PODER EXECUTIVO
MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DO AMAZONAS
INSTITUTO DE COMPUTAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

Isabel Karina Villanes Rojas

AUTOMAÇÃO DE TESTES PARA APLICAÇÕES MÓVEIS COMO SERVIÇO

AUTOMATED MOBILE TESTING AS A SERVICE AM-TaaS

Manaus
2016

ISABEL KARINA VILLANES ROJAS

AUTOMAÇÃO DE TESTES PARA APLICAÇÕES MÓVEIS COMO SERVIÇO

AUTOMATED *MOBILE TESTING AS A SERVICE* *AM-TaaS*

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Informática, PPGI, da Universidade Federal do Amazonas como requisito parcial para obtenção do grau de Mestre em Informática.

Orientador: Prof. Dr. Arilo Claudio Dias Neto

Manaus
2016

Ficha Catalográfica

Ficha catalográfica elaborada automaticamente de acordo com os dados fornecidos pelo(a) autor(a).

Rojas, Isabel Karina Villanes
R741a Automação de Testes para Aplicações Móveis como Serviço : Automated Mobile Testing as a Service AM-TaaS / Isabel Karina Villanes Rojas. 2016
119 f.: il. color; 31 cm.

Orientador: Arilo Claudio Dias Neto
Dissertação (Mestrado em Informática) - Universidade Federal do Amazonas.

1. Teste de Aplicações Móveis. 2. Testes Automatizados. 3. Teste na Nuvem. 4. Mobile Testing. 5. Mobile Cloud Testing. I. Dias Neto, Arilo Claudio II. Universidade Federal do Amazonas III. Título

FOLHA DE APROVAÇÃO

"Automação de Testes para Aplicações Móveis Como Serviços - AM-TAAS (Automated Mobile Testing As A Service)"

ISABEL KARINA VILLANES ROJAS

Dissertação de Mestrado defendida e aprovada pela banca examinadora constituída pelos Professores:

Arilo Claudio Dias Neto
Prof. Arilo Claudio Dias Neto - PRESIDENTE

Eduardo Luzeiro Feitosa
Prof. Eduardo Luzeiro Feitosa - MEMBRO INTERNO

Prof. Auri Marcelo Rizzo Vincenzi - MEMBRO EXTERNO

Manaus, 12 de Setembro de 2016

A Deus e minha família.

Agradecimentos

A Deus por me dar saúde e ser meu guia nesse caminho, por me ensinar a ser paciente e perseverante e por cuidar de minha família, sem Ele nada teria sido possível.

A meus pais, José Luis e Yolanda Salomé, ao meu pai por me ensinar que a única forma de crescer é por meio do estudo, que aquilo que aprendemos fica conosco, e que as oportunidades são apresentadas uma vez só e temos que aproveitá-las. A minha mãe pelo exemplo de força e coragem que ela me inspira.

A minhas irmãs, Erika, Gloria, Roxana e Cinthia, que sempre me incentivaram a ser perseverante e perseguir meus sonhos pelo seu apoio incondicional em todo momento.

A meu irmão Alan, pelo seu exemplo de perseverança, quem me ensinou que quem luta pelo que sonha, um dia consegue.

A meu noivo César, pelo seu amor e apoio incondicional, certamente sem seu apoio essa tarefa teria sido mais difícil, obrigado por ser meu suporte, amigo e companheiro em todo momento.

A meus sobrinhos Iker, Mía, Linmey, Dayana e Rosita que sempre que os visito irradiam luz, alegria e iluminam meus dias.

A minha avó materna Luz *(in memorian)* pelo seu infinito carinho e minha avô paterna Isabel *(in memorian)* pelo seu constante incentivo no estudo.

De forma especial ao professor Arilo, meu orientador, que me abriu as portas para esse mundo novo para mim até então. Com o senhor aprendi a fazer pesquisa, a me questionar, a buscar soluções toda vez que tinha algum problema, a sair de minha nuvem, a escrever artigos, obrigada pelos conselhos e ser meu guia nessa caminhada e entender meu português.

Aos professores Dr. Auri Vincenzi e Dr. Eduardo Feitosa, por terem aceito ser parte da banca e pelas contribuição ao longo dessa pesquisa.

A todos meus amigos do grupo ExperTS, Awdren, Silvia, Renata, Kariny, Allan, Larissa, Ingrid, Anderson, Juliana, Oswald, Laíza, Josias e todos os que já culminaram; pelo seu apoio, seu tempo e amizade, pelas conversas e conhecimento compartilhado e pela surpresa no meu aniversário, não vou esquecer. Obrigado por me fazer sentir em família.

Ao professor Eduardo Soto, pelos ensinamentos sobre pesquisa, conhecimentos e dicas compartilhadas.

À professora Tayana Uchôa, por me apresentar o mundo da experimentação, e compartilhar seus conhecimentos e dicas de pesquisa. Foram de grande ajuda para a realização desse trabalho.

Aos companheiros do PPGI, Adriana, Elton, Haline, Urique, e todos que me deram suporte e me permitiram ser parte do grupo de estudo no início do mestrado.

Ao INDT pela experiência pesquisa / indústria, por meio da parceria com a UFAM

A todos os participantes que fizeram parte do estudo experimental, obrigada pelo seu tempo e contribuição.

Aos professores e pessoal administrativo do Instituto de Computação pelo suporte durante o tempo do mestrado.

Finalmente, à CAPES pelo apoio financeiro ao longo do mestrado e todos que colaboraram com essa pesquisa.

Resumo de Dissertação apresentada à UFAM/AM como parte dos requisitos necessários para a obtenção do grau de Mestre em Informática (M.Sc.)

AUTOMAÇÃO DE TESTES PARA APLICAÇÕES MÓVEIS COMO SERVIÇO – AUTOMATED *MOBILE TESTING AS A SERVICE* *AM-TAAS*

Isabel Karina Villanes Rojas

Setembro / 2016

Orientador: Prof. Dr. Arilo Claudio Dias Neto

Devido à alta popularização de serviços em nuvem e o uso de uma ampla variedade de dispositivos móveis com diferentes ambientes e plataformas, um novo modelo para oferecer serviço de teste de software surgiu. Chamado de Teste como Serviço (do inglês *Testing as a Service* – TaaS), o qual usa a infraestrutura da nuvem para fornecer serviços de teste sob demanda para usuários finais a qualquer momento.

Muitos estudos apresentam os servidos oferecidos por TaaS e suas vantagens. Com base neste cenário, este trabalho propõe um arcabouço, chamado "*Automated Mobile Testing as a Service*" identificado com as siglas "AM-TaaS" (Automação de Testes para Aplicações Móveis como Serviço), que oferece testes automatizados para aplicações móveis baseados em critérios de teste. Os critérios de teste usados preliminarmente para esta pesquisa foram os critérios publicados pela *App Quality Alliance* (AQuA).

Esta pesquisa avaliou o arcabouço proposto por meio de dois estudos experimentais. Um primeiro, uma prova de conceito sobre a viabilidade do arcabouço proposto e um segundo estudo em ambiente controlado para avalição do arcabouço proposto em comparação a um outro ambiente local de teste de aplicações móveis.

Os resultados de ambos os estudos indicaram que é possível e viável o serviço de teste utilizando recursos na nuvem. O tempo de execução de todo o processo de teste no AM-TaaS foi inferior ao uso do Ambiente de Teste Local usualmente utilizada pelos desenvolvedores ou testadores. Além disso, uma atitude positiva foi percebida por parte dos testadores enquanto ao uso desse modelo de serviço de teste.

Palavras chave: Teste de Aplicações Móveis, Testes Automatizados, Teste na Nuvem.

Abstract of Qualify Proposal presented to UFAM/AM as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

AUTOMAÇÃO DE TESTES PARA APLICAÇÕES MÓVEIS COMO SERVIÇO – AUTOMATED *MOBILE TESTING AS A SERVICE*

AM-TAAS

Isabel Karina Villanes Rojas

September / 2016

Advisor: Prof. Dr. Arilo Claudio Dias Neto

Due high popularization of cloud services and the use of a wide range of mobile devices with different environments and platforms, a new model to offer software test service emerged, called Testing as a Service (TaaS). It uses cloud infrastructure to provide on-demand testing services on cloud for end users at any time.

Nowadays there are many studies about different services about TaaS. Based on this scenario, we propose a framework, called Automated Mobile Testing as a Service (AM-TaaS), which offers automated test for mobile applications based, preliminary, on *App Quality Alliance* (AQuA)'s test criteria.

This research evaluated the proposed framework through two empirical studies. Firstly, a proof of concept regarding the feasibility of the proposed framework and a second study in a controlled environment to evaluate the proposed framework when compared to another local environment for testing in mobile applications.

The experimental results of both studies indicate that the test service is feasible using cloud resources. The running time of the test process in AM-TaaS was less than a Local Environment Test usually used by developers or testers. In addition, a positive attitude of testers was perceived regarding the use of AM-TaaS.

Keywords: Mobile Testing, Automated Testing, Mobile Cloud Testing.

ÍNDICE

LISTA DE SIGLAS E ABREVIATURAS

API	Interface de Programação de Aplicações (em inglês *Application Programming Interface)*
APK	*Android application package*
APPs	Aplicações móveis
AVD	Dispositivo Virtual Android (em inglês, *Android Virtual Device)*
IaaS	Infraestrutura como Serviço (em inglês, *Infrastructure-as-a-Service)*
NIST	Instituto Nacional de Stándares e Tecnologia (*National Institute of Standards and Technology)*
OSS	*Open Source Software*
PaaS	Plataforma como Serviço (em inglês, *Platform-as-a-Service)*
QoS	Qualidade de Serviço (em inglês, *Quality of Service)*
SaaS	Software como Serviço (em inglês, *Software-as-a-Service)*
SDK	Kit para Desenvolvimento de Software (em inglês, *Software Development Kit)*
SLA	Acordo de Nível de Serviço (em inglês, *Service Level Agreement)*
TaaS	Teste como Serviço (em inglês, *Testing-as-a-Service*)
MTaaS	Teste Móvel como Serviço (em inglês, Mobile *Testing-as-a-Service*)
VM	Máquina Virtual (em inglês, *Virtual Machine)*

LISTA DE FIGURAS

LISTA DE TABELAS

CAPÍTULO 1 - INTRODUÇÃO

Neste capítulo são apresentados o contexto do trabalho e o que motivou esta pesquisa. São também apresentados os seus objetivos, a hipótese, a metodologia de pesquisa e a organização deste texto.

1.1. Contextualização e Motivação

O grande avanço tecnológico dos dispositivos móveis e o seu uso junto com uma ampla variedade de modelos disponíveis no mercado resultam na demanda por novas aplicações móveis que ajudem ou facilitem nas tarefas diárias dos usuários. Recentes pesquisas divulgadas por Statista (2016), afirmam que o número de *downloads* de aplicações móveis deve chegar a 268,69 bilhões em 2017. Esse número tende a crescer ainda mais no futuro, e os desenvolvedores precisarão atender a demanda na qualidade de aplicações com o objetivo de obter um melhor destaque nas lojas de aplicações móveis. Em 2015, as receitas de aplicações para dispositivos móveis ascenderam a 41,1 bilhões de dólares. Para 2020 é projetado que os usuários irão gastar mais de 101 bilhões de dólares em aplicações móveis por meio das lojas de aplicações (Statista, 2016).

Aplicações móveis são utilizadas para realizar diversas tarefas do dia-a-dia, desde troca de mensagens, jogos, ferramentas até operações financeiras. É inegável que a complexidade das tarefas realizadas pelas aplicações móveis está aumentando. A procura por aplicações móveis que auxiliem em tarefas cotidianas do usuário são fatores que influenciam no *download* e uso dessas aplicações. Os dados usados nessas aplicações também têm um grande impacto. Por exemplo, aplicações que realizam compras online usando os dados de cartão de crédito do usuário (informações confidenciais) devem ser tratadas com o maior cuidado possível e para garantir a privacidade e confidencialidade desses dados, é necessário que a aplicação passe por um controle de qualidade de forma a proporcionar segurança aos usuários no seu uso.

De acordo com o portal OpenSignal (2015), em 2015 existiam 24.093 dispositivos diferentes. Em dois anos a fragmentação de Android[1], o número de dispositivos mais do que duplicou, saindo de 11.868 dispositivos em 2013, tornando a fragmentação de dispositivos

[1] Fragmentação de Android, entendesse como a diversidade de versões do sistema operacional oferecido por distintas marcas de dispositivos móveis.

móveis em um grande desafio. Este problema tem dificultado a garantia da qualidade de aplicações móveis, pois realizar testes nessa quantidade de dispositivos para afirmar que uma aplicação é compatível com todos os dispositivos no mercado, e que se comporta de forma adequada em todos eles, é quase impossível.

Devido ao problema de fragmentação, combinado à necessidade de se avaliar a compatibilidade de uma aplicação aos diferentes modelos de dispositivos, surge a possibilidade de exploração de um novo modelo de teste em aplicações móveis, definido nesta pesquisa como *Mobile Testing as a Service* (Mobile-TaaS). Segundo Gao *et al.* (2014), os principais desafios a serem enfrentados neste cenário são:

- Testes de aplicações móveis em diferentes plataformas e vários modelos de dispositivos torna-se custoso e tedioso devido à constante atualização de dispositivos móveis e plataformas;
- Falta de infraestrutura e dispositivos disponíveis na nuvem para teste;
- Alta complexidade de testes móveis influenciada pela diversidade de plataformas e tecnologias;
- A escalabilidade de testes realizados na nuvem, como o alto número de requisições de usuários, conexões e tráfego de carga.

Segundo Parveen e Tilley, (2010), a realização de testes de aplicações móveis utilizando os conceitos de computação na nuvem, representa uma evolução na forma atual de realizar testes nestes dispositivos. De acordo com Hurwitz *et al.* (2012), a computação na nuvem (em inglês, *cloud computing*) oferece uma solução rentável e flexível por meio do poder da computação escalável e os diversos serviços (hardware, software, redes e infraestrutura), variedades de aplicações, processos de negócios, oferecidos em larga escala aos usuários, quando e onde eles precisarem.

Em outra definição, Buyya *et al.* (2009) ressaltam que a computação na nuvem está sendo transformada em um modelo que consiste em serviços que são entregues de um modo semelhante aos serviços públicos tradicionais, tais como água, eletricidade, gás e telefonia. Neste modelo, os usuários acessam aos serviços com base nas suas necessidades e pagam com base no seu uso.

A alta demanda de serviços na nuvem implica em se ter ferramentas que possam atender essa demanda, considerando que a computação na nuvem está se tornando um meio de desenvolvimento e entrega de serviços on-line (Riungu *et al.*, 2010a). Resultados estatísticos obtidos pela consultoria Gartner (2014), preveem que o mercado de serviços na nuvem irá crescer em 17,3% até 2018.

Considerando-se a demanda crescente de desenvolvimento e validação de aplicações móveis e devido ao rápido avanço da tecnologia móvel, percebe-se a necessidade de contar com ferramentas na nuvem que auxiliem no teste das aplicações móveis. Com base nessas necessidades e tendências, é que se torna oportuna a proposta do serviço de teste de aplicações móveis na nuvem, definida nesta pesquisa.

1.2. Descrição do Problema

Antes de uma aplicação ser publicada, ela é avaliada de acordo com critérios de aceitação estabelecidos pelas lojas de aplicações móveis. Somente se esses critérios forem atendidos é que a aplicação será disponibilizada aos seus usuários. Previamente a esse processo, a aplicação móvel deve ser submetida a outros testes para assegurar o cumprimento dos requisitos e funcionalidades. Como boa parte das aplicações móveis disponíveis nas lojas são desenvolvidas por pequenas equipes, geralmente o próprio desenvolvedor da aplicação é quem é o responsável por esta tarefa (Wasserman e Fosser 2010), e este nem sempre possui ou conhece as ferramentas que possam ajudá-lo.

Os desenvolvedores tomam decisões sobre o teste das aplicações baseado no seu próprio entendimento e na infraestrutura do ambiente de teste (Vilkomir *et al.*, 2015). Portanto, a tarefa de teste é muitas vezes esquecida ou postergada. Nesse contexto, o problema tratado neste trabalho está relacionado ao ambiente e as ferramentas necessárias para a configuração e execução do teste de aplicações móveis utilizando recursos disponibilizados na nuvem. Assim, os desafios que o desenvolvedor enfrenta são falta de:

- Conhecimento sobre teste de aplicações móveis;
- Conhecimento sobre ferramentas para o teste;
- Conhecimento sobre os processos ou métodos que devem ser seguidos para o teste das aplicações;
- Ambiente de teste;
- Dispositivos móveis disponíveis;
- Tempo para planejar, configurar e executar os testes.

Nesse cenário, a qualidade das aplicações móveis é seriamente comprometida, podendo causar a não aceitação da aplicação nas lojas e, principalmente, a não aceitação da aplicação por parte de seus usuários finais. Por outro lado, a execução dos testes muitas vezes está relacionada ao tempo disponível para essa fase. A automação de teste é uma prática que certamente influencia muito no avanço das tarefas de teste, gerando um impacto na qualidade das aplicações móveis, assegurando que o aplicativo irá funcionar de acordo com as características definidas no processo de desenvolvimento.

Por meio da automação, o tempo de execução do teste ocorre em proporções bem menores em relação ao tempo de execução de um processo de teste manual. Segundo Fewster e Graham (1999), testes manuais que levariam horas para serem concluídos, podem ser executados em minutos, quando automatizados.

Com isso, controlar a qualidade das aplicações móveis é uma tarefa complexa, devido à variedade de plataformas móveis e a diversidade de dispositivos que existem no mercado. Este trabalho visa auxiliar aos desenvolvedores e testadores na tarefa de teste, propondo um arcabouço de suporte para o teste de aplicações móveis.

1.3. Hipótese

A hipótese definida para este trabalho considera o seguinte cenário:

> Uma ferramenta de teste móvel e automação de testes na nuvem, oferecida como serviço, contribui para redução do esforço na execução dos testes das aplicações móveis e redução de custo em relação aos recursos humanos necessários para a execução dos testes.

1.4. Objetivos

Nessa seção serão apresentados os objetivos dessa pesquisa.

1.4.1. Objetivo geral

Desenvolver e avaliar um arcabouço que ofereça um ambiente para execução de testes funcionais automatizados de aplicações móveis como serviço na nuvem, baseado em critérios de teste previamente publicados na literatura técnica ou fornecidos por testadores, visando a detecção automática de falhas nas aplicações móveis antes de sua publicação nas lojas.

1.4.2. Objetivos específicos

Para atingir o objetivo geral desta pesquisa, pretende-se alcançar os seguintes resultados intermediários:

1 Geração do corpo de conhecimento em relação à estrutura e componentes necessários para o ambiente de teste proposto, com a finalidade de implementar um arcabouço.
2 Definição, concepção e desenvolvimento do ambiente de teste, o qual servirá de suporte para o arcabouço.

3 Análise e seleção de critérios de teste que serão parte do arcabouço proposto, levando em consideração sua viabilidade para serem automatizados e factíveis de serem implementados.

4 Análise e seleção de modelos de emuladores, os quais serão utilizados para execução dos testes.

5 Análise e desenvolvimento de scripts de teste para os critérios de testes selecionados.

6 Condução de estudos experimentais.

1.5. Metodologia de Pesquisa

Para atingir os objetivos relacionados a este trabalho, a metodologia usada será baseada na abordagem para desenvolver novas tecnologias de software proposta por Spínola *et al.* (2008). A abordagem é dividida em duas fases: concepção e avaliação. A metodologia é ilustrada na Figura 1.1.

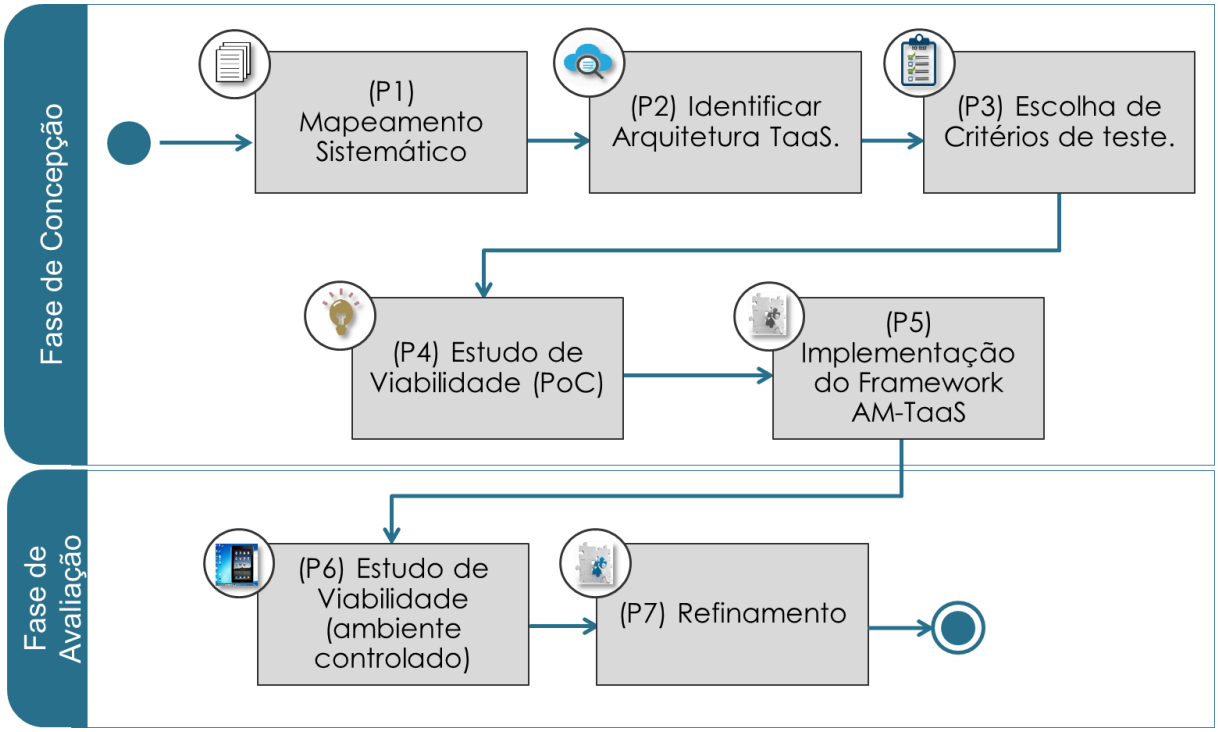


Figura 1.1 Metodologia de pesquisa adotado neste trabalho.

O primeiro passo na fase de concepção compreendeu a realização do mapeamento sistemático (P1) da literatura sobre Teste de Software como Serviço (em inglês, *Testing as a Service* - TaaS), identificando-se as diferentes abordagens (Web, Segurança, Teste web, Desktop e Mobile), assim como suas características, benefícios, problemas e desafios. Logo, identificou-se a Arquitetura do TaaS (P2) em relação aos componentes necessários para sua implementação. Em seguida, foi realizada a escolha de critérios de teste (P3) que

foram usados para o desenvolvimento da prova de conceito (P4), como forma de validar a viabilidade da proposta. Finalmente foi realizada a implementação do AM-TaaS (P5).

A fase de avaliação iniciou com a execução de um experimento com AM-TaaS em um ambiente controlado (P6) a fim de avaliar sua viabilidade. Após este estudo, foi realizado o refinamento do arcabouço proposto (P7).

1.6. Organização do Documento

Este primeiro capítulo apresentou o contexto, no qual este trabalho está inserido, a motivação para realizá-lo, o objetivo a ser atingido e a metodologia a ser seguida. Além deste capítulo, a organização do texto está estruturada em mais cinco capítulos, descritos a seguir:

Capítulo 2: aborda o referencial teórico, no qual esse trabalho é baseado com o seguinte conteúdo: (2.1) Computação na nuvem, (2.2), Teste como um Serviço, (2.3) Teste de Aplicações Móveis, (2.4) Teste de Aplicações Móveis como um Serviço e (2.5) Considerações Finais.

Capítulo 3: são apresentados o (3.1) Protocolo e mapeamento sistemático e (3.2) Trabalhos relacionados referentes ao tema desta pesquisa, os resultados obtidos neste mapeamento sistemático e, finalmente, uma comparação entre eles.

Capítulo 4: nesse capítulo são apresentadas as seguintes subseções: (4.1) Visão Geral do Arcabouço proposto, (4.2) Infraestrutura, (4.3) Componentes, (4.4) Implementação e ferramentas utilizadas e finalmente a (4.5) Prova de conceito executada e os resultados obtidos.

Capítulo 5: este capítulo apresenta o planejamento e resultados do estudo de viabilidade em ambiente controlado, nas seguintes subseções: (5.1) Definição do Estudo de Viabilidade, (5.2) Planejamento do Estudo, (5.3) Execução do Estudo, (5.4) Ameaças a Validade e finalmente os (5.5). Resultados e a análise estatística.

Capítulo 6: São apresentadas as (6.1) Considerações finais, (6.2) Contribuições, (6.3) Limitações e (6.4) Trabalhos futuros.

CAPÍTULO 2 - REFERENCIAL TEÓRICO

Neste capítulo serão apresentados os conceitos relacionados à Computação na Nuvem, Teste como serviço (TaaS), Teste Móvel como Serviço, Virtualização e Emulação de Dispositivos móveis, que são áreas envolvidas dentro desta pesquisa.

2.1. Computação na Nuvem (*Cloud Computing*)

Na década de 60s Mccarthy (1961), propôs a ideia de que a computação deveria ser organizada na forma de um serviço de utilidade pública, assim como os serviços de telefonia, em que os usuários pagam apenas pelo que usam. Já em 1997, surgiu a primeira definição acadêmica do termo “computação na nuvem”, divulgada por Chellappa (1997), como sendo um novo “paradigma de computação, no qual as fronteiras da computação serão determinadas pela racionalidade econômica, ao invés dos limites técnicos”.

O termo nuvem (em inglês, *cloud*) já havia entrado em uso no início dos anos 90. No entanto, o termo computação na nuvem (em inglês, *cloud computing*) foi usado pela primeira vez em uma conferência da empresa Google (Schmidt, 2006). O termo se referia a como eles chamavam seus *data centers* (local onde os serviços, dados e arquitetura estão armazenados) no qual, independente do hardware (computador, dispositivo móvel ou outro dispositivo) e do sistema operacional, todos poderiam ter acesso aos recursos na nuvem. Desde então, muitas definições sobre computação na nuvem foram publicadas.

Computação na nuvem é um modelo que visa permitir acesso ubíquo, conveniente e sob demanda, via rede, a um agrupamento compartilhado e configurável de recursos computacionais (por exemplo; redes, servidores, equipamentos de armazenamento, aplicações e serviços), que podem ser rapidamente fornecidos e liberados com esforços mínimos de gerenciamento ou interação com o provedor de serviços (Mell e Grance, 2011).

Segundo Riungu *et al.* (2010b), a computação na nuvem é uma tecnologia que está mudando a forma como o software é produzido e consumido, principalmente mudando o uso do tradicional computador pessoal para os serviços de software online.

A principal ideia da computação na nuvem é de utilizar, em qualquer lugar e independentemente da plataforma, as mais variadas aplicações por meio da Internet com a mesma facilidade de tê-las instaladas em nossos computadores. Computação na nuvem

suporta o modelo *qualquer-coisa como serviço* (em inglês, *Anything as a Service - XaaS*).

Segundo Mell e Grance (2011), este modelo de nuvem é composto por cinco características essenciais, definidas da seguinte forma:

1. **Autoatendimento sob demanda (*On-demand self-service*):** o consumidor deve ser capaz de alocar novos recursos automaticamente, sem interação humana com o provedor de serviços;
2. **Amplo acesso à rede (*Broad network access*):** os recursos devem estar disponíveis através da rede e devem ser acessíveis por mecanismos padrão, permitindo seu uso por diferentes dispositivos, tais como computadores pessoais, *smartphones* ou *tablets*;
3. **Agrupamento de Recursos (*Resource pooling*):** os recursos computacionais do provedor de serviços devem ser agrupados para servir a múltiplos consumidores, com recursos físicos e virtuais sendo arranjados e rearranjados dinamicamente conforme a demanda desses consumidores. Deve haver um senso de independência de localização, no qual o consumidor não tem um controle exato de onde os recursos utilizados estão localizados, mas deve ser possível especificar esse local em alto nível de abstração (país, unidade federativa ou *data center*);
4. **Elasticidade rápida (*Rapid elasticity*):** os recursos devem ser alocados e liberados de forma elástica e de forma automática em alguns casos, permitindo a rápida adaptação à demanda. Para o consumidor, os recursos disponíveis devem parecer ser ilimitados, sendo possível alocar a quantidade desejada desses recursos a qualquer momento;
5. **Serviços Mensuráveis (*Measured service*):** serviços de computação em nuvem devem controlar e otimizar os recursos de maneira automática, disponibilizando mecanismos para medir esses recursos, utilizando um sistema de medida apropriado para o tipo de recurso sendo utilizado (por exemplo, quantidade de espaço de armazenamento, velocidade de comunicação, capacidade de processamento e número de usuários ativos). Deve ser possível monitorar, controlar e consultar o uso dos recursos, provendo transparência para o consumidor e para o provedor dos serviços.

Além das características essenciais, foram definidos quatro modelos de desenvolvimento dentro do contexto da computação na nuvem que são descritos abaixo:

1. **Nuvem Privada (*Private cloud*):** a infraestrutura é provisionada para uso exclusivo por uma única organização; pode ser gerenciada e operada pela organização, por um terceiro ou uma combinação dos dois.
2. **Nuvem Comunitária (*Community cloud*):** a infraestrutura é provisionada para uso exclusivo de uma comunidade específica de consumidores de organizações com interesses comuns.
3. **Nuvem Pública (*Public cloud*):** a infraestrutura é provisionada para uso aberto ao público em geral, pode ser operada por uma organização de negócios, acadêmica ou governamental ou uma combinação entre elas.
4. **Nuvem Híbrida (*Hybrid cloud*):** a infraestrutura é a composição de dois ou mais modelos de desenvolvimento (privada, comunitária ou pública) que permanecem como entidades únicas, mas são ligados por tecnologias padronizadas ou proprietária que permite a portabilidade de dados.

E finalmente, foram definidos três modelos de serviço, que são representados por meio de uma pirâmide com três camadas. A descrição de cada modelo (camada) é descrita abaixo:

Infraestrutura como Serviço (*Infrastructure as a Service – IaaS*): é a camada inferior e é composta por plataformas para o desenvolvimento, teste, implantação e execução de aplicações proprietárias, todas oferecidas como um serviço terceirizado. O principal objetivo é tornar mais fácil e acessível o fornecimento de recursos, como servidores, redes, *firewall*, armazenamento e outros que são fundamentais na construção de um ambiente sob demanda podendo ser tanto sistemas operacionais quanto aplicações. O *Amazon EC2* (em inglês, *Elastic Cloud Computing*) (AWS, 2014) e o *Hyper-V Cloud* (Microsoft, 2014) são exemplos desse tipo de serviço.

Plataforma como Serviço (*Platform as a Service – PaaS*): é a camada intermediária do modelo conceitual, sendo composta por hardware virtual disponibilizado como serviço. Oferece tipos específicos de serviços como sistemas operacionais, banco de dados, serviços de mensagens, serviços de armazenamento de dados, dentre outros. Uma PaaS fornece ambientes de desenvolvimento de software e facilita a implantação de aplicações sem os custos e complexidades relativos a compra e gerenciamento do hardware e de software adjacentes que são necessários ao ambiente de desenvolvimento. O provedor PaaS gerencia as atualizações e manutenções de rotina. Por exemplo, *Google App Engine* (Google, 2014) permite executar suas aplicações da Web na infraestrutura da Google. No qual é possível usar tecnologia Python ou Java para criar suas aplicações Web.

Software como Serviço (*Software as a Service - SaaS*): é a camada mais externa do modelo conceitual. Está baseado no conceito de alugar software de um determinado provedor em vez de comprá-lo de maneira convencional (por exemplo, adquirindo um DVD do produto). O software está hospedado em servidores de rede centralizada para tornar a funcionalidade disponível pela web ou intranet. Também conhecido como “Software sob demanda”, é atualmente o mais popular tipo de computação em nuvem por causa de sua alta flexibilidade, serviços excelentes, maior escalabilidade e menor manutenção.

Neste modelo, o provedor do serviço cobra uma taxa para disponibilizar o serviço e dar o suporte, o software é utilizado 100% através da web, ou pode ter uma instalação local, como antivírus, serviços de backup, bancos de dados, etc. A característica principal é a não aquisição das licenças (mas sim pagar pelo uso como um "Serviço") e a responsabilidade do provedor pela disponibilização do sistema em produção. Como exemplos desses serviços, podem ser citados o *Skype*, *Google Docs* e *Dropbox*.

A pirâmide da nuvem apresentada na Figura 2.1 foi proposta por Sheehan (2008). Ela surgiu como uma proposta para diferenciar e agrupar os modelos de serviços oferecidos pela Computação na nuvem. Usando a pirâmide invertida é possível mostrar o número de provedores para cada grupo. Como é de se esperar, o número de provedores do modelo *IaaS* é menor quanto ao modelo *SaaS*, isso devido ao nível de complexidade do modelo de serviço.

