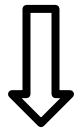
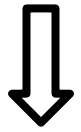
How do Developers Test android Applications?

Como Android Devs. fazem...

Design dos Testes



Desenvolvimento dos Testes



Qualidade dos testes

Tipos de Teste em Android

Teste Manual

- Análise da UI da aplicação
- Testes feitos por um time de QA
- Entender como clientes usam a aplicação
- **Rodar o código e ver o que acontece no device**

Teste Unitários

- Isolam cada componente da aplicação
- Testam o comportamento esperado
- *Servem como documentação do código*
- (Em teoria) executam rapidamente

Random Testing

(a.k.a. teste do macaco)

- Um conjunto pseudo-aleatório de eventos de usuário enviado ao dispositivo
- Útil para encontrar bugs mais escondidos, como *memory leaks*
- Difícil entender qual o fluxo

Conteúdo

- Introdução
  - Problema
  - Objetivo
- Design do estudo empírico
- Resultados e discussões

Introdução

Técnicas de automação não são usadas na prática

- **Falta** de casos de **teste reprodutíveis**
- **Efeitos colaterais** de um caso de teste em outro
- **Falta** de **suporte para debugging**

Problema

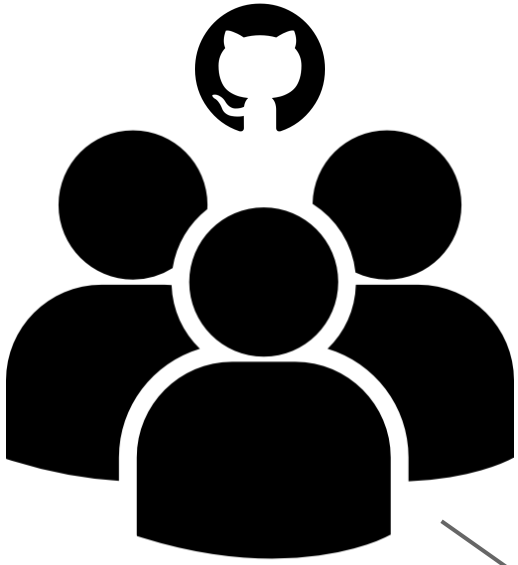
Falta de alinhamento entre as pesquisas em técnicas de testes automáticos **e as necessidades de desenvolvedores.**

Objetivo

Preencher essa lacuna através de uma **pesquisa de campo** *(survey)* **para examinar as preferências de desenvolvedores e**, portanto, **direcionar pesquisadores.**

