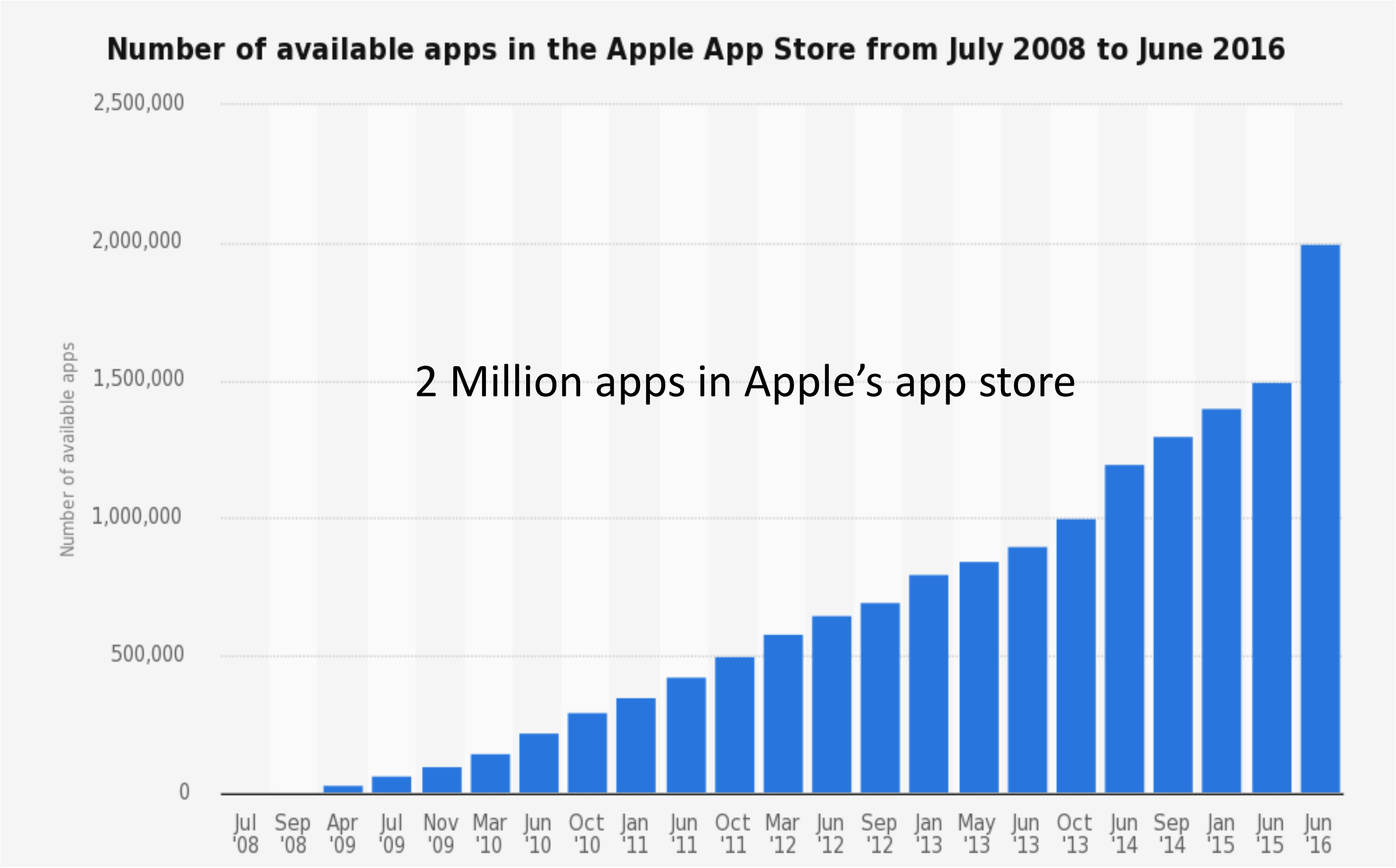
Predicting Crashing Releases of Mobile Applications

