To the Attention of Mobile Software Developers: Guess What, Test your App!

...

Apresentação do artigo por rma7 e hscs

Introdução

Os testes de software são uma técnica de desenvolvimento muito importante, pois através dela é possível identificar bugs no sistema antes dele ser entregado para os usuários.

Introdução

Existem várias pesquisas sobre testes em desenvolvimento de software geral, porém as particularidades do desenvolvimento de aplicações mobile fazem com que as aplicações mobile sejam testadas de forma *ad-hoc*, **se é que são** testadas.

Desafios em testes de aplicações mobile

- As ferramentas convencionais não são suficientes para:
  - Simular interações complexas (e.g., swipe, pinch, etc);
  - Simular dispositivos com diferentes níveis de recursos (e.g. bateria, memória) e com várias versões do OS.

Desafios em testes de aplicações mobile

- Aplicativos normalmente seguem uma estratégias de releases semanais, o que limita bastante o tempo para tarefas que não impactam diretamente a regra de negócio.
- Testes manuais, apesar de serem mais fáceis de executar, tomam muito tempo.

Por testes automatizados foram considerados:

- Testes de Unidade
- Testes de GUI
- Serviços de Teste baseados em Cloud
- Continuous Integration/ Continuous Development (CI/CD)

Esse artigo estuda a adoção de técnicas de testes automatizados por projetos Android Open Source.

Perguntas da Pesquisa

- **RQ1 -** What is the prevalence of automated testing technologies in the FOSS mobile app development community?
- **RQ2 -** Are today's mature FOSS Android apps using more automated testing than yesterday's?
- **RQ3 -** How does automated testing relates to popularity metrics in FOSS Android apps?
- **RQ4 -** How does automated testing affect code issues in FOSS Android apps?
- **RQ5 -** What is the relationship between the adoption of CI/CD and automated testing?

Coleta de Dados

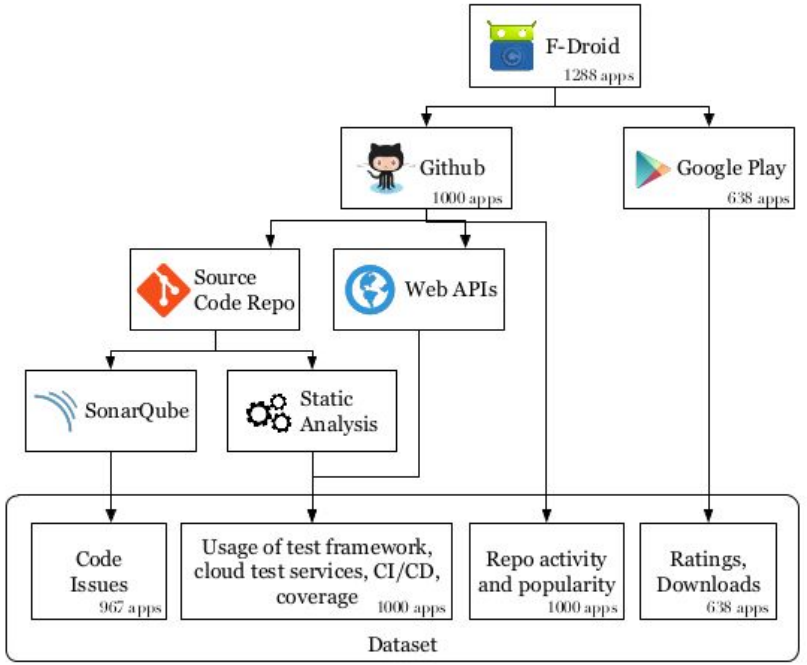
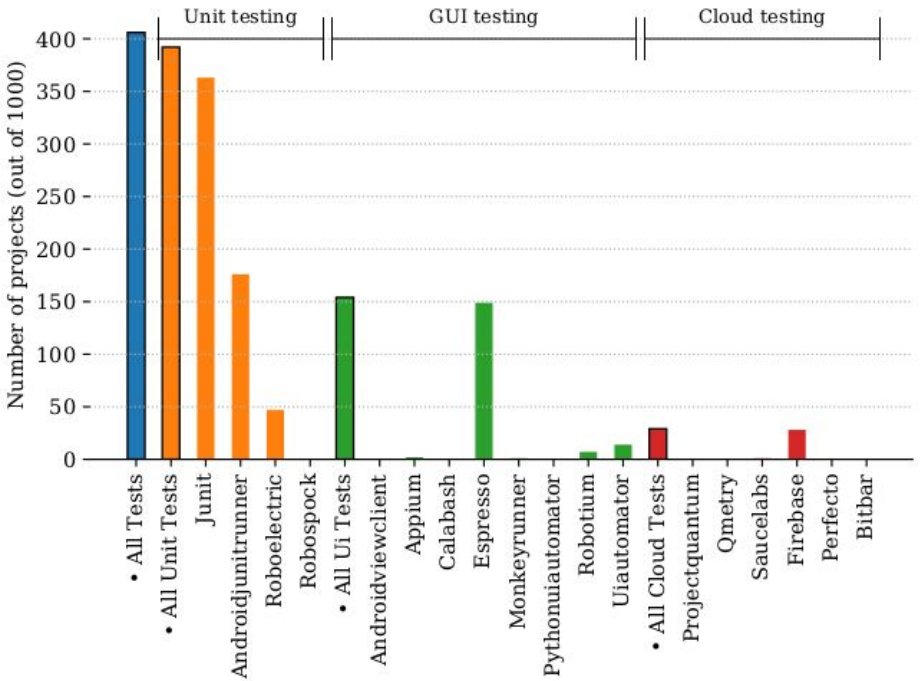


