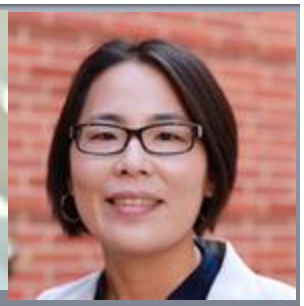
39th IEEE International Conference on Software Maintenance and Evolution (ICSME 2023) Bogota, Colombia

Most Influential Paper Award from ICSME 2013

"An Empirical Study of API Stability and Adoption in the Android Ecosystem"

by Tyler McDonnell, Baishakhi Ray, and Miryung Kim





COLUMBIA UNIVERSITY

UCLA

What was 2013 like?

Senior undergraduate student: Tyler McDonnell

2013: Senior Undergrad at University of Texas, Austin

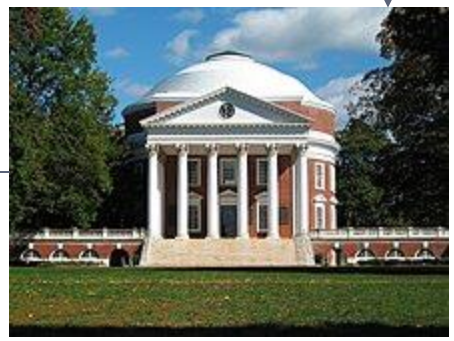
2018: PhD in AI from University of Texas, Austin

2023: Senior Manager, AI Researcher, lives in Austin area

4th Year graduate Student: Baishakhi Ray

PhD, UT Austin

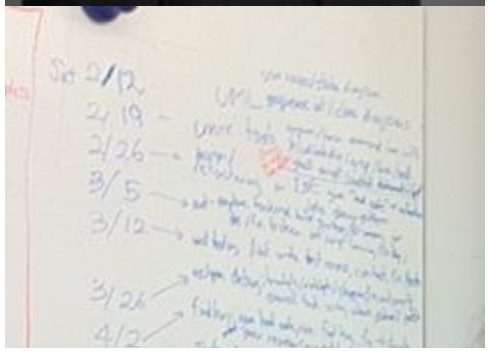
Postdoc, UC Davis



Assistant Professor,
University of Virginia

Assistant/Associate Professor,
Columbia University at New York

5th year Assistant Professor: Miryung Kim



Deadlines sketched on the white board

Feb 25 2013

David Notkin,
Jan 1, 1955-
Apr 22, 2013

Suman & Baishakhi
Farewell in Austin
Sept 17 2013

ICSM 2013
Submission Deadline
Apr 24, 2013
Conf: Sept 22-27 2013

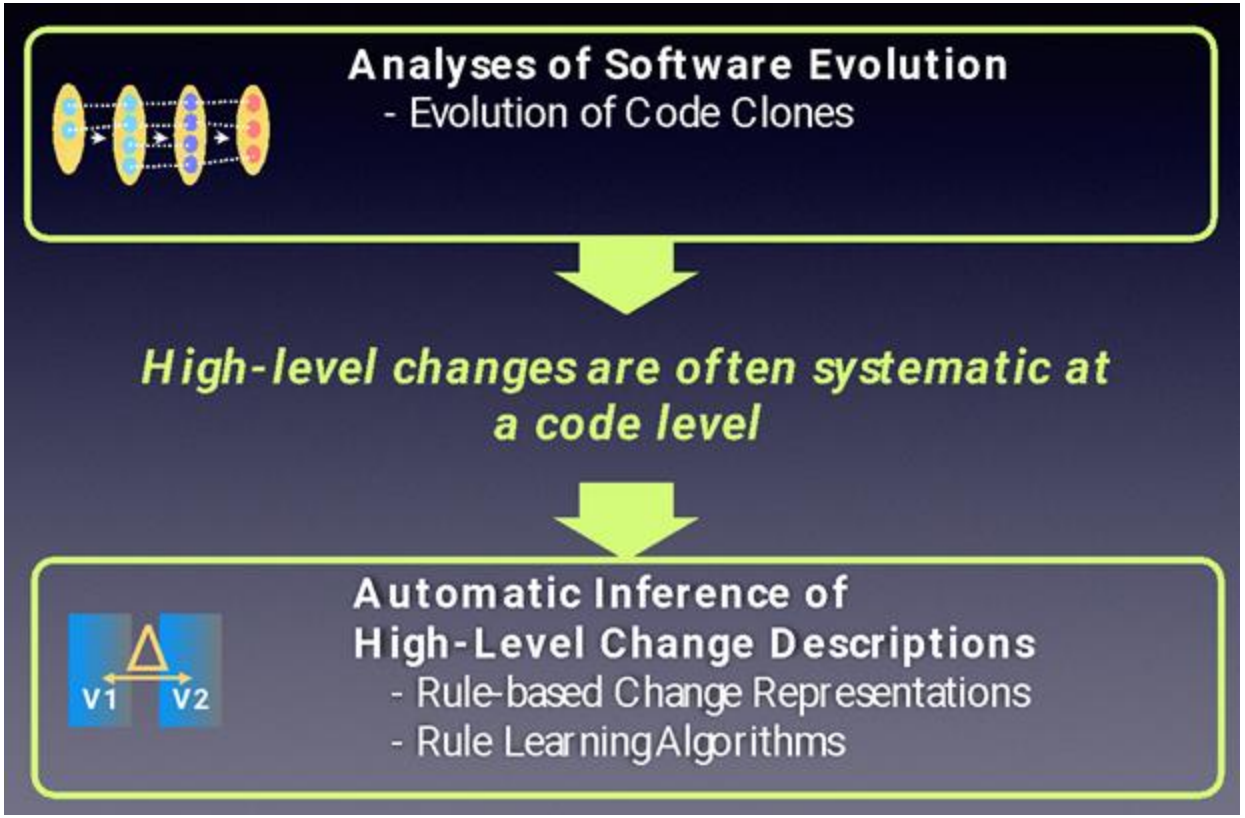
ICSM 2130 in Eindhoven, The Netherlands.

What ideas have motivated and inspired Android API Evolution?

Dagstuhl: Multiversion Program Analysis in 2005

Miryung's PhD @ University of Washington

My PhD Advisor:
David Notkin
(1 Jan 1955 – 22 Apr 2013)