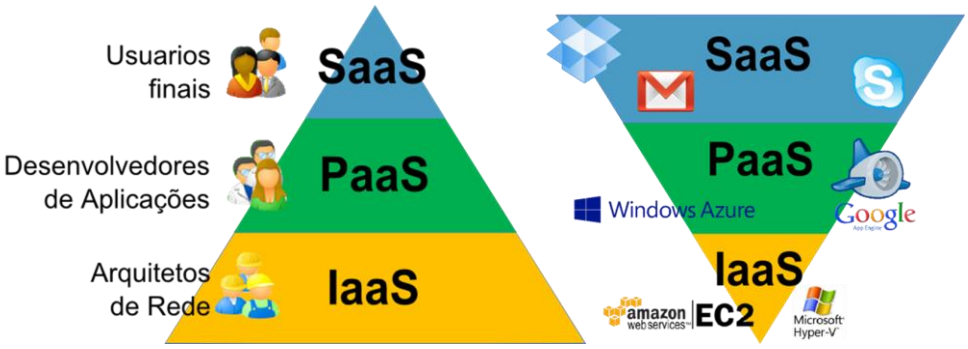


Figura 2.1. A pirâmide da nuvem e Número de provedores (Sheehan, 2008).

O novo modelo XaaS é acrescentado na pirâmide da nuvem, como é ilustrado na figura Figura 2.2, a qual que permite descrever os novos tipos de serviço na nuvem, como

DBaaS – *Database-as-a-servic*e (Banco de dados como serviço), SaaS – *Security-as-a-service* (Segurança como serviço), MaaS – *Marketing-as-a-service* (Marketing como serviço), Testing as a Service (Teste como um Serviço), Mobile Testing as a Service (Teste Móvel como um Serviço), dentre outros.

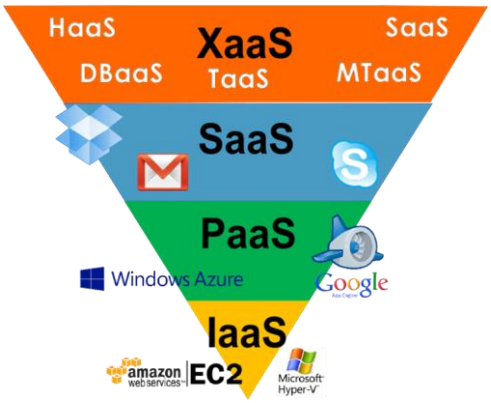


Figura 2.2. Incorporação do XaaS na pirâmide da nuvem (número de provedores).

2.2. Teste como um Serviço (*Test as a Service* – TaaS)

Desde 2009, a computação na nuvem vem ganhando uma nova área de pesquisa que oferece mais um novo modelo de serviço: ***Teste como Serviço* (*Test as a Service* – TaaS)**. TaaS é definido como um modelo de teste de software usado para testar uma aplicação oferecida como um serviço de um provedor para um cliente por meio do Internet (Aalst, 2008).

TaaS oferece a habilidade de testar o software na nuvem como um Serviço Web (em inglês, *Web Service)* de forma competitiva e de fácil acesso, permitindo aproveitar a vasta quantidade de recursos que são providos pela infraestrutura na nuvem (Candea *et al.*, 2010a). TaaS é o ato de prover serviços de teste estáticos e dinâmicos, sob demanda, sobre a nuvem em qualquer tempo e momento. Atraindo novas oportunidades de negócio, técnicas de teste, padrões de Qualidade de Serviço (*em inglês Quality of Service, QoS*), requerimentos e novos desafios (Gao *et al.*, 2011).

O TaaS é executado como parte de todas as três camadas dentro da pirâmide da nuvem, uma vez que pode ser utilizado para testar a infraestrutura, a plataforma, um software (aplicação) ou a combinação dentre elas (Roodenrijs, 2011). Esta representação é ilustrada na Figura 2.3.

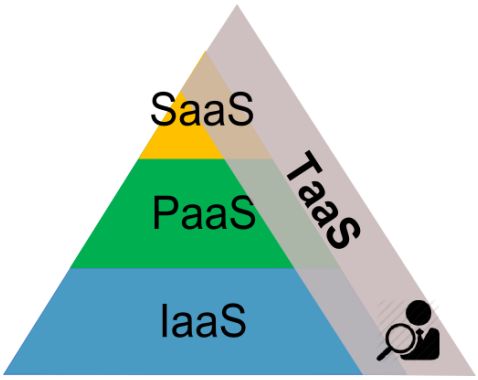


Figura 2.3. TaaS e o efeito sobre as três camadas (Roodenrijs, 2011).

O TaaS pode ser visto como a entrega de testes, ferramentas e pessoas em conjunto como parte de um serviço, que tem que ser baseado no modelo "*pay-per-use*", sem compromissos adiantados. TaaS fornece um modelo de serviço sob demanda em que as aplicações que se encontram em processos de validação de software são processadas num ambiente de teste na nuvem, baseadas em um Acordo de Nível de Serviço (*Service Level Agreements* – SLAs). O TaaS baseia-se no faturamento da utilidade, o que permite que os usuários finais obtenham diferentes tipos de serviços de teste usando o modelo "pagar pelo que você testa" (*pay-as-you-test*), garantindo assim uma redução de custos (Gao *et al.*, 2013).

2.2.1 Técnicas de Teste na Nuvem

Segundo Bai *et al.* (2011), existem diversas técnicas de teste que podem ser aplicadas na nuvem. Algumas delas são:

Simulação: para reduzir a complexidade e melhorar a qualidade de testes, é necessário um simulador de nuvem para a realização dos testes. Por meio do simulador, o testador pode se concentrar em problemas de qualidade em um determinado componente e analisar o comportamento do sistema em vários cenários.

Serviço Simulado: os sistemas hospedados na nuvem podem usar serviços de diversos fornecedores. Técnicas convencionais de simulação precisam ser adaptadas para simular as funcionalidades externas ao nível de serviço de interface.

Teste Paralelo: a programação paralela tornou-se uma prática natural para aplicações construídas para nuvem. Os testes podem se beneficiar dessa prática, dividindo as atividades de teste em trabalhos independentes e paralelizando-os, reduzindo assim o tempo de teste e o custo.

Virtualização: ao longo do ciclo de vida do software é necessário manter diferentes ambientes de teste para várias versões ou plataformas do software. Neste cenário, as máquinas virtuais podem ajudar a aliviar e acelerar o processo e reduzir seu custo. Devido ao fato de que este trabalho está baseado na virtualização, é realizado um estudo mais aprofundado.

O conceito de virtualização é amplamente usado na literatura. Segundo Barham *et al.* (2003), virtualização é a criação de recursos de computação abstratas a partir de recursos físicos. Um exemplo comum é a tecnologia de virtualização de servidores, em que uma máquina física é usada para suportar várias máquinas virtuais (*em inglês, Virtual Machines - VMs*), que podem operar em paralelo.

O componente responsável pelo gerenciamento dos recursos virtuais em um sistema de virtualização é chamado de *Hypervisor.* Ele é um programa que permite que múltiplos sistemas operacionais compartilhem o hardware de um único *host*. Cada sistema operacional simula ter processador, memória e outros recursos do hospedeiro só para ele. No entanto, a principal atividade do *Hypervisor* é controlar o processador e recursos, alocando o que é necessário para cada sistema operacional e certificando-se de que os sistemas operacionais convidados, chamados de máquinas virtuais, não possam perturbar o outro. Existem dois tipos de *Hypervisor*, *Native* e *Hosted,* ilustrados na Figura 2.4:

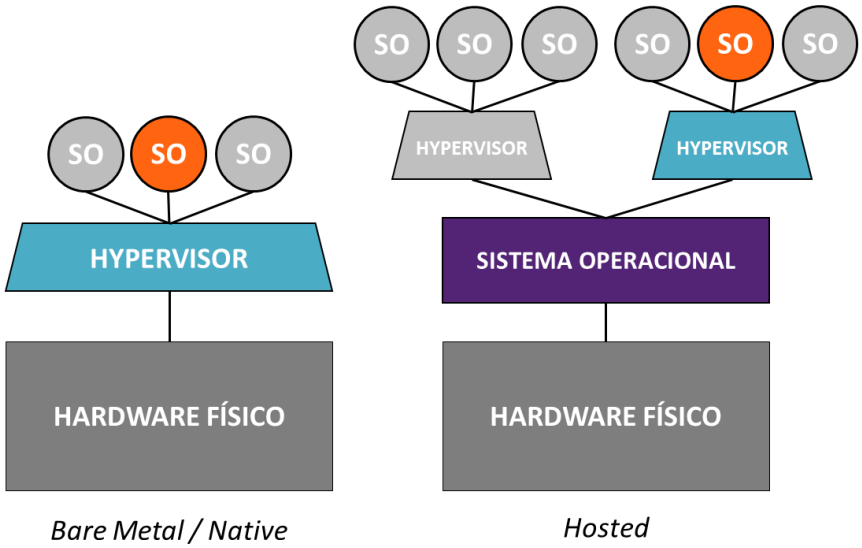


Figura 2.4. Tipos de *Hypervisor*.

- ***Hipervisor* tipo 1:** Também denominado nativo, *unhosted* ou *bare metal*, é um *software* que é executado diretamente sobre o hardware para oferecer a funcionalidade descrita. Como exemplos, podem ser citados *VMware ESXi*, *Xen*, *Microsoft Hyper-V Server*.

- ***Hipervisor* tipo 2:** Também denominado *hosted*, é um software que executa sobre um sistema operacional para oferecer a funcionalidade descrita. Como exemplos, podem ser citados *VirtualBox*, *VMware: Workstation*, *QEMU*.

Para o desenvolvimento do arcabouço desse trabalho, serão usados a virtualização e o *Hypervisor*, que será explicado no Capítulo 4.

2.3. Teste de Aplicações Móveis

O termo Teste para Dispositivo Móvel (do inglês, *Mobile Testing*) refere-se aos diferentes tipos de teste, como os testes de aplicações móveis nativos, aplicações web e dispositivos móveis. Teste de aplicações móveis utiliza diversas ferramentas para garantir a qualidade, funcionalidade, comportamento, desempenho e qualidade de serviço. Assim como, características de mobilidade, usabilidade, interoperabilidade, conectividade, segurança e privacidade (*Gao et al.*, 2014).

De acordo com Wasserman e Fosser (2010), testes de aplicações móveis têm algumas similaridades com testes de aplicações web, uma vez que os dois envolvem validação em muitos ambientes (e.g.: modelos de dispositivos móveis e navegadores, respectivamente). Porém, testes em aplicações móveis têm características específicas, tais como a necessidade de avaliar o software em diferentes plataformas, sistemas operacionais, tamanho de telas, memória, capacidade de processamento, marcas e comunicação por diversas redes. Dessa forma, realizar testes com todas as combinações de características da plataforma móvel pode ser uma tarefa inviável.

2.3.1 Critérios de Teste de Aplicações Móveis

Devido à alta diversidade de dispositivos móveis, é necessário definir critérios de teste para as aplicações móveis. Nesse contexto, existem critérios para realização de testes em aplicativos móveis publicados por grupos de desenvolvedores e empresas, como os definidos pela AQuA[2] (em inglês, *App Quality Aliance)*. AQuA é uma organização global sem fins lucrativos liderada por membros das principais indústrias de eletrônicos – AT&T, LGE, Motorola, Nokia, Oracle, Samsung e Sony Mobile – que ajudam a indústria a melhorar a qualidade das aplicações móveis por meio de diretrizes sobre os casos de teste que devem ser executados em uma aplicação. Essas diretrizes são separadas por plataforma, assim como critérios definidos por outras organizações e grupos. Esses critérios servem de base para que os desenvolvedores possam garantir a qualidade de uma aplicação e estão

[2] http://www.appqualityalliance.org

disponibilizados (em alguns casos) para o uso livre, como os da plataforma Android, iOS e Windows Phone.

Os critérios orientados para plataforma Android da AQuA sugerem os procedimentos e passos que devem ser seguidos para a realização dos testes; os desenvolvedores devem ler e seguir as instruções para saber se sua aplicação foi aprovada ou não nos testes realizados. O formato de um caso de teste contém: o título do Caso de Teste, uma descrição sobre o caso de teste, os requisitos prévios, os passos que devem ser executados, o resultado esperado e finalmente um espaço para resposta indicando se o teste passou (em inglês, *Pass*) ou falhou (em inglês, *Fail*).

Devido à variedade de aplicações móveis, existe uma necessidade de adaptar os testes de acordo com o nível de complexidade das mesmas. Para lidar com isso, AQuA define dois tipos de aplicações móveis: simples e complexas, um exemplo de caso de teste para cada tipo de aplicação é apresentada na Figura 2.5 e na Figura 2.6, respetivamente.

A. ***Aplicações simples*** são aquelas que:

- Não enviam mensagens SMS / MMS;
- Não gravam dados em arquivos de dados padrão (e.g.: contatos, calendário);
- Não gravam dados em serviços externos (e.g.: redes sociais);
- É possível acessar, mas não alteram o estado dos serviços de rede (e.g.: 3G, WiFi, *Bluetooth*);
- Podem ler arquivos de dados padrão e/ou SMS / MMS;
- Podem acessar a tela, câmera, som e teclado;
- São capazes de armazenar seus dados (e.g.: armazenar fotos, criar documentos).

Essa lista não é exaustiva, mas pode cobrir 90% das aplicações disponíveis no mercado. Alguns casos de teste que são exigidos para aplicações simples são listados abaixo. A lista completa pode ser encontrada no site da AQuA.

- *Instalação OTA - (1.1) OTA Install;*
- Tempo longo de Iniciação - *(1.2) Long launch time;*
- Enviar e receber dados - *(3.1) Send/Receive Data;*
- Descarga de Recursos - *(3.4) Resource downloading;*
- Receber mensagens - *(5.2) Message – Receive;*
- *Receber chamadas - (5.3) Incoming call;*
- Diferentes tamanhos de tela - *(7.11) Different screen sizes;*

- Erros de ortografía - *(7.14) Spelling errors;*

B. ***Aplicações* complexas** são aquelas que não se adequam à categoria simples, e são consideradas complexas. Elas são expostas a outros critérios de testes. Alguns casos de teste que podem ser aplicados a esse tipo de aplicação são:

- Mover o aplicativo móvel para a memória externa – *(1.3) Move App to external memory (SD card);*
- Outras interrupções – *(6.3) Other Interruptions;*
- Manipulação de entradas em vários formatos - *(7.12) Multiple format input handling;*
- Envio de mensagens - *(5.1) Message Send.*

Seguindo os passos da metodologia de pesquisa descrita no Capítulo 1, no Passo 3 (P3), Escolha dos Critérios de Teste, foi realizada uma análise dos 65 Casos de Teste apresentado nas diretrizes da AQuA. Destes, foram identificados apenas 6 (9%) Casos de Teste que poderiam ser automatizados. A principal dificuldade para serem automatizados é a diversidade de funcionalidades que apresenta uma aplicação. O caso de teste *OTA Install* será usado como parte da prova de conceito durante o desenvolvimento desse trabalho. Porém, o trabalho proposto poderá usar outros critérios de teste.

Test ID **5.3**	*Test Title* **Telephone call – incoming while application in use**	**Critical**
Test Description **If the user accepts an incoming phone call while the Application is running, it should be possible to resume from the same point in the Application at the end of the call, or a logical re-starting point.**		
Required for: **All applications.**		
Testing Note *Testing Steps* 1. While Application is running, make an incoming call to the test handset. 2. Accept the incoming call. 3. End the incoming call. 4. Return to the Application. *RESULT:* 1. The incoming call dialog is shown. 2. After the call is taken and ended, the Application should resume to either the point of interruption, or a point which neither inconveniences the user nor causes data loss.		
Result of Test ☐ **PASS** ☐ **FAIL**		

Figura 2.5. Exemplo de Caso de Teste "Interrupção da aplicação com ligação telefônica" para uma aplicação simples.

Test ID 1.3	*Test Title* **Move to external memory (SD Card)**	**Critical**
Test Description **The Application must move from the main memory of the device to the external memory (SD card) and back again.**		
Required for: **All devices with external memory** *Not required for:* **Application which technically requires to run from main memory**		
Testing Note *If the device supports an Internal memory (second SD card) then this test must be repeated for that location.* *Testing Steps* 1. Move the app from the main memory to the SD card using the device settings menu 2. Launch the app and check the saved data. 3. Update the saved data (if possible). 4. Exit the app 5. Move the app back to the device main memory 6. Launch the app and check the saved data *RESULT:* 1. Correct launch and function of the app 2. No loss of saved data		
Result of Test ☐ **PASS** ☐ **FAIL** **This test is not applicable where...** ☐ Application which technically requires to run from main memory.		

Figura 2.6. Exemplo de Caso de Teste " Mover o aplicativo móvel para a memória externa" para uma aplicação complexa.

2.4. Teste de Aplicações Móveis como um Serviço (*Mobile Testing as a Service – Mobile TaaS*)

Com o incremento do uso de smartphones e, consequentemente, o uso das aplicações móveis, surgiu a ideia do teste móvel como serviço. A primeira definição sobre *Mobile Testing-as-a-Service* (*Mobile TaaS*) foi dada por Gao *et al.* (2014). Neste trabalho, *Mobile TaaS* visa fornecer serviços de teste sob demanda para aplicações móveis e/ou SaaS com o objetivo de apoiar os processos de validação de software e engenharia de qualidade, aproveitando um ambiente de teste móvel escalável baseado na nuvem para garantir os requerimentos pré-definidos de Qualidade de Serviço e o SLA.

Mobile TaaS oferece um novo modelo de negócio para diversos serviços de validação de software móvel, usando o modelo *pay-as-you-test* para garantir o compartilhamento e redução de custos em recursos de computação móvel, redes e infraestrutura de armazenamento. Ele é baseado na computação em nuvem móvel (em inglês *Mobile Cloud Computing - MCC*).

Assim, segundo Al-Ahmad e Aljunid (2013), a abordagem MCC é a interseção entre computação em nuvem e móvel, permitindo aos desenvolvedores criar aplicações com serviços avançados na nuvem que têm menos limitações em relação a aplicações móveis nativas. No entanto, MCC vem com maior complexidade que afeta não só a confiabilidade e entrega do ciclo de desenvolvimento. Essa arquitetura é ilustrada na Figura 2.7. Esta abordagem também é definida como o acesso à nuvem por meio de dispositivos móveis que formam parte de uma construção nuvem maior (Warner e Karman, 2010).

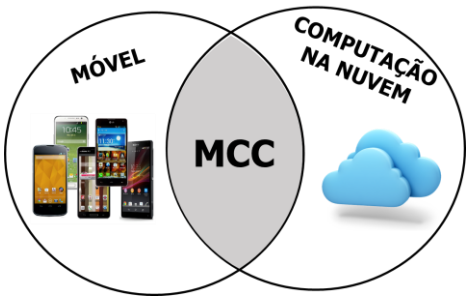


Figura 2.7. *Mobile Cloud Computing* - MCC (Al-ahmad e Aljunid, 2013).

2.5. Considerações Finais

Neste capítulo foram considerados os principais conceitos sobre Computação na Nuvem, Teste como Serviço (TaaS) e suas características. Logo é descrito o teste de aplicações móveis e seus desafios, assim como os critérios que são usados para testá-los e finalmente o Teste de Aplicações Móveis como um Serviço (*Mobile Testing as a Service*). Assim, optou-se pela realização de um mapeamento sistemático para entender o funcionamento e a implementação dos serviços de teste na nuvem. O resultado desse mapeamento sistemático e os trabalhos correlatos são apresentados no próximo capítulo.

CAPÍTULO 3 –TESTE COMO SERVIÇO E SEU USO EM DIFERENTES CONTEXTOS

Nesse capítulo serão apresentados o mapeamento sistemático e as pesquisas correlatas que foram identificadas na literatura. Os trabalhos relacionados foram classificados em três categorias: Infraestruturas para TaaS, Virtualização e Emulação de Dispositivos Móveis e no final foram apresentadas as Ferramentas para aplicações móveis.

3.1 Protocolo do Mapeamento Sistemático

3.1.1 Questões de Pesquisa

O objetivo deste estudo será projetado a partir do paradigma GQM (em inglês, *Goal, Question and Metric*) (Basili *et al.*, 1994) o qual é apresentado na Tabela 3.1. A execução do mapeamento sistemático teve como principal objetivo identificar as principais abordagens que citam as características do TaaS, assim como os principais contextos nos quais são aplicados, além dos benefícios e desafios desse novo modelo de serviço.

Tabela 3.1 Objetivo do Mapeamento Sistemático segundo o Paradigma GQM.

Analisar	As infraestruturas propostas para o contexto de Teste de Software como Serviço (*Test as a Service* - TaaS)
Com o propósito de	Identificar e caracterizar tais infraestruturas
Com relação à	Contextos, benefícios, desafios, implementação, desenvolvimento, ferramentas e abordagens das infraestruturas
Do ponto de vista do	Testador e Pesquisador
No contexto de	Acadêmico e industrial que estejam relacionadas ao TaaS

Para responder aos objetivos desta pesquisa, foram definidas as seguintes questões de pesquisa:

- QP1: Quais são as principais características do *Testing as a Service* (TaaS)?
- QP2: Quais são as principais problemas, desafios e benefícios do TaaS?

- QP3: Quais são os contextos nos quais é aplicado o TaaS?
- QP4: Quais são os processos dentro da construção/implementação de uma Infraestrutura TaaS?

3.1.2 Identificação e Seleção de Estudos Primários

A *string* de busca foi definida segundo o padrão PICO (*population, intervention, comparison, outcomes*) (Kitchenham e Charters, 2007), que está descrita abaixo:

String de Busca: ("Cloud") AND ("taas" OR "Test as a service" OR "Testing as a service" OR "Test* service*" OR "Cloud test*") AND (“Infrastructure" OR "architecture" OR "tool" OR "application" OR "Implementation" OR "Technology" OR "environment")

Foram realizadas buscas nas bibliotecas digitais online: *Scopus*[3], *IEEEXplore*[4] e *ACM Digital Library*[5]. Os critérios de inclusão de estudos para serem avaliados e/ou analisados:

- Os artigos devem estar disponíveis na Internet ou poder ser obtidas por meio do contato com os autores;
- O artigo deve estar escrito em inglês;
- Os artigos devem contemplar temas que tenham relação a desafios, limitações, infraestrutura, arquitetura, ferramentas, implementação e tecnologias referentes a *Testing as a Service* (TaaS) ou *Cloud Testing*.

Qualquer artigo que não atenda os critérios de inclusão foi excluído.

3.1.3 Formulário de Extração de Dados

Para cada artigo selecionado foram extraídas informações necessárias que permitissem responder às questões da pesquisa e obter um corpo de conhecimento sobre TssS. Essas informações são listadas na Tabela 3.2.

[3] http://www.scopus.com/

[4] http://ieeexplore.ieee.org/Xplore/home.jsp

[5] http://dl.acm.org

Tabela 3.2. Formulário de Extração de Dados.

Campos	Descrição
Título	O título do trabalho
Autor(es)	O(s) autor(es) do trabalho
Palavras-chave	Palavras-chaves do trabalho
Fonte	Onde o trabalho foi publicado
Tipo de estudo	Caso de estudo, experimento, relatório de experiência, pesquisa de opinião
Tipo de contribuição	Infraestrutura, arquitetura, protótipo, implementação, ferramenta, abordagem, técnica
Ferramentas	Ferramentas utilizadas pelos autores
Plataforma	IaaS, PaaS, SaaS
Nível de teste	Unitário, Integração, Sistema, Aceitação
Tipo de teste	Funcional, Não funcional (segurança, carga, desempenho)
Linguagem	Java, Android, C/C++ ou outra
Tipo de equipamentos	PCs, servidores, Memoria, CPU, dentre outros;
Desafios do TaaS	Desafios apontados pelo (os) autor (es);

3.1.4 Execução do Estudo de Mapeamento

A Tabela 3.3 apresenta o resumo dos artigos encontrados e selecionados. Foram encontrados 336 artigos nas bibliotecas definidas. Após a aplicação do primeiro filtro, baseado na leitura do título, palavras-chave e resumo, 67 artigos foram selecionados. No segundo filtro, baseada na análise completa do artigo, 30 artigos foram selecionados para a extração de informações.

Tabela 3.3. Resultado dos Filtros da Revisão Sistemática.

Fonte	Artigos Selecionados	Primeiro Filtro	Segundo Filtro
SCOPUS	161	48	21
IEEE	55	4	3
ACM	120	16	6
Total	**336**	**67**	**30**

Os artigos encontrados foram publicados entre 2010 e 2014, como mostrado na Figura 3.1. O fato de não se encontrar artigos em anos anteriores ao 2010 deve-se a que o tema ainda estava surgindo em 2009, enquanto que em 2010 começou a ser um tema de maior interesse na literatura técnica. A lista completa dos artigos selecionados está apresentada no Apêndice A.

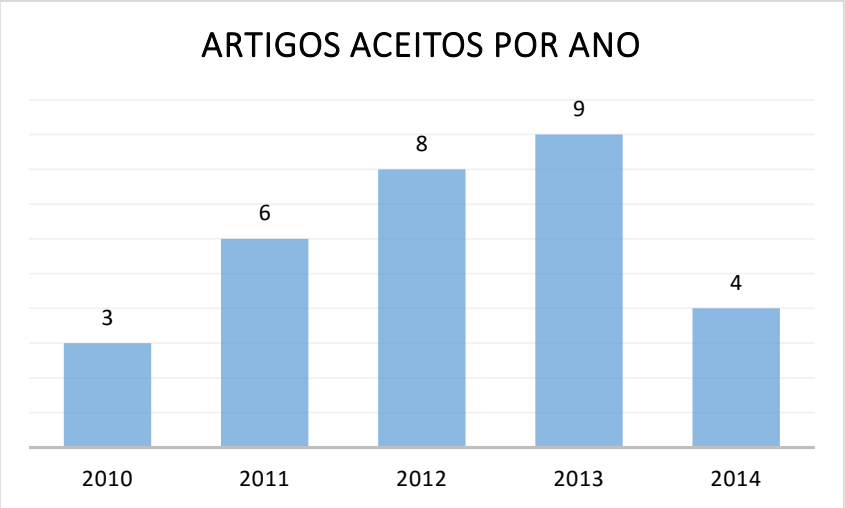


Figura 3.1. Distribuição dos artigos selecionados por ano.

3.2 Trabalhos Relacionados

Os trabalhos relacionados foram classificados seguindo as diretrizes de Priyanka *et al.* (2012) e Bai *et al.* (2011), que realizaram, respectivamente, uma revisão sistemática e uma análise sobre as técnicas de teste baseadas na nuvem.

3.2.1. Infraestruturas para TaaS

No contexto do Teste como Serviço (*Test as a Service* – TaaS), uma das primeiras infraestruturas para plataforma móvel foi proposto por Gao *et al.* (2014), na qual visa suportar os testes de aplicações móveis baseados na nuvem em três abordagens diferentes: emulação, simulação e dispositivos móveis. Destes, o mais relevante para essa pesquisa é o Teste baseado na emulação, que possui cinco componentes: *Mobile emulator manager, Mobile emulator controller, Mobile emulator test manager, Mobile network connection manager* e *Mobile emulation-based test automation manager*. Esse último componente é quem controla e gerencia a execução de scripts de teste em um ou vários dispositivos móveis emulados. No entanto, eles não consideram a automação de teste como um processo dentro da infraestrutura.

No mesmo contexto móvel, Huang (2014) apresenta uma arquitetura chamada "*Mobile App Automated Compatibility Testing Service* AppACTS". Nesta proposta, o componente principal (*Master Node)* é responsável por alocar, monitorar e executar os testes designados a cada dispositivo móvel. A plataforma usa dispositivos físicos que são espalhados em diferentes *data centers*, que são conectados por meio de uma interface padrão (USB) e podem ser acessados via a arquitetura. A desvantagem encontrada é que

cada dispositivo móvel pode ser usado só por uma aplicação e um usuário por vez. Além disso, essa solução não usa virtualização, entretanto a considera como trabalhos futuros.

Starov e Vilkomir (2013) sugerem uma proposta denominada "*Cloud Testing of Mobile Systems – CTOMS",* dividida em três camadas: (1) *Presentation*, responsável pelo acesso web; (2) *Platform*, implementada por meio da ferramenta *Google App Engine - GAE* Google (2014) à qual é responsável pelo gerenciamento e distribuição de testes nos dispositivos móveis conectados na plataforma; e (3) *Information*, responsável pelo relacionamento entre os testes e as ferramentas de teste. A proposta é executada no contexto da plataforma Android, mas, segundo os autores, a proposta pode ser adaptada a outras plataformas, como iOS e Windows Phone. Neste trabalho, os autores mencionam uma lista de ferramentas que foram usadas para a implementação da proposta. O tipo de teste aplicado é o teste funcional.

No trabalho de Tung *et al.* (2014), os autores afirmam que o teste de software é uma das melhores áreas de atuação para migrar o ambiente na nuvem, considerando a infraestrutura que oferece armazenamento ilimitado, rápida disponibilidade da infraestrutura com escalabilidade, flexibilidade e disponibilidade de ambientes de teste distribuídos, reduzindo o tempo de execução de testes de grandes aplicações e permitindo soluções rentáveis. Todos esses benefícios motivaram esta proposta, que visa usar os recursos da Computação em Nuvem para o teste de aplicações móveis.

O trabalho dos autores apresenta um framework de TaaS para Teste de Segurança (*Security TaaS* – STaaS), que consiste em oito componentes: Interface de usuário, Gerenciador de pedidos, Gerenciador de testes, Controlador de testes, Gerenciador de casos de teste de segurança, Monitoramento e rastreamento, Preços e faturamento e Scanner de segurança. Esta plataforma usa o histórico (*log*) do usuário, na qual são armazenadas as atividades solicitadas de um usuário; tornando-a uma aprendizagem que posteriormente são usadas para prever as funções que o software do sistema STaaS proverá de modo a facilitar o gerenciamento de pedidos. O componente de Gerenciador de Testes usa três tipos de algoritmos (*Round robin*, *minimal cost first*, *maximal cost first*).

Enquanto no cenário web, Wu *et al.* (2011) propõem uma plataforma de teste baseado em nuvem híbrida, que os autores definem como o uso dos recursos da nuvem e o serviço "Humanos como serviço" (*Human-as-a-Service - HaaS*), no qual, os usuários disponibilizam seus recursos móveis para serem usados na plataforma. A proposta é denominada AGARIC, que define três camadas: Camada da conexão HTTP, Camada de Comunicação (*Test Flow Control Protocol TFCP*) e a Camada de Nós. Um exemplo do conceito *Human-as-a-Service* é o *Software Testing Community* (UTEST, 2015).

No contexto de Web Services, Yan *et al.* (2012) apresentam uma plataforma para execução de Teste de Carga (*Web Service Testing as a Service* - WS-TaaS) baseada em outra ferramenta *Service4All* dos mesmos autores. A plataforma possui três camadas: IaaS, PaaS e SaaS, todas baseadas no *Service4All*. Na camada SaaS, está implementado o *Web Service Load Tester*, que executa os processos de seleção, configuração, monitoramento e amostra de resultados. Foi usado o algoritmo NFS (*Non-Shared First*) para a distribuição das tarefas de teste. Os experimentos foram rodados em 30 webservices para verificar a eficiência dos mesmos.

Gao *et al.* (2012) propõem uma infraestrutura voltada para os serviços SaaS com três camadas: Interface do Usuário (*UI Layer*), Espaço do teste (*Test Space Layer*) e Camada TaaS (*TaaS Layer*). A validação da infraestrutura foi realizada por meio de um estudo de caso executado por estudantes universitários, que testaram a aplicação OrangeHRM. Essa aplicação está disponível como *Open Source Software* (OSS) SaaS, de fácil configuração para o gerenciamento de recursos humanos. Os testes foram focados em duas funcionalidades da aplicação OrangeHRM ("*Add Employee*" e "*Personal Details*"). Com os resultados obtidos, os autores afirmam que a abordagem proposta é viável para oferecer diferentes serviços TaaS para sistemas SaaS.

Khanna (2012) propõe a solução "*Test Automation as a Service* - TAaaS" composta por quatro camadas: Hardware (servidores, armazenamento, redes); Virtualização (*firmware, software bare-metal, hypervisor*); Gestão de recursos (gerenciamento de ambiente de teste, monitoramento, métricas e processo de cobrança) e Acesso ao usuário final. Os componentes que fazem parte das camadas iniciam com o processo de requerimento do serviço. Em seguida, é feito o desenvolvimento e agendamento dos testes, execução e reporte dos resultados dos testes e finalmente a forma de pagamento. A solução não tem implementação, mas os autores descrevem os benefícios que poderiam ser obtidos.

Finalmente, Kosmatov *et al.* (2013) propõem um serviço para uso público (gratuito), a qual executa scripts de testes compilados na linguagem C. Nela são fornecidos o nome do arquivo e o nome da função a qual pretende-se testar. O método de determinar o custo dos testes é baseado no uso da CPU.

3.2.2. Virtualização e Emulação de dispositivos móveis

No contexto de Teste como Serviço (TaaS), Yu *et al.* (2010) desenvolveram um protótipo de uma plataforma de TaaS baseada no modelo de ontologia e automação de processos. A proposta tem quatro componentes principais*: Cloud Controller, Cluster Controller, Virtual Machine Controller e Monitor*. É feita, ainda, a proposta de uma arquitetura

para TaaS sobre a nuvem com cinco camadas, apresentando o fluxo de processos desde o oferecimento do serviço ao usuário, o gerenciamento das tarefas e recursos do teste, a execução dos testes e finalmente, os bancos de dados dos testes. Para validar a plataforma proposta, os autores fizeram uso do serviço de teste unitário, além do uso de algoritmos para a geração de casos de teste. As realizações dos testes foram executadas sobre máquinas virtuais priorizando o gerenciamento delas em relação ao uso dos recursos para cada projeto de teste.