Xin Xia, **Emad Shihab**, Yasutaka Kamei, David Lo, Xinyu Wang

eshihab@cse.concordia.ca

das.encs.concordia.ca

Mobile Applications are On the Rise



Source: http://www.statista.com/statistics/263795/number-of-available-apps-in-the-apple-app-store/

Mobile Apps are Different

Empir Software Eng (2016) 21:1346–1370
DOI 10.1007/s10664-015-9388-2

CrossMark

Fresh apps: an empirical study of frequently-updated mobile apps in the Google play store

Stuart McIlroy[1] · Nasir Ali[2] · Ahmed E. Hassan[1]

Published online: 7 July 2015

Abstract Mobile app stores provide a unique platform for developers to rapidly deploy new updates of their apps. We studied the frequency of updates of 10,713 mobile apps (the top free 400 apps at the start of 2014 in each of the 30 categories in the Google Play store). We find that a small subset of these apps (98 apps representing ~1 % of the studied apps) are updated at a very frequent rate — more than one update per week and 14 % of the studied apps are updated on a bi-weekly basis (or more frequently). We observed that 45 % of the frequently-updated apps do not provide the users with any information about the rationale for the new updates and updates exhibit a median growth in size of 6 %. This paper provides information regarding the update strategies employed by the top mobile apps. The results of our study show that 1) developers should not shy away from updating their apps very frequently, however the frequency varies across store categories. 2) Developers do not need to be too concerned about detailing the content of new updates. It appears that users are not too concerned about such information. 3) Users highly rank frequently-updated apps instead of being annoyed about the high update frequency.

Communicated by: Andreas Zeller

✉ Stuart McIlroy
mcilroy@cs.queensu.ca

Mobile apps **have many releases…**

…driven by two key factors:

- Competition
- App stores

Mobile App Crashes are a Challenge

FEATURE: MOBILE APPS

What Do Mobile App Users Complain About?

Hammad Khalid, Shopify

Emad Shihab, Concordia University

Meiyappan Nagappan, Rochester Institute of Technology

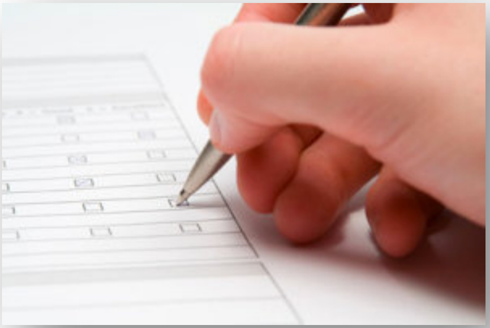
Ahmed E. Hassan, Queen's University

// A study of user reviews from 20 iOS apps uncovered 12 types of user complaints. The

from the developers' perspective.[2] However, one of the first steps in understanding the issues affecting app quality is to determine the challenges users face.

The App Store lets users review their downloaded apps. Besides assigning star ratings (all of which are aggregated and displayed on a version-level and an app-level basis), users can add comments explaining their ratings. This data source captures a unique perspective about users' perception of the apps. Such reviews, like product reviews in Web stores, correlate highly with download counts (purchases) and are a key measure of an app's success. (For more on previous research on this topic and other topics related to this article, see the sidebar.) A good understanding of these issues will help developers deal with users' concerns and avoid low ratings. Furthermore, such an understanding is crucial in guiding the software engineering community in tackling high-impact

