prejuízos financeiros, uma vez que os usuários são impacientes quando se apresenta defeitos ou perda de performance, tendo a facilidade de ir para os concorrentes sem esforço. A reputação do aplicativo também é baseado nesses comportamentos [MTPI 2014]. O custo para corrigir esses defeitos após o lançamento pode ser proibitivo [Deus 2009].

Das plataformas existentes, a plataforma Android em especial é a que possui a maior fragmentação, seja nas suas versões quanto na quantidade de dispositivos. Isso faz com que a complexidade dos testes aumente, dado que é necessário aumentar a

quantidade de testes em diferentes dispositivos [Vilkomir *et al* 2015]. Fazer esses tipos de teste de forma manual pode ser um custo inviável em muitos casos [uTest 2015].

Dado esse cenário, houve um crescimento na quantidade de soluções de infraestrutura e ferramentas para executar esses testes, desde soluções *open source* até soluções pagas [Caroline 2014]. Com o avanço da tecnologia da computação em nuvem, já existe a possibilidade de executarem os testes nesse ambiente [Nachiyappan e Justus 2015]. Pesquisas recentes já demonstram esse cenário, sendo agora uma opção viável [Starov *et al* 2015; Gao *et al* 2015; Villanes *et al* 2015].

Este artigo tem como objetivo apresentar um cenário de estudo com testes automatizados em múltiplos dispositivos, procurando responder as seguintes perguntas de pesquisa: É possível executar esses testes em paralelo? Essa execução irá ser feita em um ambiente em nuvem, com a utilização de emuladores. Se sim, qual é a eficiência desses testes em relação aos testes manuais?

Este artigo está organizado em 5 seções. Na Seção 2 é apresentada a fundamentação teórica sobre testes de software, testes *mobile*, testes automatizados e ambiente de testes. Na Seção 3 será apresentada a metodologia utilizada, incluindo as suas etapas. Na Seção 4 será descrito o cenário de estudo em si, bem como a análise dos resultados. Por fim, na Seção 5 serão apresentadas as conclusões e, em seguida, as referências do trabalho.

2. Testes de *software*

Sommerville (2011, p. 144) afirma que o teste deve mostrar que um programa faz aquilo que foi proposto e também para descobrir os defeitos do programa antes do uso. “Os resultados do teste são verificados à procura de erros, anomalias ou informações sobre os atributos não funcionais do programa.”.

Para Pressman (2011, p. 402): “Teste é um conjunto de atividades que podem ser planejadas com antecedência e executadas sistematicamente”. Seguindo essa mesma linha, os testes de software são formados por um processo sistemático que tem como principal objetivo o de encontrar erros [Bartié 2002]. Outro objetivo dos testes de software é o de verificar alguns atributos do sistema, como, por exemplo, a sua confiabilidade.

Essas definições de testes, embora diferentes, são complementares entre si e por isso são adotadas pelos autores.

2.1. Testes *mobile*

Knott (2015) afirma que os testes *mobile* possuem características especiais, e que não devem ser comparados com testes para *web* e *desktop*. Existem diversos desafios para atender as exigências dos usuários, desde as capacidades do dispositivo (GPS, acelerômetro, câmera e *Application Programming Interfaces* (APIs)) até a quantidade de dispositivos e versões do sistema operacional a serem suportados. Além disso, também deve ser eficiente em condições adversas, como, por exemplo, com baixa velocidade de conexão.

Para uTest (2015), outras preocupações são com o consumo de bateria, assim como a usabilidade e facilidade de instalação. Esses são alguns dos fatores que podem fazer os seus usuários escolherem entre o seu aplicativo ou do seu concorrente.

A seguir, são apresentados alguns conceitos sobre testes automatizados.

2.2. Testes automatizados

Segundo Bartié (2002, p. 196), os testes automatizados são definidos como "[...] a utilização de ferramentas de testes que possibilitem simular usuários ou atividades humanas de forma a não requerer procedimentos manuais no processo de execução dos testes.". Molinari (2003, p. 104), cita que a aplicação de testes automatizados "[...] permitirá aumentar a profundidade e abrangência dos casos de testes envolvidos.".

Para Bartié (2002, p. 181), um dos motivos para sua utilização é que "A automação exige um esforço inicial de criação, porém possibilita uma incomparável eficiência e confiabilidade, impossível de ser atingida com procedimentos manuais.".