Fig. 1 Flow of data collection in the study.

RQ1 - What is the prevalence of automated testing technologies in the FOSS mobile app development community?

Os 1000 aplicativos foram analisados por uma ferramenta de análise estática de código, que testava o uso de uma série de ferramentas de teste.

Resultados


Unit testing
GUI testing
Cloud testing
Number of projects (out of 1000)
400
350
300
250
200
150
100
50
0
• All Tests
• All Unit Tests
Junit
Androidjunitrunner
Roboelectric
Robospock
• All Ui Tests
Androidviewclient
Appium
Calabash
Espresso
Monkeyrunner
Pythonuiautomator
Robotium
Uiautomator
• All Cloud Tests
Projectquantum
Qmetry
Saucelabs
Firebase
Perfecto
Bitbar

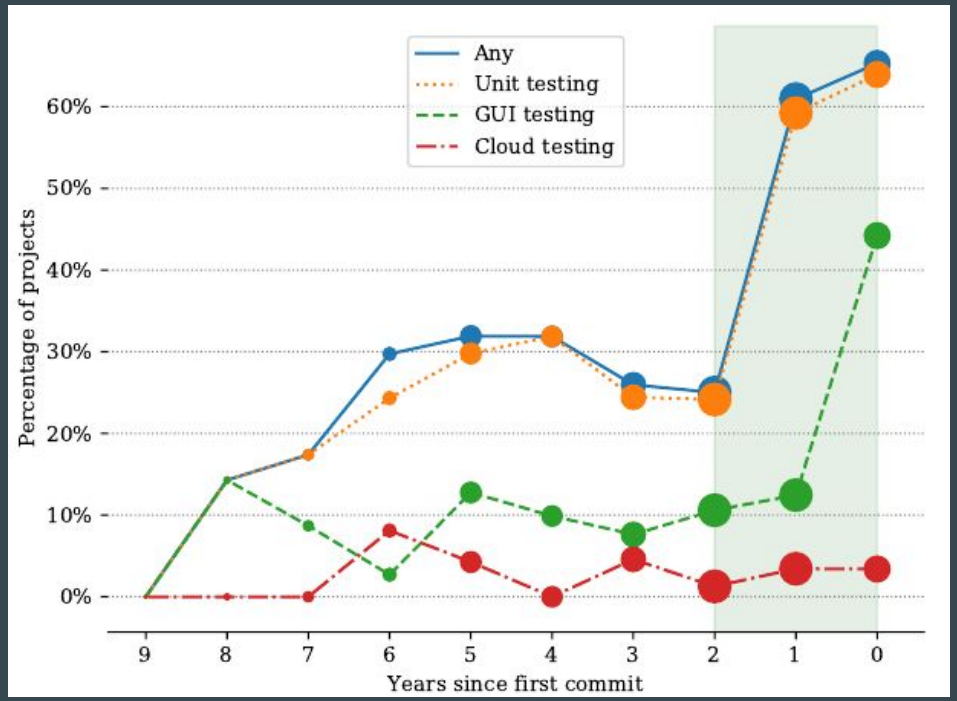
Discussão

- A maioria dos aplicativos analisados não utilizavam nenhum tipo de teste automatizado (59%);
- Por serem mais simples, testes de unidade são a forma de testes mais utilizada (39%), sendo a ferramenta de testes de unidade mais utilizada o ***JUnit***, pelo fato dela ser a ferramenta introduzida pela documentação oficial do Android.