Mobile users are **frequently and negatively impacted** by app crashes



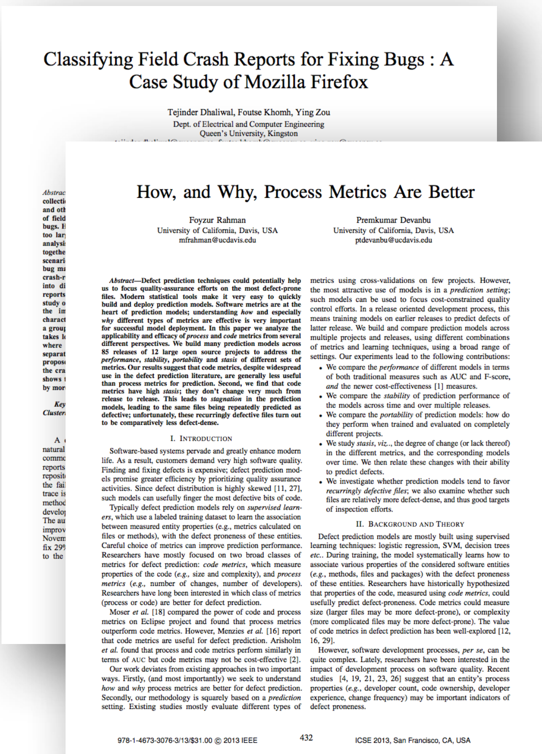
Short release cycles impact testing schedules for developers

Our Goal is to Predict Crashing Releases

RQ1. Can we **effectively predict** crashing releases?

RQ2. What are the **best indicators** of these crashing releases?

How Can we Predict Crashing Releases?


Classifying Field Crash Reports for Fixing Bugs : A Case Study of Mozilla Firefox

Tejinder Dhaliwal, Foutse Khomh, Ying Zou
Dept. of Electrical and Computer Engineering
Queen's University, Kingston

How, and Why, Process Metrics Are Better

Foyzur Rahman
University of California, Davis, USA
mfrahman@ucdavis.edu

Premkumar Devanbu
University of California, Davis, USA
ptdevanbu@ucdavis.edu

I. Introduction

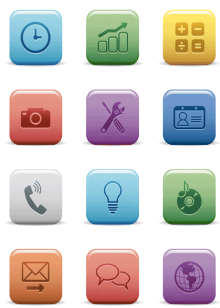
II. Background and Theory

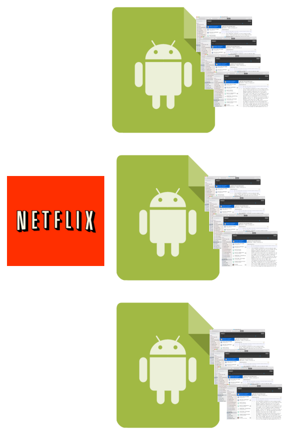
978-1-4673-3076-3/13/$31.00 © 2013 IEEE 432 ICSE 2013, San Francisco, CA, USA

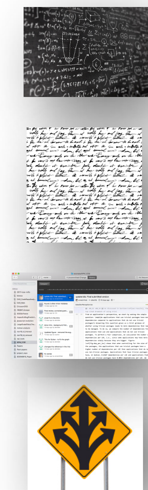
We derive metrics using **development history to predict crashing releases**

Approach

Approach Overview







Code & Complexity

Time & Text

Commit

Diffusion

.

.

Mine Apps (F-droid)

Extract & label crashing releases

Extract factors to predict crashing releases

Predict & evaluate performance

Mobile App Dataset

Mined 900 mobile apps from F-droid

- 466 of these use Git
- 22 apps are active & have 2+ yrs of dev history
- **10** have more than 100 releases