O teste unitário é implementado na infraestrutura proposta por Yu *et al.* (2011), a qual serve para o gerenciamento de tarefas de teste usando dois algoritmos LNS (*Least Mean Squares*) e HPF (*High Priority First*). São conduzidos experimentos com os dois algoritmos e com a combinação entre eles para observar a eficiência enquanto ao tempo de resposta. Esta plataforma emprega as máquinas virtuais para o desenvolvimento dos experimentos, entretanto a linguagem dos programas executados é Java.

Murugesan e Balasubramanian (2014) apresentam um modelo de teste móvel baseado na nuvem, fazendo uso da virtualização para a emulação dos dispositivos móveis. Como parte do TaaS, o modelo propõe os seguintes serviços: Dispositivos móveis físicos, Emulador e Ferramentas para automação de testes. No serviço de emulação de dispositivos móveis, os autores afirmam que os tipos de teste que podem ser executados são teste de funcionalidade e aceitação, desde que os outros tipos de teste serão melhor executados nos dispositivos reais. A proposta não apresenta a implementação, porém ressalta os desafios referentes a segurança na nuvem como privacidade, acesso a informação não autorizada e vazamento de informações sensíveis.

Baride e Dutta (2011) descrevem um framework voltado para *Mobile TaaS*, considerando cinco componentes: Emuladores; Ambiente de teste e dispositivos móveis; Automação de testes; Teste da Complexidade das Aplicações (teste de desempenho, segurança e sincronização); e Teste para diferentes plataformas. Todos esses componentes são apresentados como componentes dentro dos modelos de serviço da computação na nuvem (SaaS, PaaS, IaaS), porém eles são redefinidos como: *Mobile Testing SaaS, Mobile PaaS e Mobile IaaS*. Não obstante, não é mencionado como seria a implementação e o processo da execução dos testes.

King *et al.* (2011) definem os seguintes cenários: Um provedor "B" desenvolve uma aplicação que estendera as funcionalidades de um serviço existente, que está sendo oferecido remotamente por um provedor "A". Testar esse tipo de aplicações é limitado devido a que o controle dos serviços hospedados em servidores remotos não é total. Nesse

cenário, os autores apresentam uma nova abordagem, denominada "*Test Suport as a Service* – TSaaS", para apoiar o teste de integração de aplicações que dependem de serviços hospedados na nuvem. A infraestrutura compreende seis camadas. A virtualização é dado pelo *Hypervisor*, que é responsável pelo monitoramento das máquinas e servidores virtuais, nas quais, os serviços do TSaaS são inicializados.

A infraestrutura proposta por Kim *et al.* (2012) provê instâncias móveis virtuais na qual os usuários acessam por meio de seus próprios dispositivos para usar os recursos (CPU, memória, redes) das instâncias móveis oferecidas pela infraestrutura. É apresentada uma metodologia e arquitetura para detectar o comportamento anormal nos dispositivos móveis. A proposta é validada por meio de injeção de programas maliciosos e o uso de algoritmos de aprendizado de máquina com a finalidade de detectar os comportamentos anormais que surgiram desses programas. Como resultado, foram expostas as variações de comportamento (uso de CPU, tamanho de trafego) entre um dispositivo normal e um dispositivo infectado.

No trabalho de Bai *et al.* (2013), os autores implementaram um protótipo denominado *Vee@Cloud*, que baseia sua infraestrutura IaaS em serviços públicos como Amazon Web Services[6] e RackSpace[7] para prover recursos de teste sob demanda. O gerenciador de recursos aloca instâncias de máquinas virtuais e implementa tarefas de teste. Os testes são organizados hierarquicamente visando a execução multicamada paralela.

A proposta presentada neste trabalho usará como Plataforma base o Android, devido a ser mais acessível e ser considerada uma das mais usadas. A proposta será chamada de Automação de testes para Aplicações Móveis como Serviço "*Automated Mobile Testing as a Service,* AM-TaaS" (no decorrer do texto será denominado como "Arcabouço AM-TaaS") a qual, visa usar os recursos de virtualização com o intuito de usar os emuladores, em virtude de sua viabilidade conforme citado nos trabalhos de Baride e Dutta (2011); Gao *et al.* (2014b) e Murugesan e Balasubramanian (2014). Nesses trabalhos foram apresentadas infraestruturas para um modelo de Teste de aplicações móveis na nuvem. No entanto, essas propostas não foram implementadas nem executadas. Assim, não é possível afirmar a viabilidade das propostas.

Para o propósito desta pesquisa serão aproveitados diversos componentes dos trabalhos citados os quais foram adaptados e modificados conforme os serviços que o

[6] https://aws.amazon.com/pt/

[7] https://www.rackspace.com

Arcabouço AM-TaaS pretende oferecer. A implementação e especificação dos componentes está descrito no Capítulo 4.

3.2.3. Ferramentas para teste móvel

Existem diversas ferramentas para teste de aplicações móveis que podem ter licença comercial ou ser de livre distribuição (*open source*). Algumas delas são: *Appium*, é um framework de automação de teste de libre acesso livre (*open source*) para aplicações nativas, híbridas e web. *Monkeyrunner*, é uma ferramenta, a qual, fornece uma API para escrever scripts que controlam um dispositivo ou emulador Android de fora do código do Android. *Monkey Talk,* é uma ferramenta de testes funcionais para aplicações móveis, suporta as plataformas Android e iOS. *AndroidViewClient*, é uma ferramenta *open source* para automação de teste de aplicações móveis que permite a execução de scripts de teste em Python. *Noeload*, é uma ferramenta comercial que permite a execução de teste de carregamento e desempenho. *See Test*, é uma ferramenta comercial de automação de teste para aplicações moveis que são executadas em emuladores e dispositivos reais.

Na Tabela 3.4 são listadas essas ferramentas de forma específica para à plataforma Android. As características do arcabouço AM-TaaS também são listadas na Tabela 3.4. A proposta deste trabalho é unificar os recursos disponíveis na computação na nuvem e os recursos e ferramentas necessárias para implementação de um serviço de teste de aplicações móveis na nuvem.

Muitos dos autores propõem infraestruturas baseadas na nuvem, porém a automação de teste não é abordada. Diferentemente desse trabalho, a automação de teste é abordada amplamente quanto aos critérios de teste da AQuA abordados e testes funcionais que podem ser executados a partir do arquivo binário APK (*Android Package*) de uma aplicação, sem a necessidade de ter o código fonte.

Por outro lado, na Tabela 3.5 são apresentadas as ferramentas baseadas em nuvem (*cloud-based mobile testing tools*) para testes em aplicações móveis que foram analisadas referentes à plataforma, ambiente de teste e à abordagem de teste. Todas as ferramentas são comercias. Alguns dos serviços oferecem o “teste manual”, o qual está relacionado ao serviço de teste web. Por exemplo, são utilizados dispositivos móveis simulados os quais são disponibilizados por meio da interface web para serem manipuladas. Em relação ao tipo de aplicações que são testadas, a maioria das ferramentas disponibiliza testes para aplicações nativas e hibridas.

Tabela 3.4. Ferramentas de teste móvel na Plataforma Android.

ATRIBUTOS	Appium	Monkey Runner	Monkey Talk	Neoload	SeeTest	Android View Client	**AM-TaaS**
Open Source	X	X	X			X	**X**
Teste baseado na emulação *(Emulation-based testing)*	X	X	X	X	X	X	**X**
Teste baseado em dispositivos *(Device-based testing)*	X	X		X	X	X	
Teste de App móveis nativas (*Testing for mobile native apps*)	X	X	X	X	X	X	**X**
Teste de apps móveis web *(Testing for mobile web apps)*	X		X	X	X		
Teste baseado em GUI *(Gui-based testing)*	X	X	X		X	X	**X**
Teste de Performance *(Performance testing)*				X			
Teste de carga *(Load testing)*				X			
Suporte à Linguagem Script *(Supported script language)*	X	X	X	X	X	X	**X**

No caso da proposta deste trabalho AM-TaaS, que também é apresentado na tabela, ainda não é provido suporte a aplicações hibridas devido a que nessa primeira implementação o foco foi aplicações nativas, por elas não dependerem de outras tecnologias para funcionar, como HTML5, e são compactadas em um pacote.

Tabela 3.5. Ferramentas de teste baseadas em nuvem.

Atributos		**Saucelabs**	**SOASTA**	**Perfecto mobile**	**TestDroid**	**AWS Device Farm**	**Xamarin**	**Firebase Test Lab**	**AM-TaaS**
Plataforma	Android OS	X	X	X	X	X	X	X	X
	iOS	X	X	X	X	X	X		
	Windows OS	X		X					
	Fire OS					X			
Ambiente	Teste baseado na emulação	X							X
	Teste baseado em dispositivos	X	X	X	X	X	X	X	X
	Teste baseado em simulação				X		X		
Tipo de Teste	Teste móvel via web (Teste manual)	X	X		X		X		
	Teste de toque		X						
	Teste para aplicações nativas	X	X	X	X	X	X	X	X
	Teste para aplicações hibridas	X	X	X	X	X	X		
	Gravar e reproduzir				X			X	
	Teste móvel de performance		X						

Recentemente, em Maio de 2016, foi lançada o Firebase[8], um serviço de teste na nuvem da Google. *Firebase Test Lab* é orientado à plataforma Android e disponibiliza múltiplos dispositivos (13 dispositivos na versão livre) para a execução de testes de aplicações. É possível realizar especificações de localidade e API para a execução dos testes. Os resultados dos testes apresentam opções dos logs, vídeos e capturas de tela.

Muitas das ferramentas baseadas em nuvem empregam as ferramentas de código aberto como parte de seus serviços. Essa relação é apresentada na Tabela 3.6. Como é observado, muitos dos serviços de teste na nuvem incluem mais de uma ferramenta de código aberto dentro de suas soluções em virtude da variedade de tipos de testes que são necessários, pois uma só ferramenta não pode cobrir todas as técnicas de teste que são necessárias para um projeto de teste.

Tabela 3.6. Serviços de Teste na nuvem que usam ferramentas de código aberto.

	Appium	Calabash	Cucumber	Selenium	Ulautomator	Espresso
Saucelabs	X				X	
SOASTA				X	X	
Perfecto mobile	X		X	X		
TestDroid	X				X	X
AWS Device Farm	X	X			X	
Xamarin		X				
Firebase Test Lab					X	X

3.3 Considerações Finais

Os trabalhos apresentados abordam as diversas formas de uso e execução do TaaS em diferentes plataformas, além de indicar os níveis e tipos de teste aplicados em cada proposta. A Tabela 3.7 apresenta um resumo das principais características das propostas que foram publicadas na literatura e que foram extraídas para realizar a implementação do arcabouço proposto.

As características consideradas são as seguintes listadas abaixo, as quais respondem a uma pergunta para confirmar ou definir a característica:

[8] https://firebase.google.com/docs/test-lab/

- **Infraestrutura TaaS:** o trabalho propõe uma infraestrutura voltada para TaaS? (Sim/Não).
- **Virtualização:** o trabalho usa a virtualização para o desenvolvimento da infraestrutura TaaS proposta? (Sim/Não).
- **Emulação Móvel:** o trabalho emprega as tecnologias de emulação de dispositivos móveis para o desenvolvimento da proposta? (Sim/Não).
- **Ambiente:** qual é o ambiente em que foi desenvolvido a proposta? (Não define / Web / Mobile).
- **Nível de teste:** qual é o nível de teste aplicado na proposta?
- **Tipo de teste:** qual é o tipo de teste aplicado na proposta?

Os dados obtidos a partir da classificação de características dos trabalhos correlatos permitiram a extração de informação sobre a Infraestrutura TaaS, que corresponde ao Passo 2 (P2) da Metodologia de Pesquisa apresentada no Capítulo 1. As infraestruturas TaaS propostas por todos os autores foram analisadas e selecionadas para a instanciação no arcabouço proposto por este trabalho.

Sobre a virtualização e emulação móvel, percebeu-se que poucos autores utilizam a emulação como parte de suas propostas, porém é ressaltado sua utilidade nas primeiras fases do processo de teste e o baixo custo em relação a sua implementação. Essas características foram consideradas no arcabouço proposto, assim como o tipo de teste utilizado.

Tabela 3.7. Caracterização de trabalhos que abordam TaaS.

ID	Autores	Infraestrutura TaaS	Virtualização	Emulação Móvel	Ambiente	Nível de Teste	Tipo de Teste
1	Yu *et al.* (2010)	Sim	Sim	Não	Não define	Unitário	Funcional
2	Gao; Tsai; *et al.* (2014)	Sim	Não	Não	Mobile	Não define	Não aplica
3	Tung *et al.* (2014)	Sim	Não	Não	Web	Sistema	Segurança
4	Baride e Dutta (2011)	Não	Não	Sim	Mobile	Não define	Não aplica
5	Murugesan e Balasubramanian (2014)	Sim	Não	Sim	Mobile	Sistema	Funcional
6	Gao *et al.* (2012)	Sim	Não	Não	Web	Sistema	Funcional
7	Wu *et al.* (2011)	Sim	Não	Não	Web	Sistema	Desempenho
8	King *et al.* (2011)	Sim	Sim	Não	Não define	Integração	Funcional
9	Kim *et al.* (2012)	Sim	Sim	Não	Mobile	Sistema	Segurança
10	Yu *et al.* (2011)	Sim	Sim	Não	Não define	Unitário	Funcional
11	Yan *et al.* (2012)	Sim	Não	Não	Web	Sistema	Carga
12	Starov e Vilkomir (2013)	Sim	Não	Não	Mobile	Sistema	Funcional
13	Huang (2014)	Sim	Não	Não	Mobile	Sistema	Não define
14	Bai *et al.* (2013)	Sim	Sim	Não	Web	Sistema	Não define
15	Kosmatov *et al.* (2013)	Sim	Não	Não	Web	Unitário	Funcional

Com o cenário apresentado nesta seção, percebe-se a importância do processo de teste de aplicações móveis no contexto da computação na nuvem. No próximo capítulo é apresentado o arcabouço proposto definida para este trabalho de pesquisa.

CAPÍTULO 4 – AUTOMAÇÃO DE TESTE PARA APLICAÇÕES MÓVEIS COMO SERVIÇO

Nesse capítulo será apresentado o arcabouço proposto Automated Mobile Testing as a Service – AM-TaaS, que provê teste para aplicações móveis no contexto de computação na nuvem. Como parte do Passo 5 (P5) da metodologia de pesquisa, serão apresentadas a visão geral do arcabouço, os componentes e a relação entre eles, sua implementação e finalizando com a prova de conceito executada.

4.1. Visão Geral do Arcabouço *AM-TaaS*

Todo serviço que seja oferecido por meio da nuvem está relacionado a um modelo de serviço, seja SaaS, PaaS ou IaaS, para o qual a infraestrutura será diferente em cada modelo. A proposta do Arcabouço será chamada de Automação de Teste para Aplicações Móveis como Serviço (*Automated Mobile Testing as a Service" AM-TaaS*).

No caso do AM-TaaS, é usado o modelo de infraestrutura da nuvem e o serviço de teste como serviço oferecido. A natureza do AM-TaaS está baseada no modelo arquitetural da computação na nuvem (em inglês, *Architectural Model for Cloud Computing*), como é descrita por Rhoton e Haukioja (2011). A Figura 4.1 apresenta a visão geral da proposta deste trabalho, tendo três principais características ou componentes:

- Os usuários (realizam as requisições de serviço);
- A infraestrutura (na qual é alocada toda a plataforma);
- O serviço oferecido (AM-TaaS).

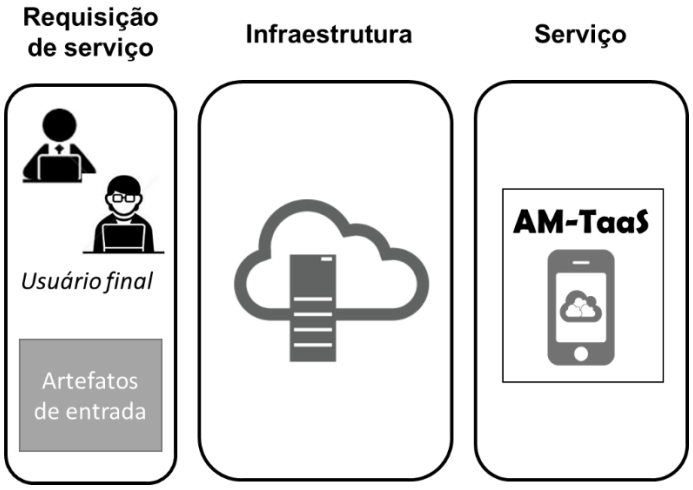


Figura 4.1. Visão Geral da proposta.

Android foi escolhida como plataforma móvel principal para esta proposta e para os experimentos de prova de conceito devido ao fato de que é a plataforma mais popular no mercado e é o sistema operacional mais difundido entre os dispositivos móveis, representando aproximadamente 84% dos dispositivos com o sistema operacional Android vendidos no primeiro trimestre de 2016 (Gartner, 2016). A diversidade que possui em quanto a fragmentação de telas e versões de Sistema Operacional fazem de Android uma plataforma com maiores possibilidades de apresentar falhas na execução de uma aplicação móvel.

4.2. Escolha da Infraestrutura para *AM-TaaS*

De forma a satisfazer os requisitos de uma infraestrutura na nuvem que seja escalável, com amplo acesso a recursos e atendimento sob demanda, a possível implementação de AM-TaaS pode ser de diferentes formas, que dependerão dos recursos que se dispõem. A lista a seguir apresenta as possíveis formas de implementação do AM-TaaS:

1) Nuvem Privada: a implementação do serviço pode ser alocada em uma nuvem privada com acesso via uma interface Web. Essa nuvem possui seus próprios recursos para banco de dados e armazenamento. Emuladores são criados ou dispositivos móveis são conectados dentro da nuvem.
2) Nuvem Pública: esse tipo de sistema contempla todos os recursos e serviços na nuvem oferecidas por terceiros, como as máquinas virtuais (do inglês, *Virtual Machine VM*), emuladores (instanciados sobre as VMs) e dispositivos (conectados a servidores específicos para o gerenciamento).

A opção 2, por ser baseada em sua totalidade em recursos da nuvem, poderia ser a melhor escolha. No entanto, isso representa um custo a ser pago pelos recursos da nuvem que tem que se usar. Por outro lado, deve-se levar em consideração também todas as tarefas necessárias para a manutenção da nuvem, como as configurações, gerenciamento das atualizações de segurança, que não é uma tarefa simples e nem barata.

A opção 1, implementação em uma nuvem privada, também é viável, pois ela permite o uso dos recursos disponíveis dentro da nuvem e permite o acesso via uma interface Web que facilita o acesso ao serviço. Porém, o nível de desempenho pode ser menor em relação à opção dois. Analisando a parte econômica, a opção 1 é menos custosa e mais factível de ser implementada quando o objetivo é apresentar o modelo de serviço.

Depois da análise das duas opções para a implementação da proposta, foi escolhida a opção 1 (Nuvem Privada), que permitia implementar e apresentar o modelo de serviço

com todas suas características, porém sem o desempenho e todas as características que a Nuvem Publica oferece.

4.3. Componentes do Arcabouço *AM-TaaS*

O Arcabouço AM-TaaS está contemplado em três camadas, de modo a ajudar a reduzir a complexidade do seu entendimento. As camadas são: Camada de Interface do Usuário (*User Interface Layer*), Camada de Informação (*Information Layer*) e Camada de Recursos (*Resources Layer*). Cada camada está composta por componentes específicos que trabalham em conjunto para a execução do todo o processo de serviço de teste. As camadas e componentes são apresentadas na Figura 4.2.

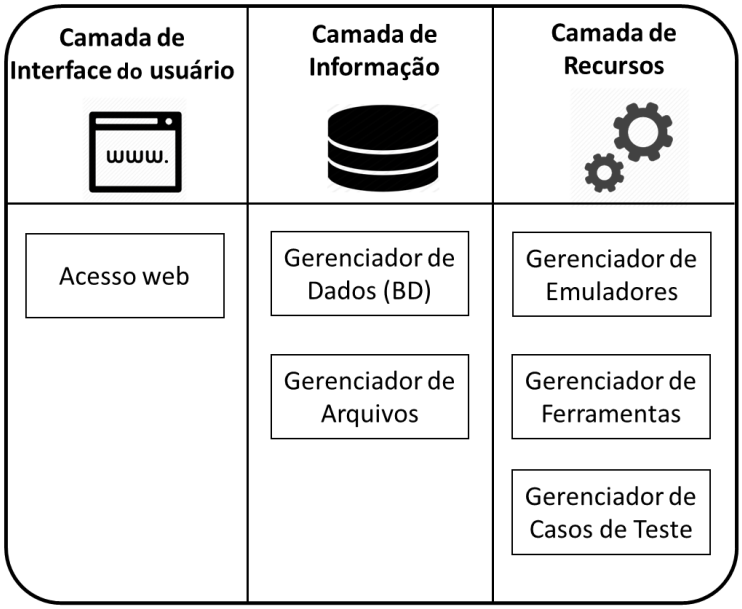


Figura 4.2. Componentes do AM-TaaS.

A relação interna entre os componentes é descrita em alto nível nos seguintes passos, os quais são ilustrados na Figura 4.3.

(1) Um usuário logado envia artefatos de entrada (Aplicação móvel e Casos de Teste) para a o Arcabouço AM-TaaS.
(2) Os dados do usuários e informações são salvos no banco de dados.
(3) É criado um diretório no qual são salvos os artefatos de entrada.
(4) São chamados os emuladores a serem usados.
(5) São chamados os scripts de teste a serem executados.
(6) Os scripts de teste são executados nos emuladores.

(7) Os resultados são salvos.

(8) Os resultados são apresentados na interface do usuário.

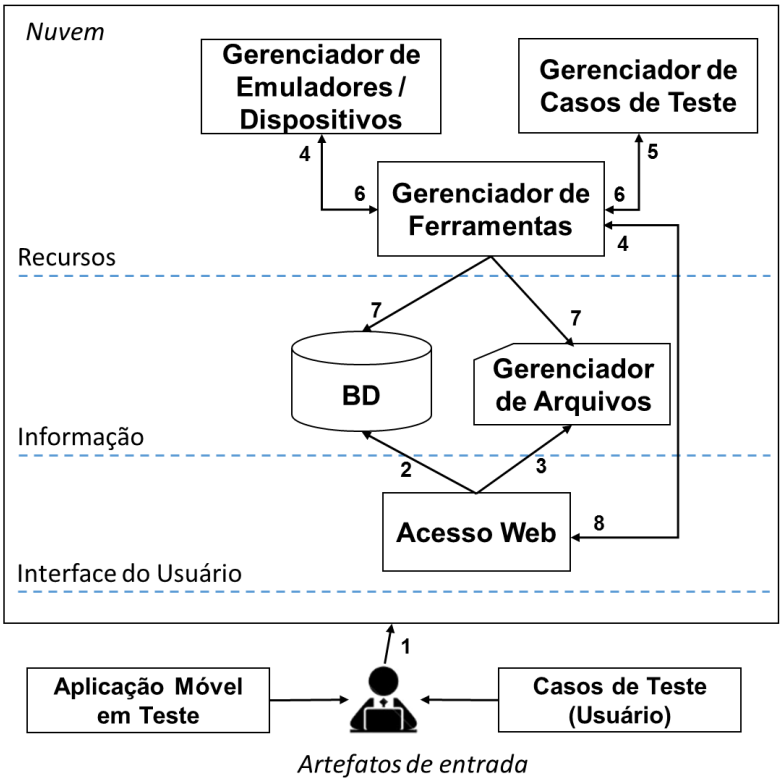


Figura 4.3. Relação entre os componentes das camadas de AM-TaaS.

4.3.1. Camada de Interface do Usuário (*User Interface Layer*)

a) Acesso web (*Web Access*): Este componente tem o propósito de expor a lógica do modelo de serviço, permitindo a interação entre os usuários que realizam a requisição de um serviço e a infraestrutura na qual está alocado o AM-TaaS. Ele possui uma interface GUI que permite a facilidade de acesso a esta camada. O acesso a um serviço por meio de esta camada apresenta as seguintes vantagens:

- Não é necessário que o usuário conheça as tecnologias usadas por trás desta camada.
- A exigência do poder computacional no usuário é mínima, devido a que a maior parte do processamento é realizado no servidor.

Os artefatos de entrada que são necessários para dar início à execução do serviço de teste do AM-TaaS são enviados por meio desta camada. Os artefatos de entrada são o arquivo binário de uma aplicação (APK) e os scripts de teste. Esses componentes são apresentados na Figura 4.4.

Figura 4.4. Artefatos de entrada do AM-TaaS.

4.3.2. Camada de Informação (*Information Layer*)

a) Gerenciador de Dados (*Data Base Manager*): Este componente é responsável pelo gerenciamento dos dados do arcabouço, os dados dos usuários, as informações das características e o status dos emuladores. Os dados dos usuários são importantes, pois quando é realizada uma requisição do serviço, os dados são usados pelo Gerenciador de Arquivos e os emuladores. Conforme um emulador é utilizado, o status dele é atualizado para que outros usuários não possam interromper o processo de execução do serviço de teste.

b) Gerenciador de Arquivos (*File Manager*): Os componentes que fazem uso deste recurso são os componentes: Acesso web e o Gerenciador de Ferramentas. Como o AM-TaaS trabalha com arquivos físicos (arquivo binário da aplicação, arquivos de scripts de teste definidos como padrões e os arquivos de scripts que os usuários enviam, logs). Este componente também é responsável pela criação de um diretório de pastas, toda vez que um usuário solicita o serviço de teste. A estrutura do diretório criado é apresentada na Figura 4.5. Todos os resultados obtidos serão apagados quando iniciar uma nova execução do serviço, em razão do armazenamento limitado do servidor local. Caso o serviço fosse oferecido na nuvem, todos os dados poderiam estar disponíveis para futuras consultas.

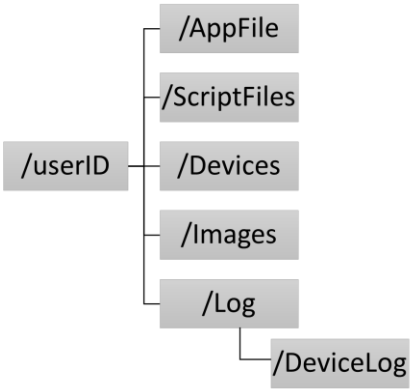


Figura 4.5. Estrutura do diretório de arquivos por usuário.

O conteúdo das pastas é como segue: A pasta *userID* identifica os dados do usuário que está fazendo uso do AM-TaaS; a pasta *AppFile* contém os arquivos binários da aplicação a ser testada (apk); a pasta *ScriptFiles* contém os scripts de teste que serão executados nos emuladores escolhidos pelo usuário; a pasta *Devices* contém informação sobre os emuladores e dispositivos que o usuário selecionou para serem testados; a pasta *Images* contém as capturas de tela dos scripts de teste; a pasta *Log* contém o log do script de teste executados e a subpasta *DeviceLog* que contém o log de cada emulador.

4.3.3. Camada de Recursos (*Resources Layer*)

a) Gerenciador de Emuladores (*Emulators Manager*): Devido a esta proposta estar baseada na plataforma Android, será usado o pacote *SDK Manager* (Kit de Desenvolvimento de Software) para gerenciar os emuladores. Este componente é responsável também pela limpeza dos dados dos usuários que tenham utilizado os emuladores e a verificação se o emulador está disponível para seu uso.

Ele está baseado no gerenciador de dispositivos móveis (*Android Virtual Devices AVD Manager*), no qual são criados emuladores com diferentes versões do SO Android e que para a implementação deste trabalho serão chamados de AVDs. Um AVD imita o comportamento de um dispositivo móvel real. Quando o emulador é criado, é gerado um arquivo chamado de “Configuração AVD” o qual descreve como deve se comportar para simular um dispositivo real. As informações que contem esse arquivo são:

- Versão da plataforma do Android que será emulada, comumente chamado de *API level.*
- Tamanho, densidade e resolução da tela.
- Tamanho do armazenamento externo da memória

- Tamanho da memória RAM do emulador.
- Tipo de CPU.

b) Gerenciador de Casos de Teste (*Test Case Manager*): O objetivo principal deste componente é hospedar e gerenciar os critérios de testes que foram automatizados neste trabalho (chamados de *casos de teste padrão*) e prover esses testes aos dispositivos móveis. Outro objetivo é gerenciar os scripts de teste que são enviados pelos usuários. AM-TaaS executa três casos de teste padrão para todas aplicações que são enviadas ao Arcabouço. A Tabela 4.1 apresenta os casos de teste que foram extraídos dos critérios de teste publicados em AQuA, os quais são considerados primordiais em qualquer tipo de aplicação.

Tabela 4.1. Casos de Teste Padrão de AM-TaaS.

ID	Caso de Teste: Título	Caso de Teste: Descrição
1	*OTA Install* (a)	Instalar a Aplicação móvel.
2	*OTA Install* (b)	Instalar a Aplicação sem espaço suficiente no emulador
3	*OTA Uninstall*	Desinstalar a Aplicação

Este componente é extensível, ou seja, é possível adicionar outros casos de teste para diferentes tipos de teste (ex: teste de segurança ou desempenho) que possam ser executados nas aplicações submetidas ao arcabouço.

c) Gerenciador de Ferramentas (Tools *Manager*): Este componente está constituído por um conjunto de ferramentas, cada uma delas realiza uma tarefa importante dentro de todo o processo de teste. Estas ferramentas são:

- ***Android Debug Bridge (ADB)***[9]***:*** ADB é uma ferramenta de linha de comando versátil que permite a comunicação com uma instância de um emulador o um dispositivo conectado. Esta ferramenta permite o controle dos emuladores por meio de comandos. Por exemplo, é possível atualizar o sistema, executar comandos *shell*, gerenciar as portas nas quais os emuladores estão conectados, copiar e ler arquivos.
- ***Monkeyrunner***[10]***:*** Esta ferramenta vem em conjunto com o framework Android SDK, a qual é projetada para a execução de testes no nível funcional

[9] https://developer.android.com/studio/command-line/adb.html

[10] https://developer.android.com/studio/test/monkeyrunner/index.html

e teste de unidade. Para interagir com a aplicação que será testada, *Monkeyrunner* usa as coordenadas de posição (X,Y). O uso desta ferramenta é viável se é usado apenas um modelo de emulador/dispositivo. No entanto, se o script gerado é usado em dois ou mais modelos de emuladores/dispositivos diferentes, não é mais viável devido à posição ou coordenadas que serão diferentes para cada modelo de dispositivo. A linguagem usada para esta ferramenta é o Python. Esta ferramenta é usada nos scripts de teste padrão do AM-TaaS.

- ***AndroidViewClient*[11]*:*** Denominado AVC, é uma ferramenta para automação de teste de aplicações móveis que permite a execução de scripts de teste em Python. A criação de scripts é feito por meio da ferramenta Culebra que vem junto com o AVC e é uma ferramenta de Captura e Reprodução. AVC é uma ferramenta independente, não é parte das ferramentas do Framework Android SDK. Uma das vantagens desta ferramenta é a interação com os elementos de uma aplicação móvel. Enquanto o *Monkeyrunner* usa as coordenadas (X,Y), *AndroidViewClient* usa as propriedades dos elementos (ID, nome e descrição) e os scripts podem ser reutilizados em outros emuladores ou dispositivos de diferentes tamanhos de tela. AVC é compatível com a ferramenta *UIAutomator* do Android SDK. Esta ferramenta é usada para geração dos scripts de teste que são oferecidos aos usuários como exemplo.
- ***UIAutomator:*** Faz parte do SDK do Android e que vem junto com o *UIAutomatorViewer*, a qual permite listar todos os objetos de uma Interface de usuário (*user interface*) de uma aplicação móvel. Uma desvantagem do *UIAutomatorViewer* é que suporta apenas dispositivos com a versão de Android API 18 ou superior.

O uso destas ferramentas não é limitado, ou seja, é possível adicionar outras ferramentas que permitam a execução de outros tipos de teste (novamente, exemplos são teste de segurança ou desempenho).

4.4. Implementação do Arcabouço *AM-TaaS*

Para suportar o modelo de serviço de teste na nuvem e visando criar uma versão demonstrativa dos principais conceitos sobre o modelo de serviço de teste para aplicações

[11] https://github.com/dtmilano/AndroidViewClient

móveis na nuvem, foi implementado o arcabouço AM-TaaS. Nessa seção serão apresentados detalhes da implementação de forma abrangente.

4.4.1 Ambiente

Em virtude da infraestrutura escolhida para a implementação do AM-TaaS, o servidor local tem as seguintes características: 8GB de memória RAM, Processador Intel Core i5 3.0 GHz, 100 GB de armazenamento, e o Sistema Operacional Linux Ubuntu de 64 Bits. O gerenciador do Android SDK foi instalado e configurado, definiram-se as variáveis locais para “ANDROID_SDK_HOME”, “ANDROID_SDK_HOME/tools” e “ANDROID_SDK_HOME/ platform-tools”. Elas são responsáveis para o correto funcionamento do SDK e das ferramentas que AM-TaaS utiliza, tais como ADB, *Monkeyrunner*. Por outro lado, foram necessárias a instalação de algumas bibliotecas (*ia32-libs*) para o suporte a aplicações de 32 bits.