- Testes de GUI não são muito populares (15%), provavelmente por serem difíceis de manter. Entre os aplicativos que usam Testes de GUI, o framework mais utilizado é o ***Espresso***, que também é apresentado pela documentação oficial do Android.
- Serviços baseados em Cloud não estão difundidos (3%) na comunidade FOSS de Android, provavelmente pelo fato de serem uma tecnologia recente.

RQ2 - Are today’s mature FOSS Android apps using more automated testing than yesterday’s?

Foi analisada a relação de testes automatizados com a ‘idade’ do app, anualmente, olhando a versão mais atual do aplicativo

Isto é, a idade é medida em relação ao primeiro commit, mas o repositório tem seu código atualizado no commit mais atual.



O tamanho dos círculos é proporcional a quantidade de apps com a respectiva idade. O autor desconsidera a significância de apps com 6 anos ou mais.

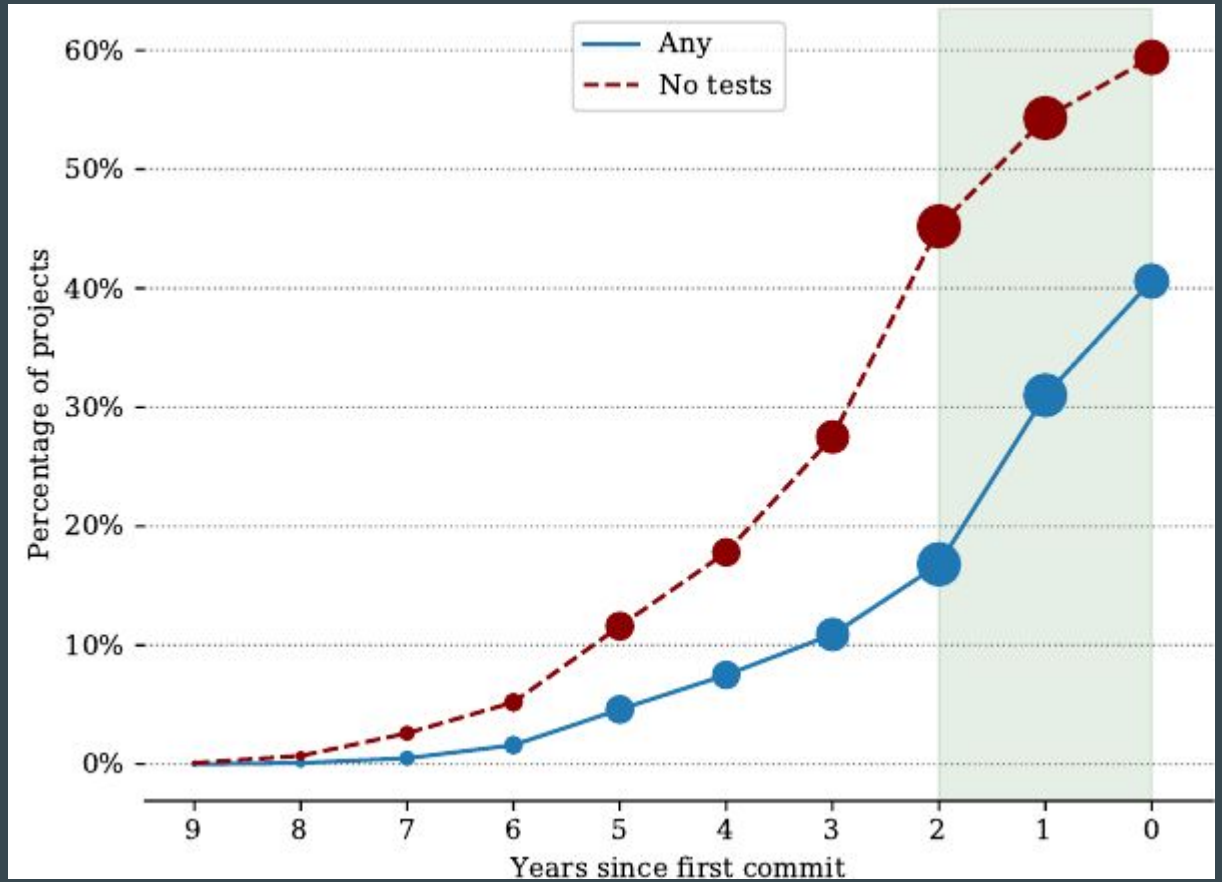
Apps com 2 anos de idade ou menos tem usado mais testes, e a quantidade de testes de GUI tem mostrado uma tendência de crescimento também no mesmo período de tempo.