Baishakhi's PhD @ UT Austin
Cross-system co-evolution

A Case Study of Cross-System Porting in Forked Projects

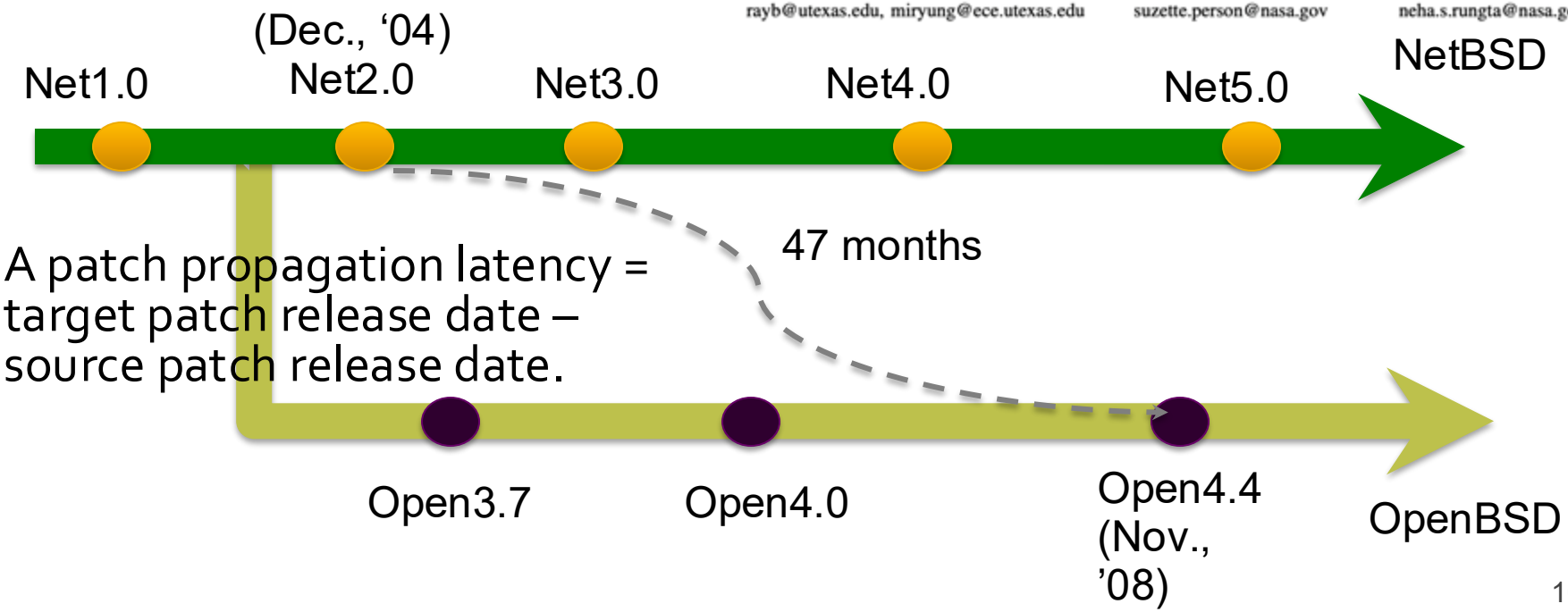
Baishakhi Ray and Miryung Kim
The University of Texas at Austin
Austin, TX USA
rayb@utexas.edu, miryung@ece.utexas.edu

Detecting and Characterizing Semantic Inconsistencies in Ported Code

Baishakhi Ray, Miryung Kim
The University of Texas at Austin
Austin, USA
rayb@utexas.edu, miryung@ece.utexas.edu

Suzette Person
NASA Langley Research Center
Hampton, USA
suzette.person@nasa.gov

Neha Rungta
NASA Ames Research Center
Mountain View, USA
neha.s.rungta@nasa.gov



Na Meng's PhD @ UT Austin
Automating Updates to Clones

Systematic Editing:
Generating Program Transformations from an Example

Na Meng Miryung Kim Kathryn S. McKinley
The University of Texas at Austin
mengna152173@gmail.com, miryung@ece.utexas.edu, mckinley@cs.utexas.edu

LASE: Locating and Applying Systematic Edits by Learning from Examples

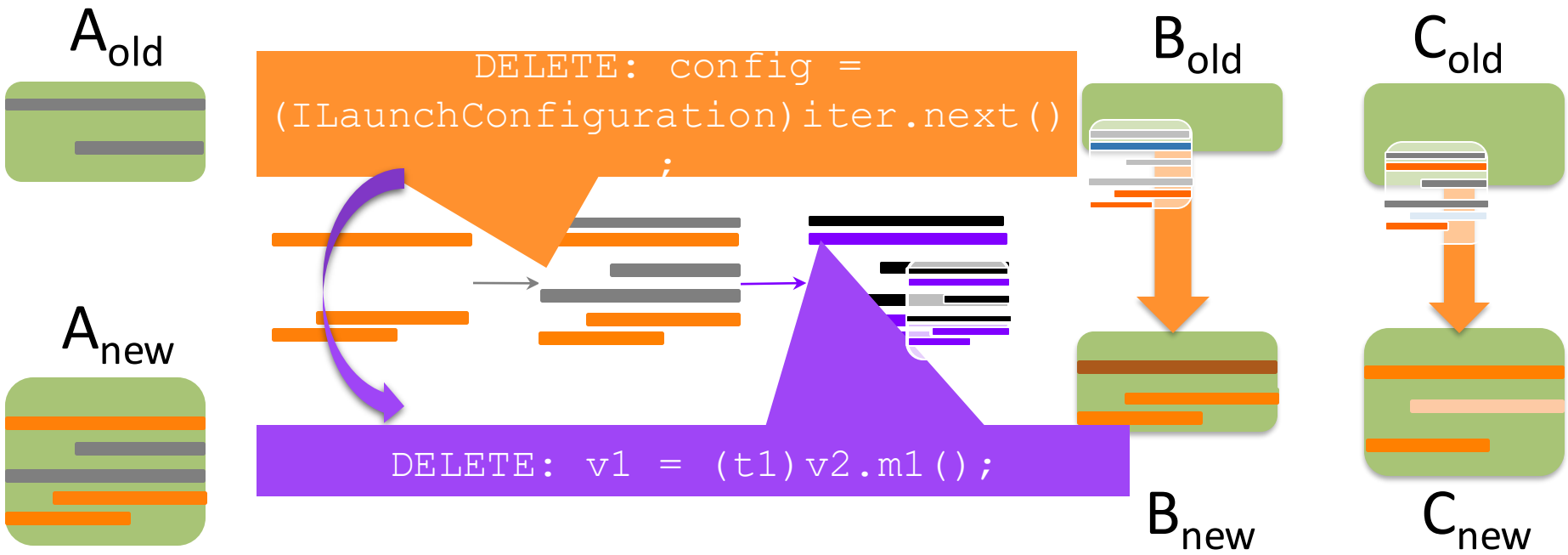
Na Meng* Miryung Kim* Kathryn S. McKinley*†
The University of Texas at Austin* Microsoft Research†
mengna09@cs.utexas.edu, miryung@ece.utexas.edu, mckinley@microsoft.com

Program differencing

Context extraction

Identifier & edit position abstraction

Abstract edit script application



Which ideas have influenced us to study evolution in software ecosystem?

Evolutionary Studies on Software

Analysis of the Linux Kernel Evolution Using Code Clone Coverage

Simone Livieri† Yoshiki Higo† Makoto Matsushita† Katsuro Inoue†

†Graduate School of Information Science and Technology, Osaka University
1-3 Machikaneyama, Toyonaka, Osaka 560-8531, Japan
E-mail: {simone, y-higo, matusita, inoue}@ist.osaka-u.ac.jp

Evolution in Open Source Software: A Case Study

Michael W. Godfrey and Qiang Tu
Software Architecture Group (SWAG)
Department of Computer Science, University of Waterloo
email: {migod,qtu}@swag.uwaterloo.ca

Understanding Collateral Evolution in Linux Device Drivers

Yoann Padioleau
OBASCO Group
Ecole des Mines de Nantes-INRIA, LINA
44307 Nantes cedex 3, France
Yoann.Padioleau@emn.fr

Julia L. Lawall
DIKU
University of Copenhagen
2100 Copenhagen Ø,
Denmark
julia@diku.dk