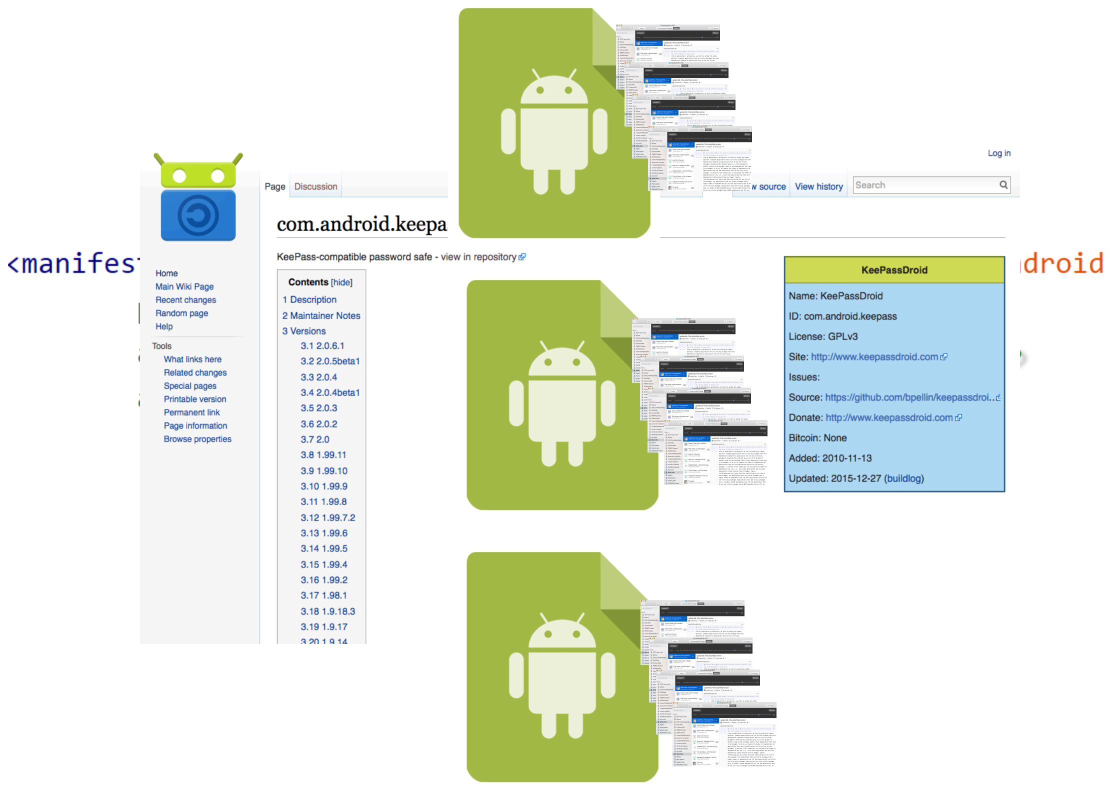
For each app, we collect:

1. Source code

2. Repository meta-data

3. Wiki page

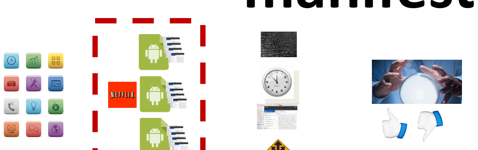
Determining Crashing Releases



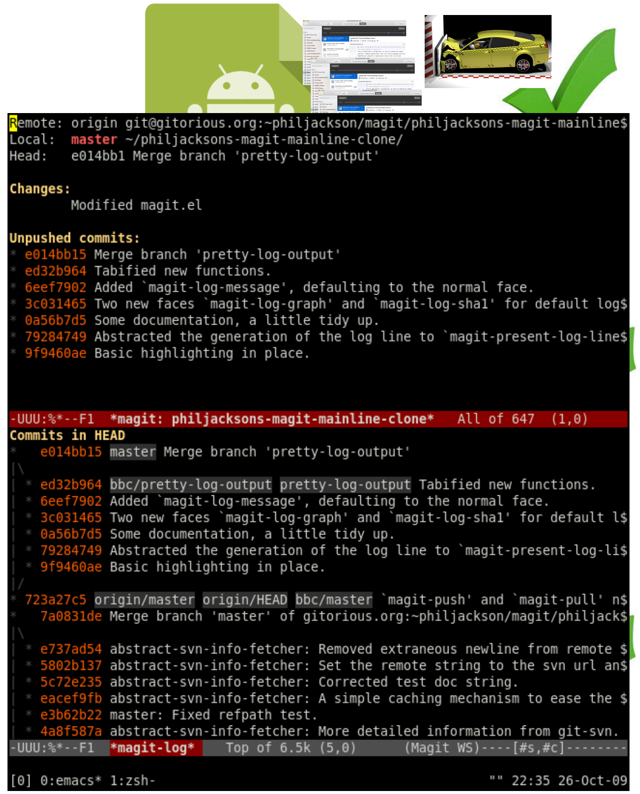
Determine all releases from manifest

Determine releases & release dates

Mine & group commits of the specific releases



Determining Crashing Releases



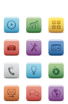
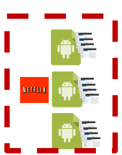
Search commits logs to determine crashing fixes