A automação de testes não faz parte de um nível específico, sendo que o mesmo garante que testes de regressão sejam feitos. Conforme Pressman (2011, p. 404) esse teste "[...] pode ser executado manualmente, reexecutando um subconjunto de todos os casos de teste ou usando ferramentas automáticas de captura/reexecução.", sendo esse um dos principais motivos para a sua automatização, o que é o foco deste trabalho.

2.3. Ambiente para os testes

Conforme Bastos *et al* (2007, p. 77-78), o ambiente para os testes deve ser preparado para cada estratégia de teste, já que ele é formado pelo *hardware* e também pela massa de dados, sistemas operacionais, ferramentas e qualquer outro recurso que dê condições de executar os testes.

Falando especialmente em ambiente de testes para dispositivos móveis, Gao *et al* (2014) exemplificam alguns tipos de ambientes, como mostra a Figura 1.

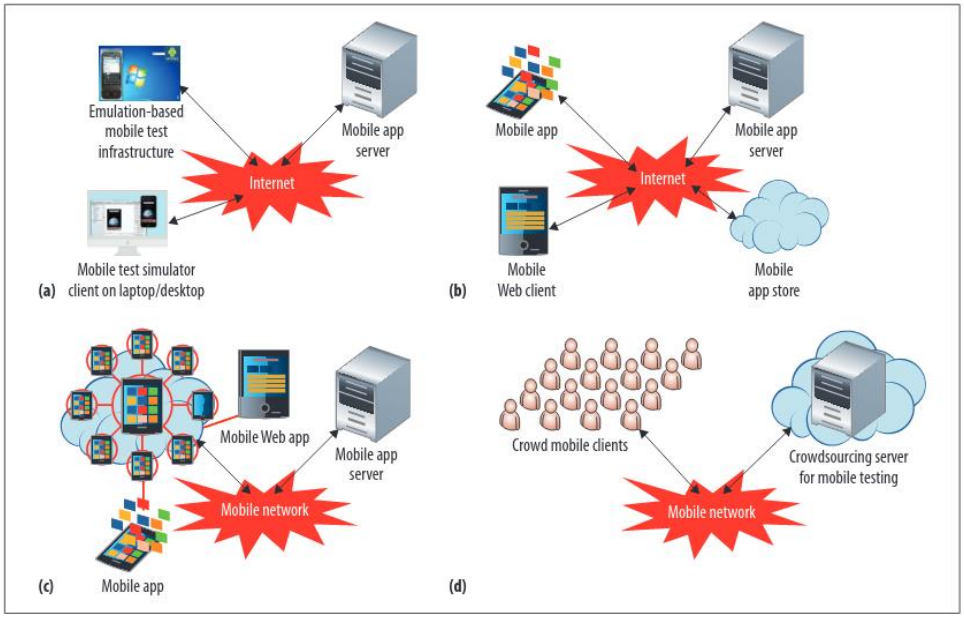

Figura 1. Diferentes ambientes de teste: (a) emuladores, (b) computação em nuvem, (c) dispositivos reais, (d) crowd testing

Fonte: Gao *et al* (2014)

Como mostra a figura acima, dos diversos tipos de ambiente de teste foram categorizados em quatro tipos: baseados em emuladores, baseados em dispositivos reais, *cloud testing* e *crowd testing*. Os dois primeiros são baseados em manter uma

infraestrutura própria, seja com servidores para os emuladores ou para manter diversos dispositivos reais conectados. Já no *cloud testing* as possibilidades de configuração e capacidade são diferentes, como será explicado a seguir. Por fim, o ambiente de *crowd testing* é dependente de quem vai executar os testes, sejam eles profissionais autônomos ou uma empresa terceira, o que não se tem certeza de qual ambiente será utilizado [Gao *et al* 2015].

Segundo Starov *et al* (2015), a computação em nuvem já está sendo extensivamente utilizada para testes, especialmente testes automatizados. Um dos motivos para isso é a facilidade em provisionar ambientes que possuam bom desempenho computacional e suportem a carga exigida, tanto com emuladores quanto com dispositivos reais. Isso também ajuda em diminuir os custos ao manter uma infraestrutura física.

Dados esses conceitos, a seguir será apresentado a metodologia utilizada neste artigo.