Pesquisa de campo

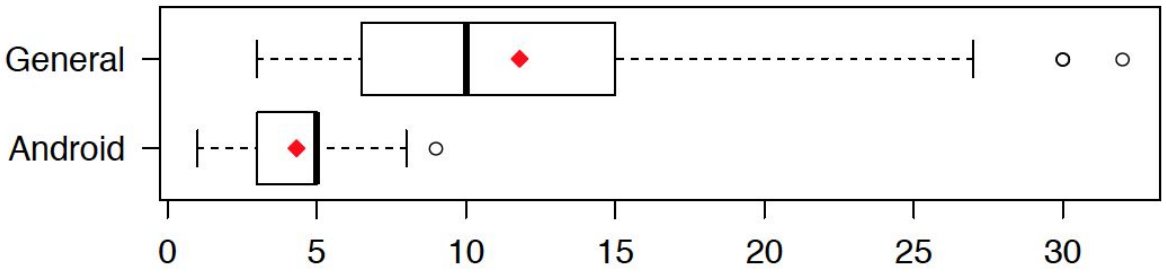
10.000 emails enviados

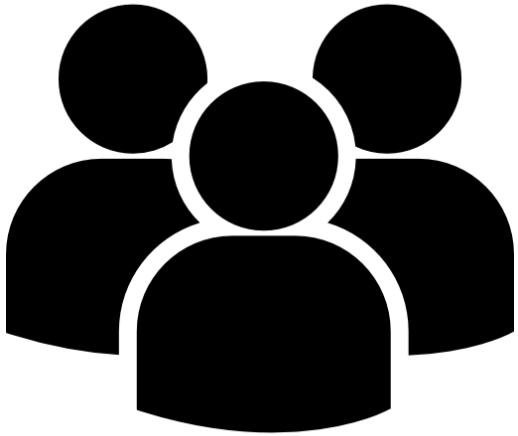


485 respostas

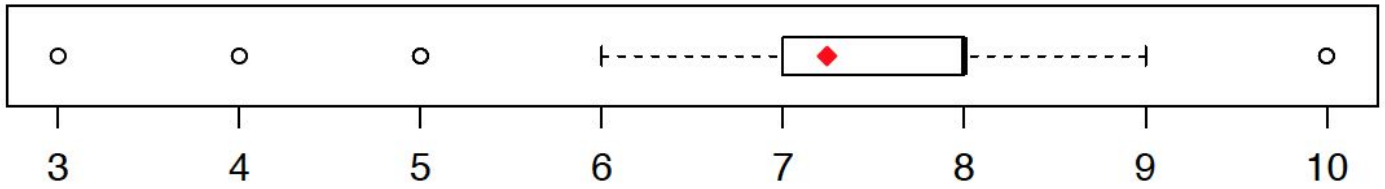
102 foram consideradas

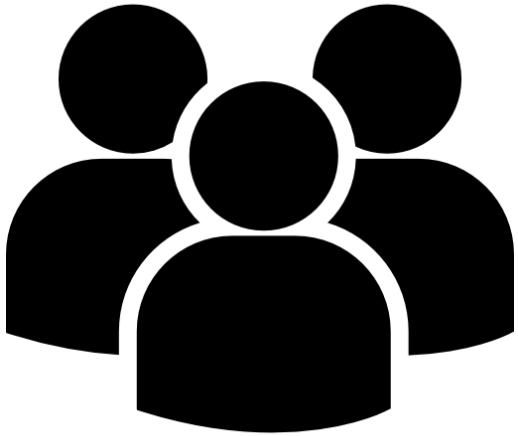
Experiência em programação (em anos)





Quão experiente em programação você se considera?

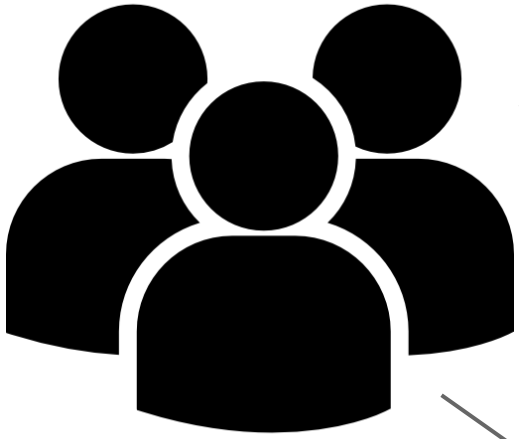




Nível acadêmico dos participantes

51 Bacharelado

36 Mestrado

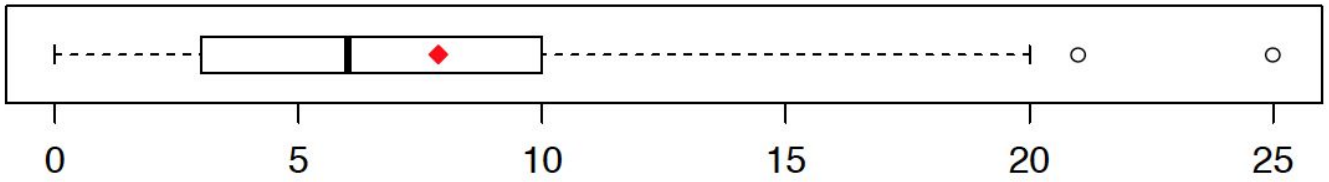


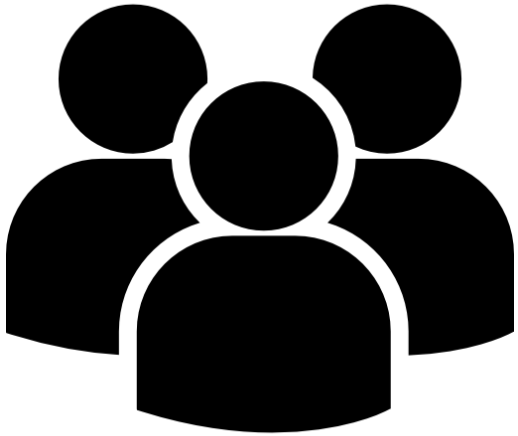
13 High School

2 Pós-Doutorado

1 PhD

Experiência na indústria (em anos)