Para a criação dos emuladores foi utilizado o gerenciador de dispositivos do Android (*AVD Manager*). Nele foram necessários especificar as características dos emuladores que seriam usados. A lista de características é apresentada na Tabela 4.2. É importante ressaltar que para a escolha de estas características foi realizada uma análise sobre as versões de Android que são as mais usadas e os tamanhos de tela mais comuns. Essas informações foram obtidas do próprio site de Android[12].

Tabela 4.2. Características de emuladores - AM-TaaS.

MODELO	Nexus One	Galaxy Nexus	Nexus 5	Nexus 6	Nexus 7
Target (API)	Google API Level 17	Google API Level 17	Google API Level 18	Google API Level 19	Google API Level 19
API Nome	Jelly Bean	Jelly Bean	Jelly Bean	KitKat	KitKat
CPU/ABI	ARM (armeabi-v7a)	ARM (armeabi-v7a)	ARM (armeabi-v7a)	ARM (armeabi-v7a)	ARM (armeabi-v7a)
Teclado	Sim	Sim	Sim	Sim	Sim
Skin	Nexus One	Galaxy Nexus	Nexus 5	Nexus 6	Nexus 7
Ram	768	1024	1024	1024	1024
VM Heap	32	64	64	128	64
Armazenamento Interno	200	200	200	200	200
SD tamanho	100 MB	100 MB	100 MB	100 MB	100 MB
Snapshot	Não	Não	Não	Não	Não
Use Host GPU	Sim	Sim	Sim	Sim	Sim

[12] https://developer.android.com/about/dashboards/index.html

Para a execução dos Casos de Teste, a ferramenta *Monkeyrunner* vem junto com o framework do Android SDK, já a ferramenta *AndroidViewclient* teve que ser configurada para se comunicar com o Android SDK. Também foram instaladas as librarias necessárias para o suporte de *Python*, o qual é a linguagem para a escrita dos scripts de teste e *Python image* para o uso da ferramenta Culebra, que é uma ferramenta de Captura e Reprodução de Scripts de teste.

Apache é o servidor web usado no sistema operacional do servidor e permite a interação do usuário com a infraestrutura de AM-TaaS. Com o objetivo de atender às necessidades de AM-TaaS, foi necessário configurar algumas características como o tamanho limite de arquivos para serem enviados ao arcabouço (*upload de arquivos*), assim como o tamanho máximo de tais arquivos. Essas configurações foram necessárias em razão de que o tamanho em MB (*MegaBytes*) dos arquivos binários (APK) dos usuários é muito variável sendo importante cobrir ao máximo todas as aplicações que usem o AM-TaaS.

4.4.2 Acesso web (*Front End*)

Com o objetivo de seguir um padrão de desenvolvimento web, a camada de Interface de usuário (*User Interface Layer*) foi implementada baseada na ferramenta *MTControtool* (Nascimento, dos Santos e Dias-Neto, 2014). Esta ferramenta foi implementada usando o Framework Yii[13] que permite a criação de interfaces web baseados no modelo MVC (*Model View Control*), PHP é a linguagem usada.

4.4.3 Gerenciador de Dados e Arquivos

O componente Gerenciador de Dados é composto por um banco de dados MySQL. Em contrapartida, os arquivos (*arquivos binários, logs, scripts de teste*) foram armazenados diretamente no disco rígido organizado em diretórios, como foi descrito no componente de Gerenciador de Arquivos.

4.4.4 Gerenciador de Casos de Teste

A camada Gerenciador de Casos de Teste é composta pelos casos de teste padrão do AM-TaaS. Os scripts foram escritos na linguagem Python e funcionam junto com a ferramenta *Monkeyrunner*. Parte do código do caso de teste "*OTA Install*" é apresentado na Figura 4.6 e o código completo está disponível no Apêndice C.

[13] http://www.yiiframework.com

```
# Funcion: get_package_activity_name(apk_address)
def get_package_activity_name(apk_address):
    command = defPath.sdk_plaTool +  "aapt dump badging %s" %apk_address
    aapt_result = subprocess.Popen(command, stdout=subprocess.PIPE, shell=True).communicate()[0]
    lines = aapt_result.split("\n")
    myDic = {}
    for line in lines:
        splitedline=line.split(":")
        if len(splitedline)==2:
            myKey,myValue=line.split(":")
            myDic[myKey]=myValue
    package = myDic['package'].split("'")[1]
    activity = myDic['launchable-activity'].split("'")[1]
    return package, activity
.....................................................................................
#Execute test
    def run(self):
        eraseLog(self.device)
        fs_1 = check_free_space(self.device)

        for apk in glob.glob(self.apkfolder + '/*.apk'):
            print "Getting package and activity name..."
            packagename,activity = get_package_activity_name(apk)
            componentname = packagename + "/." + activity
            apk_path = self.device.shell('pm path ' + packagename)
            if not apk_path:
               print "Installing " + str(packagename) + " ..."
               result_install = self.device.installPackage(apk)
            else:
               if apk_path.startswith('package:'):
                  print "Apk has already installed, re-installing..."
                  result_install = self.device.installPackage(apk)
```

Figura 4.6. Parte do código Caso de Teste - *OTA Install*.

Os Casos de teste que são enviados pelos usuários também são gerenciados e executados. A estrutura é variável em virtude da variedade de aplicações móveis que podem ser testadas usando o AM-TaaS. O usuário está facultado a enviar seja um só arquivo de teste ou um conjunto de casos de teste. Eles são enviados em formato comprimido em um arquivo com extensão “zip”. Para serem executados, eles serão descomprimidos e executados para todos os emuladores que o usuário tenha selecionado.

4.4.5 Processo de Serviço do Arcabouço AM-TaaS

O processo para o funcionamento do arcabouço contém seis passos que podem ser observados na Figura 4.7.

- Passo 1: é iniciado quando o usuário acessa a plataforma por meio da interface web.
- Passo 2: o usuário envia os scripts de teste indicando quais os testes que serão executados nos emuladores.
- Passo 3: o usuário seleciona os dispositivos móveis nos quais serão executados os testes.

- Passo 4 e 5: é um processo interno (não visível ao usuário) os scripts de teste são executados nos emuladores selecionados no Passo 3.
- Passo 6: finalmente, um relatório com informação sobre o estado dos testes executados é enviado e mostrado na tela do usuário.

Os testes que são oferecidos como automatizados no Passo 2 podem ser definidos pelo próprio desenvolvedor ou pelo gerente do projeto dos testes, pois o Arcabouço poderá aceitar scripts de testes que já estão automatizados apenas para execução ou podem ser critérios de teste que um time de teste já tenha definido com antecedência.

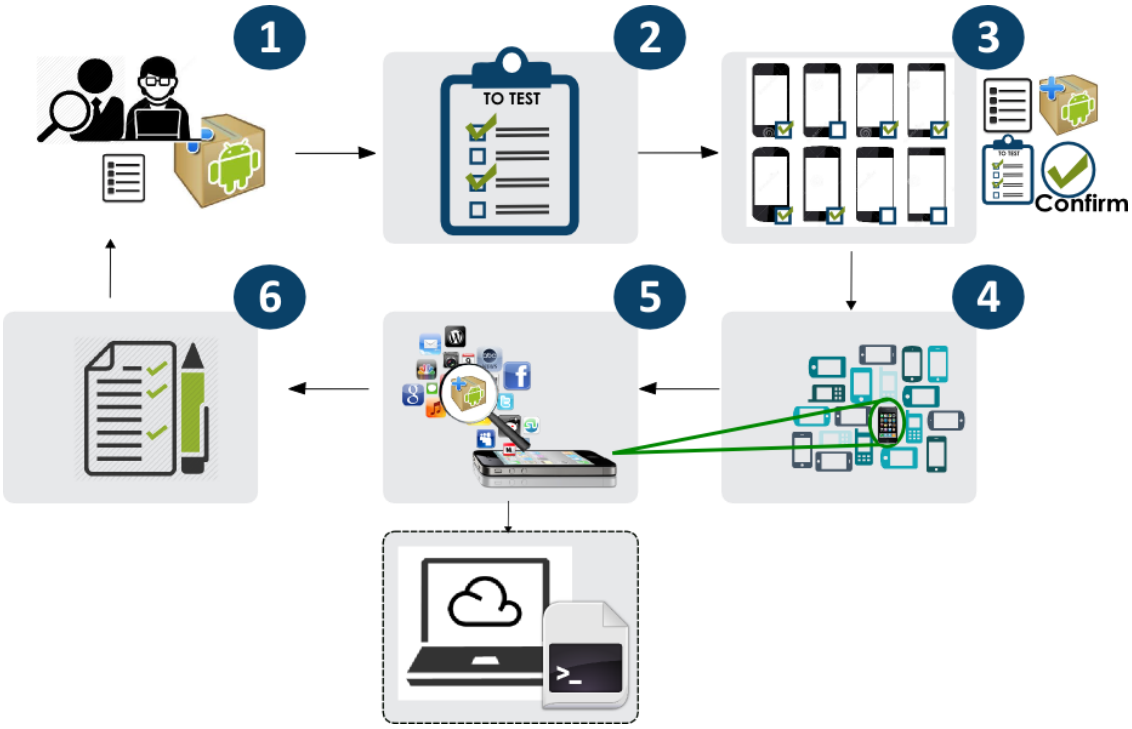


Figura 4.7. Processo de Serviço do AM-TAAS.

4.5. Prova de Conceito

Prova do Conceito é um modelo prático que visa a provar o conceito (teórico) de uma pesquisa (Carsten, 1989). Para a execução da prova de conceito, foram definidos os seguintes componentes: o ambiente de teste, o critério de teste a ser executado e as aplicações móveis a serem testadas. Após a execução será apresentada a análise dos resultados. Esse processo é apresentado na Figura 4.8.

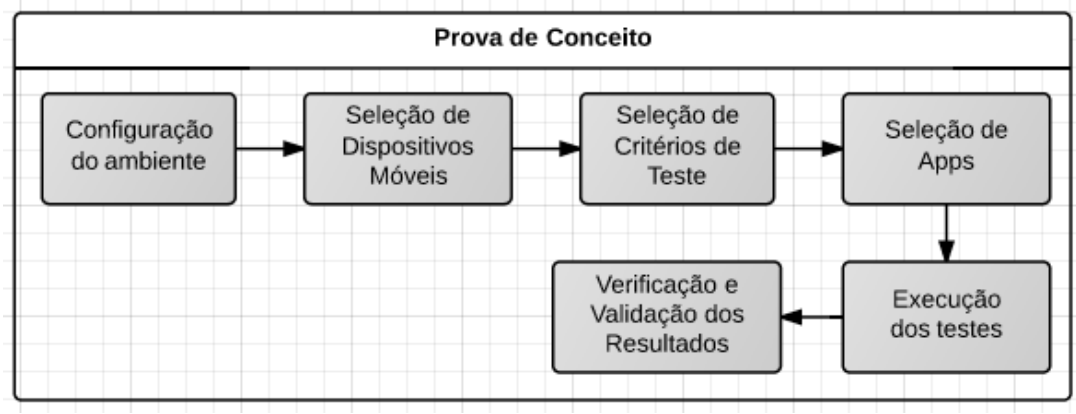


Figura 4.8. Passos seguidos para a prova de conceito.

O Passo 4 (P4), Estudo de Viabilidade – Prova de Conceito, da metodologia de pesquisa apresentada no Capítulo 1 é descrito nessa seção. O principal objetivo desta prova de conceito é avaliar a viabilidade da proposta.

Foram selecionados nove emuladores, cinco aplicações móveis e três casos de teste para analisar o tempo de execução, e com isso verificar a viabilidade do arcabouço proposto. O objetivo não é verificar o tempo entre os dispositivos, mas sim verificar a viabilidade de introduzir esses casos de teste automatizados no arcabouço proposto nessa pesquisa.

4.5.1 Ambiente de Teste

O ambiente foi implementado com uma máquina Ubuntu 14.04, 8GB Ram, 50 GB HDD. Foram usados dispositivos móveis emulados que possuem diferentes versões de Sistema Operacionais Android, tamanhos de tela, capacidade de memória e diversos modelos de dispositivos.

A classificação dos dispositivos móveis foi baseada de acordo com Kirubakaran e Karthikeyani (2013), que definiram três categorias de dispositivos móveis de acordo a suas características. Para o desenvolvimento do trabalho foram redefinidos os valores para se ajustarem as novas características dos dispositivos móveis atuais. A lista de categorias é ilustrada na Tabela 4.3.

Tabela 4.3. Classificação de Dispositivos Móveis.

Grupo	Prioridade	Descrição
Grupo 1	C	Dispositivos com baixo desempenho em CPU, RAM e baixa resolução de tela.
Grupo 2	B	Dispositivos com desempenho médio em CPU, RAM (<=512 MB), tamanho de tela e resolução boa.
Grupo 3	A	Dispositivos de alto desempenho com dual/quad-core CPU, RAM (>512 MB) e alta resolução de tela.

Além da classificação usada por Kirubakaran e Karthikeyani, (2013), os dispositivos foram baseados nas informações divulgadas por (Gartner, 2013b) e em base nas estatísticas de uso sobre as versões coletadas do site do Android[14] no ano de 2015. A Figura 4.9 apresenta essas estatísticas. No final, foram definidos nove diferentes dispositivos para serem parte da prova de conceito. Os dados são apresentados na Tabela 4.4.

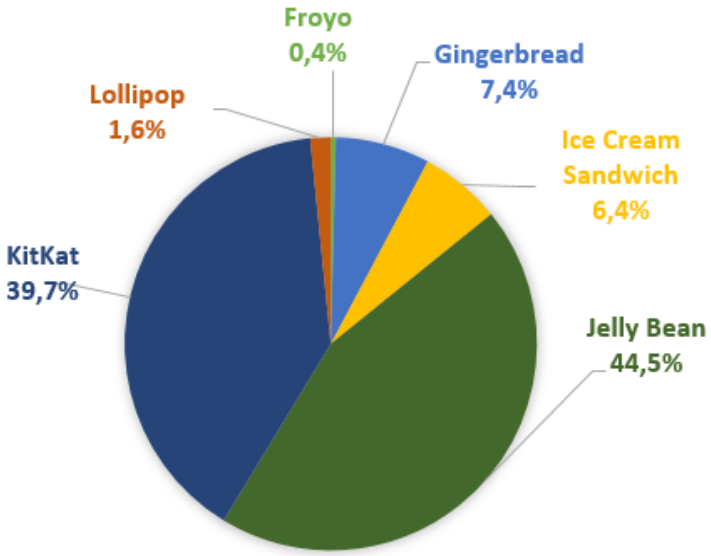


Figura 4.9. Uso das Versões do SO Android.

Tabela 4.4. Dispositivos Móveis Emulados.

Modelo	ID-Tipo	Versão Android	API	Tamanho de Tela	Ram
Xperia ZR	1-A	4.1	16	4.55	2 GB
Nexus 5	2-A	4.4.4	19	4.95	2 GB
Galaxy S4	3-A	4.4.2	19	5.0	2 GB
Galaxy S3	4-B	4.3	18	4.8	1 GB
LG f60	5-B	4.4.2	19	4.5	1 GB
Galaxy S Duos	6-B	4.2	17	4.0	768 MB
Xperia Go	7-C	4.0	14	3.5	512 MB
LG Optimus l7	8-C	4.0.3	15	4.3	512 MB
Galaxy Star	9-C	4.1.2	16	3.0	512 MB

4.5.2 Critério de Teste

Para a execução do teste, foi selecionado o critério da AQuA "*OTA Install*". Ele verifica se uma aplicação foi instalada com sucesso, caso contrário devolve uma mensagem indicando o erro. Na Figura 4.10 é ilustrado este caso de teste publicado por AQuA.

[14] http://developer.android.com/about/dashboards/index.html

Para a execução do critério de teste foi necessária a criação de três casos de teste.

- **CT1-*OTA Install*:** O primeiro verifica se à aplicação (APK) existe no dispositivo móvel e, se não existir, realiza a instalação, caso exista, é exibida uma mensagem informando que a aplicação já está instalada.
- **CT2-*OTA Install*:** O segundo caso é quando o dispositivo não tem memória suficiente e tenta-se iniciar a instalação da aplicação. Feito isso, verifica se a instalação foi realizada com sucesso, se não for, mostra-se uma mensagem sobre o espaço insuficiente na memória.
- **CT3-*OTA Install*:** O terceiro caso é quando a aplicação possui alguma dependência para sua instalação que não está previamente instalado no emulador. O resultado esperado é uma mensagem reportando a falta de dependências para a instalação.

Test ID 1.1	*Test Title* **OTA install**	**Critical**
Test Description **The Application must install via OTA to the main memory of the device.**		
Required for: **All applications.**		
Testing Note 1. If errors occur at installation time, any corresponding messages must be reported by the tester in the test report. 2. If the device does not display the icon, then the user must be able to start the Application using other means. 3. For carriers that will only accept the installation of Applications from Android Market, this test cannot be performed until the Application is in Android Market. 4. If the device supports installation to SD card or other internal memory then this test must be repeated for each one supported by the device.		
Testing Steps 1. Open the browser Application of the device; 2. Type the URL of the Application file, or navigate to it graphically; 3. Connect to the typed URL / application icon; 4. Accept the installation of the Application to the main memory of the device *RESULT:* 1. The Application installs to the device. 2. The icon for the Application can be found from the device. 3. If there is insufficient space, the user is informed. 4. The application name is correctly displayed in the menu and in the application manager.		
Result of Test ☐ **PASS** ☐ **FAIL**		

Figura 4.10. Critério de teste "OTA Install" da AQuA.

4.5.3 Aplicações móveis

Foram selecionadas 5 aplicações móveis do repositório *Google Play Store*, baseadas nas aplicações mais populares. A Tabela 4.5 apresenta os dados e características de cada

aplicação. Cada aplicação representa uma categoria diferente dentro do repositório e a propriedade *Dependência* indica se a aplicação precisa de outra aplicação previamente instalada para ser executada. No caso da aplicação *Easy Taxy*, ela depende da instalação da aplicação *Google Maps* para funcionar.

Tabela 4.5. Lista de Aplicações móveis.

Aplicação móvel	Categoria	Dependência
Duolingo	Educação	Não
Pou	Jogos	Não
Easy Taxi	Transporte	Sim (Google Maps)
Instagram	Social	Não
Viber	Comunicação	Não

4.5.4 Execução da Prova de Conceito

Para o processo de execução da prova de conceito, os emuladores descritos na Tabela 4.4 foram criados com suas respectivas características. Logo, os casos de teste foram escritos na linguagem Python e foram executados uma vez com a ferramenta *Monkeyrunner*. Cada caso de teste foi executado uma vez para cada emulador por meio do terminal (*linha de comando*). A Figura 4.11 apresenta um extrato do código para os casos de teste, usando a ferramenta *Monkeyrunner*.

```
from com.android.monkeyrunner import MonkeyRunner, MonkeyDevice
import os, threading, glob, subprocess
from datetime import datetime

startTime = datetime.now()
def main():
    devices = os.popen('adb devices').read().strip().split('\n')[1:];
    count = 0
    for d in devices:
        deviceid = d.split('\t')[0];
        print "Identified device %s" % deviceid
        device = MonkeyRunner.waitForConnection('',deviceid)
        count = count + 1
        droidTesting = DefaultDroidTest(device, count)
        droidTesting.run()
-------------------------------------------------------------------------
    def run(self):
        eraseLog(self.device)
            self.device.startActivity(component=componentname)
            MonkeyRunner.sleep(5)
            image = self.device.takeSnapshot()
            image.writeToFile(self.imagefolder + 'screenshot_' + packagename  +
                str(self.count) + '.png','png')
            log(self.logfolder + 'test' + str(self.count) +'.log', self.device);
            print 'Execution Time', datetime.now() - startTime
```

Figura 4.11. Parte do caso de teste criado usando *Monkeyrunner*.

4.5.5 Resultados

O experimento foi realizado com o objetivo de avaliar a viabilidade do arcabouço proposto neste trabalho. Ao total, foram executados 99 casos de teste, no qual 84 casos de teste, que representam 85%, passaram os testes e 15 casos de teste falharam. Esses resultados podem ser observados na Figura 4.12.

Alguns casos de teste falharam por causa de um problema existente na versão API 19 do Android relacionado ao armazenamento interno em AVDs. Este problema não permite mudar o tamanho da memória interna em AVDs, o que impossibilitou a execução do teste CT2-OTA para alguns dispositivos/aplicações.

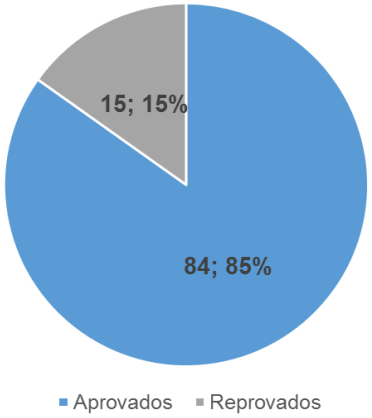


Figura 4.12. Prova de Conceito: Resultados Totais dos Casos de Teste.

Os resultados dos casos de teste ***CT1-OTA Install, CT2-OTA Install e CT3-OTA Instal***l podem ser observados na Tabela 4.6, Tabela 4.7 e Tabela 4.8, respectivamente. Os resultados são mostrados em segundos. Os casos de teste foram executados em cada dispositivo móvel emulado, para as cinco aplicações escolhidas. Na Tabela 4.7 podem ser observados com o símbolo * quais casos de teste não puderam ser executados para cada dispositivo/aplicação, como citado anteriormente.

Tabela 4.6. Resultados em segundos do CT1-*OTA Install*.

MODEL	DUOLINGO	POU	INSTAGRAM	VIBER	EASYTAXI
1 - A	46.29	89.06	63.37	127.47	50.94
2 - A	36.22	52.16	44.34	79.20	36.13
3 - A	35.14	52.64	42.89	82.56	39.94
4 - B	36.29	56.52	45.57	82.72	93.48
5 - B	36.08	52.70	273.80	294.35	61.36
6 - B	44.11	79.86	61.22	125.77	63.06
7 - C	46.21	80.73	55.88	124.36	42.06
8 - C	43.75	50.13	55.50	129.35	43.55
9 - C	43.20	72.86	45.21	108.94	72.87
Total	**367.30s**	**586.65s**	**687.78s**	**1174.72s**	**503.38s**

Tabela 4.7. Resultados em segundos do CT2 OTA Install.

MODEL	DUOLINGO	POU	INSTAGRAM	VIBER	EASYTAXI
1 - A	19,63	30,36	20,67	45,54	14,08
2 - A	*	*	*	*	*
3 - A	*	*	*	*	*
4 - B	20,06	32,84	21,58	47,03	14,81
5 - B	*	*	*	*	*
6 - B	18,89	29,28	19,37	43,10	14,80
7 - C	17,53	29,80	19,79	42,91	14,05
8 - C	18,34	29,88	19,97	43,36	13,80
9 - C	16,32	29,28	20,01	42,54	14,84
Total	**110.77s**	**181.44s**	**121.39s**	**264.49s**	**86.39s**

Tabela 4.8. Resultados em segundos do CT3 OTA Install.

MODEL	EASYTAXI
1 - A	27,02
2 - A	23,98
3 - A	27,60
4 - B	18,55
5 - B	30,02
6 - B	24,14
7 - C	21,67
8 - C	26,62
9 - C	21,00
Total	**220.60**

Finalmente, na Figura 4.13 pode ser observado o tempo de execução total em minutos por cada aplicação. Cada aplicação foi testada nos 9 dispositivos virtuais.

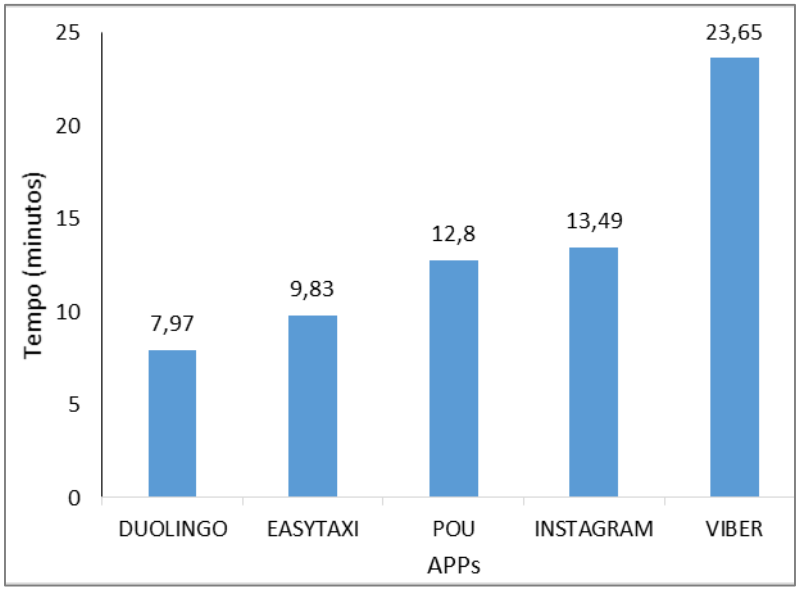


Figura 4.13. Prova de Conceito: Tempo total de execução em minutos dos Casos de teste CT1, CT2 e CT3 por cada aplicação.

O tempo médio para executar todos os testes em nove dispositivos emulados, para uma aplicação foi de aproximadamente 14 minutos, sendo executados 18 casos de teste para aplicações sem dependências e 27 casos de teste para aplicações com dependência. Como os dispositivos móveis foram emulados, isso contribuiu para a redução do esforço e custo dos recursos necessários para os testes, pois não foi necessário adquirir dispositivos móveis físicos e nem a contratação de pessoas para realizar testes em cada dispositivo.

Os critérios de teste usados para esta prova de conceito foram os critérios da AQuA. No entanto, a escolha dos critérios de teste a serem automatizados depende de cada projeto de aplicação móvel, sendo os desenvolvedores ou gerentes de projetos quem definem quais testes podem ser automatizados e executados. No cenário atual, são os desenvolvedores que criam as aplicações e quem tomariam essas decisões.

Os resultados mostraram a possibilidade de executar os casos de teste automatizados para aplicações com e sem dependências em diversos dispositivos emulados, os quais poderiam ser introduzidos no arcabouço proposto e a partir deles adicionar outros que possam ser automatizados.

4.6. Considerações Finais

Nesse capítulo foram apresentados os detalhes do processo de concepção do arcabouço AM-TaaS, assim como sua implementação e os componentes que formam parte. Esse arcabouço foi desenvolvido com o objetivo de apresentar o modelo de serviço de teste para aplicações móveis.

Em seguida, foi apresentada a prova de conceito realizado com o objetivo de verificar a viabilidade do arcabouço proposto. Os resultados apresentaram a viabilidade da proposta para sua implementação. Porém, é necessário realizar um novo estudo de viabilidade com a intervenção de usuários no uso do arcabouço proposto. O próximo capítulo irá explicar o detalhe do estudo de viabilidade que foi planejado e executado, assim como os resultados obtidos.

CAPÍTULO 5 – ESTUDO DE VIABILIADE

Neste capítulo é apresentado um estudo experimental para avaliar a viabilidade do arcabouço que oferece de serviço para teste para aplicações móveis "AM-TaaS", avaliado do ponto de vista da eficiência em relação ao tempo, a corretude e utilidade. O experimento foi conduzido em dois ambientes de teste: o ambiente de teste oferecido por AM-TaaS e um ambiente de teste local que é utilizado frequentemente por desenvolvedores de aplicações móveis. Esse capítulo representa o Passo 6 (P6) da metodologia de pesquisa.

5.1 Definição do Estudo de Viabilidade

5.1.1 Propósito

O propósito desde estudo foi responder à questão: "*A utilização do arcabouço AM-TaaS é viável analisando sua utilidade e eficiência comparados com um ambiente de teste local?*" Por utilidade se entende neste estudo como a percepção dos usuários sobre a utilidade do AM-Taas e por eficiência, o tempo de execução e corretude em relação ao número de Casos de Teste que são executados.

Nesse trabalho, o ambiente de teste local refere-se ao ambiente provido pelo *Framework Android SDK* que vem junto com o *Android Studio*[15]. Nele, é possível criar emuladores e executar os testes de forma local. Esse ambiente apoia as atividades de testes por desenvolvedores da plataforma Android. Assim, no decorrer desse capítulo será chamado de "Amb. Local".

5.1.2 Perspectiva

A perspectiva é do ponto de vista dos usuários. Nesse contexto, os usuários mais próximos são os desenvolvedores/testadores que desejam analisar a utilidade e eficiência do AM-TaaS nas atividades de teste de aplicações móveis.

5.1.3 Objetivos Específicos

- **Analisar** o arcabouço de apoio à execução de testes automatizados em aplicações móveis (AM-TaaS);
- **Com o propósito de** caracterizá-lo;

[15] https://developer.android.com/studio/index.html

- **Com relação à** utilidade e eficiência;
- **Do ponto de vista do** testador;
- **No contexto** de teste de aplicações móveis em múltiplos dispositivos.

5.1.4 Questões e métricas

- Q1. Qual é a eficiência do Arcabouço AM-TaaS em relação ao uso de um Ambiente de Teste Local no que diz respeito ao tempo de execução de um conjunto de Casos de Teste?
    - **Métrica**: Tempo gasto na execução de um Conjunto de Casos de Teste.

- Q2. Qual é a eficiência do Arcabouço AM-TaaS em relação ao uso de um Ambiente de Teste Local no que diz respeito ao número de Casos de Teste que falharam?
    - **Métrica**: Número de Casos de Teste executados que falharam.

- Q3. Qual é a utilidade do Arcabouço AM-TaaS em relação ao uso de um Ambiente de Teste Local no que diz respeito à percepção do usuário?
    - **Métrica**: Número de participantes favoráveis ao uso do Arcabouço proposto.

5.1.5 Questões em aberto

Há alguma mudança na atitude positiva ou negativa sobre a utilidade percebida e facilidade de uso na utilização do arcabouço AM-TaaS?

5.2 Planejamento do Estudo

Este estudo tem como objetivo a execução de testes para aplicações móveis por meio da utilização do Arcabouço AM-TaaS, no qual foram usados aplicações móveis e scripts de testes automatizados previamente selecionados. Com os resultados obtidos a partir da realização desse estudo é possível caracterizar o arcabouço proposto em relação ao uso de um ambiente de teste local. Entretanto, a comparação entre os recursos computacionais usados nos dois ambientes de teste não é o objetivo principal, devido ao fato de que os dois ambientes usaram os mesmos recursos computacionais, porém os ambientes de teste eram diferentes.

Portanto, o objetivo dessa comparação é de caracterizar AM-TaaS em relação a outro ambiente de teste o “Amb. Local”. Esse estudo de viabilidade busca construir corpo de conhecimento que permita avaliar a viabilidade do arcabouço proposto. Com isso, espera-se que os resultados obtidos e o corpo de conhecimento decorrente de sua condução forneçam informações que permitam evoluir o arcabouço e otimizar seu uso na execução de teste para aplicações móveis.

Para avaliar o arcabouço, foi analisado o tempo de execução dos Casos de Teste em duas partes. A primeira parte foi chamada de Tempo de Execução (TE), o qual é medido em

relação ao tempo gasto na execução de um conjunto de quatro Casos de Teste que foram executados nos dois ambientes. A segunda parte é chamada de Tempo de Execução Total (TT), que considera todo o processo de teste, desde a configuração dos ambientes de teste (emuladores), passando pela execução dos scripts até a apresentação dos resultados.

Por outro lado, também foi analisado o número de Casos de Teste que falharam nos dois ambientes de teste. Finalmente foi analisada a utilidade percebida em relação ao uso dos dois ambientes de teste (AM-TaaS e Ambiente Local).

5.2.1 Formulação das Hipóteses

A premissa por trás de um serviço de teste para aplicações móveis é o tempo que leva a execução, o nível de utilidade e a facilidade de uso que os usuários percebem. Assim, foi definida a seguinte hipótese nula para este estudo:

Hipótese nula (H0): Não há diferença entre os resultados obtidos usando AM-TaaS e o Ambiente Local em relação ao Tempo Total (TT) gasto na execução dos testes e o número de Casos de Teste que falharam (CTF).

Para se obter resultados que venham a refutar ou confirmar a hipótese H0, a eficiência dos ambientes de teste foi avaliada. Portanto, para cada característica, as seguintes hipóteses foram definidas:

Hipótese nula A (H0 A): O tempo gasto na execução total dos testes usando o arcabouço AM-TaaS é o mesmo tempo usando o Ambiente de Teste Local.

H0 A: $TT_{Amb.\ Local} = TT_{AM\text{-}TaaS}$

- **Hipótese Alternativa 1 A (H1 A):** O tempo gasto na execução total dos testes usando o arcabouço AM-TaaS e o Ambiente de Teste Local são diferentes.
  H_{1A}: $TT_{Amb.\ Local} \neq TT_{AM\text{-}TaaS}$

Hipótese nula B (H0 B): O número de Casos de Teste executados que falharam usando o arcabouço AM-TaaS é o mesmo que usando o Ambiente de Teste Local.

H0 B: $CTF_{Amb.\ Local} = CTF_{AM\text{-}TaaS}$

- **Hipótese Alternativa 1 B (H1 B):** O número de Casos de Teste executados que falharam usando o arcabouço AM-TaaS e o Ambiente de Teste Local são diferentes.
  H_{1B}: $CTF_{Amb.\ Local} \neq CTF_{AM\text{-}TaaS}$

- $TT_{Amb.\ Local}$ = Tempo médio na execução total das atividades de teste usando o Ambiente de Teste Local.
- $TT_{AM\text{-}TaaS}$ = Tempo médio na execução total das atividades de teste usando AM-TaaS
- $CTF_{Amb.\ Local}$ = Número de Casos de Teste executados que Falharam usando o Ambiente de Teste Local.
- $CTF_{AM\text{-}TaaS}$ = Número de Casos de Teste executados que Falharam usando AM-TaaS.