Mark releases as crash-inducing releases

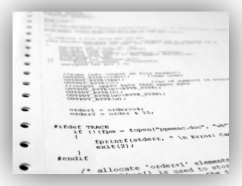
Manually examine (flagged) crashing releases

Determining Crashing Releases

Project	# Releases	# Crashing
App1	597	97
App2	149	23
App3	156	28
App4	392	19
App5	230	36
App6	233	26
App7	262	34
App8	241	34
App9	205	39
App10	123	8
Total	**2,638**	**344**

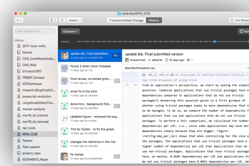



Factors Used in Prediction



Code & Complexity

- Lines added
- Lines del.
- Cyclomatic comp.
- Rel. size
- No. files (curr & prev) release

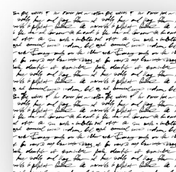


Commit

- No. of commits
- No. bug fixing commits

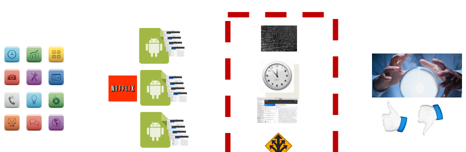
Diffusion

- No. subsystems modified
- Entropy of modified files
- Entropy of code churn

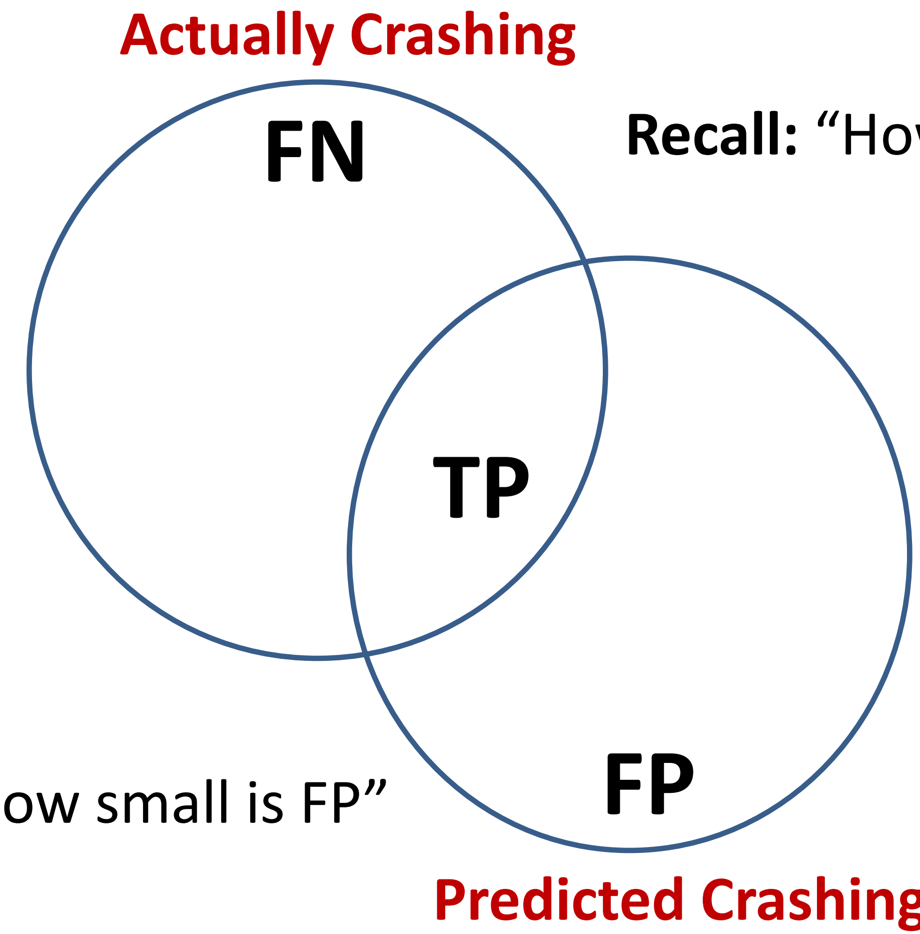


Time & Text

- Days since last rel.
- Fuzzy and Naïve Bayes scores of commit messages from prior release



