Inferring Likely Mappings Between APIs

Amruta Gokhale - amrutag@cs.rutgers.edu
Vinod Ganapathy - vinodg@cs.rutgers.edu
Yogesh Padmanaban - ypadmana@cs.rutgers.edu

Department of Computer Science
Rutgers University

ICSE'13, the 35th International Conference on Software Engineering
San Francisco, California, May 2013

“There is an app for that!”

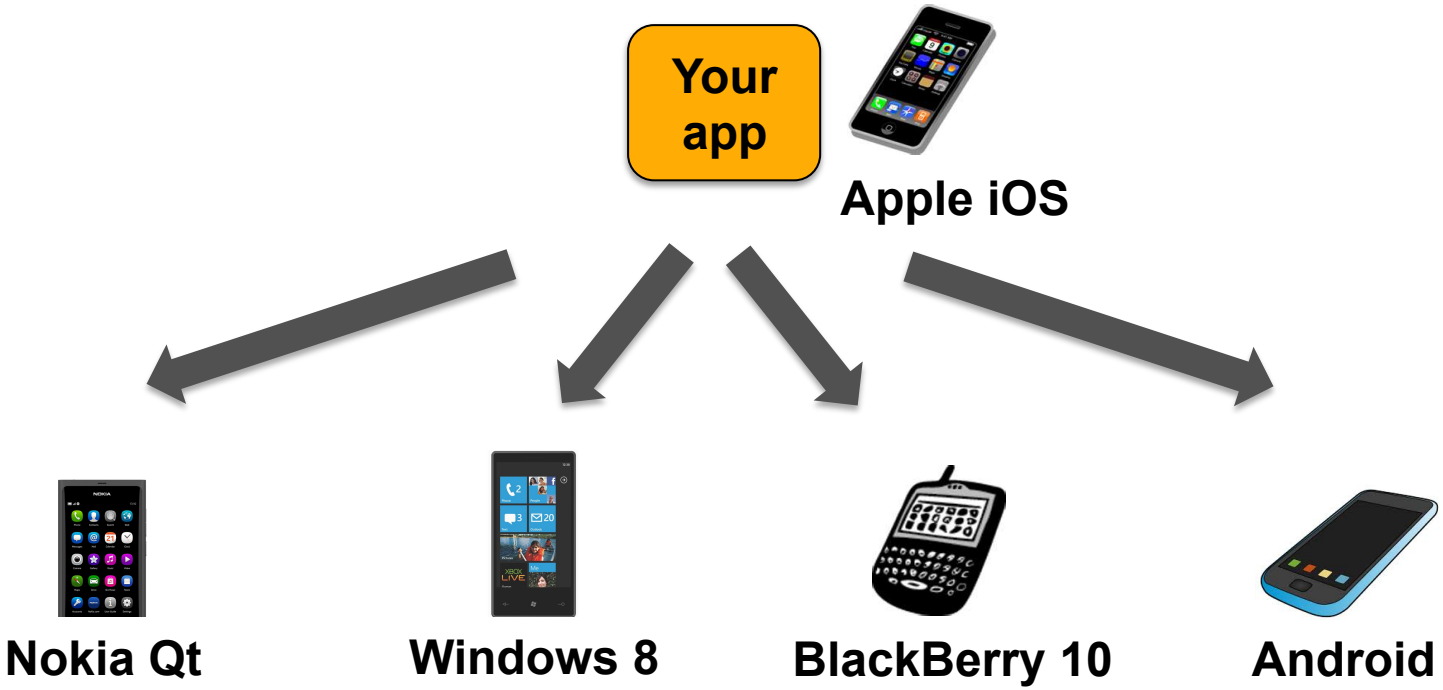
Approximately 1 million apps available on Google Play and Apple app stores

Developing a new mobile app

Your app

Apple iOS

Challenge: Porting across platforms



"Publishing an app in an enterprise today means supporting around four platforms to cover all of your employees."

Information Week Jan 2013 "Tips for Multi-Platform App Development"

Challenges in porting apps

Every mobile platform exposes its own programming API

- Different SDKs for app development
- Different programming languages
- Different development environments
- Different debugging aids

Platform	Language	Development Tools
Android	Java	Eclipse
Apple iOS	Objective C	XCode
Windows 8	C#	Visual Studio

Each API has different features

Android class	Android method name
`android.graphics`	`void` **`drawRect`**`(Rect r, Paint paint)` Draws the specified Rect using specified Paint
`android.graphics`	`bool` **`contains`**`(int x, int y)` Returns true if (x,y) is inside the rectangle.