5.2.2 Testes Estatísticos

A análise dos resultados obtidos do experimento foi feita baseado em Bisquerra *et al.* (2004), seguindo a execução dos seguintes testes estatísticos:

1. Primeiramente foi executado o Teste de Normalidade (*Shapiro-Wilk*) com o objetivo de avaliar se a distribuição obtida era normal ou não;
2. Para as distribuições normais, aplicou-se o Teste de Homocedasticidade (*Levene Test*) para verificar se as variâncias obtidas dos resultados do experimento eram homogêneas ou heterogêneas entre si;
    - Para as distribuições homogêneas (homocedásticas), foi aplicado o teste estatístico Análise de Variância (ANOVA);
    - Para as distribuições heterogêneas, foi aplicado o teste estatístico *Welch ANOVA*.
3. Para as distribuições não normais e amostras independentes, foi usado o teste Não Paramétrico *Mann-Whitney* como forma de testar a hipótese;
4. Para apoiar a análise estatística foi utilizada a ferramenta SPSS v22.0[16];
5. O nível de confiança usado foi de 95% (α = 0,05).

5.2.3 Seleção de Participantes

Os participantes para esse estudo foram provenientes da lista de estudantes matriculados na disciplina Verificação & Validação nos cursos de graduação e pós-graduação do Instituto de Computação (IComp) da Universidade Federal do Amazonas no período 2016/01. Entretanto, para participar do estudo, os estudantes tiveram que:

1. Manifestar interesse em participar do estudo, concordando com o Termo de Consentimento Livre e Esclarecido e Formulário de caracterização, para termos

[16] http://www-03.ibm.com/software/products/pt/spss-statistics

conhecimento do grau de experiência de cada aluno. Assim como, ambiente de teste e plataformas usadas, ferramentas e tipos de serviços de teste para aplicações na nuvem. Instrumentos disponíveis no Apêndice B e em https://pt.surveymonkey.com/r/DQTCMHG.

2. Participar de um treinamento nas ferramentas de teste que seriam usadas no estudo e o uso do Arcabouço AM-TaaS.

5.2.4 Seleção de Grupos

Um total de 18 participantes foram divididos em dois grupos, Grupo A e Grupo B, levando em consideração as informações que foram preenchidas por eles mesmos, sobre o grau de conhecimento e experiência em desenvolvimento e teste de aplicações móveis.

O formulário de caracterização apoiou divisão dos participantes e balanceamento nos grupos. Analisando o perfil dos participantes, 44.4% (8) deles tinham conhecimento sobre desenvolvimento de aplicações móveis e 33.3% (6) tinham alguma experiência em teste de software. Essas informações ajudaram na execução do treinamento de maneira a uniformizar os conhecimentos sobre teste de aplicações móveis.

5.2.5 Materiais

Devido ao experimento envolver execução dos testes na plataforma Android, foi necessário o uso dos seguintes elementos:

1) Infraestrutura: para a execução desse experimento foi necessário o uso do Laboratório de Informática do ICOMP-UFAM. Esse ambiente foi utilizado quanto para a execução dos testes utilizando o Ambiente de Teste Local e o arcabouço AM-TaaS;

2) Aplicações Móveis: o critério de seleção das aplicações móveis envolveu as seguintes características: (1) A aplicação não deve precisar de conexão à Internet para seu funcionamento, pois isso evita possíveis problemas de conexão que possam impactar na execução do experimento. (2) A aplicação deve ter elementos que possam ser acessíveis (e.g.: caixas de texto, lista de opções e botões). As duas aplicações móveis escolhidas cumprem com esses requisitos e apresentam ações semelhantes como adicionar, editar, deletar, procurar. Essas aplicações foram selecionadas a partir do repositório F-Droid[17]. São elas:

[17] https://f-droid.org/repository/browse/

- **Daily Money (Aplicação 1 – APP-1):** é uma aplicação financeira que permite gerenciar as despesas diárias, ativos e passivos. A Figura 5.1 apresenta as principais telas da aplicação.

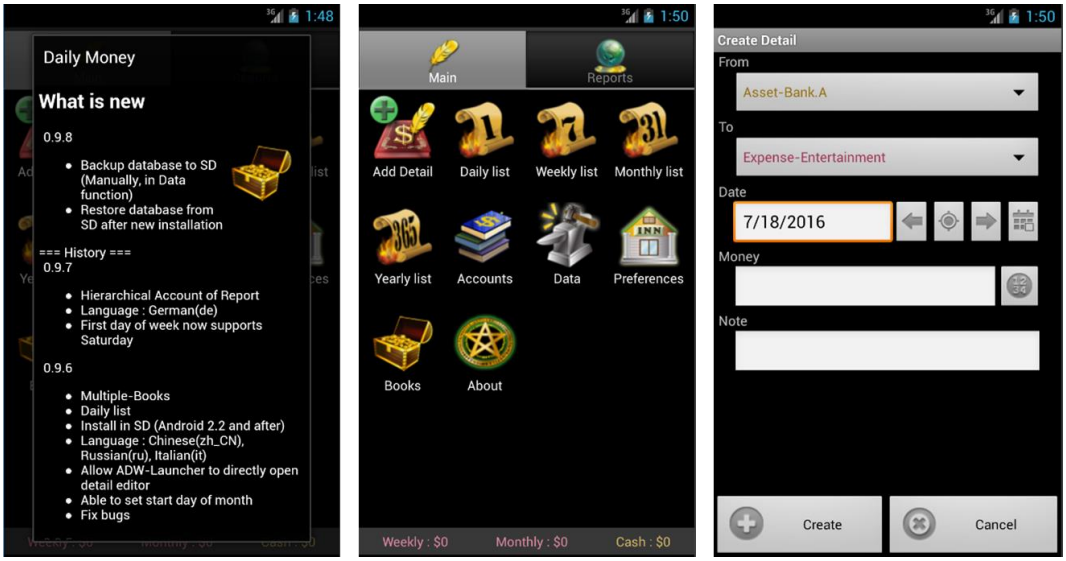


Figura 5.1. Telas da Aplicação A: *Daily Money*.

- **Gradr (Aplicação 2 – APP-2):** é uma aplicação que permite acompanhar de forma simples e fácil as notas do aluno. As principais opções são gerenciar cursos e notas para cada curso, além de permitir prever os pontos necessários para obter a nota desejada. A Figura 5.2 apresenta as principais telas da aplicação.

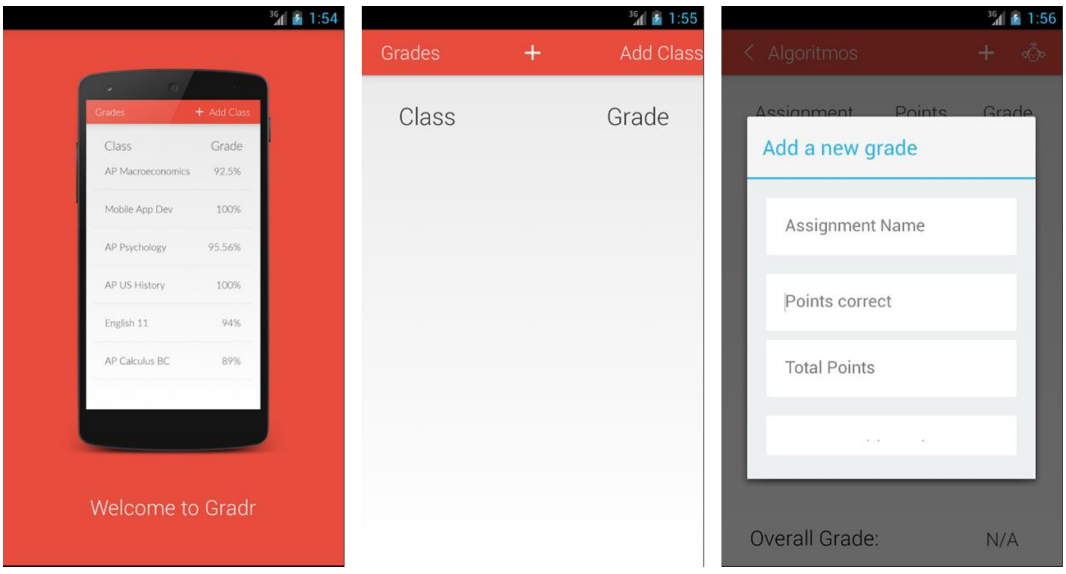


Figura 5.2. Telas da Aplicação B: Gradr.

3) Casos de Teste: para as duas aplicações móveis foi criado um conjunto de quatro Casos de Teste automatizados. Os testes estão relacionados a ações de criação, edição, atualização e remoção de registros. Os scripts foram automatizados na linguagem Python.

4) Emuladores: considerando os recursos computacionais do laboratório disponível, foi designado o uso de três emuladores para a execução dos experimentos. As características desses emuladores foram escolhidas de forma a ter uma variedade nas versões de APIs e tamanho de tela, o skin usado para cada emulador correspondia ao modelo de dispositivo móvel, assim a visualização dos emuladores foi o mais próximo possível ao dispositivo real. Devido aos recursos necessários para o funcionamento, o modelo Nexus 6 precisou de 128MB de memória RAM a ser alocada (VM *Heap)* para um melhor desempenho do emulador, no caso dos outros dois emuladores não foi necessário acrescentar esse tamanho. Essas características são descritas na Tabela 5.1.

Tabela 5.1. Emuladores utilizados para o experimento.

MODELO	Galaxy Nexus	Nexus 5	Nexus 6
Target (API)	Google API Level 17	Google API Level 18	Google API Level 19
API Nome	Jelly Bean	Jelly Bean	KitKat
CPU/ABI	ARM (armeabi-v7a)	ARM (armeabi-v7a)	ARM (armeabi-v7a)
Teclado	Sim	Sim	Sim
Skin	Galaxy Nexus	Nexus 5	Nexus 6
Ram	1024	1024	1024
VM Heap	64	64	128
Armazenamento Interno	200	200	200
SD tamanho	100 MB	100 MB	100 MB
Snapshot	Não	Não	Não
Use Host GPU	Sim	Sim	Sim

5.3 Execução do Estudo

O estudo de viabilidade foi conduzido de forma online e executado em duas sessões. Em cada sessão os participantes foram divididos nos grupos Grupo A e Grupo B e foram usadas as duas aplicações (APP-1 e APP-2), que foram usadas de forma alternada. Este cenário é ilustrado na Figura 5.3.

5.3.1 Primeira Sessão

Nessa primeira sessão foi realizado um treinamento sobre teste de aplicações móveis e as ferramentas disponíveis para a execução dos testes. Foi ensinado o uso da ferramenta *AndroidViewClient*, que permite a execução dos testes, e *Culebra*, que é uma ferramenta que ajuda na criação dos scripts de teste a partir da gravação das interações com uma aplicação.

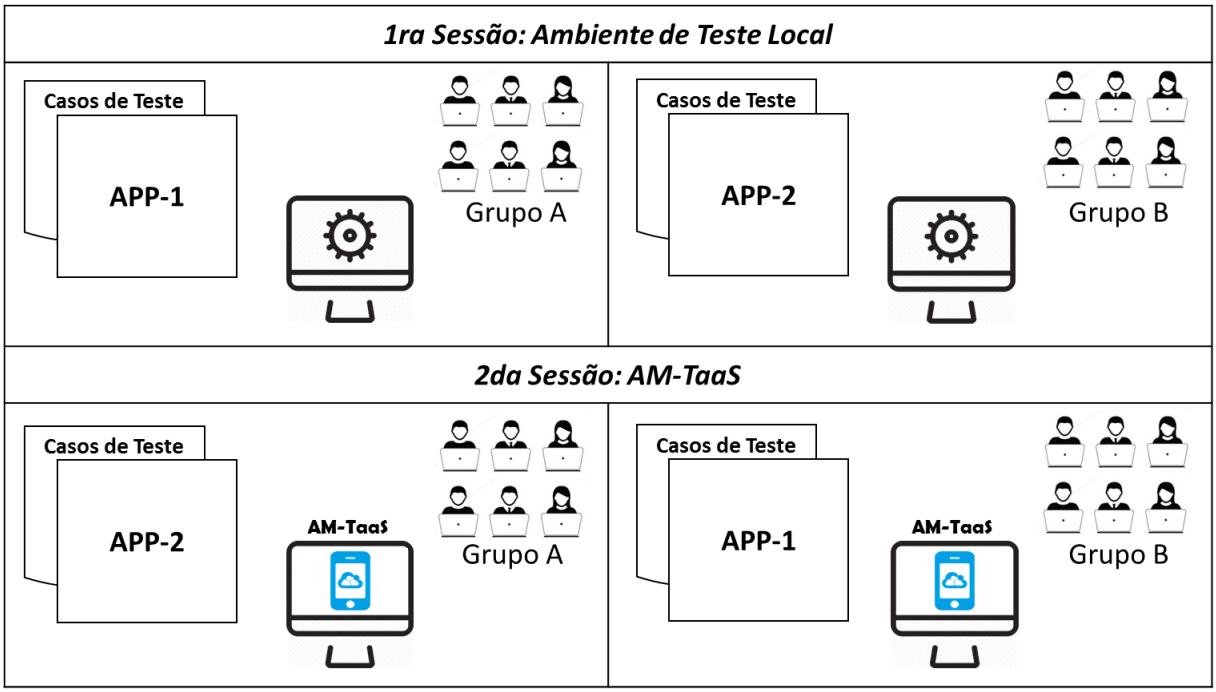


Figura 5.3. Cenários de Execução dos Testes.

O ambiente de teste usado nessa seção foi o Ambiente de Teste Local (Amb. Local) provido pelo Framework Android SDK. Para cada grupo foram disponibilizados os seguintes artefatos: O arquivo binário de uma aplicação (*<nomeApp>.apk*), um script de teste contendo um conjunto de quatro Casos de Teste (*fullteste<nomeApp>.py*), um guia com o passo a passo das atividades e os artefatos na qual seriam preenchidos os resultados dos testes. O guia completo pode ser visualizado no Apêndice F: Artefatos utilizados e guia das atividades - Sessão 1, um extrato dos passos da primeira atividade "*Criação de emuladores*" são:

Passo 1. Abra o AVD manager
Passo 2. Clique na aba "*Device Definitions*"
Passo 3. Selecione o modelo de dispositivo (*device*)
Passo 4. Clique no botão "*Create AVD*"
Passo 5. Registre a hora de inicio
Passo 6. Preencha as configurações
Passo 7. Clique no botão "OK"
Passo 8. Registre a hora de término

O treinamento partiu da ideia de que os participantes possuem uma aplicação desenvolvida e é necessária a execução dos testes. Portanto, a primeira atividade executada foi a "*Criação de três emuladores*". As características e os passos foram disponibilizados dentro dos artefatos entregues na sala; cada participante registrou o tempo de criação para cada emulador. A segunda atividade foi a "*Inicialização dos emuladores*", no

qual o tempo de inicialização também foi registrado. Finalmente, foi registrado o tempo de execução dos Casos de Teste em cada emulador. Esse tempo é chamado de Tempo de Execução dos Casos de Teste (TE). A soma dos tempos das três atividades realizadas é considerada como o Tempo Total do processo de teste (TT).

Após a conclusão das atividades, os participantes responderam a um questionário de Pós-treinamento (disponível no Apêndice D e em https://pt.surveymonkey.com/r/LMTM5G2), como forma de identificar as percepções sobre as atividades realizadas, utilizando a Escala de Likert (em quatro pontos, desde Discordo Totalmente a Concordo Totalmente), e identificar alguma mudança de atitude positiva (Concordo Totalmente ou Parcialmente) ou negativa (Descordo Totalmente ou Parcialmente) sobre o treinamento.

5.3.2 Segunda Sessão

Nessa segunda sessão o treinamento foi relacionado ao teste de aplicações na nuvem. Assim, foi explicada a estrutura do arcabouço AM-TaaS e suas funcionalidades. O ambiente de teste usado nessa seção foi o arcabouço AM-TaaS. Para cada grupo foram disponibilizados os seguintes artefatos: O arquivo binário de uma aplicação (*<nomeApp>.apk*), um script de teste contendo um conjunto de quatro Casos de Teste (*fullteste<nomeApp>.zip*), um guia com o passo a passo das atividades e os artefatos onde seriam preenchidos os resultados dos testes. É importante ressaltar que AM-TaaS já disponibiliza os emuladores prontos para serem utilizados.

O arcabouço AM-TaaS foi implementado em cada computador do laboratório, com o objetivo de todos os participantes usarem na mesma sessão. Dessa forma, todos podiam acessar por meio do link http://localhost/siteamtaas/. Seguindo os passos do funcionamento do arcabouço, os participantes iniciaram a atividade de teste. O primeiro passo foi submeter os artefatos de entrada do arcabouço (arquivo binário.*apk* e arquivo dos scripts de teste.*zip*). Em seguida, deveriam ser selecionados três emuladores (os mesmos que foram usados na primeira seção) e então iniciar o teste. Os guias utilizados podem ser visualizados no Apêndice G: Artefatos utilizados e guia das atividades - Sessão 2.

O AM-TaaS por padrão executa três Casos de Teste básicos para todos os emuladores selecionados. Esses testes estão relacionados ao critério *OTA Install* do AQuA, que contém o teste de instalação, instalação com memória insuficiente e desinstalação. Com isso, foram registrados o Tempo de execução dos testes do critério *OTA Install* e o Tempo de Execução dos Casos de Teste submetidos pelo usuário (TE), que são os quatro testes previamente definidos neste estudo. Assim, a soma desses tempos é considerada o Tempo Total da atividade de teste (TT).

Após a conclusão das atividades, os participantes responderam a um questionário Pós-treinamento como forma de identificar as percepções sobre as atividades realizadas com o arcabouço AM-TaaS, utilizando a Escala de Likert (em quatro pontos, desde Discordo Totalmente a Concordo Totalmente). Ainda como parte do Pós-Questionário, foram incluídas questões sobre a percepção de utilidade e facilidade de uso para serem analisados seguindo a metodologia do Modelo de Aceitação Tecnológica (em inglês, *Technology Acceptance Model - TAM*) (Davis *et al.*; 1989), disponível no Apêndice E, e em https://pt.surveymonkey.com/r/JDB6JLB.

A Figura 5.4 apresenta as ferramentas e artefatos usados em cada sessão. Todas as atividades em cada sessão tiveram uma duração de 2 horas. Em cada sessão os participantes receberam informações de como seria a dinâmica de treinamento e deveriam ficar atentos em relação ao tempo de execução das atividades designadas. Além disso, os participantes foram orientados a preencher os formulários preparados para cada atividade. Após a conclusão do experimento os resultados foram coletados para análise.

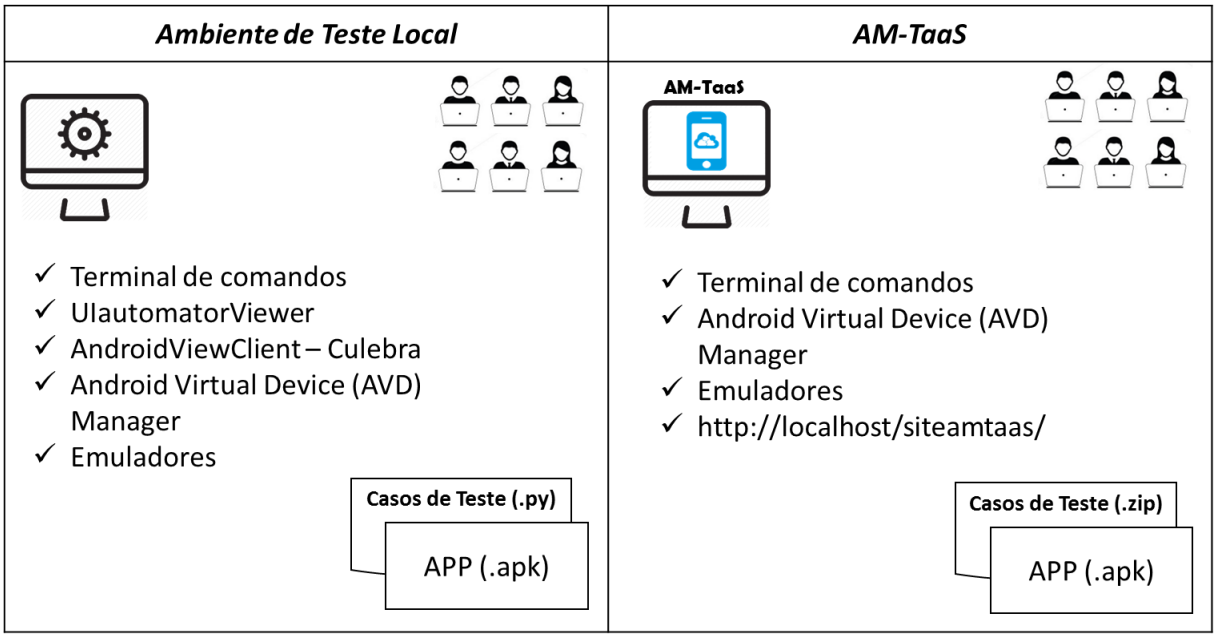


Figura 5.4. Ferramentas utilizadas no experimento.

5.4 Resultados do Estudo de Viabilidade

Neste estudo foram realizadas análises quantitativas e qualitativas em cima dos dados obtidos. Os dados quantitativos foram obtidos a partir do tempo registrado nas atividades de cada sessão. Os resultados qualitativos, por sua vez, foram obtidos a partir das respostas dos participantes aos questionários Post-Sessão.

5.4.1 Análise Quantitativa

Nesta seção será apresentada a análise do tempo de execução dos testes divididas em duas variáveis: primeiramente é analisado o Tempo de Execução (TE) de quatro Casos de Teste em ambos os ambientes, em seguida é analisado o Tempo Total (TT) de toda a atividade de teste em cada ambiente. A variável TT é utilizada para analisar a hipótese definida para este estudo (Hipótese H0 A).

5.4.1.1 Análise do tempo

Os dados do tempo foram coletados separadamente. Na primeira parte é considerado o Tempo de Execução (TE) a partir dos quatro casos de teste gerados para cada aplicação. Esse conjunto de casos de teste foi executado nos dois ambientes. Na segunda parte é considerado o Tempo Total (TT) o qual refere-se à atividade completa do teste que inclui o TE e as atividades de cada ambiente. No caso do Ambiente de Teste Local (Amb. Local), o Tempo Total inclui a criação de emuladores, inicialização e execução dos testes; e no caso do AM-TaaS o Tempo Total inclui os testes padrão do AM-TaaS e a execução dos quatro casos de teste gerados para cada aplicação, considerando que no ambiente proposto não existe atividade de criação e inicialização de emuladores, pois eles já estão configurados em AM-TaaS. Conforme a Seção 5.2.5, os casos de teste foram executados nos três emuladores utilizados neste estudo: *Galaxy Nexus*, *Nexus 5* e *Nexus 6*.

Após os resultados serem tabulados, a Tabela 5.2 e a Tabela 5.3 apresentam os resultados em cada sessão, respectivamente em relação a métrica analisada em este estudo. O tempo registrado (mm:ss) para cada emulador é o Tempo de Execução (TE) dos quatro casos de teste projetados para cada aplicação e o Tempo Total (TT) em que foi concluída toda a atividade. Ao todo foram executados 144 Casos de Teste na Primeira Sessão (Amb. Local) e 144 Casos de Teste na Segunda Sessão (AM-TaaS).

Tabela 5.2. Resultados da Primeira Sessão (Amb. Local).

Ambiente	APP	Participante	Nome do AVD	Tempo Execução TE	Tempo Total TT
Amb. Local	APP A	A1	AVD17_GalaxyNexus	05:00	07:00
			AVD18_Nexus5	02:00	06:00
			AVD19_Nexus6	02:00	10:00
		A2	AVD17_GalaxyNexus	12:00	17:00
			AVD18_Nexus5	02:00	07:00
			AVD19_Nexus6	00:00	07:00
		A3	AVD17_GalaxyNexus	03:00	08:00
			AVD18_Nexus5	06:00	11:00
			AVD19_Nexus6	02:00	11:00
		A5	AVD17_GalaxyNexus	05:00	10:00
			AVD18_Nexus5	06:00	11:00
			AVD19_Nexus6	02:00	12:00
		A6	AVD17_GalaxyNexus	11:00	15:00
			AVD18_Nexus5	06:00	11:00
			AVD19_Nexus6	04:00	17:00
		A7	AVD17_GalaxyNexus	09:00	19:00
			AVD18_Nexus5	05:00	12:00
			AVD19_Nexus6	08:00	17:00
	APP B	B1	AVD17_GalaxyNexus	03:00	10:00
			AVD18_Nexus5	01:00	06:00
			AVD19_Nexus6	00:00	08:00
		B3	AVD17_GalaxyNexus	04:00	08:00
			AVD18_Nexus5	06:00	11:00
			AVD19_Nexus6	07:00	15:00
		B4	AVD17_GalaxyNexus	04:00	08:00
			AVD18_Nexus5	05:00	09:00
			AVD19_Nexus6	07:00	14:00
		B6	AVD17_GalaxyNexus	05:00	11:00
			AVD18_Nexus5	04:00	12:00
			AVD19_Nexus6	05:00	13:00
		B7	AVD17_GalaxyNexus	08:00	12:00
			AVD18_Nexus5	04:00	10:00
			AVD19_Nexus6	07:00	15:00
		B8	AVD17_GalaxyNexus	05:00	07:00
			AVD18_Nexus5	02:00	06:00
			AVD19_Nexus6	03:00	09:00

Tabela 5.3. Resultados da Segunda Sessão (AM-TaaS).

Ambiente	APP	Participante	Nome do AVD	Tempo Execução TE	Tempo Total TT
AM-TaaS	APP B	A1	AVD18_Nexus5	01:20	02:20
		A2	AVD17_GalaxyNexus	03:52	05:10
			AVD18_Nexus5	01:23	02:39
		A3	AVD17_GalaxyNexus	03:43	04:52
			AVD18_Nexus5	01:03	02:14
			AVD19_Nexus6	04:48	05:50
		A4	AVD18_Nexus5	02:10	07:16
			AVD19_Nexus6	04:46	05:45
		A5	AVD17_GalaxyNexus	03:42	04:51
			AVD18_Nexus5	01:21	02:33
			AVD19_Nexus6	06:01	09:42
		A6	AVD17_GalaxyNexus	03:41	04:56
			AVD18_Nexus5	01:22	02:40
			AVD19_Nexus6	06:05	11:17
		A7	AVD17_GalaxyNexus	03:43	05:01
			AVD18_Nexus5	01:19	02:34
			AVD19_Nexus6	06:03	11:56
	APP A	B1	AVD18_Nexus5	04:56	06:05
			AVD19_Nexus6	06:45	12:28
		B2	AVD17_GalaxyNexus	06:17	07:33
			AVD18_Nexus5	05:04	06:21
			AVD19_Nexus6	06:30	11:39
		B3	AVD17_GalaxyNexus	06:18	07:29
			AVD18_Nexus5	05:00	06:10
			AVD19_Nexus6	06:32	10:31
		B4	AVD17_GalaxyNexus	06:11	07:19
			AVD18_Nexus5	04:59	06:07
			AVD19_Nexus6	06:29	10:08
		B5	AVD17_GalaxyNexus	06:17	07:31
			AVD18_Nexus5	05:01	06:33
		B6	AVD17_GalaxyNexus	02:47	03:53
			AVD18_Nexus5	06:28	07:33
			AVD19_Nexus6	06:37	10:47
		B8	AVD17_GalaxyNexus	06:12	08:04
			AVD18_Nexus5	02:27	03:37
			AVD19_Nexus6	06:45	11:53

1) Análise das Médias por Ambiente

Para analisar a média em relação ao Tempo de Execução (TE) e o Tempo Total (TT), foram levados em consideração os tempos registrados nas duas sessões desse estudo. Com esta informação calculou-se a média, mediana e o desvio padrão paras as

duas variáveis. Os resultados são apresentados na Tabela 5.4 e Tabela 5.5, respectivamente.

Tabela 5.4. Média, Mediana e Desvio padrão do Tempo de Execução (TE) nos Quatro Casos de Teste por Ambiente.

Emuladores	Amb. Local			AM-TaaS		
	Média	Mediana	Desvio Padrão	Média	Mediana	Desvio Padrão
AVD17_GalaxyNexus	06:10	05:00	03:04	04:48	03:52	01:25
AVD18_Nexus5	04:05	04:30	01:53	03:08	02:18	01:58
AVD19_Nexus6	03:55	03:30	02:51	06:07	06:29	00:43
TOTAL	**14:10**			**14:03**		

Analisando a variável Tempo de Execução dos 4CTs, observou-se que existe uma pequena diferença entre os dois ambientes, sendo que o "Amb. Local" demorou 14min10seg em executar os 4CTs nos três emuladores, e AM-TaaS levou 14min03seg. Assim, o tempo é aproximadamente similar para os dois ambientes de teste. É possível perceber que o desempenho dos emuladores também é variável. No Amb. Local o emulador *AVD18_Nexus5* teve a menor variação (01:53), já no AM-TaaS o emulador *AVD19_Nexus6* teve a menor variação (00:43). O desempenho dos emuladores está sujeito aos recursos computacionais onde são executados os testes. Não é possível afirmar que os dois emuladores que tiveram menor variação serão sempre os melhores.

Tabela 5.5. Média, Mediana e Desvio padrão do Tempo Total de Execução (TT) por Ambiente.

Emuladores	Amb. Local			AM-TaaS		
	Média	Mediana	Desvio Padrão	Média	Mediana	Desvio Padrão
AVD17_GalaxyNexus	11:00	10:00	04:01	06:04	05:10	01:31
AVD18_Nexus5	09:20	10:30	02:25	04:37	04:51	02:06
AVD19_Nexus6	12:20	12:30	03:24	10:11	10:47	02:19
TOTAL	**32:40**			**20:51**		

Analisando a variável Tempo Total de Execução, observou-se que existe uma diferença de 11min49seg entre o Tempo Total de AM-TaaS (20min51seg) e o Amb. Local (32min40seg). Isso indica que AM-TaaS obteve um tempo melhor do que o ambiente Amb. Local. Sobre o desempenho dos emuladores, no Amb. Local o emulador *AVD18_Nexus5* teve a menor variação e em AM-TaaS a menor variação foi no emulador

AVD17_GalaxyNexus. Com isso, é possível perceber que dependendo do desempenho dos emuladores, os resultados para os dois ambientes podem variar.

A Figura 5.5 apresenta graficamente os resultados das médias nas duas variáveis Tempo de Execução dos 4 Casos de Teste e o Tempo Total.

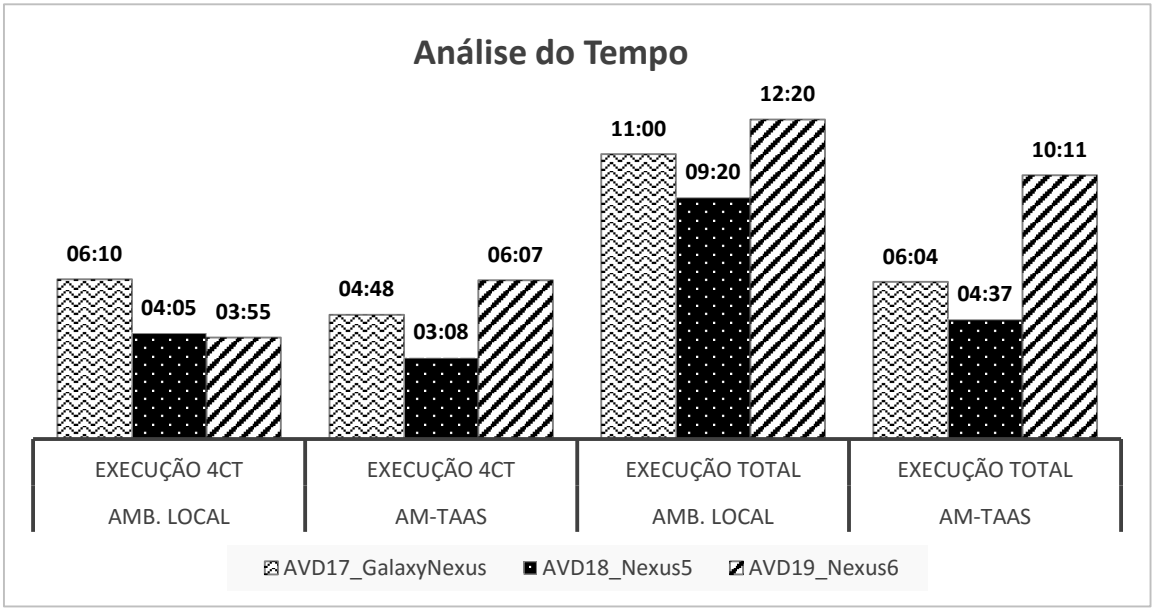


Figura 5.5. Análise da Média do Tempo por Ambiente.

2) Análise das Médias por Aplicação

As aplicações utilizadas nesse estudo são de características similares. Isso é refletido na similaridade de tempos de execução. O Tempo médio para as duas aplicações no Tempo Total (TT) levam menos tempo se executados no ambiente AM-TaaS do que no Ambiente Local (Amb. Local), apresentado na Figura 5.6.