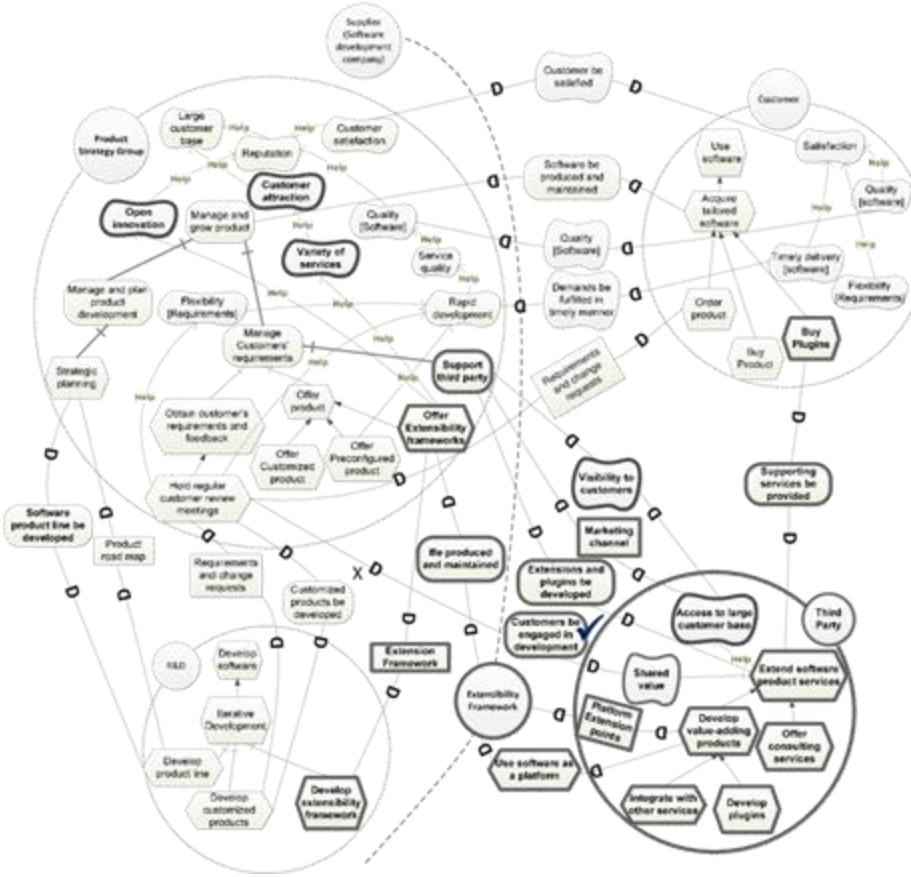
Gilles Muller
OBASCO Group
Ecole des Mines de Nantes-INRIA, LINA
44307 Nantes cedex 3, France
Gilles.Muller@emn.fr

Notion of Software Ecosystem

Lungu et al.'s definition—*a collection of software projects which are developed and co-evolve in the same environment.*

Robbes et al. found that 14% of deprecated methods produce non-trivial API change effects.

Studies in Smalltalk



Impact of API Refactoring on Client Applications

Dig and Johnson found that 80% of the code changes that break client-side code are API refactorings.

Xing and Stroulia studied Eclipse evolution history and found that 70% of structural changes are due to refactorings and existing IDEs lack support for complex refactoring.

Kim et al. found the number of bug fixes increases after API refactorings

Fast-paced Android Ecosystem Evolution

Excerpts from Original ICSM 2013 Talk

An Empirical Study of API Stability and Adoption in the Android Ecosystem

Tyler McDonnell, Baishakhi Ray, Miryung Kim
Department of Electrical and Computer Engineering
The University of Texas at Austin
Austin, TX, USA

An Empirical Study of API Stability and Adoption in the Android Ecosystem

Tyler McDonnell, Baishakhi Ray and Miryung Kim
The University of Texas at Austin

Motivation

- Despite the benefit of new or updated APIs, developers are often slow to adopt new APIs.
- API evolution and its associated ripple effect throughout software ecosystems are still under-studied.

Study Findings

- We study the **co-evolution** of Android APIs and applications using the github data
  - Android is evolving fast at a rate of 115 API updates per month.
  - 28% of API references in client apps are outdated with a median lagging time of 16 months.
  - API usage adaptation code is **defect prone** than other code.

Outline

- Motivation & Related Work
- Study Approach
- Research Questions and Results
- Limitations
- Conclusions

Study Approach

1. **Android API Version History**
2. **Mobile Apps in Github**
3. **Correlate changes in mobile apps with changes in Android OS**

API Version:
14

Release date: October 19, 2011

Class: android.widget.RemoteViews

void setRemoteAdapter(int, Intent)

Android API Version History

Client Code : Remote.java

Commit Date: January 26, 2012

import android.widget.RemoteViews;

```
int viewID = settings.getViewID();
Intent I = new Intent(this,
ActivityTwo.class);
```

setRemoteAdapter(viewID, I);

Client Source Code

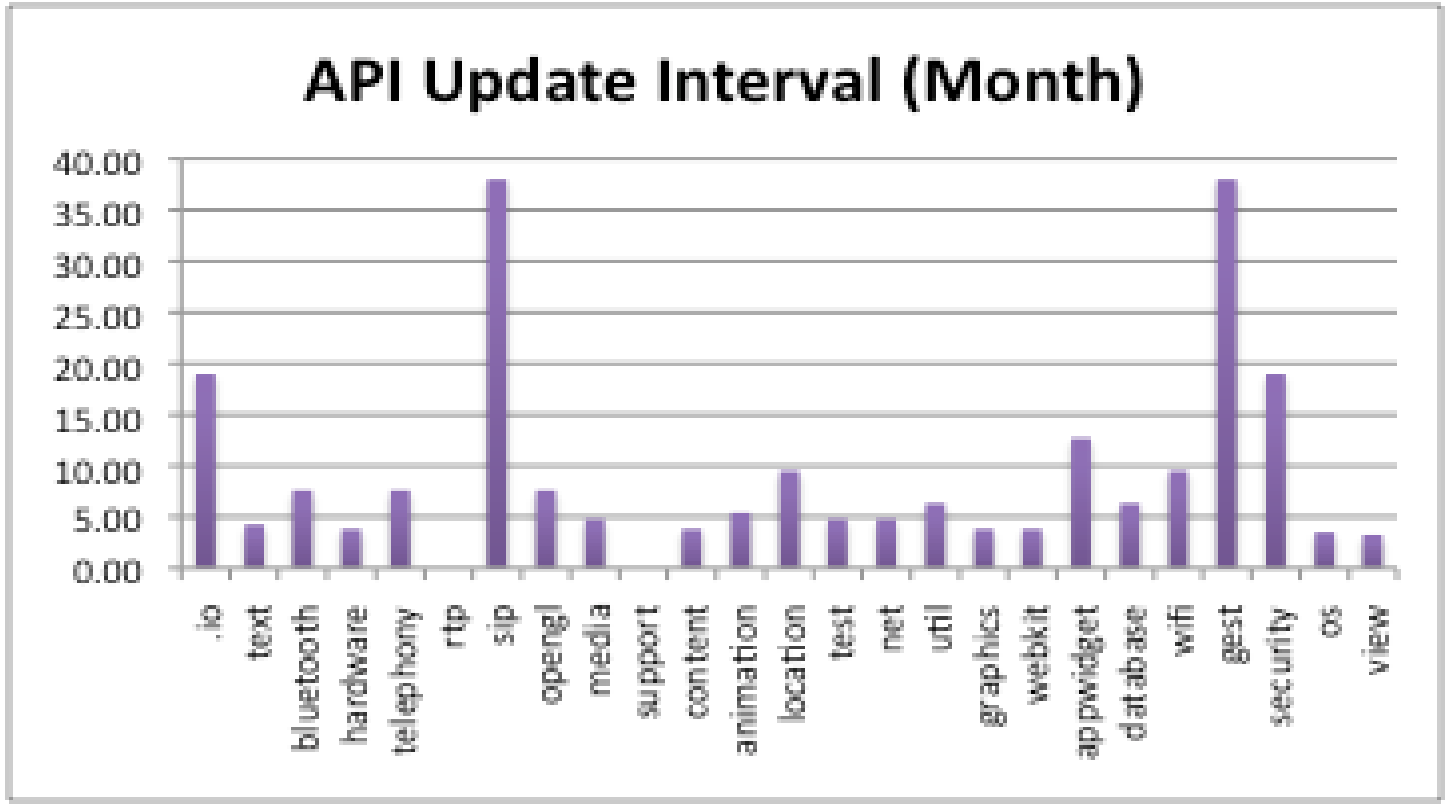
Android OS API Evolution Characteristics

- API Version 3 to 15

	Class	Method			Fields		
	Δ	Δ	+	-	Δ	+	-
Min	37	0	0	0	7	0	0
Max	269	416	98	9	619	205	0
Avg	149	158	37	2	179	32	0
Rate	42	44	11	<1	51	9	0

Android OS is evolving fast at the rate of 115 API updates per month.