3. Metodologia

Em termos de metodologia, este trabalho pode ser caracterizado como uma pesquisa aplicada, utilizando dados qualitativos [Marconi e Lakatos 2003]. Sendo assim, o artigo está dividido nas seguintes etapas, como mostra a Figura 2.

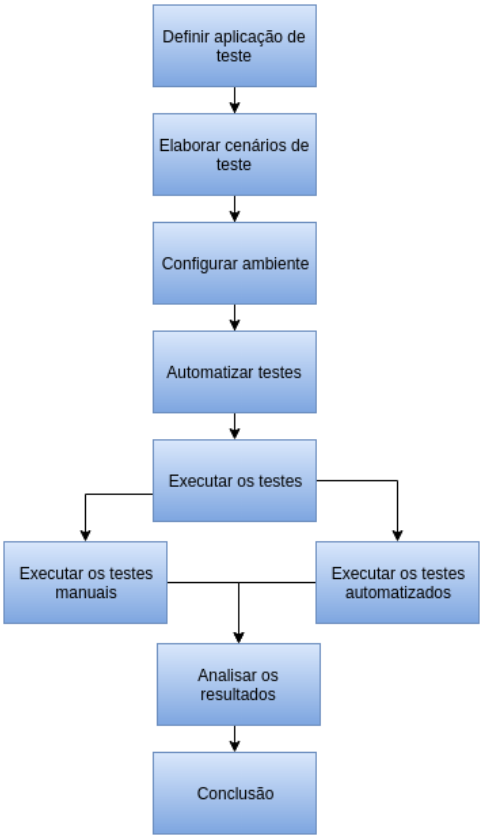

Figura 2. Etapas metodológicas

A primeira etapa foi de definir qual aplicação de teste seria utilizada, sendo nesse caso escolhida uma aplicação da categoria de finanças. Esta aplicação, segundo a loja

Google Play[1] possui entre 100.000 até 500.000 instalações, estando na versão 0.11.2 até o momento da publicação desse artigo.

Definido isso, a etapa de definição de cenários de teste foi realizada com base nas situações mais comuns do uso da aplicação, que vão desde o cadastro/exclusão de despesas e receitas, até a configuração de uma nova categoria das transações. Ao todo foram definidos cinco cenários, sendo estes escritos no formato BDD (*Behavior Driven Development*).

Após definidos os cenários, a etapa de configuração consistiu em configurar o ambiente do qual os testes serão executados, sendo este um serviço em nuvem chamado SauceLabs[2]. Para a execução dos testes, será utilizada uma conta para projetos *open source*. Foram escolhidos 5 entre 16 emuladores Android disponíveis na plataforma com a versão 4.4 (API 19). Esses emuladores foram escolhidos com base em dispositivos de fabricantes que são utilizados nessa versão, como, por exemplo, das empresas Samsung, Google e LG.

Com o ambiente configurado, a etapa de automatizar os testes é feita com os cenários já definidos previamente. Para este trabalho, os testes foram automatizados com ferramentas *open source*, sendo estas apresentadas na Seção 4.

Na etapa de execução de testes, serão executados tanto os testes manuais quanto os testes automatizados. Ambos serão executados contra cinco emuladores Android, sendo que o primeiro será executado um de cada vez e o segundo será executado em paralelo.

Por fim, na etapa de análise de resultados será realizada com base nos tempos de execução obtidos entre os dois tipos de testes, sendo comparada a eficiência entre um e outro. Em seguida o artigo será finalizado com a etapa de conclusão.

4. Cenário de estudo

Esta seção apresenta o desenvolvimento e a análise do cenário de estudo.

4.1. Ferramentas utilizadas

Para realizar a automação de testes desse trabalho, foram utilizadas algumas ferramentas *open source*, sendo que um breve resumo será descrito a seguir.

O RSpec[3] é um *framework* para a escrita de testes que segue o padrão *xUnit*. A ferramenta foi utilizada para automatizar os cenários de teste, utilizando alguns recursos como o de *Setup* (mandar as requisições para o SauceLabs) e *Teardown* (fechar a sessão do teste). Outro motivo para ser utilizada é de que se trata de uma das ferramentas mais usadas na comunidade de Ruby.