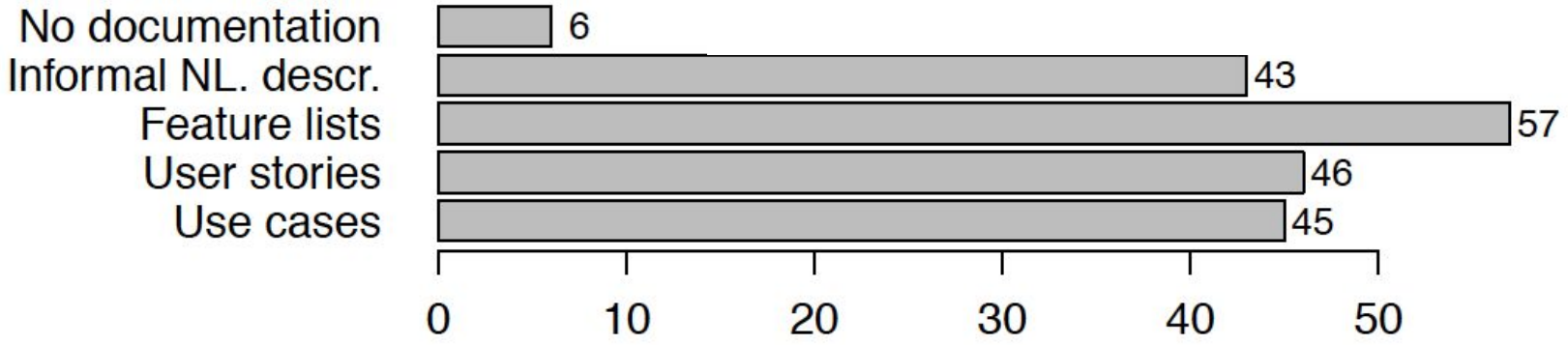


Perguntas de pesquisa

- **RQ1:** Estratégias para design de testes
- **RQ2:** Preferências para casos de teste
- **RQ3:** Atuais práticas de teste
- **RQ4:** Métricas para qualidade dos testes

RQ1: Estratégias de teste

Documentação de requisitos do App


No documentation
Informal NL. descr.
Feature lists
User stories
Use cases
6
43
57
46
45
0
10
20
30
40
50

RQ1: Estratégias de teste

Distribuição do esforço nos testes

Por que tantos preferem testes manuais?

- Mudança constante de requisitos
- Falta de tempo para teste devido a decisões de gerência/clientes
- Custo, em termos de investimento/tempo, para criar e manter
- Falta de conhecimento das ferramentas existentes

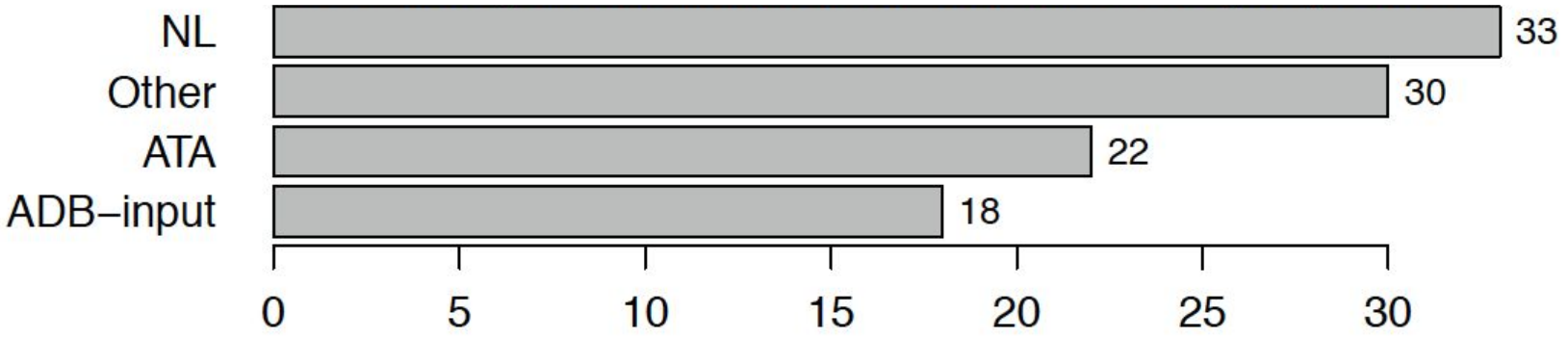
Eu sou mais rápido(a) testando todas as possibilidades do app no próprio dispositivo do que criando casos de testes separados.