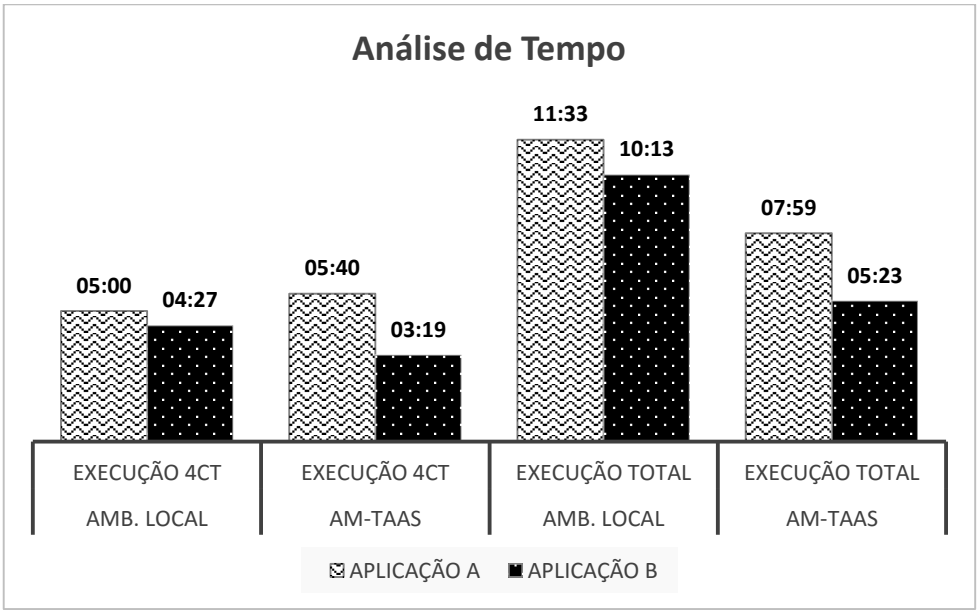


Figura 5.6. Análise de Média do Tempo por Aplicação.

Com os resultados obtidos, calculou-se a média, mediana e desvio padrão para as variáveis Tempo de Execução dos quatro Casos de Teste (TE) e para o Tempo Total (TT), apresentado na Tabela 5.6 e Tabela 5.7, respectivamente.

Tabela 5.6. Média, Mediana e Desvio padrão do Tempo de Execução dos 4CT (TE) por Aplicação.

Aplicações	Amb. Local			AM-TaaS		
	Média	Mediana	Desvio Padrão	Média	Mediana	Desvio Padrão
APLICAÇÃO A	**05:00**	05:00	03:20	**05:40**	06:17	01:16
APLICAÇÃO B	**04:27**	04:30	02:09	**03:19**	03:42	01:50

Analisando a variável Tempo de Execução dos 4CTs, observa-se que a variação no tempo usando o ambiente do AM-TaaS é menor nas duas aplicações (01:16 e 01:50). Porém, a Aplicação A teve um melhor tempo médio de execução usando o Amb. Local, enquanto que a Aplicação B teve um melhor tempo usando AM-TaaS.

Tabela 5.7. Média, Mediana e Desvio padrão do Tempo Total de Execução (TT) por Aplicação.

Aplicações	Amb. Local			AM-TaaS		
	Média	Mediana	Desvio Padrão	Média	Mediana	Desvio Padrão
APLICAÇÃO A	**11:33**	11:00	03:58	**07:59**	07:31	02:35
APLICAÇÃO B	**10:13**	10:00	02:52	**05:23**	04:56	03:04

Analisando a variável Tempo de Execução Total, observa-se que usando o ambiente do AM-TaaS obteve-se menor tempo de execução em comparação à execução no Ambiente Local. No entanto, a variação dos tempos em cada aplicação é diferente nos dois ambientes de teste.

Como forma de entender a variabilidade do Tempo de Execução dos testes (TE) e o Tempo de Execução Total (TT) por Ambiente e Aplicação, foi utilizado o diagrama de caixa (*boxplot*) conforme na Figura 5.7 e Figura 5.8, respectivamente.

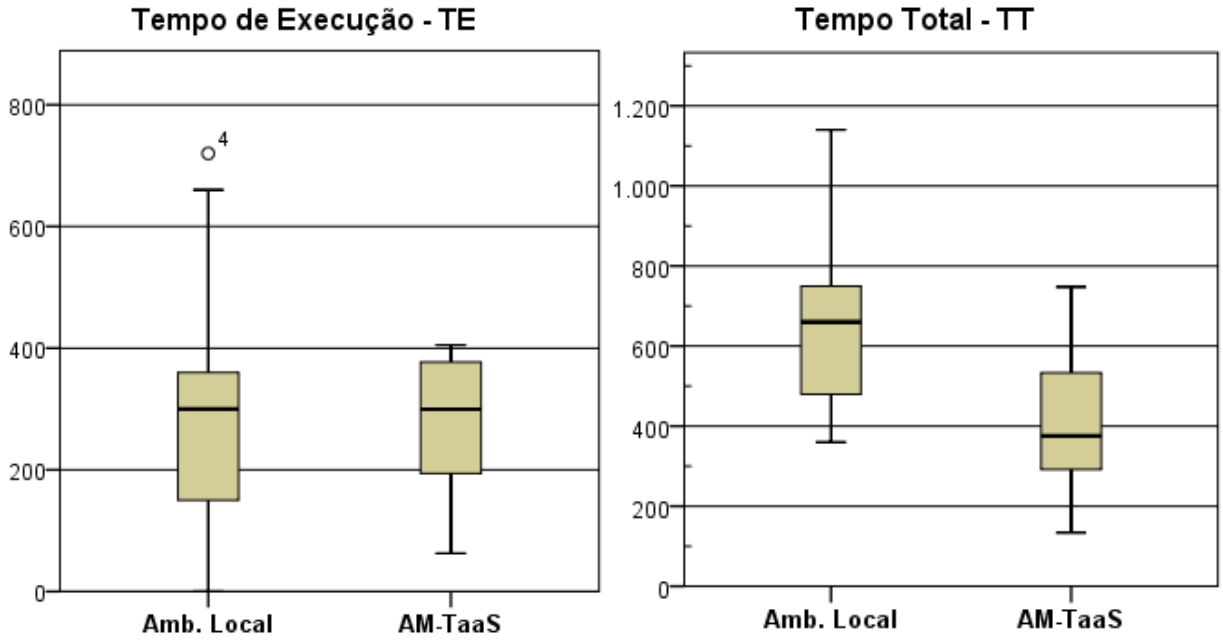


Figura 5.7. Diagrama de Caixa – Variabilidade do Tempo por Ambiente.

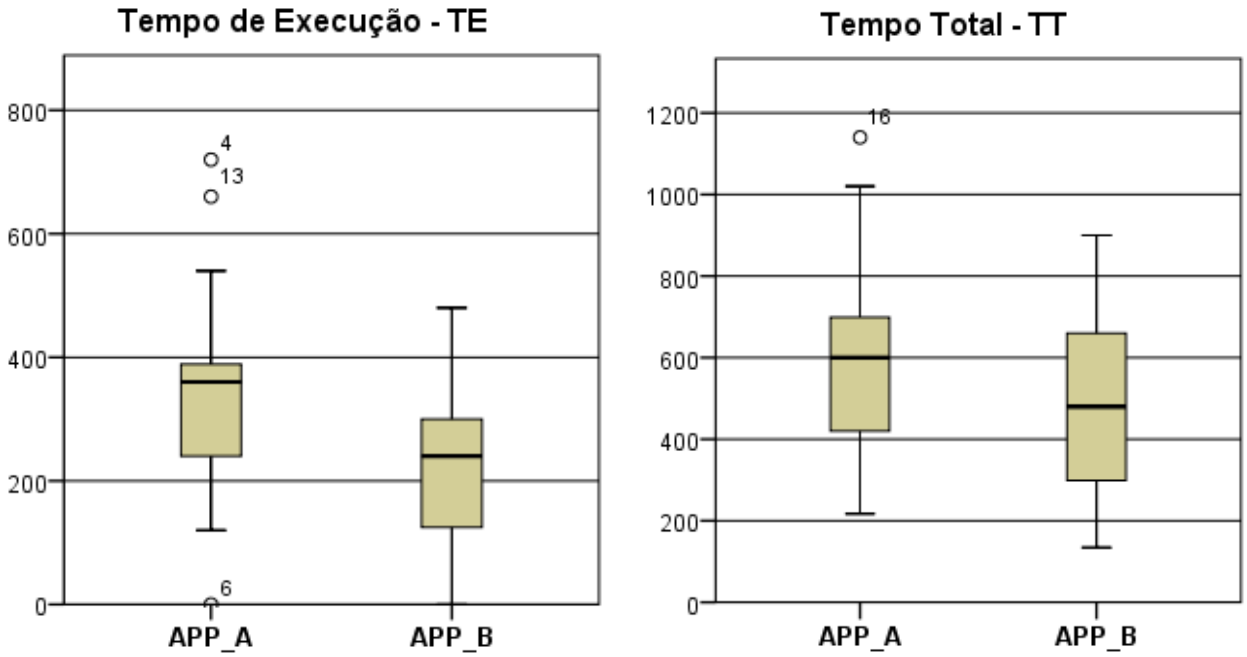


Figura 5.8. Diagrama de Caixa – Variabilidade do Tempo por Aplicação.

3) Teste de Normalidade

Como forma de testar a hipótese nula H0 definida na Seção 5.2.1, seguiram-se os procedimentos para o teste estatístico indicados na Seção 5.2.2. A Tabela 5.8 apresenta os resultados dos testes de normalidade *Shapiro-Wilk* para as variáveis analisadas neste estudo (Tempo de Execução e Tempo Total). Considerando o resultado do valor de *p-value*, qualquer valor menor do que 0,05 indica que os dados não possuem distribuição normal

nem homocedasticidade; a homocedasticidade é testado só em caso dos dados possuam distribuição normal, caso contrário não são testados e identificados com a sigla NT (*Not Tested*). Para as variáveis analisadas neste estudo na análise por ambiente e por aplicação, os dados não possuem distribuição normal. Nesse caso devido a que as amostras são independentes foi utilizado o teste não paramétrico *Mann-Whitney* apresentado na Tabela 5.9.

Tabela 5.8. Teste de Normalidade e Homocedasticidade.

	Análise por Ambiente			Análise por Aplicação		
Variável Dependente	**Shapiro-Wilk**		**Levene Test**	**Shapiro-Wilk**		**Levene Test**
	Amb. Local	**AM-TaaS**		**APP A**	**APP B**	
Tempo Execução 4CTs	0,2082	0,0005	NT	0,0253	0,2679	NT
Tempo Total	0,0651	0,0497	NT	0,0187	0,1269	NT

Tabela 5.9. Resultados do teste Mann-Whitney.

	Mann Whitney	
Variável Dependente	**Por Ambiente**	**Por Aplicação**
Tempo Execução 4CTs	*p-value* = 0,87900	*p-value* = 0,00690
Tempo Total	*p-value* = 0,00001	*p-value* = 0,05316

4) Análise do Tempo de Execução 4CTs

Na análise por ambiente o *p-value* obtido foi de 0,87900, o que indica que não há diferença estatística significativa. Esse resultado era esperado, pois os quatro casos de teste que foram executados são os mesmos em cada ambiente. Porém, na análise por aplicação é possível perceber que existe diferença significativa (*p-value = 0,00690*). Esse resultado não era esperado. Logo, foi realizada uma análise das aplicações e uma das aplicações (Aplicação B) possui um defeito ocasionando um erro intermitente na primeira execução da aplicação. Se for executada uma segunda vez existe a possibilidade de o erro não ser executado outra vez. Portanto é possível que esse defeito tenha influenciado nos resultados obtidos na análise por aplicação.

5) Análise do Tempo Total

Considerando que o Tempo Total (TT) é a variável para a avaliação da hipótese desse estudo, na análise por ambiente o *p-value* obtido foi de 0,00001, o que indica que existe uma diferença estatística significativa entre os dois ambientes ao nível de 5% de significância (α = 0,05). Logo, a hipótese H0 A é rejeitada (*o tempo gasto na execução total*

dos testes usando o arcabouço AM-TaaS é o mesmo tempo usando o Ambiente de Teste Local). Já na análise por aplicação, não existe diferença significativa (*p-value = 0,05316*).

5.4.1.2 Analise de Casos de Teste

Em cada emulador foram executados quatro casos de teste e foram definidos três estados: Casos de Teste que Passaram (P), Falharam (F) e os que Não foram executados (X). No total foram executados 144 Casos de Teste na Primeira Sessão usando o ambiente "Amb. Local" e 144 casos de teste utilizando AM-TaaS. Na Figura 5.9 percebe-se a maior diferença no número de Casos de Teste que passaram entre os dois ambientes, no qual 66.6% dos Casos de Teste passaram usando AM-TaaS,

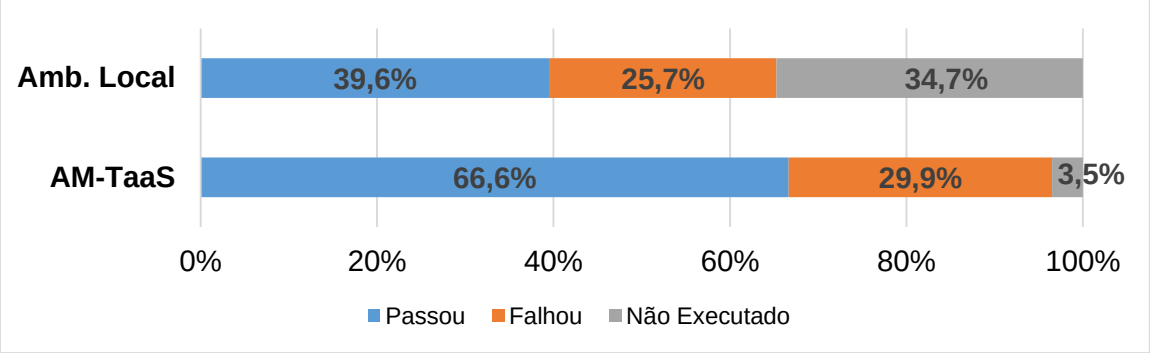


Figura 5.9. Número de Casos de Teste executados em cada ambiente.

Uma análise foi realizada sobre os Casos de Teste que falharam, a causa encontrada foi erro "*App crash*" enquanto que os que não foram executados (3.5%) foi devido a um erro na criação do arquivo XML da ferramenta AVC utilizada.

Já utilizando o Ambiente Local, obteve-se um maior índice que Casos de Teste Não executados (34.7%) que foram causados pela ferramenta AVC utilizada. Com isso, o número de Casos de Teste que falharam não são iguais, rejeitando a hipótese nula H0B *(O número de Casos de Teste executados que falharam usando o arcabouço AM-TaaS é o mesmo que usando o Ambiente de Teste Local*).

Com esses resultados foi percebido que a ferramenta AVC não está totalmente amadurecida e ainda possui bugs que comprometem seu funcionamento, esse erro já foi reportado, mas ainda não foi solucionado pelos autores, as futuras melhoras da ferramenta serão atualizadas em AM-TaaS.

1) Análise dos Casos de Teste por Emuladores

Cada emulador possui características diferentes e principalmente APIs diferentes, o que faz com que os resultados sejam diferentes entre eles, como é apresentado na Figura 5.10.

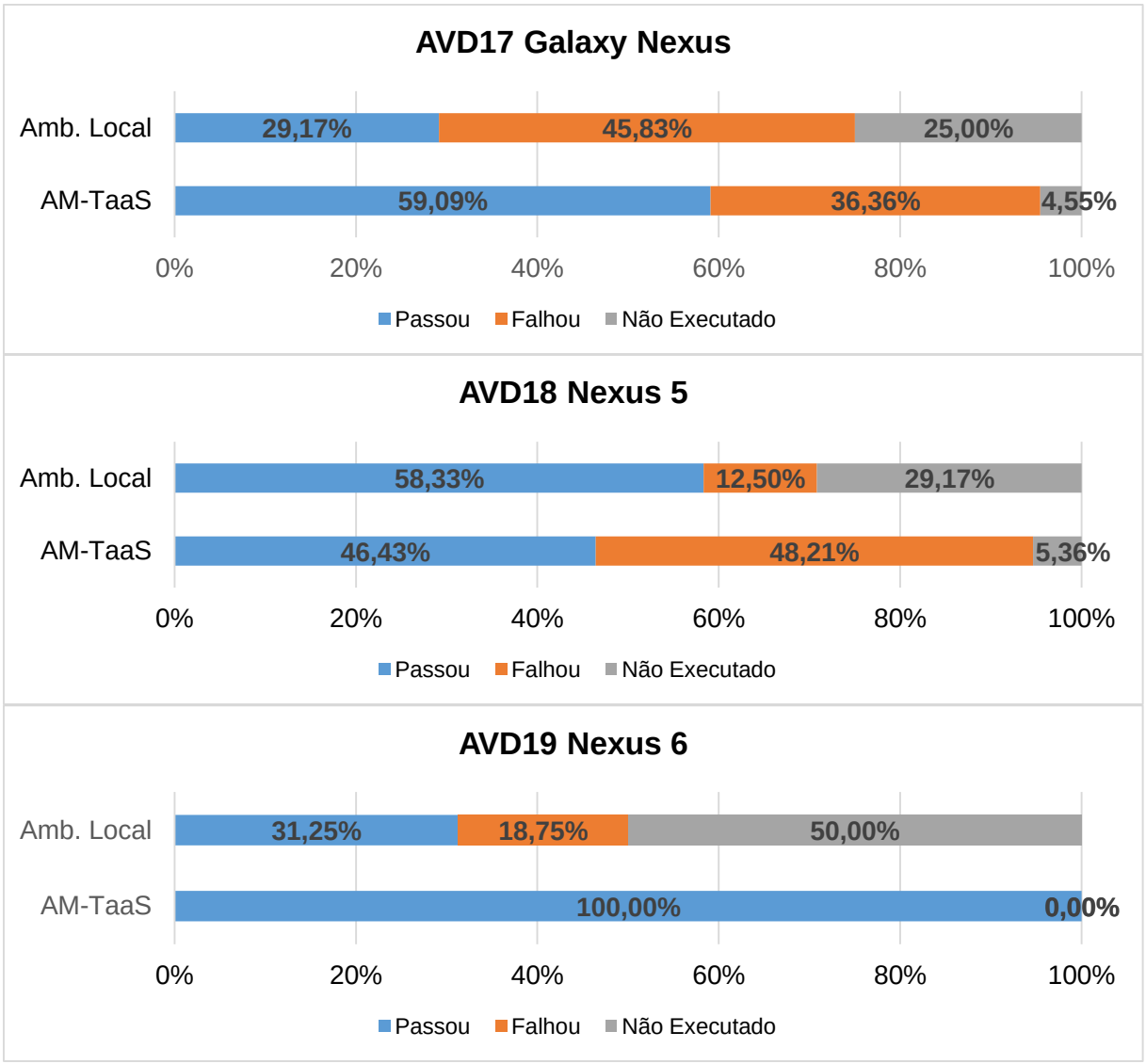


Figura 5.10. Casos de Teste por Emuladores.

As falhas que foram percebidas nos dois ambientes são: App crash e o erro na criação de um arquivo XML da ferramenta AVC, o número de ocorrências em cada emulador é apresentado na Tabela 5.10. O erro App Crash está relacionado aos Casos de Teste que falharam e o erro XML está relacionado aos Casos de Teste que não foram executados.

Tabela 5.10 Ocorrências registradas por Ambiente e Emulador.

Emuladores	Falhas	Amb. Local	AM-TaaS
AVD17_GalaxyNexus	App Crash	2	4
AVD18_Nexus5	App Crash	2	5
	AVC: erro XML	3	1
AVD19_Nexus6	AVC: erro XML	8	0

Os resultados para o emulador *AVD18_Nexus5* foram desfavoráveis para o ambiente proposto. Nesse caso será executado outro estudo para verificar se o erro tem a ver com a

versão do API do emulador ou com a aplicação. Finalmente, no *AVD19_Nexus6* surgiu o erro da ferramenta *AndroidViewClient* no Ambiente Local, fazendo com que muitos dos casos de teste não pudessem ser executados.

2) Análise dos Casos de Teste por Aplicação

A Aplicação A teve um melhor resultado utilizando AM-TaaS comparado com o Ambiente Local. Porém, no Ambiente Local foram percebidos um erro da aplicação (*crash*) e seis erros da ferramenta *AndroidViewClient* (AVC). Este erro está relacionado à criação de um arquivo XML gerado como resultado do comando *dump*, os quais ainda não foram resolvidOs pelos autores da ferramenta.

Na Aplicação B foram observadas cinco erros da ferramenta (AVC) e três erros da aplicação (*crash*) no Ambiente Local, enquanto que no AM-TaaS foram observados nove erros da aplicação (*crash*) e um erro da ferramenta (AVC). Estes dados influenciam muito na execução dos casos de teste, pois muitos deles não passaram ou não foram executados. O resumo dessas informações poder ser visualizado na Figura 5.11.

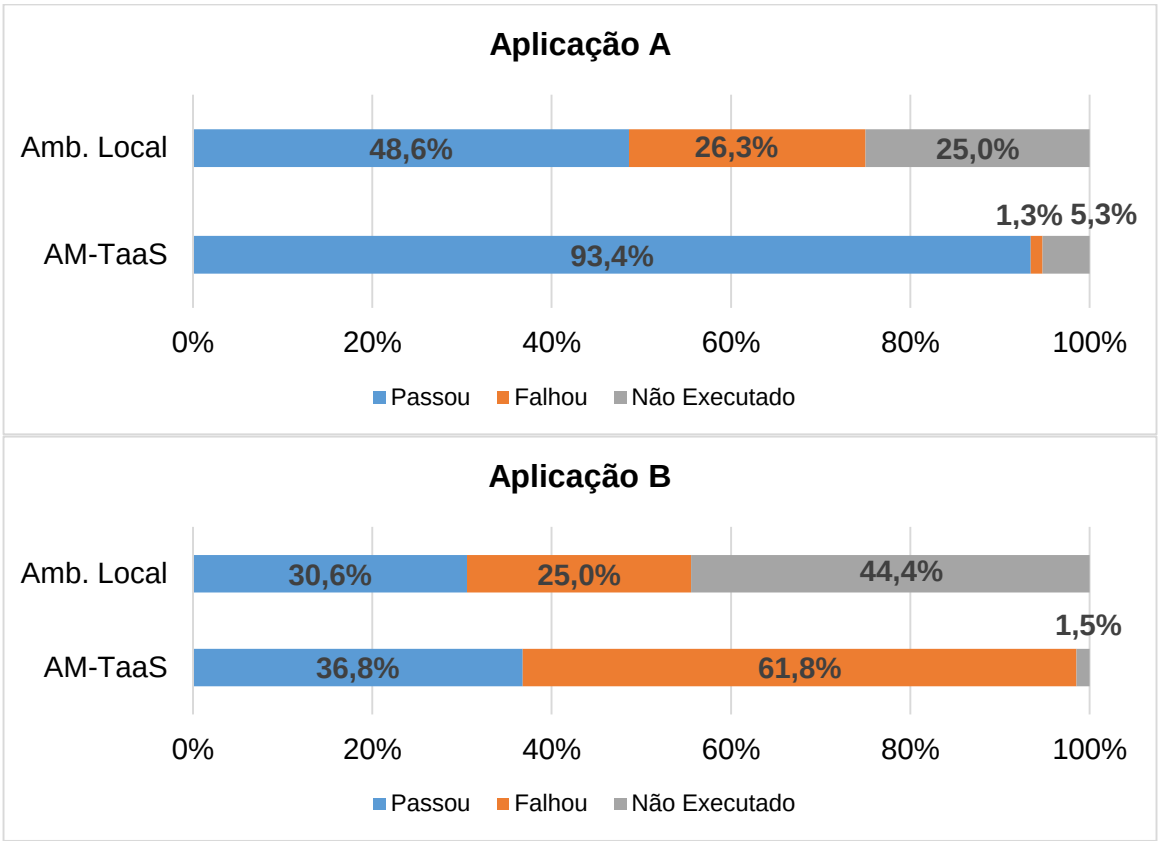
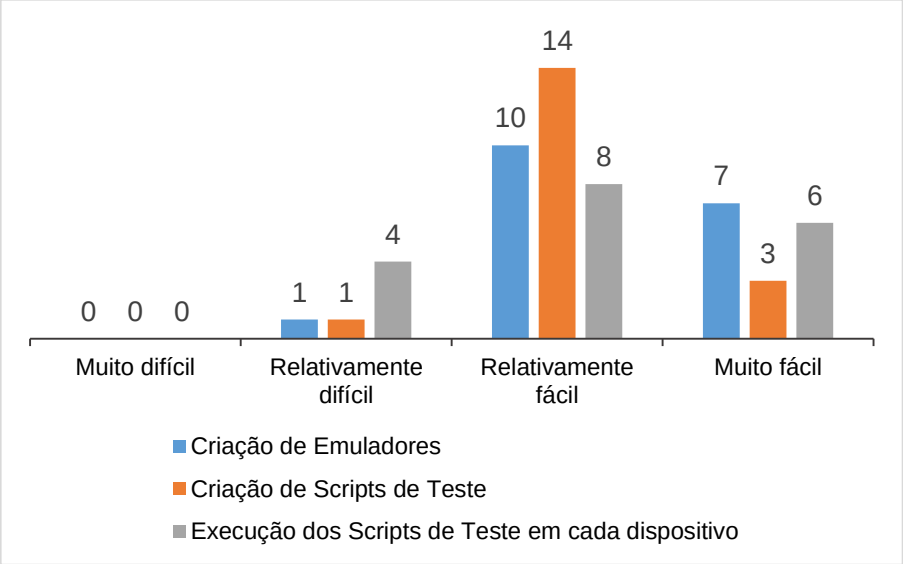


Figura 5.11. Casos de Teste por Aplicação Móvel.

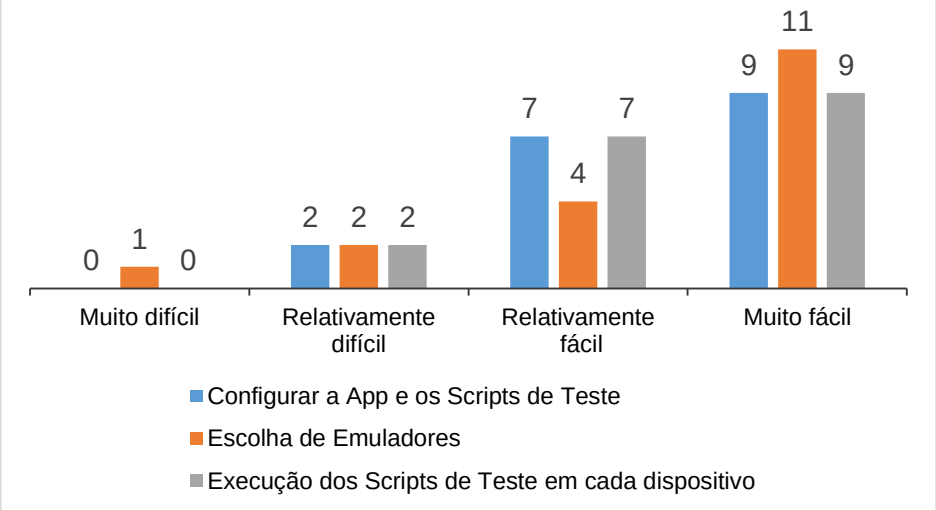
5.4.2 Análise Qualitativa

Nesta seção, analisamos de forma qualitativa a opinião dos participantes sobre as atividades executadas em cada ambiente. O resultado e as atividades são apresentados na

Figura 5.12. Na primeira sessão é possível perceber que a atividade relacionada à Criação e Execução de scripts não é considerada fácil; já na segunda sessão, as atividades Escolha de emuladores e Execução de scripts são consideradas atividades muito fáceis.



(a) Ambiente Local



(b) AM-TaaS

Figura 5.12. Opinião sobre as atividades executadas nas sessões nos dois ambientes (a) Ambiente Local e (b) AM-TaaS.

Como forma de medir a atitude dos participantes em relação ao nível de dificuldade das atividades, eles foram questionados a partir de duas questões: (1) Qual é a sua atitude frente a uma nova ferramenta para automação de teste? (2) Qual é a sua atitude em relação ao nível de dificuldade do processo de automação de testes (Emuladores + Scripts + Execução). As respostas seguem a escala de Likert em quatro pontos. Estas respostas foram agrupadas em “Atitude negativa” (considerando as respostas Relativamente difícil e

Muito difícil) e a “Atitude positiva” (considerando as respostas Relativamente fácil e Muito fácil). Assim, percebeu-se que depois da segunda sessão foi identificado uma atitude positiva maior em relação ao nível de dificuldade sobre o aprendizado de uma nova ferramenta. Estes resultados podem ser observados na Figura 5.13 (a) e (b), respectivamente, que respondem as seguintes questões:

(a) Qual é a sua atitude frente a uma nova ferramenta para automação de teste?
(b) Qual é a sua atitude em relação ao nível de dificuldade do processo de automação de testes (Emuladores + Scripts + Execução)?

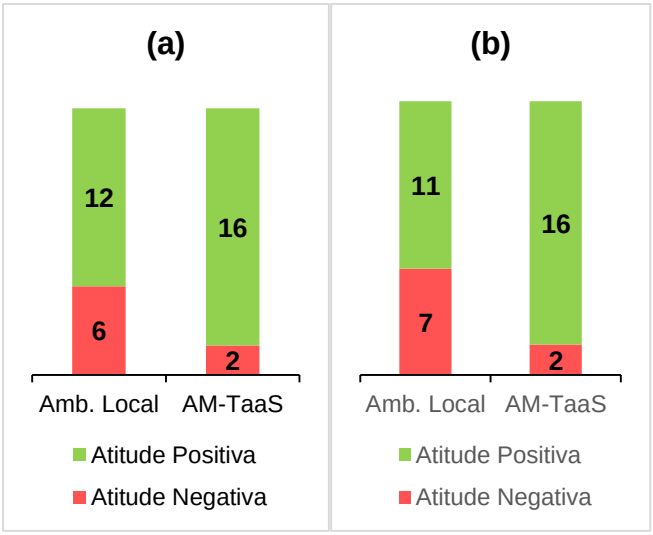


Figura 5.13. Mudança de atitude Post-Sessão.

5.4.3 Análise da Percepção do Usuário

Depois da análise qualitativa, foi executado um questionário relacionado à aceitação da tecnologia proposta, as respostas seguem a escala de Likert e as perguntas do questionário podem ser visualizadas no Apêndice E. Este questionário foi baseado nos indicadores do Modelo de Aceitação de Tecnologias (*Technology Acceptance Model - TAM*) (Davis *et al.*, 1989). Os indicadores definidos foram Utilidade Percebida (PU), Facilidade de Uso Percebida (EOU) e Intenção de Uso (IU), como é visualizado na Figura 5.14.

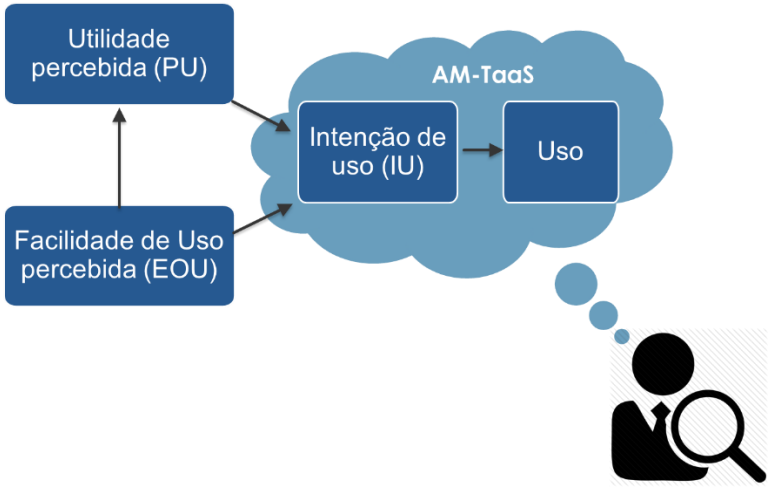


Figura 5.14. Aplicando TAM na Avaliação do AM-TaaS.

1) Análise da Utilidade Percebida (PU): é o grau em que um usuário acredita que utilizando os serviços providos pelo AM-TaaS terá uma melhoria na execução de suas atividades e no seu desempenho. Para avaliar este indicador as seguintes perguntas foram realizadas e os resultados são apresentados na Tabela 5.11.

[PU-1]: A utilização do AM-TaaS permite a execução de teste mais facilmente?

[PU-2]: A utilização do AM-TaaS permite reduzir o esforço dos usuários nas atividades de teste?

[PU-3]: A utilização do AM-TaaS iria melhorar meu desempenho nas atividades de teste?

[PU-4]: Utilizar o AM-TaaS não adiciona valor à atividade de teste?

[PU-5]: De forma geral, AM-TaaS é útil?

Tabela 5.11. Resultados sobre a Utilidade Percebida (PU).

Perguntas	Discordo plenamente	Discordo parcialmente	Concordo parcialmente	Concordo plenamente
PU-1	1 5,6%	0 0,0%	4 22,2%	13 72,2%
PU-2	2 11,1%	0 0,0%	3 16,7%	13 72,2%
PU-3	1 5,6%	0 0,0%	3 16,7%	14 77,8%
PU-4	15 83,3%	0 0,0%	0 0,0%	3 16,7%
PU-5	1 5,6%	0 0,0%	3 16,7%	14 77,8%

Dos resultados, 77,8% (14) dos participantes concordam que AM-TaaS é útil (PU-5) e 88,3% (15) discorda sobre a afirmação que AM-TaaS não adiciona valor a atividade de Teste (PU-4). Além disso, 72,2% (13) concordam que a AM-TaaS permite reduzir o esforço dos usuários nas atividades de teste (PU-2). Esses resultados nos permitem ter boas expectativas sobre a utilidade que usuário tem sobre o ambiente de teste proposto.