Android App

Android phone

Each API has different features

iOS class	iOS method name
`CGGeometry`	`CGRect` **`CGRectMake`** `(CGFloat x, y, width, height)` Returns a rectangle with the specified coordinate and size values.
`CGGeometry`	`bool` **`CGRectContainsPoint`** `(CGRect rect, CGPoint point)` Returns whether a rectangle contains a specified point.

iPhone App

iPhone

But API features often map to each other

API mapping databases store target API methods that map to a source API method

Android class name	Android method name
android.graphics	void **drawRect**(Rect r, Paint paint)
android.graphics	bool **contains**(int x, int y)

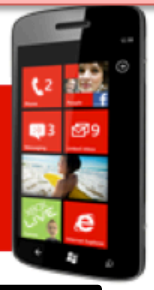
iOS class name	iOS method name
CGGeometry	CGRect **CGRectMake** (CGFloat x, y, width, height)
CGGeometry	bool **CGRectContainsPoint** (CGRect rect, CGPoint point)

Microsoft's API mapping database

windowsphone.interoperabilitybridges.com/porting

Windows Phone

Windows Phone
Interoperability



HOME | PORTING | DOWNLOAD | DISCUSSIONS | ABOUT

Windows Phone Guides for Android, iPhone& Qt Developers

Windows Phone API Mapping

Got a porting question?

Assistance for Android, iPhone & Qt developers

Help forums for developers

Windows Phone 7 development to your skill set, while leveraging what you have already learned building iPhone and/or android:

- Guide for Android Developers
- Guide for iOS/iPhone Developers
- Guide for Nokia Symbian Qt Developers

The API mapping tool now includes Nokia Symbian Qt, in addition to Android (added in June 2011) and iOS (released in April 2011). The API Mapping tool helps developers find their way around the API when they discover the Windows Phone platform.

Go to the API Mapping tool:

- Android to WP7
- iOS/iPhone to WP7
- Qt to WP7

Nokia's API mapping database

developer.nokia.com/Develop/Porting/

NOKIA Developer

Log in | Join | English

Design | Develop | Distribute | Devices | Resources | Community

Search with Bing

Share

Porting

Overview | Documentation | Code examples | Videos

Extend your outreach by porting to Nokia platforms

Porting to Qt

API Mapping

Porting to Series 40

Overview

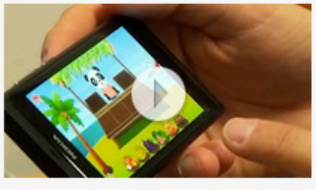
Porting to Windows Phone

API Mapping

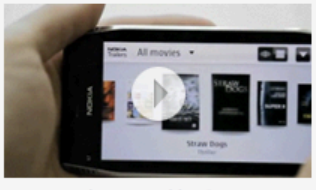
Use cases of apps ported to Qt
Developer stories about porting

App spotlights

SpaceBlok
This video demonstrates the Space Blok showcase example app, which has been ported from Windows Phone to Qt for the Nokia N9 and Symbian phones.



Porting to Qt: Lola's Fruit Shop Sudoku
Conor Lennon from Symbio talks about the experience of porting an iOS game, from games studio BeiZ, to Qt. Conor highlights how Qt Quick was fast and



LiveJournal: Powered by Qt
Lina Udovenko from SUP Media talks about their LiveJournal Qt/QML app. Mikhail Nosov from Teleca explains that the app was first developed for Symbian

Latest News

Playing the field: Updated tools to help you port your apps between platforms
By Jason Black 25 October 2011

Learn Windows Phone faster: Qt to WP API Mapping and Guide available
By JC Cimetiere 21 September 2011

Creating API mapping databases

Mapping databases are populated manually by domain experts

- Microsoft and Nokia's app interoperability Web sites shown earlier
- ***Painstaking***, ***error-prone*** and ***expensive***
  - Involves reading and understanding API docs
  - Or crowdsourcing, asking on help forums, *etc.*
  - Hard to evolve API mapping databases as the corresponding APIs evolve

Our contribution

A methodology to automatically create API mapping databases

- Prototyped in a tool called **Rosetta**
  - Infers mapping between Java2 Mobile Edition graphics API and Android graphics API
- Leverages a novel probabilistic inference approach to identify *likely* API mappings

What are API mappings?

Source API

Target API

`android.graphics.drawRect`

`CGGeometry.CGRectMake`

API Mapping

=

Similar?

Yes → API mapping

How to find API mappings?

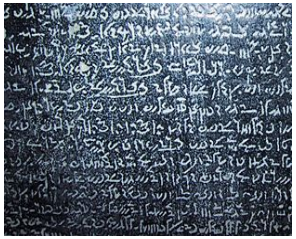
Source API

Target API

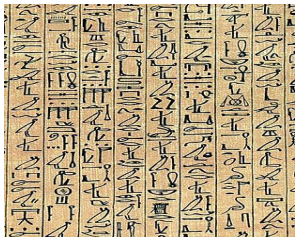
- Consider apps with similar functionality on source and target platforms
- Respective developers must have exercised knowledge of source/target APIs
- **Idea:** Harvest knowledge by tracing apps