Evaluating Prediction Models



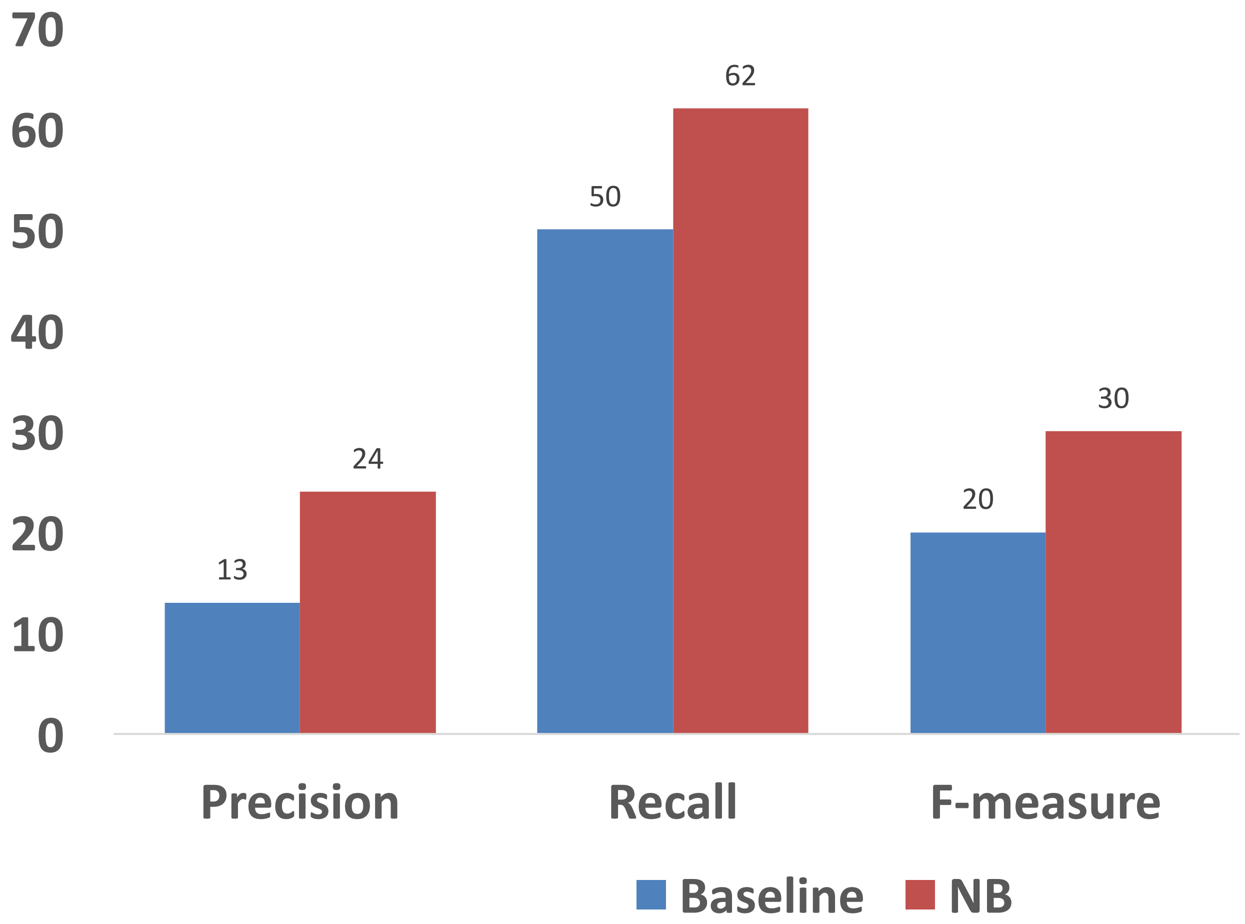
Recall: “How small is FN”

Precision: “How small is FP”

F-measure: Harmonic mean of Precision and Recall

Empirical Results

RQ1. How well can we predict crashing release?