O Appium[4] é um *framework* para a automação de testes *mobile*, podendo ser utilizada para aplicativos nativos, híbridos e *web*. Utiliza como base a API do Selenium

---

[1] Aplicativo Grana: https://play.google.com/store/apps/details?id=com.renanferrari.grana

[2] SauceLabs: https://saucelabs.com/

[3] RSpec: http://rspec.info/

[4] Appium: http://appium.io/

WebDriver, ou seja, os mesmos recursos para automação *web* podem ser utilizados, como, por exemplo, encontrar elementos pelo *id*, *class*, *name* e *XPath*. Esse *framework* possui um bom suporte para o Android, além de permitir que os testes sejam escritos em diversas linguagens, tais como Java, Ruby, JavaScript, C#, Perl e PHP.

O aplicativo utilizado nesse trabalho é nativo para o Android, sem acesso ao código fonte. Para identificar os elementos foi utilizada a ferramenta UI Automator Viewer[5], fornecida pelo próprio Android. Basicamente, essa ferramenta apresenta uma interface gráfica que permite fazer o mapeamento dos elementos do aplicativo de forma hierárquica, mostrando informações de algumas propriedades, tais como o *resource-id*, *class*, *name* e *text*.

4.2. Matriz de testes

A complexidade da execução dos testes é representada na Tabela 1.

Tabela 1. Matriz de testes (dispositivos x cenários de teste)

	Samsung Galaxy S3	**Samsung Galaxy S4**	**Samsung Galaxy Nexus**	**Google Nexus 7C**	**LG Nexus 4**
Cadastrar uma receita					
Cadastrar uma despesa					
Excluir uma transação					
Cadastrar um orçamento					
Cadastrar uma categoria					

Como se trata de uma matriz 5x5, para cada um dos cinco emuladores serão executados cinco cenários de testes, totalizando 25 execuções. A adição de mais cenários e/ou mais dispositivos fazem com que a quantidade de testes cresça exponencialmente, aumentando assim a complexidade em executar os testes em todos esses ambientes.

4.3 Análise dos resultados

A análise desse trabalho será dividida entre os testes manuais, testes automatizados e após isso será feito um comparativo entre as execuções.

[5] UI Automator Viewer: https://developer.android.com/topic/libraries/testing-support-library/index.html

Os testes manuais apresentaram os seguintes tempos de execução (em segundos), conforme a Tabela 2.

Tabela 2. Tempos de execução dos testes manuais

Dispositivo	*Setup*	Testes	Total
Samsung Galaxy S3	73	97	170
Samsung Galaxy S4	76	77	153
Samsung Galaxy Nexus	79	88	167
Google Nexus 7C	69	69	138
LG Nexus 4	78	87	165

Como mostra a Tabela 2, os testes tiveram um tempo parecido de *Setup*, com a média de 75 segundos para carregar o dispositivo e abrir o aplicativo. O tempo de execução dos testes teve um pouco mais de variação devido ao tempo de resposta do dispositivo, como foi o caso do emulador Google Nexus 7C, com 69 segundos. O tempo total apresentou uma média de 158,6 segundos, sendo este um tempo aceitável para os testes de cinco cenários.

Os testes automatizados apresentaram os seguintes tempos de execução (em segundos), conforme a Tabela 3.

Tabela 3. Tempos de execução dos testes automatizados

Dispositivo	*Setup*	Testes	Total
Samsung Galaxy S3	78	197	215
Samsung Galaxy S4	78	144	222
Samsung Galaxy Nexus	79	123	202
Google Nexus 7C	80	152	232
LG Nexus 4	79	150	229

Como mostra a Tabela 3, os testes automatizados tiveram um tempo praticamente igual de *Setup*, com 78,8 segundos na média. O tempo de execução dos testes apresentou uma variação, sendo que o menor tempo foi no emulador Samsung Galaxy Nexus, com 123 segundos. Já os testes no emulador Samsung Galaxy S3 apresentaram o maior tempo, com 197 segundos. O tempo total apresentou uma média de 220 segundos, o que pode ser considerado alto em relação aos testes manuais.

Em termos comparativos entre os testes manuais e automatizados, a Figura 3 apresenta os tempos de *Setup*.