Mesmo assim, a maioria dos aplicativos android ainda não estão fazendo testes automatizados.

E quanto aos apps antigos?

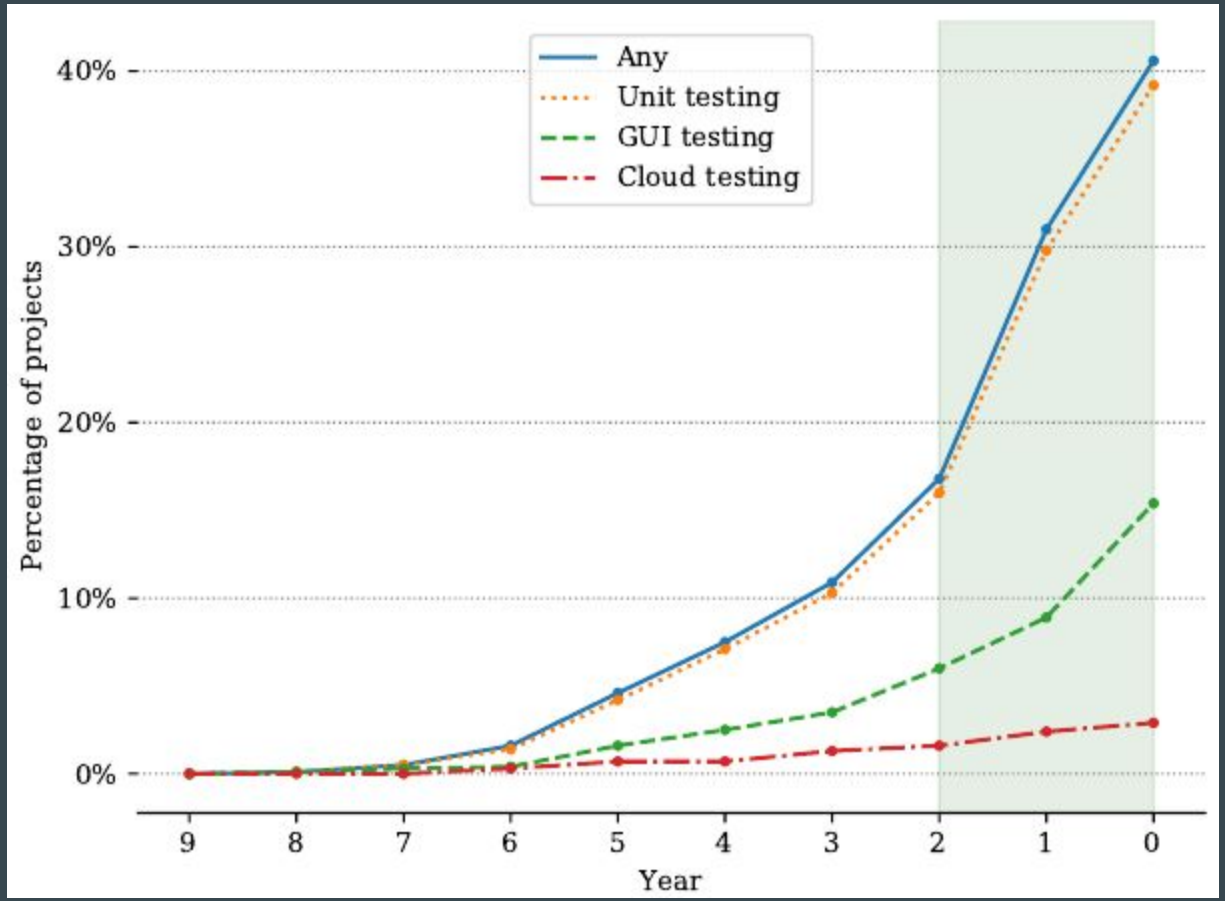
2 Fenômenos podem explicar a falta de testes:

- Testes automatizados estão ficando **mais acessíveis** atualmente e os desenvolvedores **cientes dos benefícios**, aplicando em projetos recentes.
- Desenvolvedores vêem que o ***overhead*** de manter o app com testes **não vale a pena**, e **retiram** a implementação deles do projeto.



No entanto esse gráfico nos dá a ideia de que o primeiro fenômeno, mais positivo, é o mais provável.

Apps com 2 anos de idade ou menos tem usado mais testes, e a quantidade de testes de GUI tem mostrado uma tendência de crescimento também no mesmo período de tempo.



Adoção de tipos de teste pela idade do app. (hscs) - A relação do uso crescente dos testes de GUI parece justificável pelo surgimento do framework ***Espresso,*** 6 anos atrás.

O crescimento da implementação de testes de GUI e Unidade é bastante positivo. Primeiro, porque apenas testes de unidade não permitem que seja alcançada uma cobertura de testes elevada em apps mobile. Segundo, porque isto fornece mais casos de uso para que pesquisadores estudem novos testes (e.g. bateria, segurança...)

RQ3 - How does automated testing relates to popularity metrics in FOSS Android apps?

Foram comparadas a seguintes medidas de popularidade de um aplicativo com a adoção ou não de testes

- Quantidade de Estrelas (no Github)
- Quantidade de Forks (no Github)
- Quantidade de Contribuidores (no Github)
- Número de Commits
- Nota (na Google Play store)
- Número de avaliações(na Google Play store)

Resultados

Table 2 Statistical analysis of the impact of tests on the popularity of apps.

	p-value	$\Delta\bar{x}$	ΔMd	CL (%)
Stars	0.2130	54.78	3.00	52.74%
Forks	0.4525	11.39	1.00	51.40%
Contributors	0.0052	**2.17**	**0.00**	**55.80%**
Commits	**0.0008**	**247.58**	**49.00**	**57.13%**
Rating Value	0.0835	0.05	0.05	54.77%
Rating Count	0.2465	−894.26	−56.00	47.03%

Só existem evidências estatísticas que projeto que aplicam testes automatizados tem mais **commits** e **contribuidores**.

Discussão

- O aumento no número de commits pode ser explicado pelo esforço necessário para manter e configurar testes.
- Na questão de quantidade de contribuidores, foi levantado que testes ajudam novos contribuidores, e conseguir contribuidores é uma preocupação grande de mantenedores open source.

RQ4 - How does automated testing affect code issues in FOSS Android apps?

Uso da ferramenta de análise estática **Sonar** no dataset de apps, para avaliar o número de issues, classificadas por impacto:

Blocker, Critical, Major, Minor

Apps que não implementam testes automatizados tem um aumento de 18% em *minor issues*. Demais tipos de issues não tem muita significância, já que testes manuais, análise estática, etc, podem ser suficientes para prevenção.

Apenas 41% dos apps implementavam testes automatizados.