Android API Evolution Characteristics



Hardware, user interface and web support are evolving fast.

Data Sets : Mobile Apps

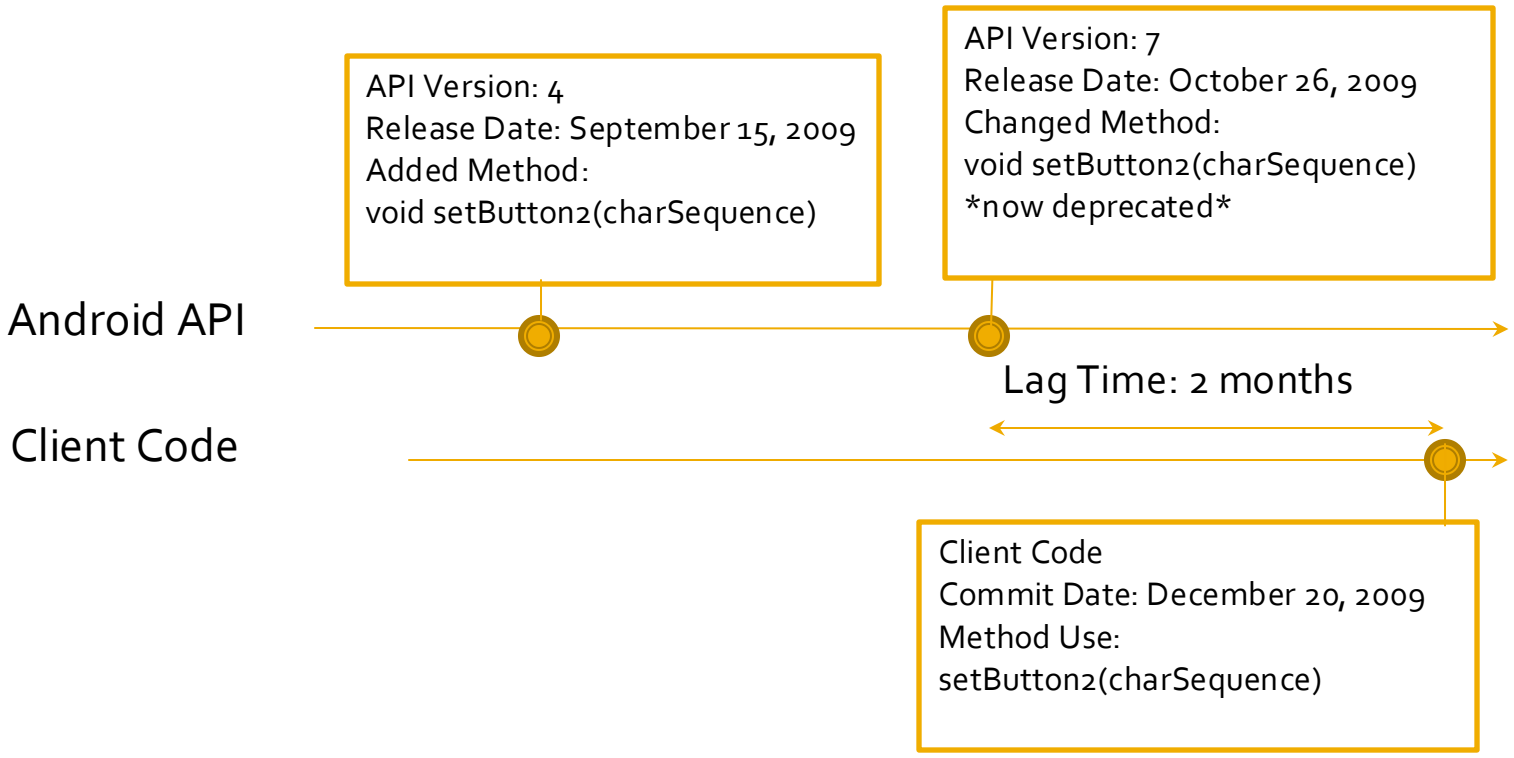
	Revision	LOC	Author	% Android Refs.
Congress Tracker	1359	13349	7	30%
Apollo M	9	15783	1	35%
Cyanogen	109	28972	20	24%
Google Analytic	926	52932	23	26%
LastFM	212	9771	7	16%
mp3Tunes	104	9608	1	22%
OneBusAway	497	51784	5	22%
ownCloud	665	25109	12	30%
RedPhone	116	21315	5	19%
XMBCremote	928	92893	24	22%

Around 25% of all method and field references in client code use Android APIs.

Research Questions

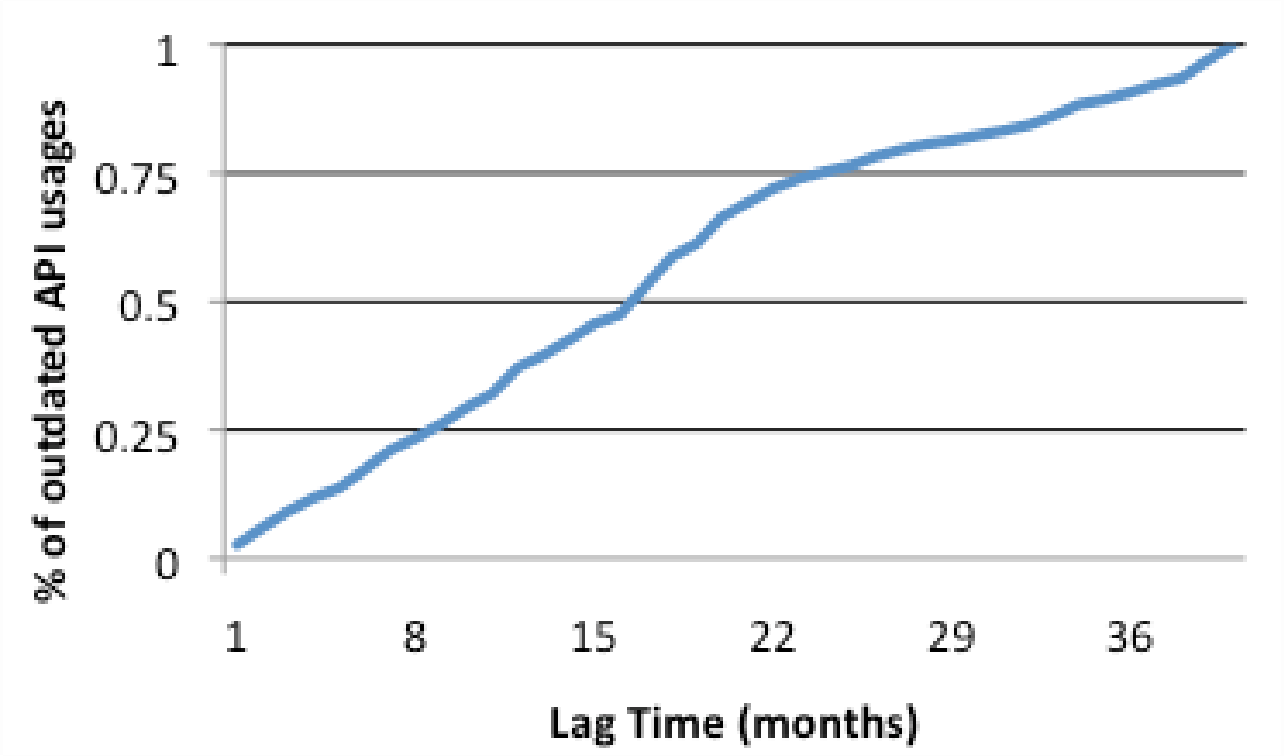
- Q1: What is the lag time between client code and the most recent Android API?
- Q2: How quickly do API changes propagate throughout client code?
- Q3: What is the relationship between API updates and bugs in clients?
- Q4: What is the relationship between API stability and adoption?