Nunca me envolvi num projeto grande o suficiente para envolver testes unitários. Não valia a pena investir nisso.

RQ2: Preferências para testes gerados automaticamente

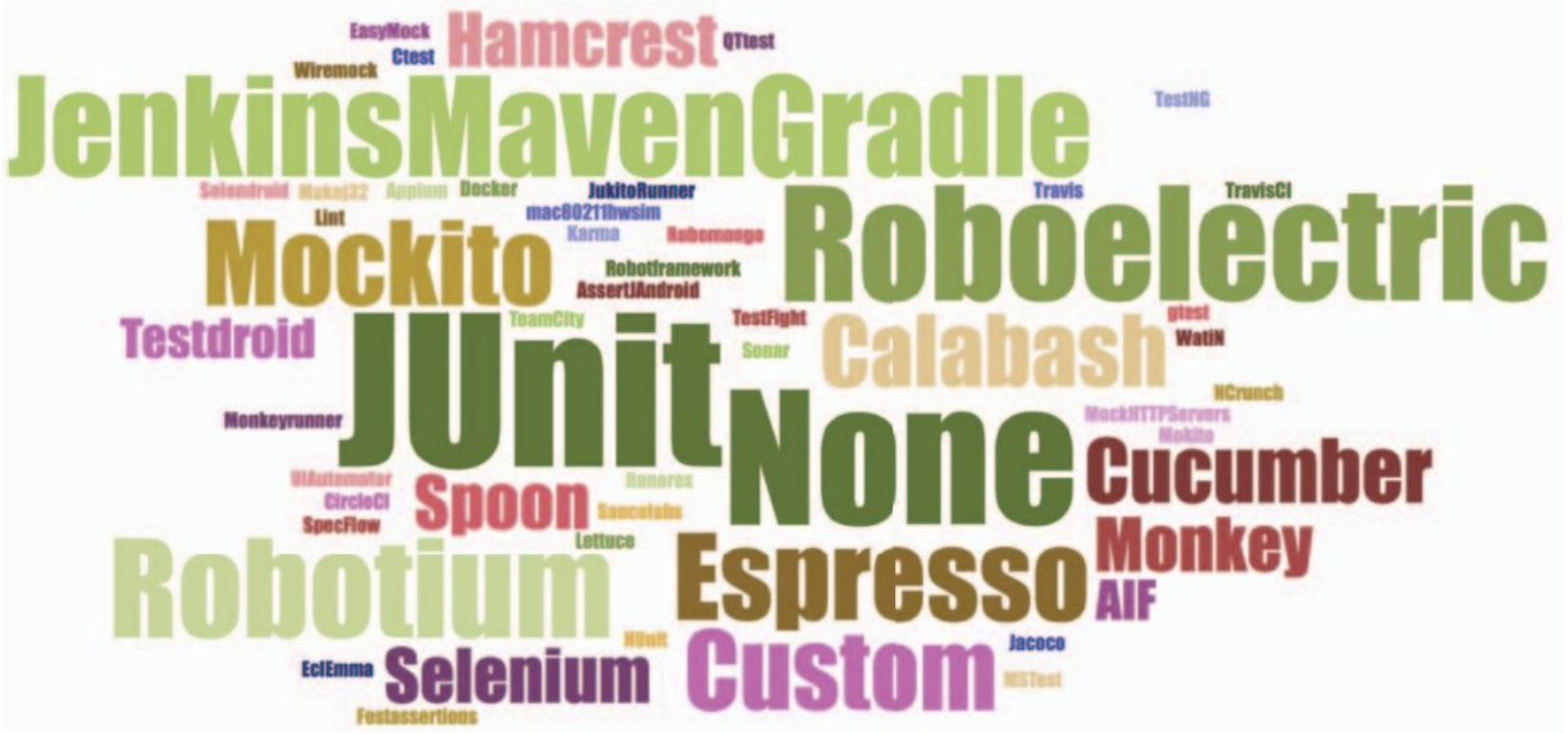
Quais ferramentas vocês preferem?