2) Análise da Facilidade de Uso percebida (EOU): é o grau em que um usuário acredita que o uso dos serviços providos pelo AM-TaaS não envolverá esforço. Para avaliar este indicador as seguintes perguntas foram realizadas e os resultados são apresentados na Tabela 5.12.

[EOU -1]: Eu achei AM-TaaS muito complicado de usar?

[EOU -2]: O uso do AM-TaaS é claro e compreensível?

[EOU -3]: De forma geral, AM-TaaS é fácil de usar?

Tabela 5.12. Resultados sobre a Facilidade de Uso percebida (EOU).

Perguntas	Discordo plenamente	Discordo parcialmente	Concordo parcialmente	Concordo plenamente
EOU-1	**12** **66,7%**	3 16,7%	3 16,7%	0 0,0%
EOU-2	0 0,0%	1 5,6%	5 27,8%	12 66,7%
EOU-3	0 0,0%	2 11,1%	5 27,8%	11 61,1%

Sobre a facilidade de uso, 66,7% (12) dos participantes discordam com a afirmação que o AM-TaaS é muito complicado de usar (EOU-1). Da mesma forma, 66,7% (12) afirmam que o uso do AM-TaaS é claro e compreensível. No entanto, 5.6% (1) discordam dessa afirmação e 11,1% (2) afirmam que o AM-TaaS não é fácil de usar. Neste estudo não foram coletados os comentários das respostas, porém em outro estudo será realizado uma verificação sobre a usabilidade da proposta.

3) Análise da Intenção de Uso (IU): é o grau em que um usuário tem a intenção de usar AM-TaaS, os resultados são apresentados na Tabela 5.13 e as perguntas realizadas são as seguintes:

[IU -1]: Sou favorável em relação ao uso do AM-TaaS;

[IU -2]: Faz sentido usar AM-TaaS;

[IU -3]: Os usuários (desenvolvedores/testadores) devem adotar o uso de AM-TaaS.

Tabela 5.13. Resultados sobre a Intenção de Uso (IU).

Pergunta	Discordo plenamente	Discordo parcialmente	Concordo parcialmente	Concordo plenamente
IU-1	1 5,6%	0 0,0%	4 22,2%	13 72,2%
IU-2	1 5,6%	0 0,0%	2 11,1%	15 83,3%
IU-3	0 0,0%	1 5,6%	4 22,2%	13 72,2%

Sobre a intenção de uso, 72,2% (13) dos participantes concordam plenamente que são favoráveis ao uso do AM-TaaS e que os desenvolvedores/testadores deveriam adotar o uso de AM-TaaS como suporte a execução dos testes.

5.5 Ameaças à Validade

Nesta seção são discutidas as possíveis ameaças à validade dos resultados deste estudo de viabilidade.

5.5.1 Validade Interna

Relacionada a questões que ameaçam a habilidade de traçar conclusões corretas sobre o relacionamento entre tratamentos e resultados de um estudo experimental.

- **Instrumentação:** os instrumentos que foram utilizados em ambas sessões (formulários online e os formulários de registros de resultados) passaram por revisão e foram submetidos a um estudo piloto;
- **Seleção:** como mencionado na Seção 5.2.3, para seleção dos participantes levou-se em consideração sua caracterização. Os grupos foram distribuídos de maneira igualitária;
- **Maturação**: para não ocorrer desmotivação durante o andamento das sessões, foi informado que seriam disponibilizados brindes (chocolates) para todos os participantes no final de cada sessão, independentemente de seu desempenho;
- **Comportamento competitivo:** como forma de reduzir a competição entre os dois grupos, foi explicado que não haveria avaliação entre o resultado dos grupos;
- **Efeito da expectativa do sujeito:** para reduzir o efeito positivo ou negativo por parte dos participantes, foi esclarecido que não haveria avaliação dos resultados de forma individual nem de forma grupal, a utilização do uso dos guias passo a passo também ajudou a reduzir esse efeito;

- **Efeito da expectativa do experimentador:** para reduzir esta ameaça, foi utilizado um guia com o passo-a passo das ações que deveriam ser executadas e como elas deviam ser registradas, de forma que o experimentador não interviesse.

5.5.2 Validade Externa

Questões relacionada à ameaça dos resultados do estudo não serem generalizáveis a projetos reais da indústria. As principais ameaças identificadas foram:

- **Participantes:** os participantes selecionados refletem o comportamento da população de desenvolvedores, que de forma geral adquirem conhecimento sobre desenvolvimento de aplicações móveis, mas não sobre como testá-las. No entanto, é necessário outro estudo com outros participantes com maior experiência em desenvolvimento e também em testes para tratar essa ameaça;
- **Tempo:** o tempo em que foi executado o experimento foi o mesmo para as duas abordagens, considerando que foram executados os mesmos Casos de Testes para as duas abordagens;
- **Ambiente do experimento:** o ambiente utilizado para a execução do experimento foi um ambiente acadêmico, porém a infraestrutura computacional representa as mesmas que são usadas na indústria por desenvolvedores e testadores.

5.5.3 Validade de *Constructo*

Problemas relacionados à ameaça de generalizar os resultados do estudo à teoria que o sustenta. As principais ameaças são:

- **Projeto do Experimento:** Como descrito nos Capítulos 2 e 3, o arcabouço AM-TaaS passou por uma fase de concepção, baseado num corpo de conhecimento construído por meio de um mapeamento sistemático e sob uma prova de conceito para validar os conceitos propostos. E enquanto ao Ambiente de Teste Local no Laboratório, este foi utilizado pelo fato se ter semelhança ao ambiente que um desenvolvedor utiliza como ambiente de teste na indústria;
- **Fatores humanos (ou sociais):** os participantes não foram informados sobre o objetivo do experimento para eles não serem influenciados, participando assim de forma voluntária como uma forma de adquirir conhecimento. Por outro lado, nenhum dos participantes estava participando de outro experimento nos dias que foram executadas as duas sessões;
- **Apreensão de Avaliação:** uma possível ameaça é a probabilidade de os participantes influenciarem seus resultados devido a expectativas pessoais sobre

a avaliação. Como forma de suprimir esta ameaça, foi informado que não será feito nenhum tipo de avaliação sobre os resultados, como parte da disciplina.

5.5.4 Validade de Conclusão

Relacionada a questões que ameaçam a habilidade de traçar conclusões corretas sobre o relacionamento entre tratamentos e resultados de um estudo experimental.

- **Confiabilidade nas medidas:** Percebendo-se que o uso de um único servidor local não seria suficiente para todos os participantes, as duas abordagens foram implementadas em cada computador, garantindo assim que as medidas não sejam influenciadas pelos recursos computacionais. Na primeira sessão, foram usados artefatos para o registro dos resultados, enquanto que na segunda sessão os resultados foram obtidos do arquivo de *log* de testes;
- **Heterogeneidade Aleatória dos Participantes:** os participantes foram alunos de graduação. Portanto, não se espera participantes com níveis de competência tão discrepantes que possam comprometer a validade do estudo;
- **Confiabilidade na Implementação do Tratamento:** este risco está relacionado à aplicação do tratamento não ser similar entre os diferentes participantes do estudo, como forma de reduzir esta ameaça foram entregues guias com o passo-a-passo das atividades a serem executadas em cada sessão.

5.6 Considerações Finais

Este capítulo apresentou o estudo de viabilidade do ambiente de teste AM-TaaS, correspondente à fase de avaliação da metodologia adotada nesta pesquisa. Seu objetivo foi analisar a viabilidade, por meio de um experimento controlado, com respeito ao Tempo de Execução, Casos de Teste que falharam e a percepção de utilidade do usuário referente à o uso do AM-TaaS. A proposta conseguiu impactar na mudança de atitudes positivas em relação ao seu uso como forma de suporte as atividades de teste.

Mesmo com os resultados que confirmam a viabilidade da proposta, percebeu-se a necessidade de refinar a ferramenta em termos de usabilidade e suporte a outros tipos de teste necessários (desempenho, regressão), assim como definir um padrão de scripts de teste segundo a ferramenta utilizada.

No próximo capítulo serão apresentadas as considerações finais e contribuições deste trabalho, assim como as perspectivas de trabalhos futuros para continuação desta pesquisa.

CAPÍTULO 6 – CONCLUSÕES E TRABALHOS FUTUROS

Neste capítulo são apresentadas as considerações finais e conclusões sobre este trabalho de pesquisa, também são indicados os trabalhos futuros como oportunidades de investigação para dar continuidade a esta pesquisa.

6.1 Considerações Finais

A atividade de teste de aplicações móveis tem se tornado uma tarefa extenuante para os desenvolvedores e testadores, devido à ampla diversidade de modelos de dispositivos móveis, sistemas operacionais, versões de APIs, tamanhos de tela e tamanhos de memória diferentes nos dispositivos móveis. Como resultado dessa variedade, os usuários (desenvolvedores e testadores) precisam de um ambiente apropriado que permita refletir um cenário próximo à realidade, no qual, existem milhares de modelos de dispositivos com todas as diversas características mencionadas antes.

O ambiente adequado de teste, além dos dispositivos, requer de conhecimento de ferramentas que permitam o suporte na execução dos testes. No entanto, essas ferramentas precisam de um tempo para conhecer e aprender o funcionamento e configuração delas e esse tempo muitas vezes não está considerado dentro do tempo destinado ao teste. O lançamento das aplicações é realizado em curtos períodos de tempo, isso impacta diretamente a atividade de teste nas aplicações.

Como forma de oferecer suporte aos desafios descritos acima, esta pesquisa propôs o Serviço de Teste para Aplicações na Nuvem. Assim, o arcabouço de Automação de Teste para Aplicações Móveis como Serviço (*Automated Mobile Testing as a Service AM-TaaS*) é proposto nesta pesquisa.

Para a identificação da estrutura da proposta, foi realizado um mapeamento sobre *Cloud Testing*, e nele foram identificadas características relacionadas ao ambiente móvel e os componentes necessários para sua implementação. Logo, foram analisados os Casos de Teste proposto por AQuA com a finalidade de verificar se era possível sua automatização. Observou-se que muitos casos de teste que são exigidos precisam de ações externas (por exemplo, inserir e retirar o cartão de memória ou bateria) e outras de percepção humana (por exemplo, ouvir se o som é claro). No entanto, foram identificados outros Casos de Teste que sim era possível sua automatização, os quais foram adicionados ao arcabouço proposto.

Em seguida, foi realizado ainda uma prova de conceito de forma a verificar a viabilidade da pesquisa. Com os resultados obtidos foi possível definir a estrutura de AM-TaaS que suporte um ambiente de teste para aplicações móveis.

Finalmente, foi desenvolvida uma ferramenta como meio de acesso ao arcabouço, a qual foi implantada numa nuvem privada. Como forma de avaliar a proposta, foi realizado um estudo de viabilidade que permitiu observar a utilidade e eficiência do arcabouço e a atitude dos usuários frente a uma nova tecnologia.

Os resultados obtidos do estudo de viabilidade no Capítulo 5, permitiu afirmar a hipótese definida para esta pesquisa. A proposta do arcabouço AM-TaaS ajuda na redução do esforço na execução dos testes em aplicações móveis.

O arcabouço AM-TaaS proposto permite ser implantado para diversos tipos de usuários, listados abaixo:

- Usuários finais: é possível ser implementado no ambiente local de cada usuário, utilizando os recursos disponíveis pelos usuários (desenvolvedores ou testadores);
- Ambiente acadêmico: se implementado dentro de um servidor local ou uma nuvem privada, é possível que o AM-TaaS seja usado por alunos de um ambiente acadêmico;
- Ambiente industrial: a implantação também é possível ser realizada dentro de um ambiente industrial. No entanto, nessa primeira versão, ainda precisa ser realizada mais ajustes e outro estudo de viabilidade para ter um amadurecimento da proposta.

6.2 Contribuições

As principais contribuições desta pesquisa são as seguintes:

1) Corpo de Conhecimento sobre *Cloud Testing as a Service*, obtido por meio de um Mapeamento Sistemático da Literatura.
2) Corpo de Conhecimento e análise de componentes de uma infraestrutura para *Cloud Testing* voltado a Aplicações Móveis.
3) Arcabouço de teste para aplicações móveis “AM-TaaS”.
4) Como resultado dessa pesquisa, os seguintes artigos foram aceitos para publicação:
    - Villanes, Costa e Dias-Neto, “*Automated Mobile Testing as a Service (AM-TaaS)*”, 2015 IEEE World Congress on Services, New York.
    - Villanes, Meireles e Dias Neto, “*Cloud-Based Mobile App Testing Framework: Architecture, Implementation and Execution*”, VII Congresso Brasileiro de

Software: Teoria e Prática - CBSoft 2016 - I Simpósio Brasileiro de Teste de Software Sistemático e Automatizado, Maringá.

6.3 Limitações

Algumas limitações foram observadas no decorrer desta pesquisa:

- O componente Gerenciador de emuladores usa o *SDK Manager* de Android, que apresenta menor desempenho se comparado a outros gerenciadores de emuladores como *Genymotion*. Nessa primeira versão do AM-TaaS, foi usado o SDK por ser parte da família Android, que foi a plataforma escolhida para esse trabalho.
- O uso de emuladora ajuda na execução de testes para quem não tem acesso a dispositivos reais, porém tem limites de desempenho.
- Os participantes que avaliaram a proposta são do ambiente académico. Assim, é necessário ainda avaliar a utilização da proposta com usuários mais experientes na área de teste e que representem a indústria.
- A ferramenta utilizada como base do ambiente de teste (*AndroidViewClient*) para os testes ajudaram na execução em diversos modelos de emuladores e versões de APIs, porém ainda continua em evolução, e não tem a maturidade suficiente para ser estável. No entanto, outras ferramentas estão surgindo que poderiam ser inclusas na proposta visando sua melhoria.

6.4 Trabalhos Futuros

- No decorrer desta pesquisa surgiram novas ferramentas para dar suporte a execução dos testes, como o Expresso no Android. É necessário incluir essa nova ferramenta na proposta para avaliar o desempenho do AM-TaaS.
- Os testes inclusos em AM-TaaS são os de funcionalidade e compatibilidade, porém se faz necessária a implementação de novos tipos de teste que permitam a execução de mais testes.
- O uso de emuladores foi a base de AM-TaaS, e pretende-se avaliar o desempenho do arcabouço com o uso de outros gerenciadores de emuladores, como *Genymotion*.
- A execução de testes usando dispositivos reais é possível. No entanto, é necessário realizar ajustes em AM-TaaS para diferenciar emuladores de dispositivos reais e permitir a execução dos testes.
- Incluir outros serviços de teste no ambiente do AM-TaaS (e.g. Teste de segurança, desempenho, carga) que possam cobrir mais tipos de teste, necessários para garantir a qualidade da aplicação móvel.
- Realizar um estudo de observação no ambiente industrial o que permitiria o amadurecimento do AM-TaaS.

REFERÊNCIAS BIBLIOGRÁFICAS

AALST, L. V. D. *"Software Testing as a Service - STaaS"*, In: 26th Annual Pacific Northwest Software Quality Conference, Portland, Oregon, pp. 113–126, 2008.

AL-AHMAD, A. S.; ALJUNID, S. A. *"Mobile Cloud Computing Testing Review"*, In: International Conference on Advanced Computer Science Applications and Technologies, pp. 176–180, 2013.

AWS *Amazon EC2,* Página WEB, disponível em: <http://aws.amazon.com/pt/ec2>, Acesso em: 23/10/2014.

BAI, X.; LI, M.; CHEN, B.; TSAI, W.-T.; GAO, J. "*Cloud testing tools"*, In: Proceedings of 6th International Symposium on Service Oriented System (SOSE), pp. 1–12, 2011.

BAI, X.; LI, M.; HUANG, X.; TSAI, W.-T.; GAO, J. "*Vee@Cloud: The virtual test lab on the cloud"*, 8th International Workshop on Automation of Software Test (AST), pp.15–18, 2013.

BARHAM, P.; DRAGOVIC, B.; FRASER, K.; et al. "*Xen and the Art of Virtualization Categories and Subject Descriptors"*, In: Proceedings of the 19th ACM symposium on Operating sys- tems principles - SOSP, New York, USA, pp. 164 – 177, 2003.

BARIDE, S.; DUTTA, K. A *"Cloud based Software Testing Paradigm for Mobile Applications"*, ACM SIGSOFT Software Engineering Notes, v. 36, n. 3, pp. 1–4, 2011.

BASILI, V. R.; CALDIERA, G.; ROMBACH, H. D. *"The Goal Question Metric Approach"*, v. 2, pp. 1–10, 1994.

BISQUERRA, R.; SARRIERA, J. C.; MARTINEZ, F. *"Introdução à estatística: enfoque informático com o pacote estatístico SPSS"*, Porto Alegre, 2004.

BUYYA, R.; YEO, C. S.; VENUGOPAL, S.; BROBERG, J.; BRANDIC, I. *"Cloud computing and emerging it platforms: Vision, hype, and reality for delivering computing as the 5th utility"*, Future Generation Computer Systems, v. 25, n. 6, pp. 599–616, 2009.

CANDEA, G.; BUCUR, S.; ZAMFIR, C. *"Automated Software Testing as a Service"*, In: SoCC '10 Proceedings of the 1st ACM symposium on Cloud computing, pp.155–160, 2010.

CARSTEN, B. Carsten's Corner. *"Revista Power Conversion and Intelligent Motion"*, pp. 38, Novembro, 1989.

CHELLAPPA, R. *"Intermediaries in cloud-computing: A new computing paradigm"*, Dallas, USA, 1997.

DAVIS, F. D.; BAGOZZI, R. P.; WARSHAW, P. R. *"User acceptance of computer technology: a comparison of two theoretical models"*, In: Management Science, 1989.

FEWSTER, M.; GRAHAM, D. *"Software Test Automation: Effective Use of Test Execution Tools"*, Addison Wesley, 1999.

GAO, J.; BAI, X.; TSAI, W. *"Cloud Testing- Issues, Challenges, Needs and Practice"*, In: Software Engineering: An International Journal (SEIJ), v. 1, no. 1, September 2011.

GAO, J.; BAI, X.; TSAI, W.-T. *"Testing as a Service (TaaS) on Clouds"*, In: Seventh International Symposium on Service-Oriented System Engineering, pp. 212–223, 2013.

GAO, J.; BAI, X.; TSAI, W.-T.; UEHARA, T. *"Mobile Application Testing: A Tutorial"*, Computer, v. 47, n. 2, pp. 46–55, 2014.

GAO, J.; MANJULA, K.; ROOPA, P.; et al. *"A cloud-based TaaS infrastructure with tools for SaaS validation, performance and scalability evaluation"*, In. 4th International Conference on Cloud Computing Technology and Science Proceedings, pp.464–471, 2012.

GAO, J.; TSAI, W.-T.; PAUL, R.; BAI, X.; UEHARA, T. *"Mobile Testing-as-a-Service (MTaaS) -- Infrastructures, Issues, Solutions and Needs",* In: 15th International Symposium on High-Assurance Systems Engineering, pp.158–167, 2014.

GARTNER. *"Worldwide Smartphone Sales to End Users by Vendor",* Página WEB, disponível em: <http://www.gartner.com/newsroom/id/2665715>. Acesso em: 3/2/2015, 2013.

GARTNER. *"Forecast Analysis: Public Cloud Services, Worldwide, 1Q14 Update",* Página WEB, disponível em: <https://www.gartner.com/doc/2738817>. Acesso em: 10/10/2014, 2014.

GARTNER. *"Gartner Says Worldwide Smartphone Sales Grew 3.9 Percent in First Quarter of 2016",* Página WEB, disponível em: <http://www.gartner.com/newsroom/id/3323017>. Acesso em: 1/8/2016, 2016.

GOOGLE. *"Google App Engine",* Página WEB, disponível em: <https://cloud.google.com/appengine/docs/whatisgoogleappengine>. Acesso em: 23/1/2015, 2015.

HUANG, J. *"AppACTS: Mobile App Automated Compatibility Testing Service",* In: 2nd International Conference on Mobile Cloud Computing, Services and Engineering, pp.85–90, 2014.

HURWITZ, J.; KAUFMAN, M.; HALPER, F. *"Cloud Services for Dummies",* IBM Limite ed. Hoboken, New Jersey: John Wiley & Sons, Inc., 2012.

KHANNA, R. *"Making the Most of Test Automation as a Service",* In: International Conference on Cloud Computing in Emerging Markets (CCEM). pp.1–4, 2012.

KIM, T.; CHOI, Y.; HAN, S.; et al. *"Monitoring and detecting abnormal behavior in mobile cloud infrastructure",* In: Network Operations and Management Symposium, pp.1303–1310, 2012.

KING, T. M.; GANTI, A. S.; FROSLIE, D. *"Enabling Automated Integration Testing of Cloud Application Services in Virtualized Environments",* In: Conference of the Center for Advanced Studies on Collaborative Research, pp. 120–132, 2011.

KIRUBAKARAN, B.; KARTHIKEYANI, V. *"Mobile application testing — Challenges and solution approach through automation",* In: International Conference on Pattern Recognition, Informatics and Mobile Engineering, pp.79–84, 2013.

KITCHENHAM, B.; CHARTERS, S. *"Guidelines for performing Systematic Literature Reviews in Software Engineering",* 2007.

KOSMATOV, N.; WILLIAMS, N.; BOTELLA, B.; ROGER, M. *"Structural Unit Testing as a Service with PathCrawler-online.com",* In: Seventh International Symposium on Service-Oriented System Engineering, pp. 435–440, 2013.

MCCARTHY, J. *MIT Centennial*, 1961.

MELL, P.; GRANCE, T. *"The NIST -National Institute of Standars and Technology- Definition of Cloud Computing",* 2011.

MICROSOFT. *"Hyper-V Cloud",* Página WEB, disponível em: <http://technet.microsoft.com/en-us/cloud/gg680488>. Acesso em: 20/10/2014.

MURUGESAN, L.; BALASUBRAMANIAN, P. *"Cloud Based Mobile Application Testing",* In: 13th International Conference on Computer and Information Science (ICIS), Taiyuan, pp. 287 – 289, 2014.

DO NASCIMENTO, J.; DOS SANTOS, J.; DIAS-NETO, A. C. *"MTControol: Ferramenta de Apoio à Gestão de Testes de Aplicativos Móveis Baseada nas Diretrizes do AQuA",* In: X Workshop Anual do MPS (WAMPS 2014), pp.234–241, 2014.

OPENSIGNAL. *OpenSignal*, Página WEB, disponível em: <http://opensignal.com/reports/2015/08/android-fragmentation/>. Acesso em: 1/1/2016.

PARVEEN, T.; TILLEY, S. *"When to migrate software testing to the cloud?",* In: 3rd International Conference on Software Testing, Verification, and Validation Workshops - ICSTW, pp. 424–427, 2010.

PRIYANKA; CHANA, I.; RANA, A. *"Empirical evaluation of cloud-based testing techniques: a systematic review",* In: ACM SIGSOFT Software Engineering Notes, v. 37, pp.1–9, 2012.

RHOTON, J.; HAUKIOJA, R. *"Cloud Computing Architected: Solution Design Handbook",* United States, 2011.

RIUNGU, L. M.; TAIPALE, O.; SMOLANDER, K. *"Software Testing as an Online Service : Observations from Practice"*, In: Third International Conference on Software Testing, Verification, and Validation Wokshops. pp. 418 - 423, 2010a.

RIUNGU, L. M.; TAIPALE, O.; SMOLANDER, K. *"Research Issues for Software Testing in the Cloud",* In: Second International Conference on Cloud Computing Technology and Science, pp. 557–564, 2010b.

ROODENRIJS, E. *"Testing Clouds",* Groningen: LINE UP boek en media, 2011. Sogeti Netherlands.

SCHMIDT, E. *"Conversation with Eric Schmidt hosted by Danny Sullivan"*, In: Search Engine Strategies Conference, Agosto de 2006, disponível em: <http://www.google.com/press/podium/ses2006.html>. Acesso em: 23/10/2014.

SHEEHAN, M. *"Cloud computing expo: Introducing the cloud pyramid",* Página WEB, disponível em: <http://cloudcomputing.sys-con.com/node/609938>, 2008.

SPÍNOLA, R. O.; DIAS-NETO, A. C.; TRAVASSOS, G. H. *"Abordagem para Desenvolver Tecnologia de Software com Apoio de Estudos Secundários e Primários",* In: Experimental Software Engineering Latin American Workshop, Salvador, Brasil, 2008.

STAROV, O.; VILKOMIR, S. *"Integrated TaaS platform for mobile development: Architecture solutions",* In: 8th International Workshop on Automation of Software Test (AST), pp. 1–7, 2013.

STATISTA. *"Forecast for the number of mobile app downloads",* Página WEB, disponível em: <http://www.statista.com/statistics/266488/forecast-of-mobile-app-downloads/>. Acesso em: 1/1/2016.

TUNG, Y.-H.; LIN, C.-C.; SHAN, H.-L. *"Test as a Service: A Framework for Web Security TaaS Service in Cloud Environment",* In: 8th International Symposium on Service Oriented System Engineering. pp. 212–217, 2014.

UTEST. *uTest*, Página WEB, disponível em: <http://www.utest.com/about-us>. Acesso em: 31/1/2015.

WARNER, S. A.; KARMAN, A. F. *"Defining the Mobile Cloud",* In: NASA IT Summit 2010, 16-18 Agosto, 2010.

WASSERMAN, A. I.; FOSSER. *"Software Engineering Issues for Mobile Application Development"*, In: FoSER '10 Proceedings of the FSE/SDP workshop on Future of software engineering research, p 397-400, 2010.

WU, J.; WANG, C.; LIU, Y.; ZHANG, L. *"Agaric - A hybrid cloud based testing platform",* In: International Conference on Cloud and Service Computing, pp.87–94, 2011.

YAN, M.; SUN, H.; WANG, X.; LIU, X. WS-TaaS: *"A Testing as a Service Platform for Web Service Load Testing",* In: 18th International Conference on Parallel and Distributed Systems, pp.456–463, 2012.

YU, L.; LI, X.; LI, Z. *"Testing tasks management in testing Cloud environment",* In: Proceedings - International Computer Software and Applications Conference. pp. 76–85, 2011.

YU, L.; TSAI, W.-T.; CHEN, X.; et al. *"Testing as a Service over Cloud",* In: Fifth International Symposium on Service Oriented System Engineering, pp. 181–188, 2010.

Apêndice A: Artigos Selecionados no Mapeamento Selecionado

[1] S. Baride and K. Dutta, “A Cloud based Software Testing Paradigm for Mobile Applications,” ACM SIGSOFT Software Engineering Notes, vol. 36, no. 3, pp. 1–4, 2011.

[2] J. Gao, K. Manjula, P. Roopa, E. Sumalatha, X. Bai, W. T. Tsai, and T. Uehara, “A cloud-based TaaS infrastructure with tools for SaaS validation, performance and scalability evaluation,” in 4th IEEE International Conference on Cloud Computing Technology and Science Proceedings, pp. 464–471, 2012.

[3] G. S. De Oliveira and A. Duarte, “A Framework for Automated Software Testing on the Cloud,” in 2013 International Conference on Parallel and Distributed Computing, Applications and Technologies, 2013, pp. 344–349.

[4] J. Wu, C. Wang, Y. Liu, and L. Zhang, “Agaric — A hybrid cloud based testing platform,” in 2011 International Conference on Cloud and Service Computing, 2011, pp. 87–94.

[5] J. Huang, “AppACTS: Mobile App Automated Compatibility Testing Service,” in 2014 2nd IEEE International Conference on Mobile Cloud Computing, Services, and Engineering, 2014, pp. 85–90.

[6] G. Candea, S. Bucur, and C. Zamfir, “Automated Software Testing as a Service,” in SoCC ’10 Proceedings of the 1st ACM symposium on Cloud computing, 2010, pp. 155–160.

[7] L. Murugesan and P. Balasubramanian, “Cloud Based Mobile Application Testing,” in 2014 IEEE/ACIS 13th International Conference on Computer and Information Science (ICIS), 2014, pp. 287 – 289.

[8] X. Bai, M. Li, B. Chen, W.-T. Tsai, and J. Gao, “Cloud testing tools,” Proceedings of 2011 IEEE 6th International Symposium on Service Oriented System (SOSE), pp. 1–12, Dec. 2011.

[9] L. Ciortea, C. Zamfir, S. Bucur, V. Chipounov, and G. Candea, “Cloud9 : A Software Testing Service.”, in ACM SIGOPS Operating Systems Review, v.43, pp. 5-10, 2010

[10] Priyanka, I. Chana, and A. Rana, “Empirical evaluation of cloud-based testing techniques: a systematic review,” in ACM SIGSOFT Software Engineering Notes, 2012, vol. 37, no. 3, pp. 1–9.

[11] T. M. King, A. S. Ganti, and D. Froslie, “Enabling Automated Integration Testing of Cloud Application Services in Virtualized Environments,” Proceedings of the 2011 Conference of the Center for Advanced Studies on Collaborative Research, pp. 120–132, 2011.

[12] O. Starov and S. Vilkomir, “Integrated TaaS platform for mobile development: Architecture solutions,” in 2013 8th International Workshop on Automation of Software Test (AST), 2013, pp. 1–7.

[13] R. Khanna, “Making the Most of Test Automation as a Service,” in 2012 IEEE International Conference on Cloud Computing in Emerging Markets (CCEM), 2012, pp. 1–4.

[14] S. Bucur, J. Kinder, and G. Candea, “Making automated testing of cloud applications an integral component of PaaS,” in Proceedings of the 4th Asia-Pacific Workshop on Systems - APSys '13, pp. 1–7, 2013.

[15] J. Gao, W.-T. Tsai, R. Paul, X. Bai, and T. Uehara, “Mobile Testing-as-a-Service (MTaaS) -- Infrastructures, Issues, Solutions and Needs,” in 2014 IEEE 15th International Symposium on High-Assurance Systems Engineering, 2014, pp. 158–167.

[16] T. Kim, Y. Choi, S. Han, J. Y. Chung, J. Hyun, J. Li, and J. W.-K. Hong, “Monitoring and detecting abnormal behavior in mobile cloud infrastructure,” in 2012 IEEE Network Operations and Management Symposium, 2012, pp. 1303–1310.

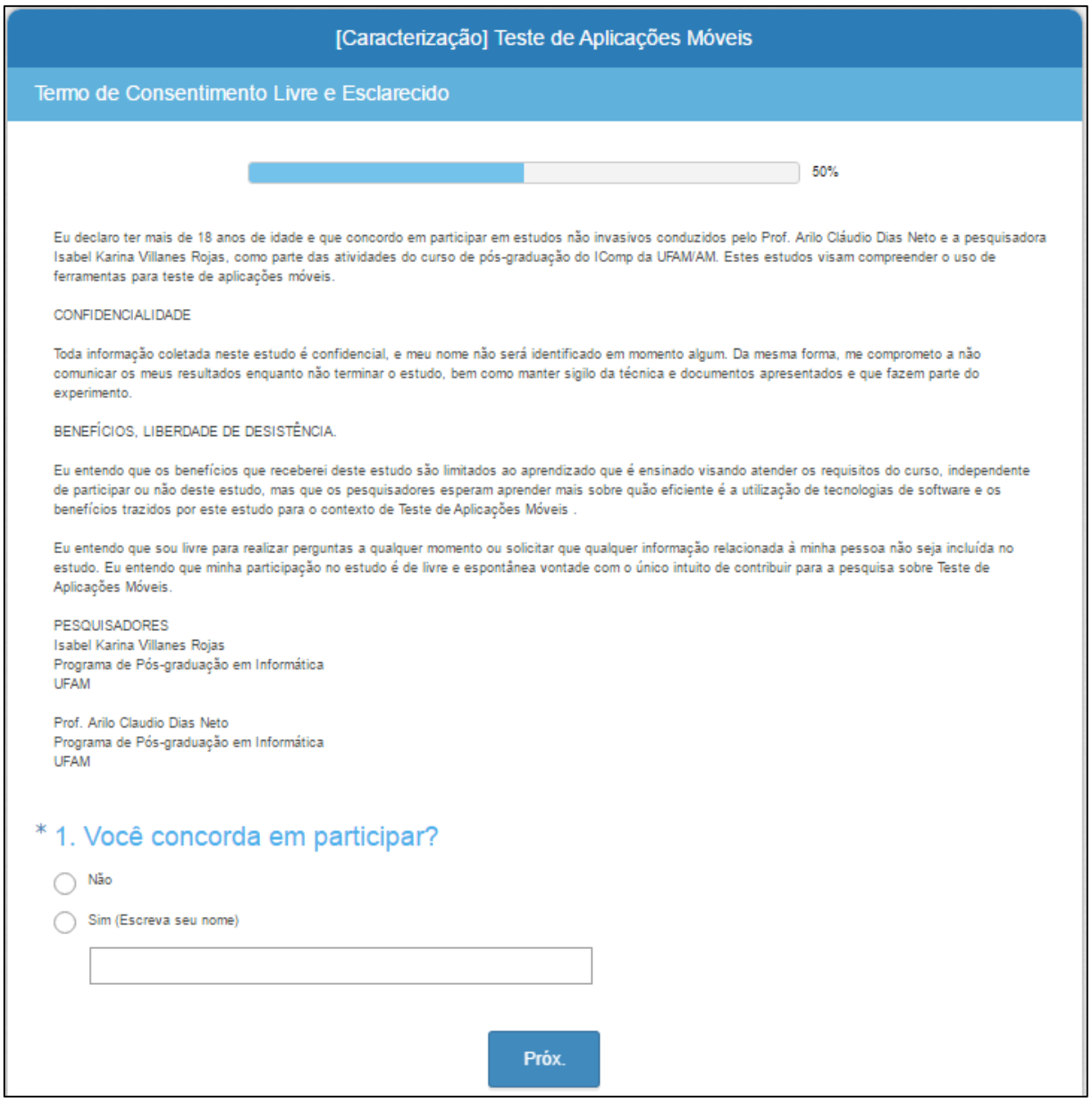
[17] S. Huang, Z. J. Li, Y. Liu, and J. Zhu, “Regression Testing as a Service,” in 2011 Annual SRII Global Conference, 2011.