Q1: What is the lag time between client code and the most recent Android API?



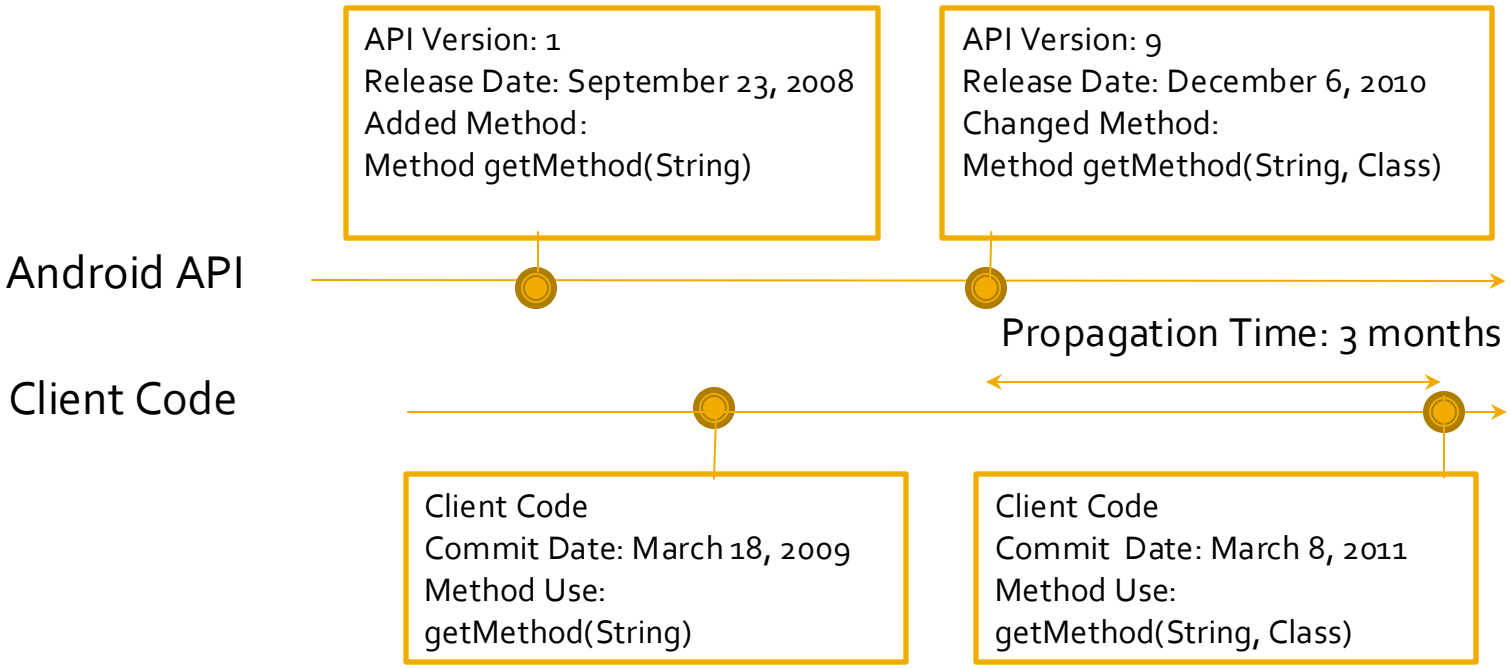
Lag time: the number of months elapsed between the release of the new version and the commit time of the outdated API usage

Q1: What is the lag time between client code and the most recent Android API?



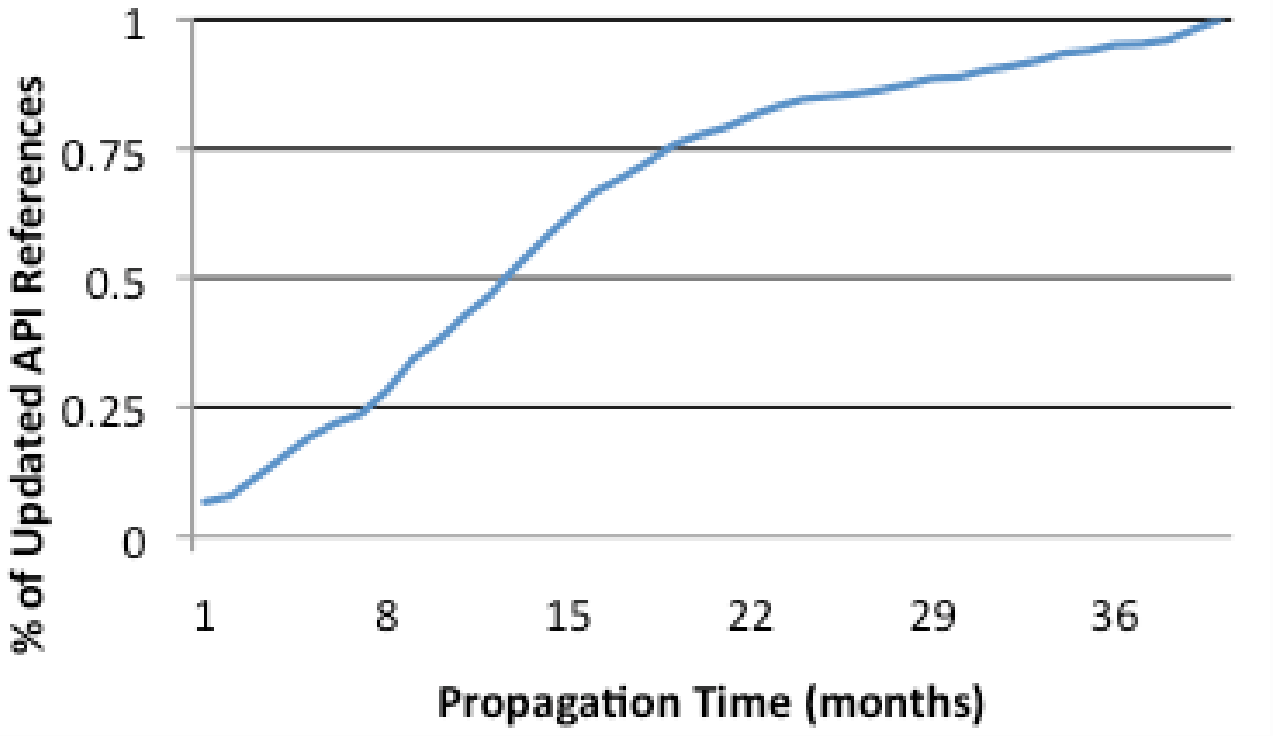
A half of all outdated API references are lagging behind by 16 months or more.

Q2: How quickly do API changes propagate throughout client code?