***Evaluated on a 100 times 10-fold cross validation.**

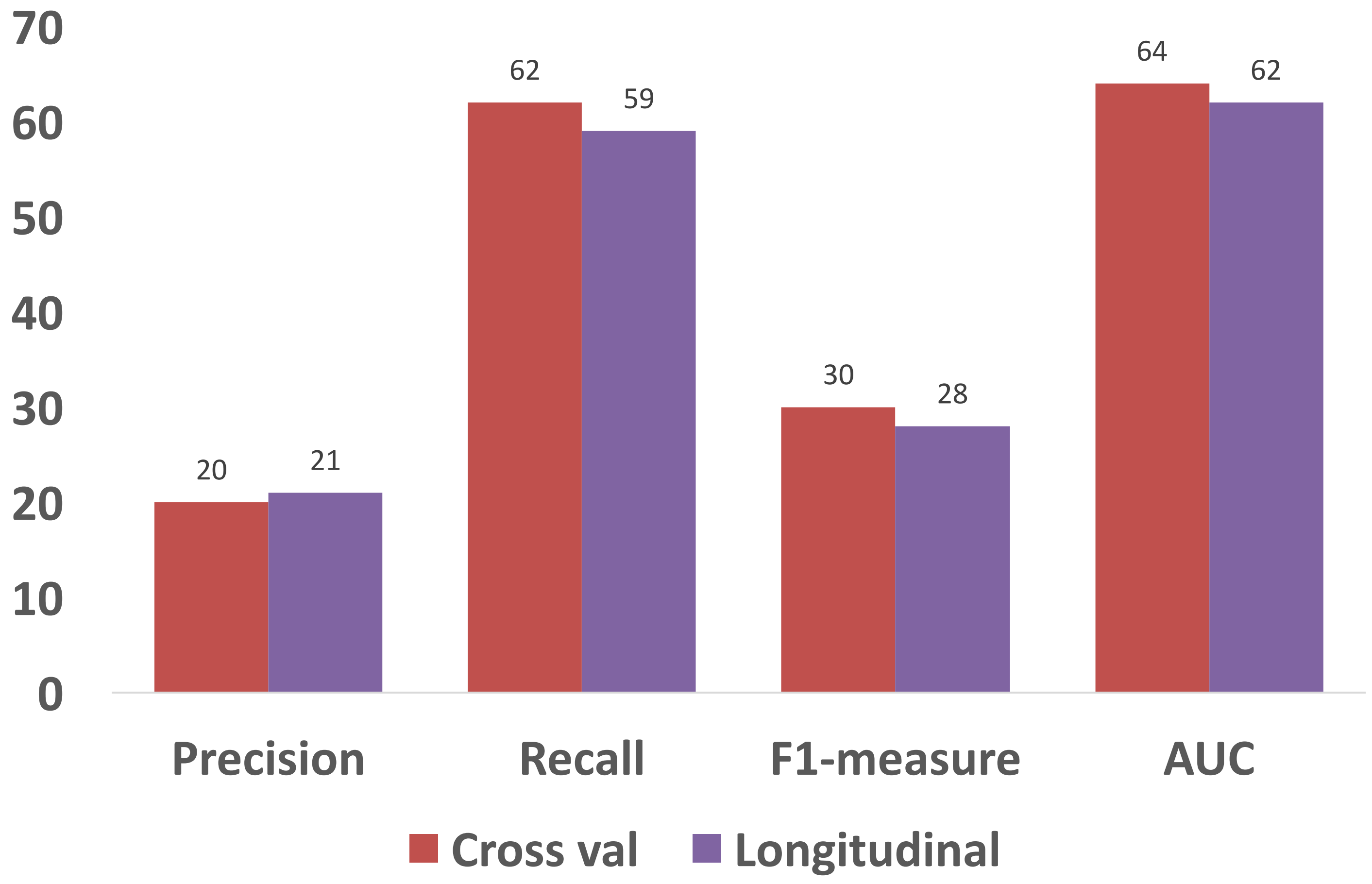
RQ2. What are the best indicators of a crashing release?