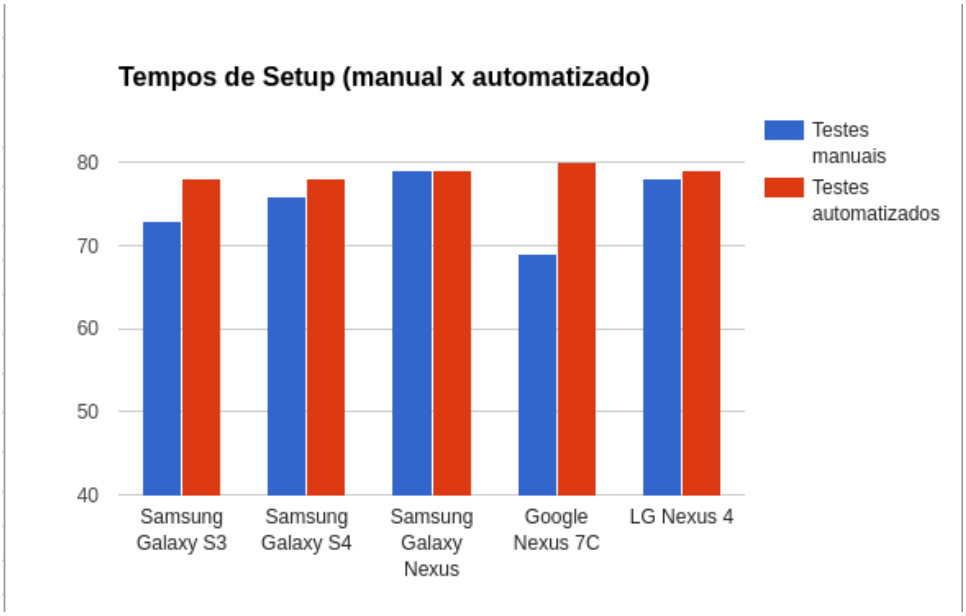

Figura 3. Comparativo nos tempos de *Setup*

Como mostra a figura acima, os tempos de *Setup* foram similares, até pelo motivo de que foram executados no mesmo ambiente de testes. Dos cinco emuladores, dois deles apresentaram uma maior diferença. O emulador Samsung Galaxy S3 apresentou uma diferença de 5 segundos, enquanto o emulador Google Nexus 7C apresentou 11 segundos de diferença entre o teste manual e o teste automatizado.

De uma maneira geral, os testes manuais apresentaram um menor tempo de *Setup*, com algumas variações para cada dispositivo, tendo uma média de 3,8 segundos inferior aos testes automatizados.

Além do tempo de *Setup*, outro comparativo a ser feito é com o tempo de execução de todos os testes, conforme mostra a Figura 4.

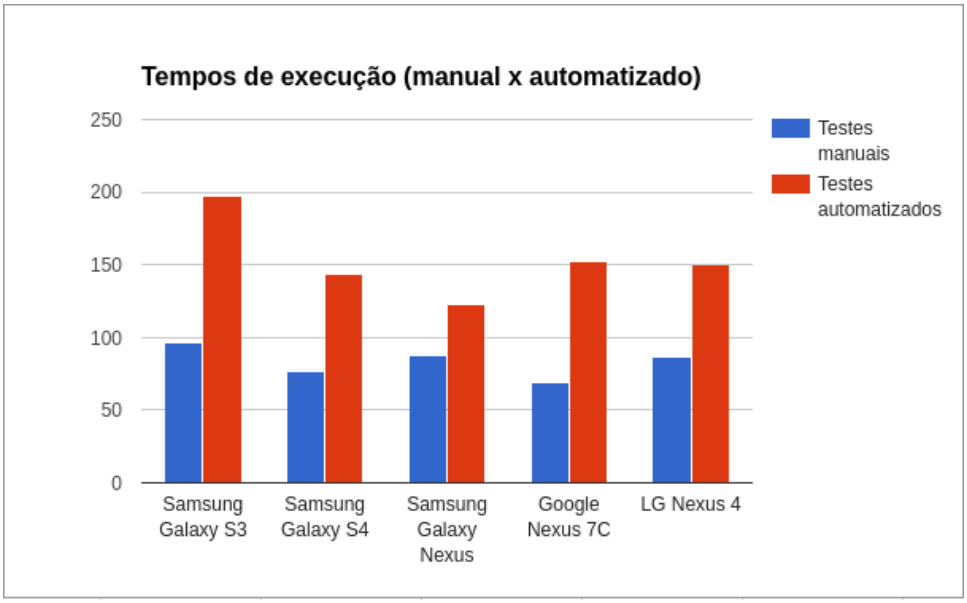

Figura 4. Comparativo nos tempos de *execução*

Nesse comparativo entre os tempos de execução houve uma diferença considerável. O emulador Samsung S3 apresentou uma diferença de 121 segundos. O emulador Google Nexus 7C apresentou a segunda maior diferença, com 72 segundos.