Propagation time: time difference in months between the API release and the timing of client adaptation

Q2: How quickly do API changes propagate throughout client code?



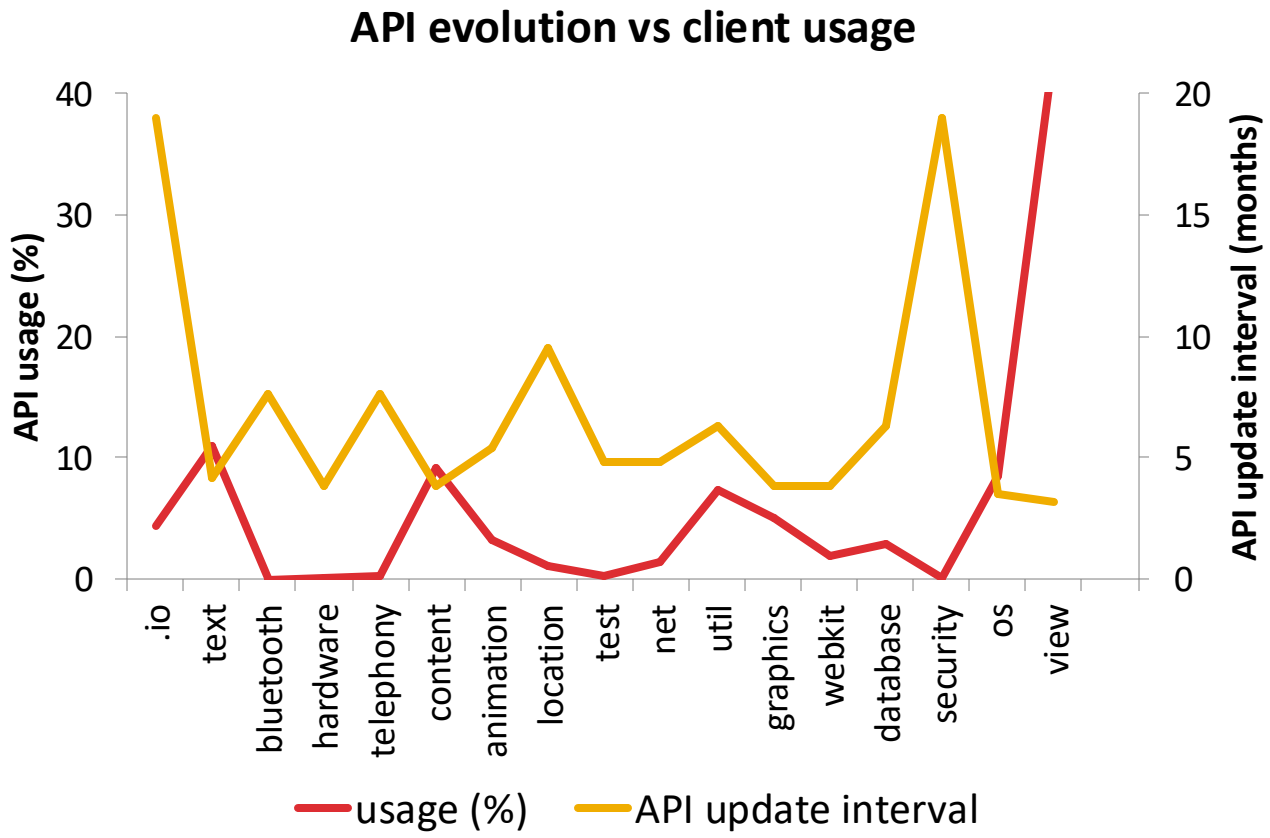
The median propagation time is 14 months. Outdated API usages upgrade to newer APIs but at a much slower pace than the API release rate.

Q3: What is the relationship between API updates and bugs?

	Spearman Correlation with bugs		
	CLOC	**API Update**	**Non API Update**
Congress Tracker	0.39	0.56	0.39
OneBusAway	0.26	0.46	0.25
RedPhone	0.23	0.24	0.23
XMBCremote	0.34	0.62	0.33
Google Analytic	0.36	0.54	0.31
ownCloud	0.43	0.55	0.42
Cyanogen	0.58	0.63	0.58
LastFM	0.42	0.37	0.43

Files with API usage adaptations are defect-prone in all applications except LastFM.

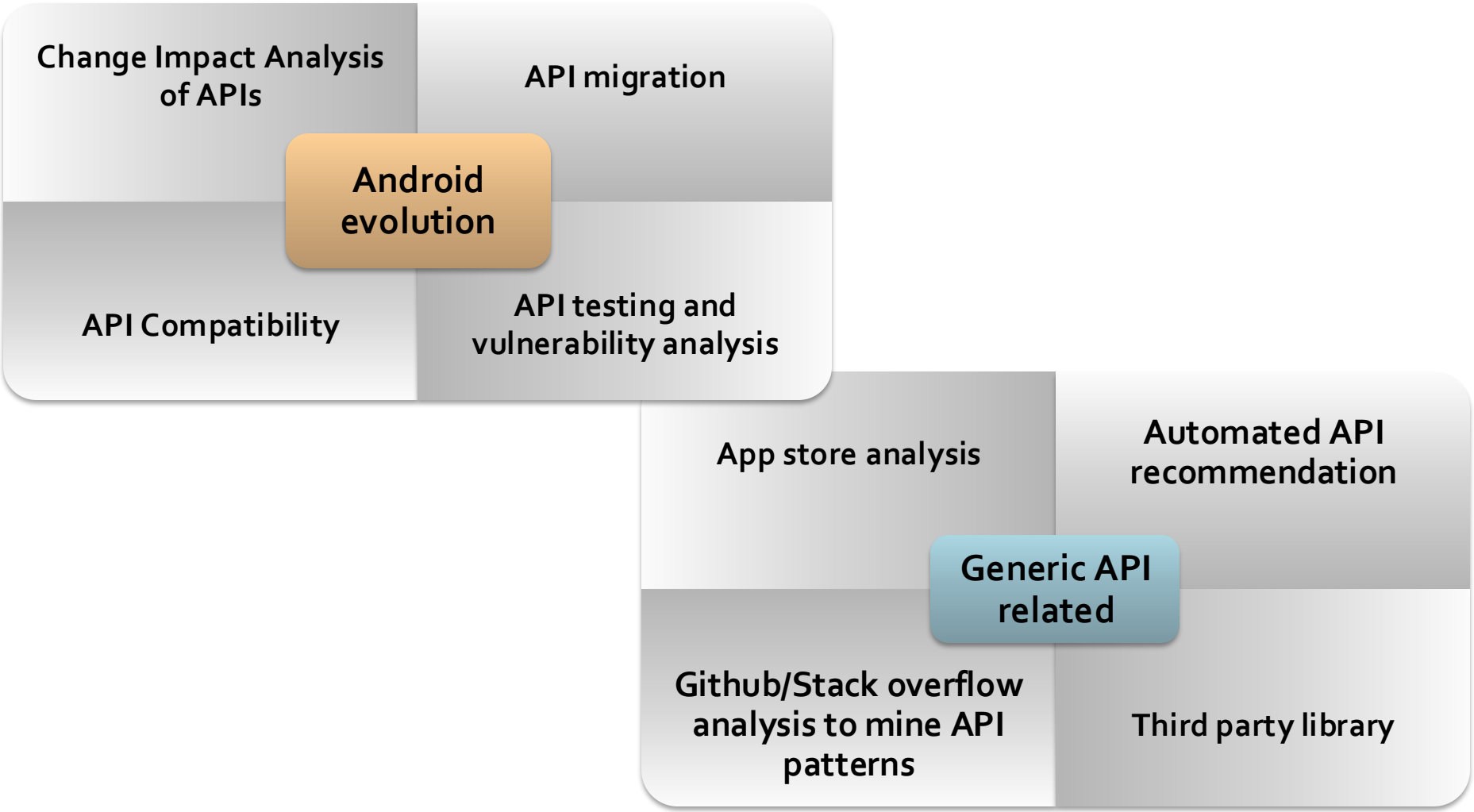
Q4: What is the relationship between API stability and usage?