Se você usa (ou pretende usar) ferramentas para geração automática de casos de teste, que formato de casos de teste você prefere?



RQ3: Atuais práticas de teste

Ferramentas para testes automáticos


EasyMock
Hamcrest
QTtest
Ctest
Wiremock
JenkinsMavenGradle
TestNG
Selendroid
Make[32
Appium
Docker
JukitoRunner
Travis
TravisCI
Lint
mac80211hwsim
Karma
Robomonga
Roboelectric
Mockito
Robotframework
AssertJAndroid
TeamCity
TestFight
gtest
Testdroid
JUnit
Sonar
Calabash
WatiN
NCrunch
Monkeyrunner
MockHTTPServers
Mokito
None
Cucumber
UIAutomator
Robotex
Spoon
CircleCI
Saucelabs
SpecFlow
Lettuce
Monkey
Robotium
Espresso
AIF
NUnit
Jacoco
EclEmma
Selenium
Custom
MSTest
Festassertions

RQ3: Atuais práticas de teste

Experiências com ferramentas de testes aleatórios

Monkey é muito útil para fazer stress test na aplicação, ou verificar que não tem leaks. Às vezes, encontra bugs estranhos também.

Bom para stress testing, resultados não são muito consistentes.

Eu executei Android Monkey e ele encontrou alguns defeitos, mas os desenvolvedores disseram ‘quase não acontece’ ou ‘nunca acontece’. Portanto, os defeitos foram ignorados.

RQ4: Métricas para qualidade dos testes

Métricas para qualidade dos testes

Cobertura do código pelos testes

- **64%** disseram que é **inútil**

Métricas consideradas úteis por alguns participantes

- Cobertura de features
- Revisão do código de teste
- Quantidade de falhas não-cobertas

Nós medimos o número de bugs não capturados e regressões ao longo do tempo que os desenvolvedores tiveram que gastar tempo consertando.

Nos baseamos em features, elementos cobertos etc, para calcular cobertura total.

Usamos cobertura do código mais como um guia para qual parte do código pode precisar de mais atenção em termos de escrita de testes. Não temos outra métrica para medir qualidade dos casos de teste.

Resumo dos resultados

RQ1: Design dos testes

- Usage model
- Testes baseados em features e casos de uso.

RQ2: Preferências pros casos de teste

- Testes em linguagem natural
- Informação contextual

RQ3: Atuais práticas de teste

- JUnit, Roboelectric e Robotium são mais usadas
- Ferramentas automáticas não são usadas
- Falta expressividade e facilidade de manutenção

RQ4: Métricas para qualidade dos testes

- Cobertura do código é considerada inútil
- Cobertura de features, revisão do código de teste são consideradas úteis.

Em que os pesquisadores devem focar?

- **Reduzir overhead** introduzido por testes automáticos
- Levar em consideração **outras estratégias além de testes aleatórios**
- **Utilizar outras métricas** para qualidade dos casos de teste

Ameaças à Validade

Validação de construção: responde o que se propõe a responder?

- Respostas incompletas/inválidas filtradas
- Participantes sem experiência filtrados

Validação externa: as conclusões podem ser generalizadas?

- Pode não ser para devs de outras plataformas
- Só foca em desenvolvedores open-source no Github
- Participantes na indústria?
- Participantes representam toda a comunidade de android devs?