Um dos motivos para essa diferença no tempo de execução é que nos testes manuais existe pouca diferença de tempo ao enviar os comandos, uma vez que a interação com o aplicativo é mais direta. Para a execução dos testes automatizados, cada comando de teste é enviado em uma requisição para o servidor, sendo que esse tempo é variável de acordo com a velocidade de conexão entre o cliente (máquina que envia os comandos) e o servidor (que vai executar o comando de fato). Outro fator importante é que o tempo de execução também depende do emulador em si, já que eles possuem capacidades de memória e processamento diferentes.

O tempo total de todos os testes executados (em segundos) é mostrado na figura 5.

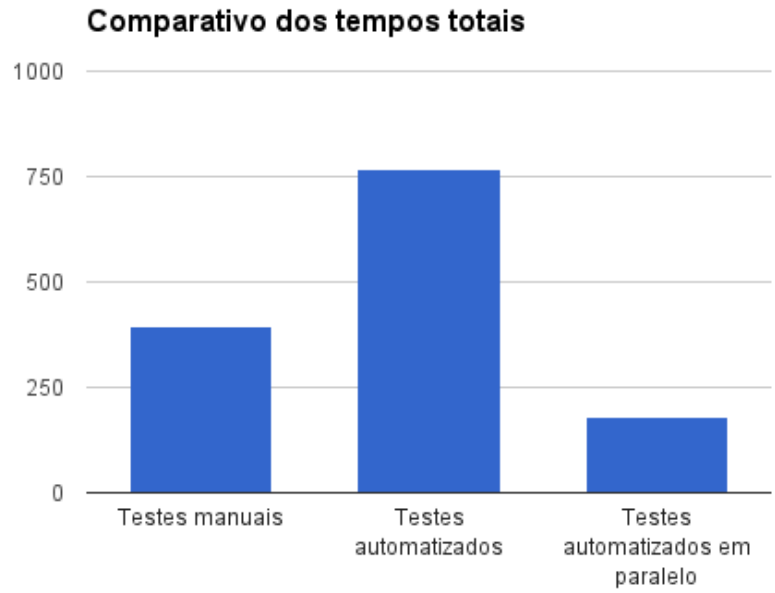

Figura 5. Comparativo dos tempos totais

Observando os tempos de execução de forma individual, os testes manuais apresentaram um melhor desempenho, com a diferença de 61,4 segundos no tempo médio. Avaliando os tempos de execução por outra visão, os testes manuais tiveram o tempo total de 394 segundos, uma vez que os mesmos foram executados para cada emulador, um de cada vez. Já para os testes automatizados, os mesmos foram executados em paralelo, com 5 instâncias. Isso acabou cortando o tempo de execução total de 766 segundos para cerca de 180 segundos, o que representa uma eficiência de pelo menos 45,86%, quase metade do tempo gasto com a execução dos testes manuais.

5. Conclusões

A importância da qualidade de software está cada vez maior, não sendo diferente com aplicativos móveis. Com um mercado tão exigente e com muitos concorrentes, qualquer problema apresentado pode ser o motivo para os usuários desistirem de usar o seu aplicativo, além de ter a reputação prejudicada.

Além do fator da qualidade, um outro desafio é o de realizar testes em sistemas operacionais fragmentados, com vários dispositivos (emuladores ou reais) e diferentes versões, como é o caso do Android. A necessidade de ter velocidade nesses testes também é crucial para o lançamento de novas funcionalidades e/ou correções de erros. Outro problema seria o de adquirir e manter esses dispositivos para testes, o que pode apresentar um alto custo. Essa seria uma das motivações para se utilizar serviços de computação em nuvem, o que facilitam a configuração e execução desses testes.

Levando em consideração esses problemas, este trabalho buscou aplicar testes *mobile*, tanto manuais quanto automatizados. De acordo com as perguntas de pesquisa, foi possível realizar a execução dos testes automatizados em paralelo, que mesmo apresentando uma diferença de tempo considerável com os testes manuais, provou-se mais eficiente, uma vez que os testes manuais não são possíveis de paralelizar da mesma forma. Esse tempo de execução dos testes manuais pode ser melhor aproveitado para cobrir outros tipos de teste, como, por exemplo, testes exploratórios e de usabilidade.