Project	Factor 1
App1	Text
App2	Text
App3	Text
App4	Text
App8	Text
App9	Code & Comp
App10	Text

Text and code & complexity factors are the best indicators of crashing releases

Longitudinal Analysis

Train on the <u>earliest 70%</u> of the data and test on the last 30%



Developer Feedback

Majority of developers indicated that the factors and machine learning technique are practical

"we can easily extract these factors from commits in a release…"

"given that releases tend to be small, inspecting an entire release is not impossible…"

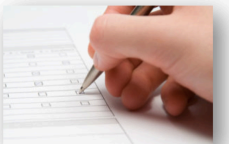
"although F1-score is not high, recall score is good, in practice we are interested to find all the crash releases as possible, thus recall is more important…"

Limitations

- Our approach is **heavily dependent on commit logs**
- Examine what **textual features best indicates crashing releases**
- **Improve prediction accuracy**, possibly by adding more/better factors
- **Replicate** on more applications

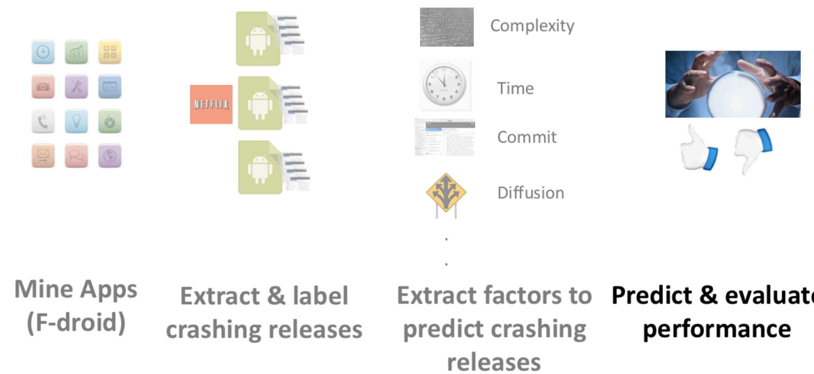
Mobile App Crashes are a Challenge

Mobile users are **frequently and negatively impacted** by app crashes

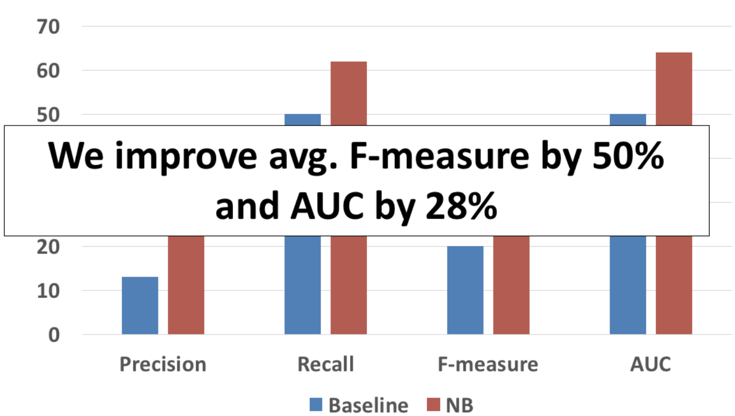


Short release cycles impact testing schedules for developers

Approach Overview



RQ1. How well can we predict crashing release?



We improve avg. F-measure by 50% and AUC by 28%

*Evaluated on a 100 times 10-fold cross validation.

RQ2. What are the best indicators of a crashing release?

Project	Factor 1	Factor 2
App1	Text	Complexity
App2	Text	Complexity
App3	Text	Code
App7	Text	Code
App8	Text	Code
App9	Code	Text
App10	Text	Text

Text, code and complexity factors are the best indicators of crashing releases