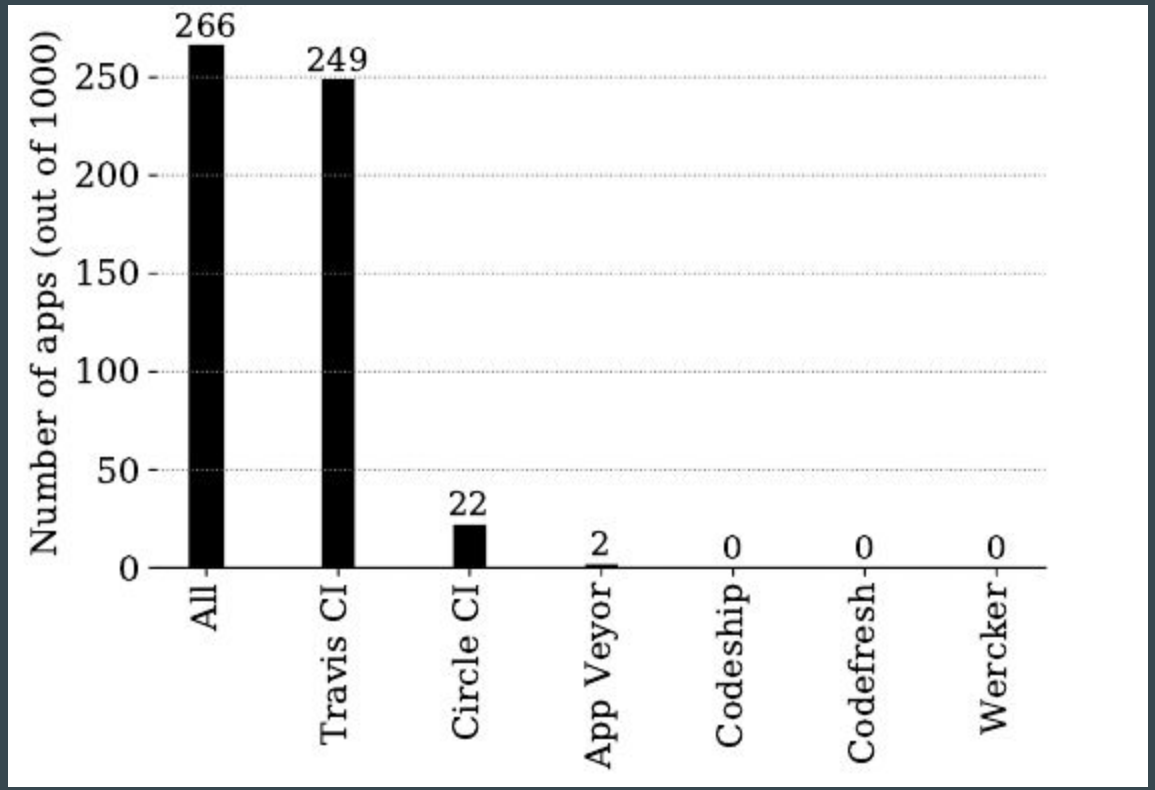
RQ5 - What is the relationship between the adoption of CI/CD and automated testing?