Outro fator a ressaltar neste trabalho foi em relação a execução dos testes utilizando a computação em nuvem. Isso trouxe bastante facilidade com a criação e execução dos testes em diferentes dispositivos, sem precisar manter nenhum tipo de configuração desses emuladores no ambiente local. Mesmo apresentando um tempo maior de *Setup* e execução, essa solução se mostrou mais escalável, com a possibilidade de executar vários testes em diferentes configurações, com pouco esforço.

Como trabalhos futuros, propõe-se o estudo em um cenário com uma matriz de testes mais complexa, seja com mais testes, mais emuladores ou diferentes versões. Além disso, outro estudo poderia ser feito com aspectos não funcionais, como, por exemplo, testes de carga e *performance*. A criação e execução desses testes em um ambiente que utiliza metodologia ágil seria outra opção de estudo.

Referências

Bartié, Alexandre (2002), Garantia da qualidade de *software:* adquirindo maturidade organizacional, Campus.

Bastos, Aderson, Rios, Emerson. Cristalli, Ricardo e Moreira, Trayahú (2007). Base de conhecimento de teste de software, Martins, 2ª Edição.

Caroline, A. (2014) "Ferramentas de Testes em Aplicativos para Celular*"* in Grupo de testadores de Software. Disponível em http://gtsw.blogspot.com.br/2014/08/ferramentas-de-testes-em-aplicativos.html. Maio.

Dantas, V. L. L. (2009), Requisitos para Testes de Aplicações Móveis, Universidade Federal do Ceará. Dissertação de Mestrado em Ciência da Computação, 132 p.

Deus, G. D. de (2009), Avaliação de Técnicas de Teste para Dispositivos Móveis por Meio de Experimentação, Universidade Federal de Goiás. Dissertação de Mestrado em Ciência da Computação, 119 p.

Gao, J., Bai, X., Tsai, W.-T. e Uehara, T (2014) "*Mobile Application Testing: A Tutorial"* in *Computer (Long. Beach Calif)*., vol. 47, no. 2, p. 46-55.

Knott, D. (2015), *Hands-On Mobile App Testing: A Guide For Mobile Testers and Anyone Involved in the Mobile App Business*, Addison-Wesley Professional.

Marconi, Marina de Andrade e Lakatos, Eva Maria (2003), Fundamentos da Metodologia Científica, Atlas, 5ª edição.

Molinari, Leonardo (2003), Testes de Software: Produzindo Sistemas Melhores e Mais Confiáveis, Érica.

MTPI (2014), *"*Testes de qualidade em Aplicações Mobile*",* in MTP Digital Business Assurance. Disponível em http://www.mtpi.com.br/noticias/407-testes-de-qualidade-em-aplicacoes-mobile. Maio.

Nachiyappan, S. e Justus, S. (2015) *"Cloud Tools Tests and its Challenges: A Comparative Study"* in *2nd International Symposium on Big Data and Cloud Computing (ISBCCC '2015)*, p. 482-489.

Pressman, Roger S. (2011), Engenharia de software: Uma abordagem profissional, AMGH, 7ª edição.

Sommerville, Ian (2011), Engenharia de software*,* Pearson Prentice Hall, 9ª edição.

Starov, O., Vilkomir, S., Gorbenko A. e Kharchenko, V. (2015) "*Testing-as-a-Service for Mobile Application: State-of-the-art Survey"* in *Dependability Problems of Complex Information Systems*, p. 55-71.

uTest (2015), *"The Essencial Guide to Mobile App Testing"* in Applause. Disponível em http://go.applause.com/rs/539-CKP-074/images/The-Essential-Guide-to-Mobile-App-Testing.pdf. Maio.

Villanes, I. K., Costa, E. A. B. e Dias-Neto, A. C. (2015) "*Automated Mobile Testing as a Service (AM-TaaS)"* in *IEEE World Congress on Services*, p. 79-86.

Vilkomir, S., Marszalkowski, K., Perry, C. e Mahendrakar, S. (2015) "*Effectiveness of Multi-device Testing Mobile Applications"* in *2nd ACM International Conference on Mobile Software Engineering and Systems*, p. 44-47.