[18] J. Gao, W. T. Tsai, and T. Uehara, “SaaS Testing on Clouds - Issues, Challenges and Needs,” in 2013 IEEE Seventh International Symposium on Service-Oriented System Engineering, pp. 409–415, Mar. 2013.

[19] E. Lamas, L. A. V. Dias, and A. M. Da Cunha, “Software architectural drivers for cloud testing,” in VALID 2012 - 4th International Conference on Advances in System Testing and Validation Lifecycle, 2012, pp. 114–120.

[20] B. Floss and S. Tilley, “Software Testing as a Service: An Academic Research Perspective,” 2013 IEEE Seventh International Symposium on Service-Oriented System Engineering, pp. 421–424, Mar. 2013.
[21] S. Chen, J. Huang, and Y. Gong, “Static Testing as a Service on Cloud,” in 2013 27th International Conference on Advanced Information Networking and Applications Workshops, 2013, pp. 638–642.
[22] N. Kosmatov, N. Williams, B. Botella, and M. Roger, “Structural Unit Testing as a Service with PathCrawler-online.com,” in 2013 IEEE Seventh International Symposium on Service-Oriented System Engineering, 2013, pp. 435–440.
[23] Y.-H. Tung, C.-C. Lin, and H.-L. Shan, “Test as a Service: A Framework for Web Security TaaS Service in Cloud Environment,” in 2014 IEEE 8th International Symposium on Service Oriented System Engineering, 2014, pp. 212–217.
[24] J. Gao, X. Bai, and W.-T. Tsai, “Testing as a Service (TaaS) on Clouds,” in 2013 IEEE Seventh International Symposium on Service-Oriented System Engineering, pp. 212–223, Mar. 2013.
[25] L. Yu, W.-T. Tsai, X. Chen, L. Liu, Y. Zhao, L. Tang, and W. Zhao, “Testing as a Service over Cloud,” in 2010 Fifth IEEE International Symposium on Service Oriented System Engineering, pp. 181–188, 2010.
[26] L. Riungu-Kalliosaari, O. Taipale, and K. Smolander, “Testing in the cloud: Exploring the practice,” IEEE Software, vol. 29, no. 2, pp. 46–51, 2012.
[27] L. Yu, X. Li, and Z. Li, “Testing tasks management in testing Cloud environment,” in Proceedings - International Computer Software and Applications Conference, 2011, pp. 76–85.
[28] P. Zhenlong, O. Y. Zhonghui, and H. Youlan, “The Application and Development of Software Testing in Cloud Computing Environment,” in 2012 International Conference on Computer Science and Service System, pp. 450–454, Aug. 2012.
[29] X. Bai, M. Li, X. Huang, W.-T. Tsai, and J. Gao, “Vee@Cloud: The virtual test lab on the cloud,” in 2013 8th International Workshop on Automation of Software Test (AST), 2013, pp. 15–18.
[30] M. Yan, H. Sun, X. Wang, and X. Liu, “WS-TaaS: A Testing as a Service Platform for Web Service Load Testing,” in 2012 IEEE 18th International Conference on Parallel and Distributed Systems, 2012, pp. 456–463.

Apêndice B: Formulário de Consentimento



[Caracterização] Teste de Aplicações Móveis

Termo de Consentimento Livre e Esclarecido

50%

Eu declaro ter mais de 18 anos de idade e que concordo em participar em estudos não invasivos conduzidos pelo Prof. Arilo Cláudio Dias Neto e a pesquisadora Isabel Karina Villanes Rojas, como parte das atividades do curso de pós-graduação do IComp da UFAM/AM. Estes estudos visam compreender o uso de ferramentas para teste de aplicações móveis.

CONFIDENCIALIDADE

Toda informação coletada neste estudo é confidencial, e meu nome não será identificado em momento algum. Da mesma forma, me comprometo a não comunicar os meus resultados enquanto não terminar o estudo, bem como manter sigilo da técnica e documentos apresentados e que fazem parte do experimento.

BENEFÍCIOS, LIBERDADE DE DESISTÊNCIA.

Eu entendo que os benefícios que receberei deste estudo são limitados ao aprendizado que é ensinado visando atender os requisitos do curso, independente de participar ou não deste estudo, mas que os pesquisadores esperam aprender mais sobre quão eficiente é a utilização de tecnologias de software e os benefícios trazidos por este estudo para o contexto de Teste de Aplicações Móveis .

Eu entendo que sou livre para realizar perguntas a qualquer momento ou solicitar que qualquer informação relacionada à minha pessoa não seja incluída no estudo. Eu entendo que minha participação no estudo é de livre e espontânea vontade com o único intuito de contribuir para a pesquisa sobre Teste de Aplicações Móveis.

PESQUISADORES
Isabel Karina Villanes Rojas
Programa de Pós-graduação em Informática
UFAM

Prof. Arilo Claudio Dias Neto
Programa de Pós-graduação em Informática
UFAM

* 1. Você concorda em participar?

Não

Sim (Escreva seu nome)

Próx.

Formulário de Caracterização

* 2. Qual é a sua idade?

- ◯ 17 ou menos
- ◯ 18 a 20
- ◯ 21 a 29
- ◯ 30 a 39
- ◯ 40 ou mais

* 3. Qual é sua experiência em desenvolvimento de aplicações móveis?

- ◯ Sem experiência em desenvolvimento de aplicações móveis.
- ◯ Tenho desenvolvido aplicações móveis de forma individual ou na Universidade
- ◯ Tenho desenvolvido aplicações móveis como parte de uma equipe ou como parte de uma empresa.

* 4. Qual é sua experiência em teste de aplicações móveis?

- ◯ Sem experiência em teste de aplicações móveis.
- ◯ Tenho testado aplicações móveis de forma individual ou na Universidade
- ◯ Tenho testado aplicações móveis como parte de uma equipe ou como parte de uma empresa.

5. Quanto tempo possui de experiência em teste de aplicações móveis?

- ◯ Nenhuma.
- ◯ Menos de 1 ano.
- ◯ De 1 a 2 anos.
- ◯ De 3 a 5 anos.
- ◯ Mais de 5 anos.

6. Em quais plataformas, você já realizou teste de aplicações móveis?

- ☐ Android (Google)
- ☐ IOs (Apple)
- ☐ Windows e/ou Windows Phone (Microsoft)
- ☐ Outro (especifique)

7. Em quais dos seguintes ambientes, você já executou os testes?

- ☐ Emuladores
- ☐ Serviço de emuladores na nuvem
- ☐ Devices (físicos)
- ☐ Serviço de devices na nuvem
- ☐ Outro (especifique)

8. Quais destas ferramentas de teste, você já usou?

- ☐ Appium
- ☐ Cucumber
- ☐ UIAutomator
- ☐ AndroidViewClient - Culebra
- ☐ MonkeyRunner
- ☐ Calaba.sh
- ☐ Robotium
- ☐ Outro (especifique)

Apêndice C: Código do Caso de Teste OTA Install

```
from com.android.monkeyrunner import MonkeyRunner, MonkeyDevice
import os, threading, glob, subprocess
from datetime import datetime
import sys
import re
import defPath #File with path of directoeries
startTime1 = datetime.now()

##
# Function: main
#   Open and read files with name of emulators from
#   server and from users selected
# @emulatorid = receive emulator's name
# @deviceid = receive internal name of emulated devices online
# @count = receive number(index) of emulators
def main():
   print "_____   [Starting Devices]   _____"
   deviceid = sys.argv[1]
   print "[Emulator Internal Name] %s" % deviceid
   emulatorid = sys.argv[2]
   print "         [Emulator Name] %s" % emulatorid
   count = sys.argv[3]
   device = MonkeyRunner.waitForConnection('',deviceid)
   droidTesting = DefaultDroidTest(device, count, emulatorid, deviceid)
   droidTesting.run()
#
# Funcion: get_package_activity_name(apk_address)
def get_package_activity_name(apk_address):
    command = defPath.sdk_plaTool +  "aapt dump badging %s" %apk_address
    aapt_result          =          subprocess.Popen(command,          stdout=subprocess.PIPE,
shell=True).communicate()[0]
    lines = aapt_result.split("\n")
    myDic = {}
    for line in lines:
        splitedline=line.split(":")
        if len(splitedline)==2:
            myKey,myValue=line.split(":")
            myDic[myKey]=myValue
    package = myDic['package'].split("'")[1]
    activity = myDic['launchable-activity'].split("'")[1]
    return package, activity

def saveresult(fr, device, msg):
    f_result = open(fr, 'wt')
    if msg is None:
        msg = 'None'
    f_result.write(msg.encode('utf-8'))
```

```
    f_result.close()
def log(fn, device):
    msg = device.shell('logcat -d')
    f_log = open(fn, 'at')
    if msg is None:
        msg = 'None'
    f_log.write(msg.encode('utf-8'))
    f_log.close()
    device.shell('logcat -c')

def eraseLog(device):
    device.shell('logcat -c')

def verify_package_in_avd(device,packagename):
    command = device.shell("pm list packages -3")
    splitedline=re.split(':|\r|\n',command)
    if any(packagename in s for s in splitedline):
       msg = "failed,"
    else:
       msg = "passed,"
    return msg

def verify_package_in_normal_avd(device,packagename):
    command = device.shell("pm list packages -3")
    splitedline=re.split(':|\r|\n',command)
    if any(packagename in s for s in splitedline):
       msg = "passed,"
    else:
       msg = "failed,"
    return msg

def check_free_space(device):
    r = device.shell("df|grep data")
    pos = r.rfind("M")
    s = r[pos-6:pos].strip()
    mf = int(float(s))
    return mf

def shutdown_avd(deviceid):
    #command = "/home/experts/android-sdk-linux/platform-tools/adb -s " + deviceid + " emu
kill"
    command = defPath.sdk_plaTool + "adb -s " + deviceid + " emu kill"
    subprocess.call(command, shell=True)

def uninstall_apk(device,packagename):
    print "Starting App Uninstaller... "
    device.shell ("pm uninstall " +  packagename)

def select_files(fmemsize):
    lsf = {}
    fms= int(fmemsize)
```

```
    if fms>0:
      for p in 100, 50, 20, 10, 5, 2, 1:
         if fms >= p:
            n = fms/p
            r = fms-p*n
            lsf[p]=n
            fms = r
    else:
      return (lsf)
    return (lsf)

def copy_files(lsf,device,deviceid):
    src_path = defPath.src_filesmb
    dst_path = " /data/"
    for v, q in lsf.items():
      src = 'filetxt' + str(v) + 'mb.txt'
      x=1
      while x <= q:
       dstfile = src_path + str(x) + src
       command = defPath.sdk_plaTool + "adb -s "+ deviceid + " push " + dstfile + dst_path
       subprocess.call(command, shell=True)
       x+=1
      src = ""

def list_files_del(device):
    command = device.shell("ls /data/ | grep filetxt")
    splitedline=re.split('\r|\n',command)
    for filetxt in splitedline:
       res=device.shell("rm -f /data/"+filetxt)

class DefaultDroidTest:
    deviceid = sys.argv[1]
    emulatorid = sys.argv[2]
    usr_id = sys.argv[4]

    #apkfolder = rootfolder + 'app_files/'
    apkfolder = defPath.src_rootFolder + usr_id + defPath.src_app_files
    imagefolder = defPath.src_rootFolder + usr_id + defPath.src_img
    #logfolder = rootfolder + 'log/'
    logfolder = defPath.src_rootFolder + usr_id + defPath.src_log
    #resfolder = rootfolder + 'log/' + emulatorid + '/'
    resfolder = defPath.src_rootFolder + usr_id + defPath.src_log + emulatorid + '/'
    startTime1 = datetime.now()
    if not os.path.exists(resfolder):
        os.makedirs(resfolder)

    def __init__(self, device, count, emulatorid, deviceid):
        self.device = device
        self.count = count
        self.emulatorid = emulatorid
        self.deviceid = deviceid
```

```
    def run(self):
        eraseLog(self.device)
        fs_1 = check_free_space(self.device)

        for apk in glob.glob(self.apkfolder + '/*.apk'):
            print "Getting package and activity name..."
            packagename,activity = get_package_activity_name(apk)
            componentname = packagename + "/." + activity
            apk_path = self.device.shell('pm path ' + packagename)
            if not apk_path:
               print "Installing " + str(packagename) + " ..."
               result_install = self.device.installPackage(apk)
            else:
               if apk_path.startswith('package:'):
                  print "Apk has already installed, re-installing..."
                  result_install = self.device.installPackage(apk)

        print "Verifying if Apk was installed"
        if (result_install):
           msg = "passed,Installation,"
        else:
           msg = "failed,Installation,"
        fs_2 = check_free_space(self.device)

        time1 = datetime.now() - startTime1
        msg+=str(time1)

        print "Saving results..."
        saveresult(self.resfolder + self.emulatorid + 'ota1' + '.txt',self.device, msg)
        msg=""
        startTime2 = datetime.now()
        MonkeyRunner.sleep(10)

#Uninstall APK
        print "Uninstalling...."
        print packagename
        print activity
        print componentname
        uninstall_apk(self.device,packagename)

#vrik: verifica se package esta no dispositivo
        print "Verifying if Apk was uninstalled"
        msg = verify_package_in_avd(self.device,packagename)
        msg+= "Uninstall,"
        time2 = datetime.now() - startTime2
        msg+=str(time2)

        print "Saving results..."
        saveresult(self.resfolder + self.emulatorid + 'ota2' + '.txt',self.device, msg)
```

```
        startTime3 = datetime.now()
        fs_3 = (fs_1 - fs_2)
        fs_4 = check_free_space(self.device)
        free_mem = fs_4 - (fs_3 - 1)

#Copy files based on free memory size
        print "Filing memory..."
        lf = select_files(free_mem)
        copy_files(lf,self.device,self.deviceid)

#TC-memory test - install
        print "Installing Apk without enought memory..."
        for apk in glob.glob(self.apkfolder + '/*.apk'):
            self.device.installPackage(apk)
#Verify if Package is on device
        print "Verifying if Apk was installed"
        msg = verify_package_in_avd(self.device,packagename)
        msg+= "No space left on device,"
        time3 = datetime.now() - startTime3
        msg+=str(time3)
        print "Saving results..."
        saveresult(self.resfolder + self.emulatorid + 'ota3' + '.txt',self.device, msg)
        print "Freeing up memory space"
        list_files_del(self.device)
# Install App for User Test
        print "Installing Apk for User Test..."
        for apk in glob.glob(self.apkfolder + '/*.apk'):
            self.device.installPackage(apk)
        res = verify_package_in_normal_avd(self.device,packagename)
        print res

        log(self.resfolder + self.emulatorid + 'logtest' +'.log', self.device);
        print "__________ FINISHED ___________"
if __name__ == "__main__":
    main()
```

Apêndice D: Questionário Pós-treinamento - Sessão 1

[Sessão 1] Formulário Pós-Treinamento

Responda conforme sua opinião sobre os itens e atividades realizadas durante o treinamento.

* 1. Qual é seu nome?

* 4. Indique o nível de dificuldade das seguintes atividades executadas no treinamento.

	Muito difícil	Relativamente difícil	Relativamente fácil	Muito fácil
Criação de Emuladores	◯	◯	◯	◯
Criação de Scripts de Teste	◯	◯	◯	◯
Execução dos Scripts de Teste em cada dispositivo (vários emuladores)	◯	◯	◯	◯

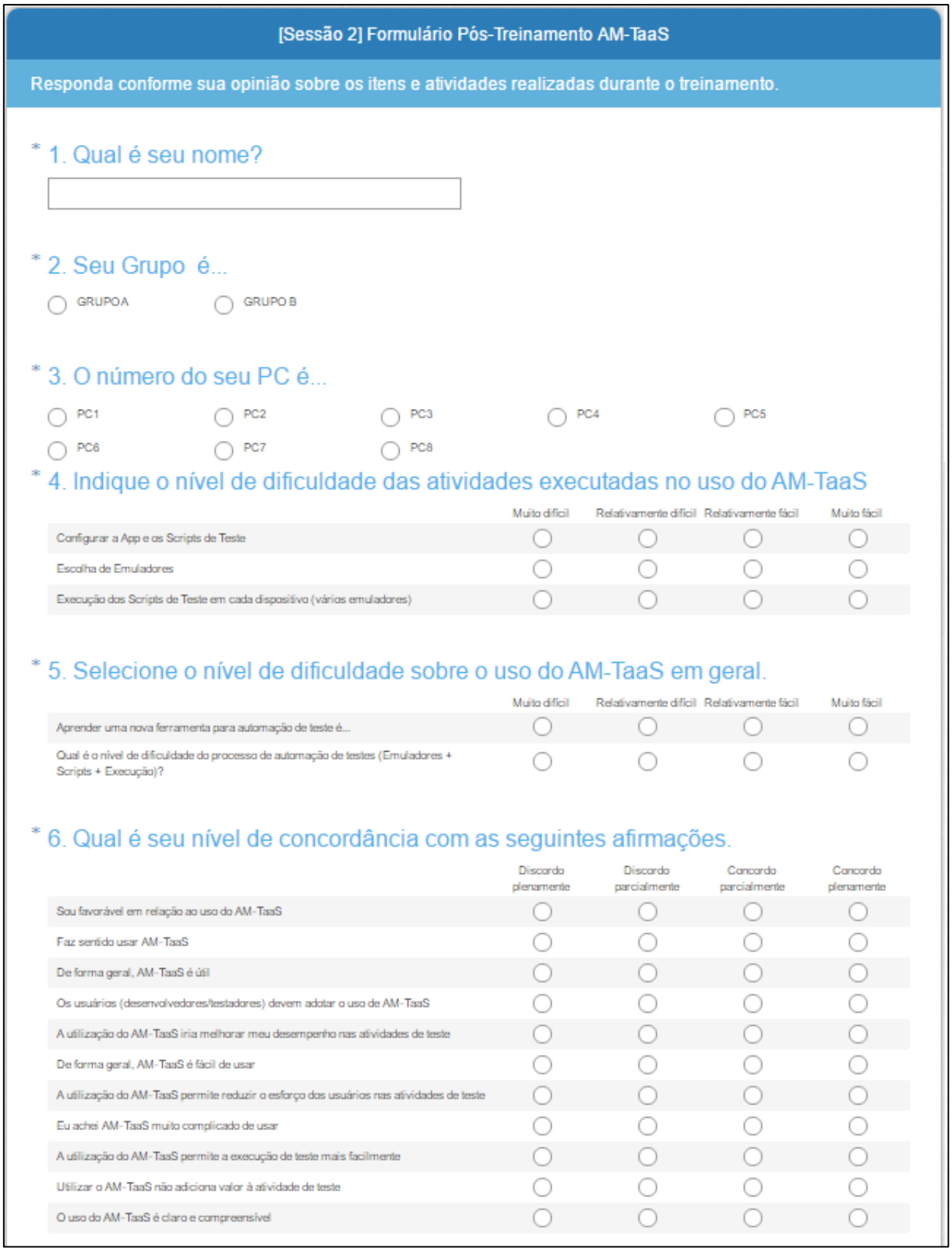
* 5. Selecione o nível de dificuldade das seguintes afirmações sobre o treinamento em geral.

	Muito difícil	Relativamente difícil	Relativamente fácil	Muito fácil
Aprender uma nova ferramenta para automação de teste é...	◯	◯	◯	◯
Qual é o nível de dificuldade do processo de automação de testes (Emuladores + Scripts + Execução)?	◯	◯	◯	◯

* 6. Qual é seu nível de concordância com as seguintes afirmações.

	Discordo plenamente	Discordo parcialmente	Concordo parcialmente	Concordo plenamente
O uso de ferramentas para automação de teste é importante	◯	◯	◯	◯
A automação de testes ajuda na melhoria da execução de testes	◯	◯	◯	◯
Me senti engajado com as atividades de automação de Teste de Apps Móveis.	◯	◯	◯	◯

Apêndice E: Questionário Pós-treinamento - Sessão 2



[Sessão 2] Formulário Pós-Treinamento AM-TaaS

Responda conforme sua opinião sobre os itens e atividades realizadas durante o treinamento.

* 1. Qual é seu nome?

* 2. Seu Grupo é...

GRUPO A GRUPO B

* 3. O número do seu PC é...

PC1 PC2 PC3 PC4 PC5

PC6 PC7 PC8

* 4. Indique o nível de dificuldade das atividades executadas no uso do AM-TaaS

	Muito difícil	Relativamente difícil	Relativamente fácil	Muito fácil
Configurar a App e os Scripts de Teste				
Escolha de Emuladores				
Execução dos Scripts de Teste em cada dispositivo (vários emuladores)				

* 5. Selecione o nível de dificuldade sobre o uso do AM-TaaS em geral.

	Muito difícil	Relativamente difícil	Relativamente fácil	Muito fácil
Aprender uma nova ferramenta para automação de teste é...				
Qual é o nível de dificuldade do processo de automação de testes (Emuladores + Scripts + Execução)?				

* 6. Qual é seu nível de concordância com as seguintes afirmações.

	Discordo plenamente	Discordo parcialmente	Concordo parcialmente	Concordo plenamente
Sou favorável em relação ao uso do AM-TaaS				
Faz sentido usar AM-TaaS				
De forma geral, AM-TaaS é útil				
Os usuários (desenvolvedores/testadores) devem adotar o uso de AM-TaaS				
A utilização do AM-TaaS iria melhorar meu desempenho nas atividades de teste				
De forma geral, AM-TaaS é fácil de usar				
A utilização do AM-TaaS permite reduzir o esforço dos usuários nas atividades de teste				
Eu achei AM-TaaS muito complicado de usar				
A utilização do AM-TaaS permite a execução de teste mais facilmente				
Utilizar o AM-TaaS não adiciona valor à atividade de teste				
O uso do AM-TaaS é claro e compreensível				

Apêndice F: Artefatos utilizados e guia das atividades - Sessão 1

Ferramentas de Teste para Aplicações Móveis

GRUPO X

ATIVIDADE 1 – CRIAÇÃO DE EMULADORES

I. Descrição da Atividade:

Criar cinco emuladores usando os modelos de dispositivo descritos na tabela abaixo. Registre a hora do início e a hora de fim da atividade. Use a hora do seu computador.

Início: quando inicia o preenchimento as características do emulador.

Fim: quando finaliza o preenchimento das características do emulador.

Exemplo: Inicio: 18:20 Fim: 18:25

Id	Modelo de Dispositivo
1	**GalaxyNexus**
2	**Nexus5**
3	**Nexus6**

II. Passos da Atividade:

1. Abra o AVD manager
2. Clique na aba **"Device Definitions"**
3. Selecione o modelo de dispositivo (device)
4. Clique no botão **"Create AVD"**
5. Registre a hora de inicio
6. Preencha as configurações
7. Clique no botão **"OK"**
8. Registre a hora de término

ID	Nome	Inicio	Fim	Observação
1	AVD17_GalaxyNexus			
2	AVD18_Nexus5			
3	AVD19_Nexus6			

III. Características dos Emuladores:

1) Modelo do dispositivo (*device*): Galaxy Nexus

Características	Galaxy Nexus
AVDs Name	**AVD17_GalaxyNexus**
Device	Galaxy Nexus
Target (API)	Google API Level 17
CPU/ABI	ARM (armeabi-v7a)
Keyboard	✓
Skin	Galaxy Nexus
Ram	1024
VM Heap	64
Internal Storage	200
SD Card Size	100
Use Host GPU	✓

2) Modelo do dispositivo (*device*): Nexus 5

Características	Nexus 5
AVDs Name	**AVD18_Nexus5**
Device	Nexus 5
Target (API)	Google API Level 18
CPU/ABI	ARM (armeabi-v7a)
Keyboard	Sim
Skin	Nexus 5
Ram	1024
VM Heap	64
Internal Storage	200
SD Card Size	100
Use Host GPU	Sim

3) Modelo do dispositivo (*device*): Nexus 6

Características	Nexus 6
AVDs Name	**AVD19_Nexus6**
Device	Nexus 6
Target (API)	Google API Level 19
CPU/ABI	ARM (armeabi-v7a)
Keyboard	✓
Skin	Nexus 6
Ram	1024
VM Heap	128
Internal Storage	200
SD Card Size	100
Use Host GPU	✓

ATIVIDADE 2 – INICIALIZAÇÃO DE EMULADORES

IV. Descrição da Atividade:

Devido aos recursos do computador **PRIMEIRO devem-se inicializar APENAS 3 emuladores** (que estão em negrito na tabela) depois de terminar a atividade 3 (execução de scripts). Fechar os 3 emuladores usados e inicialize os 2 que faltam. Registre o tempo que demorou a inicialização do emulador indicando a hora do início e fim. Use a hora do seu computador.

Exemplo: Inicio: 18:30 Fim: 18:38

> *Nota: Um emulador termina a inicialização quando você é capaz de abrir e fechar o App Calculadora*

Id	Modelo do Dispositivo
1	**GalaxyNexus**
2	**Nexus5**
3	**Nexus6**

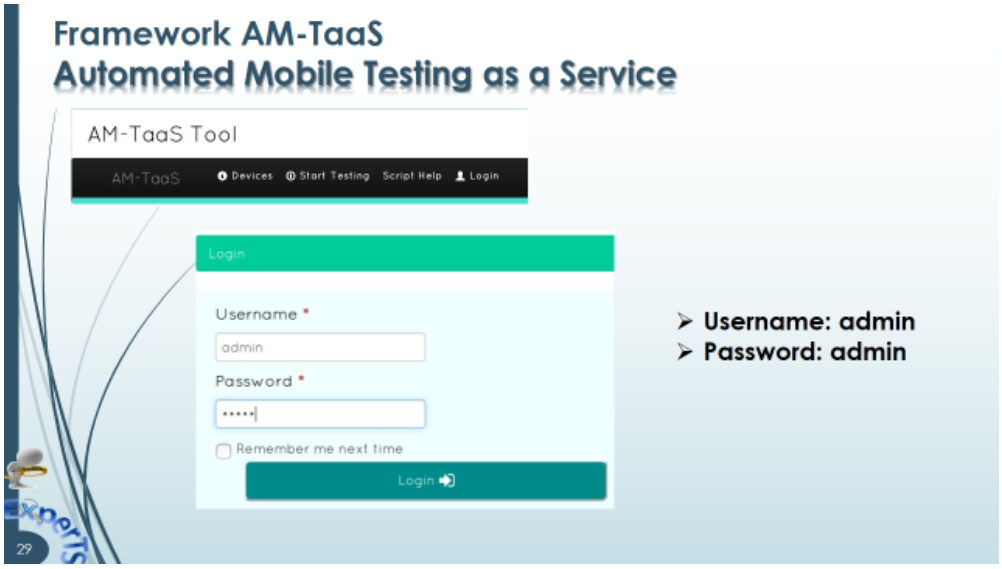
V. Passos da Atividade:

1. Abra o AVD Manager, com o comando: **$ android avd**
2. Clique na aba **"Android Virtual Devices"**
3. Selecione um emulador
4. Clique no botão "**Start**"
5. Registre a hora de Início e esperar finalizar a inicialização do emulador.
6. Registre a hora de término.
7. Se ocorrer alguma falha registre-o na coluna Observação.

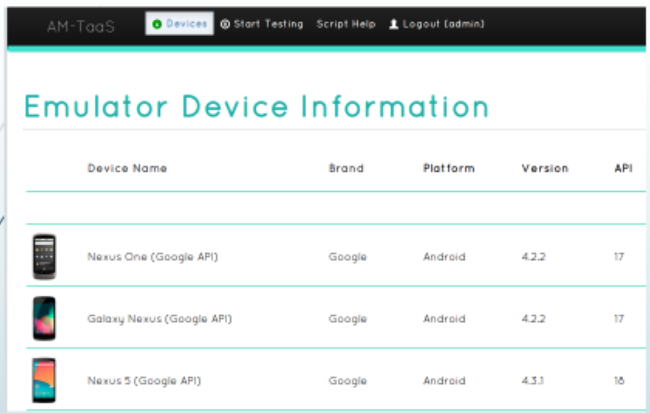
Se o emulador demorar mais de 5 minutos em inicializar feche-o e volte a iniciar, atualize a hora do início e a hora término, logo registre-o na coluna observação.

ID	Nome	Inicio	Fim	Observação
1	AVD17_GalaxyNexus			
2	AVD18_Nexus5			
3	AVD19_Nexus6			

Apêndice G: Artefatos utilizados e guia das atividades - Sessão 2

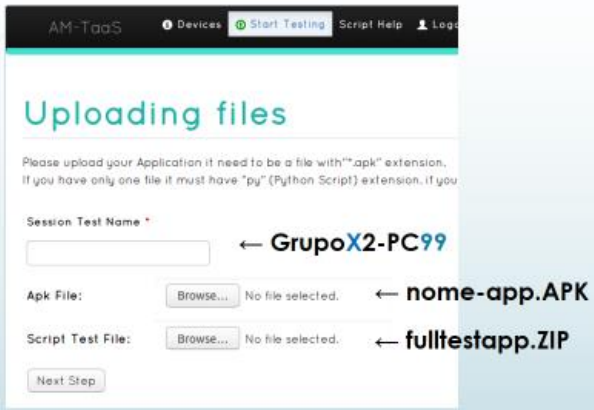


Framework AM-TaaS
Automated Mobile Testing as a Service



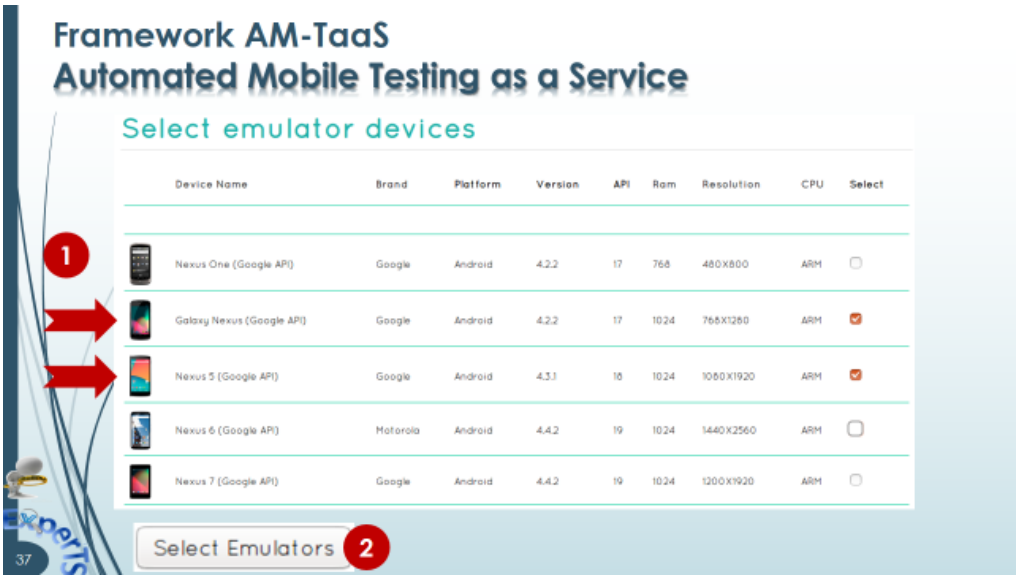
33

Framework AM-TaaS
Automated Mobile Testing as a Service



35

Framework AM-TaaS
Automated Mobile Testing as a Service



37

Framework AM-TaaS
Automated Mobile Testing as a Service

Select emulator devices

	Device Name	Brand	Platform	Version	API	Ram	Resolution	CPU	Select
	Nexus One (Google API)	Google	Android	4.2.2	17	768	480X800	ARM	☐
	Galaxy Nexus (Google API)	Google	Android	4.2.2	17	1024	768X1280	ARM	☑
	Nexus 5 (Google API)	Google	Android	4.3.1	18	1024	1080X1920	ARM	☑
	Nexus 6 (Google API)	Motorola	Android	4.4.2	19	1024	1440X2560	ARM	☑
	Nexus 7 (Google API)	Google	Android	4.4.2	19	1024	1200X1920	ARM	☐

1

Select Emulators 2

39

Framework AM-TaaS
Automated Mobile Testing as a Service

http://localhost/siteamtaas/

41

Framework AM-TaaS
Automated Mobile Testing as a Service

Test Results #

Nexus One (Google API)

OTA INSTALL

Id	Test	Time	Status
1	Installation	0.00:10.475999	passed
2	No space left on device	0.00:43.688000	passed

OTA UNINSTALL

Id	Test	Time	Status
1	Uninstall	0.00:16.334000	passed

USER SCRIPTS

Id	Test	Time	Status

Device.Log

TEST-LOG
Debug.log
3292 bytes

43