Comparações feitas

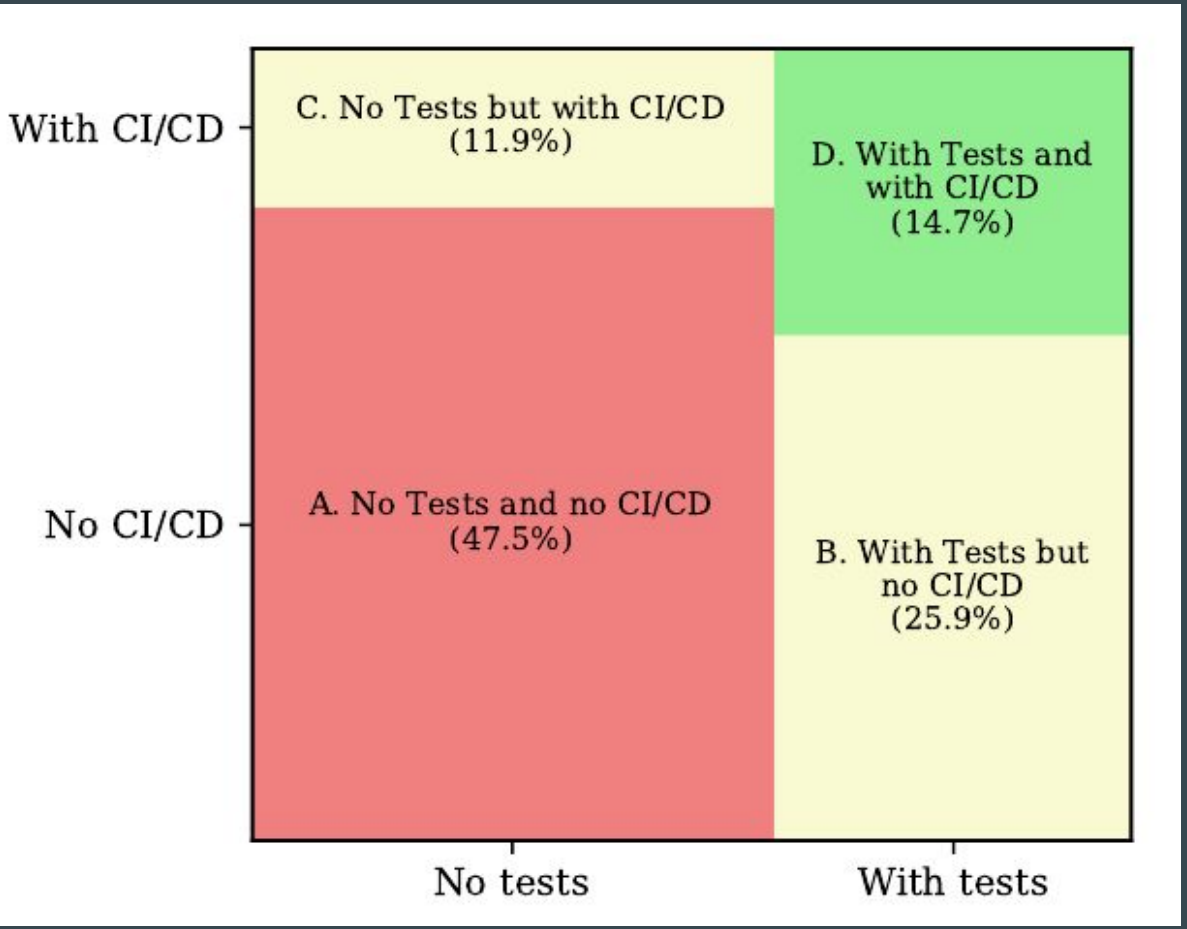
- Adoção de diferentes tecnologias de CI/CD
- Frequência de projetos usando CI/CD

x

- Frequência de projetos usando Testes



Adoção de diferentes tecnologias de CI/CD - Travis CI é a preferida para projetos Open Source, provavelmente pela facilidade de uso e fácil integração com Github.



A porcentagem de apps **sem Testes e sem CI/CD** é quase 50% de todos os apps analisados. Testes se mostraram **mais prevalentes** em projetos com CI/CD

Discussão

- CI/CD não é amplamente adotado para apps como em projetos de software tradicionais. Apenas 26% dos apps adotam CI/CD, enquanto a fatia de software no Github que usa essa tecnologia é de 40%

Discussão

- Maior obstáculo de implementar CI/CD: desenvolvedores não são familiarizados com isso

- Necessidade: mais treinamento e suporte para os desenvolvedores. Incentivo para apps que já implementam testes, implementarem CI/CD

Threats to Validity

- A pesquisa foi feita com aplicativos open source e gratuitos, o que pode não representar bem os aplicativos pagos e de código fechado, que por terem um orçamento maior, podem aplicar melhor essas técnicas.

- A análise dos apps é feita com um software desenvolvido e testado pelos autores do artigo, mas ainda podem haver *corner cases* que levem a falsos positivos ou falsos negativos.

- O uso do *SonarQube* para analisar as issues dos códigos não leva em consideração que algumas guidelines de desenvolvimento podem não estar sendo seguidas devido a requisitos do projeto

Conclusion

- Testes são cruciais para entregar apps de qualidade

- Apps mobile ainda são testados de maneira muito ***ad hoc,*** se é que são testados.

- Houve aumento no uso de testes, com os frameworks favoritos sendo **Espresso** e **JUnit**.

- Pipelines CI/CD são raridade - provavelmente pela própria ausência de testes automatizados.

Future Work

1) Como projetos de apps mobile testam para requisitos específicos como energia, segurança...?;
2) Baseado nos testes coletados, elaborar um conjunto de boas práticas para desenvolvedores;
3) Verificar que as descobertas também valem para outras plataformas.

E é isso aí.

Dúvidas? Questões? Comentários? Críticas?
Sugestões? Adições? Colaborações?