Correlation between API usage (%) and API update interval: -0.47
Fast evolving APIs are used more by clients.

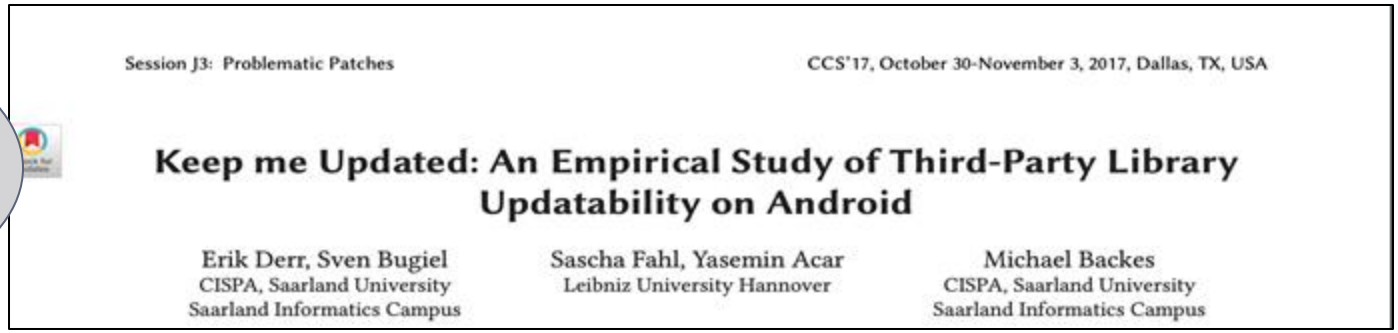
Reflections on the paper

SE community took this work to several directions



Some follow-up studies on Android API evolution

More empirical analysis on API fragmentation



Session J3: Problematic Patches CCS'17, October 30-November 3, 2017, Dallas, TX, USA

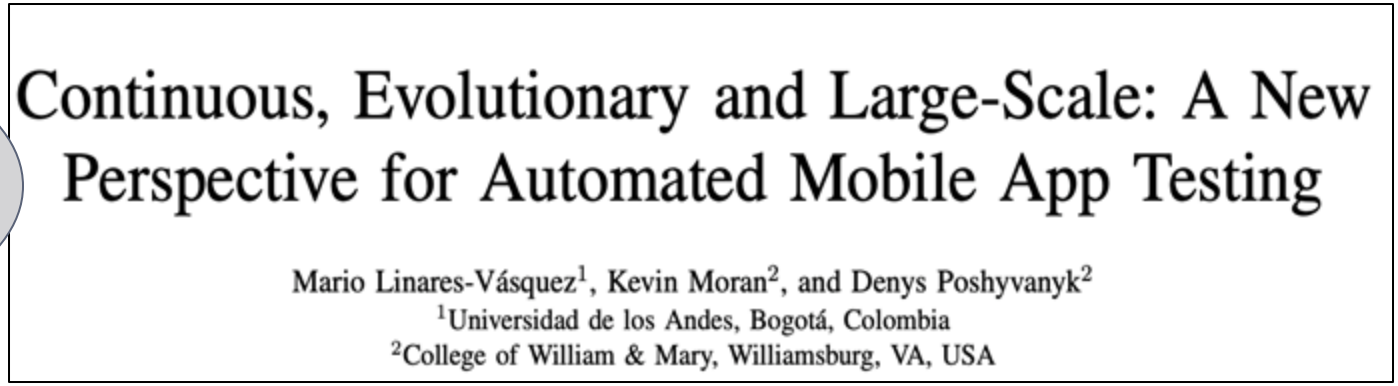
Keep me Updated: An Empirical Study of Third-Party Library Updatability on Android

Erik Derr, Sven Bugiel
CISPA, Saarland University
Saarland Informatics Campus

Sascha Fahl, Yasemin Acar
Leibniz University Hannover

Michael Backes
CISPA, Saarland University
Saarland Informatics Campus

Testing to handle API fragmentation



Continuous, Evolutionary and Large-Scale: A New Perspective for Automated Mobile App Testing

Mario Linares-Vásquez[1], Kevin Moran[2], and Denys Poshyvanyk[2]
[1]Universidad de los Andes, Bogotá, Colombia
[2]College of William & Mary, Williamsburg, VA, USA

Fixing API fragmentation

Taming Android Fragmentation: Characterizing and Detecting Compatibility Issues for Android Apps

Lili Wei, Yepang Liu, Shing-Chi Cheung
Department of Computer Science and Engineering
The Hong Kong University of Science and Technology, Hong Kong, China
{lweiae, andrewust, scc}@cse.ust.hk

Some follow-up studies on API Evolution in Ecosystem

Understanding the Test Automation Culture of App Developers

Pavneet Singh Kochhar[1], Ferdian Thung[1], Nachiappan Nagappan[2], Thomas Zimmermann[2], and David Lo[1]
[1]Singapore Management University
[2]Microsoft Research
{kochharps.2012,ferdiant.2013,davidlo}@smu.edu.sg,{nachin,tzimmer}@microsoft.com

Can Automated Pull Requests Encourage Software Developers to Upgrade Out-of-Date Dependencies?

Samim Mirhosseini
North Carolina State University
Raleigh, NC, USA
smirhos@ncsu.edu

Chris Parnin
North Carolina State University
Raleigh, NC, USA
cjparnin@ncsu.edu

When and How to Make Breaking Changes: Policies and Practices in 18 Open Source Software Ecosystems

CHRIS BOGART, CHRISTIAN KÄSTNER, and JAMES HERBSLEB,
Carnegie Mellon University, USA
FERDIAN THUNG, Singapore Management University, Singapore

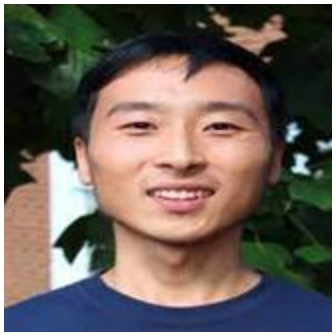
Thanks to Miryung's Students

From Right to Left
Baishakhi Ray (PhD 2013 ⇒Assistant Prof @ Columbia) *Detecting Recurring Changes and Errors*
Na Meng (PhD 2014 ⇒ Assistant Prof @ Virginia Tech) *Automating Recurring Changes & Clone Removal*
Tianyi Zhang (PhD 2019, Postdoc @ Harvard) *Leveraging Redundancy for Code Review, Testing, API Usage Mining*
Muhammad Ali Gulzar (PhD 2020 ⇒ Assistant Prof @ Virginia Tech) *Debugging and Testing for Big Data Analytics*
Myoungkyu Song (Postdoc 2015 ⇒ Assistant Prof @ Nebraska, Omaha) *Error Detection in Refactoring Edits*

Thanks to Baishakhi's Students

Saikat Chakraborty (PhD 2022 ⇒Senior Researcher @ Microsoft Research RiSE group)

Kexin Pei (PhD 2023/Postdoc ⇒Assistant Professor, U Chicago)



Yuchi Tian(PhD 2021 ⇒Facebook Research)

Thankful to ICSME "*Community*"

ICSM 2009 Edmonton
My first PC

ICSM 2011 Williamsburg
My first OC/ ERA co-chair

ICSM 2013 Einhoven

ICSM 2012 Riva del Garda

ICSME 2019
My first PC co-chair

ICSME

ICSME 2019-2022
SC membership

ICSM 2013 "Friendly" Memories

Meet, Learn, and Share Monday 9/23/13 Inbox ×

Emily Hill hillem@mail.montclair.edu via ece.utexas.edu Sun, Sep 22, 2013, 2:44 PM

to Lori, Lori, Anca.Ionita, U.Tikhonova, a.farcasi, anne.etien, aschwar2, bazelli, camargo, carolina.chiao, Dawı

Greetings!

Lori and I are excited to join everyone tomorrow for our Meet, Learn, and Share Session at ICSM 2013. Since we have this great opportunity with over a dozen ladies in software maintenance attending, **please try to think ahead about what you'd like to get out of the event** (e.g., what questions you'd like answered or partnerships you'd like to form). We will have an opportunity during the pre-dinner session to network and think about how best to address the questions & issues you bring with you.

The schedule:
4-5:45 pm Session in 2.03 Zwarte Doos (building #55 on the map http://www.tue.nl/fileadmin/content/universiteit/Over_de_universiteit/Route_Plattegrond/plattegrond/actuele_plattegrond/90--GIP0__20130815_.pdf)
5:45 pm walk to Usine, Lichttoren 6
6-8:30 pm Dinner

We look forward to seeing you at ICSM!
Emily & Lori

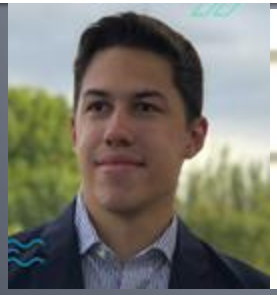


39th IEEE International Conference on Software Maintenance and Evolution (ICSME 2023) Bogota, Colombia

Most Influential Paper Award from ICSME 2013

"An Empirical Study of API Stability and Adoption in the Android Ecosystem"
by Tyler McDonnell, Baishakhi Ray, and Miryung Kim

David Notkin
(1955 –2013)



COLUMBIA UNIVERSITY

UCLA

Study Limitations and Future Work

- False negatives and positives in detecting API usage updates.
- Our method of detecting lagging methods does not take into account multi-version API support.
- We study the correlation between API usage, adoption, and bugs, but not causation.
- External validity beyond studied mobile apps from github.

Summary and Future Work

- We study on the co-evolution of Android OS and its clients.
  - 28% of Android references are lagging behind the latest version with a median lagging time of 16 months.
  - 22% of outdated API references upgrade to use newer APIs. The median propagation time is 14 months.
  - Fast-evolving APIs are used more.
  - API updates are more defect prone than other types of changes in client code.

Summary and Future Work

- Various stakeholders affect the process of API adoption in the software ecosystem. We need to identify factors affecting API adoption.
- Our goal is to automate required API adaptations in client applications using our example-based program transformation approach [Meng et al. 2013.]

Q1: What is the lag time between client code and the most recent Android API?

	Lagging API references(%)
Congress Tracker	18%
Apollo M	72%
Cyanogen	12%
Google Analytic	37%
LastFM	43%
mp3Tunes	5%
OneBusAway	3%
ownCloud	18%
RedPhone	43%
XMBCremote	15%
Average	28%

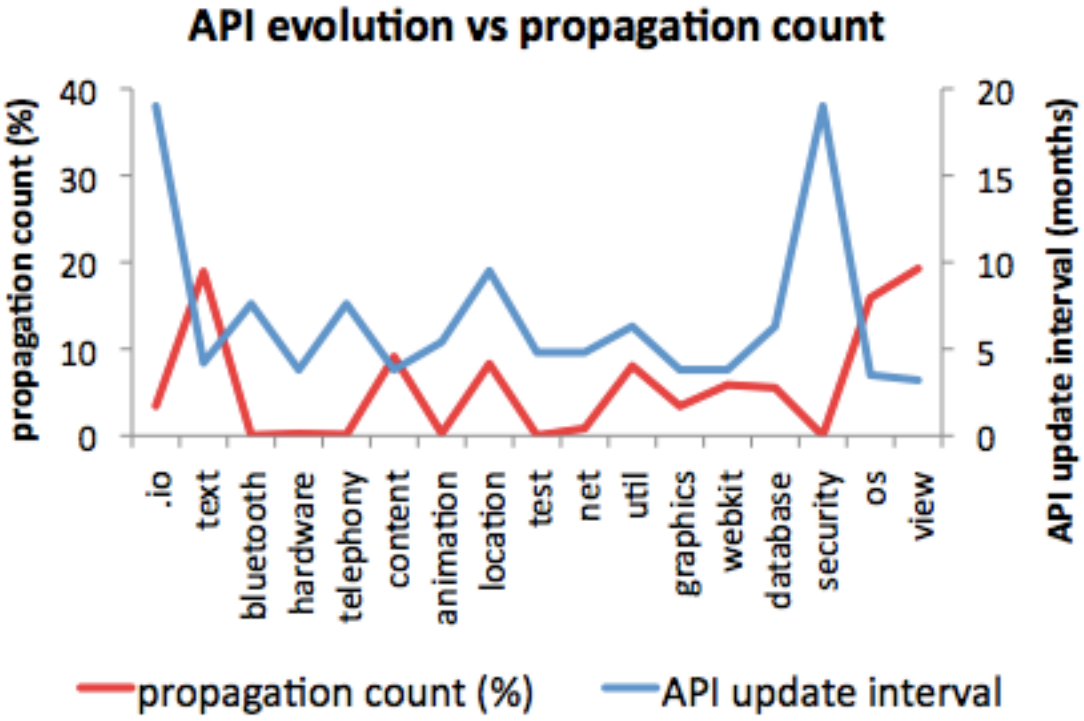
Q2: How quickly do API changes propagate throughout client code?

	% of outdated usages that were upgraded to use newer APIs
Congress Tracker	45%
Apollo Music	0%
Cyanogen	27%
Google Analytic	34%
LastFM	5%
mp3Tunes	0%
OneBusAway	12%
ownCloud	29%
RedPhone	39%
XMBCremote	33%
Average	22%

Q1: What is the lag time between client code and the most recent Android API?

	Lag (# Method)	Lagging API references(%)
Congress Tracker	216	18%
Apollo M	964	72%
Cyanogen	171	12%
Google Analytic	1409	37%
LastFM	181	43%
mp3Tunes	26	5%
OneBusAway	14	3%
ownCloud	489	18%
RedPhone	498	43%
XMBCremote	537	15%
Average	451	28%

Q5: What is the relationship between API stability and adoption?



Correlation between API usage (%) and API update interval: -0.47
Clients update to faster evolving APIs more frequently.

Client Applications	Android API	Total API	% Android API	Unique Android API
Apollo Music	1332	3820	35%	155
Congress Tracker	1007	3396	30%	82
Cyanogen	1439	5992	24%	144
Google A	3164	12145	26%	336
LastFM	371	2122	16%	64
mp3Tunes	510	2275	22%	101
OneBusAway	2416	10932	22%	297
ownCloud	1838	6132	30%	194
RedPhone	830	4303	19%	160
XMBCremote	3209	14626	22%	275

Figure 3. Degree of Android API dependence of client code

Related Work

- Many techniques have been proposed to ease API update and version incompatibilities
- API evolution and its associated ripple effect through ecosystems are under-studied
  - Robbes et al. study how API deprecation affects client applications in Smalltalk.
- Kim et al. study the relationship between API refactoring and bugs in libraries.