An analogy: The Rosetta Stone

- Created circa 196BC
- Found circa 1799AD
- **Has same decree in 3 different scripts**



Egyptian



Ancient Egyptian

Ancient Greek

Known

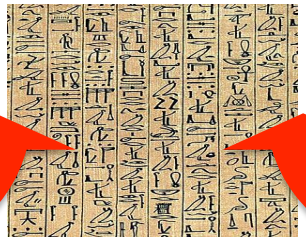
Unknown

Known

An analogy: The Rosetta Stone

- Created circa 196BC
- Found circa 1799AD
- **Has same decree in 3 different scripts**

Egyptian



Ancient Egyptian

Ancient Greek

Known

Known

Known

Rosetta of ICSE 2013

Source API

Target API

Familiar

Unfamiliar

App corpus

=

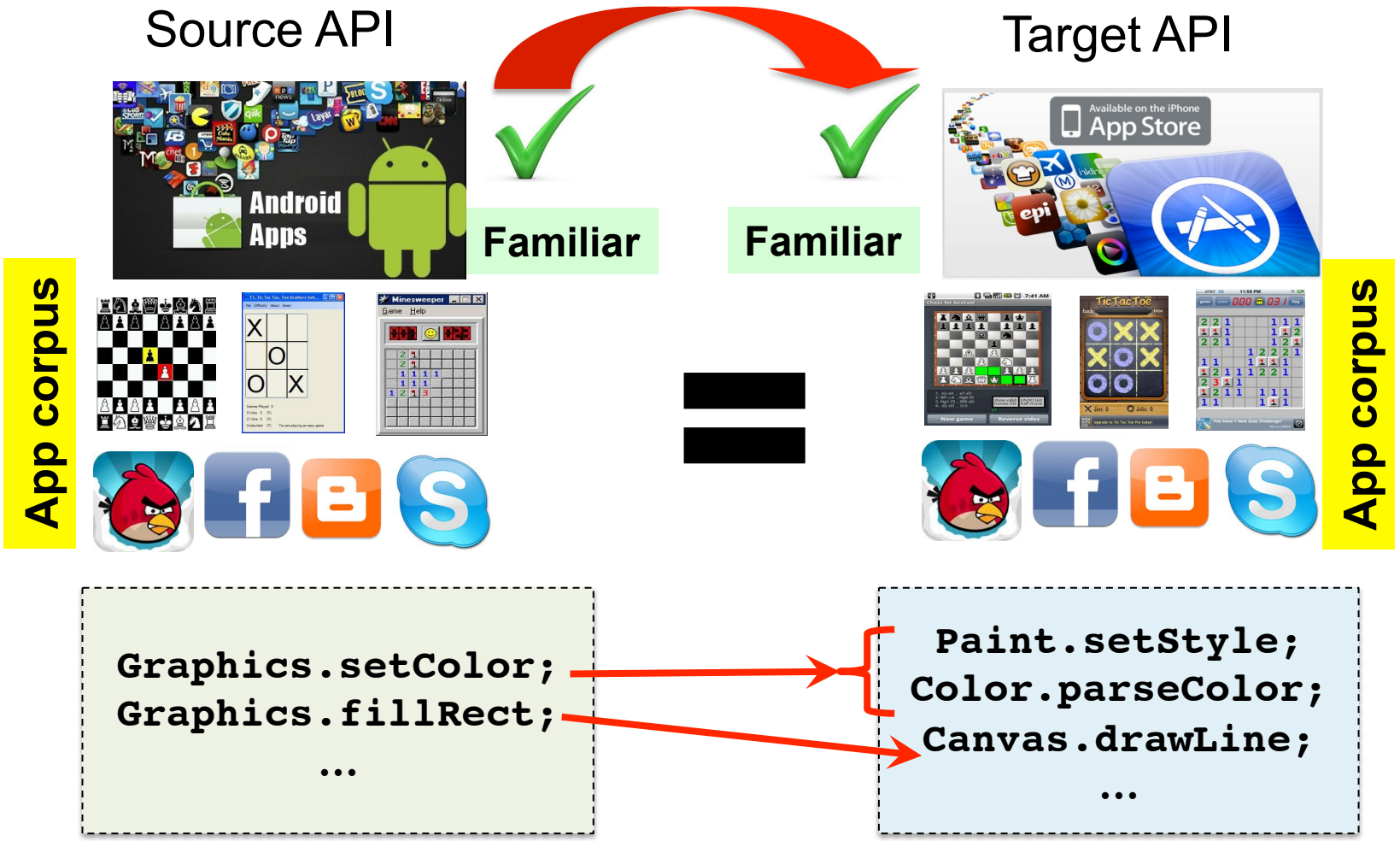
App corpus

```
Graphics.setColor;
Graphics.fillRect;
...
```

```
Paint.setStyle;
Color.parseColor;
Canvas.drawLine;
...
```

Traces under similar workloads

Rosetta of ICSE 2013



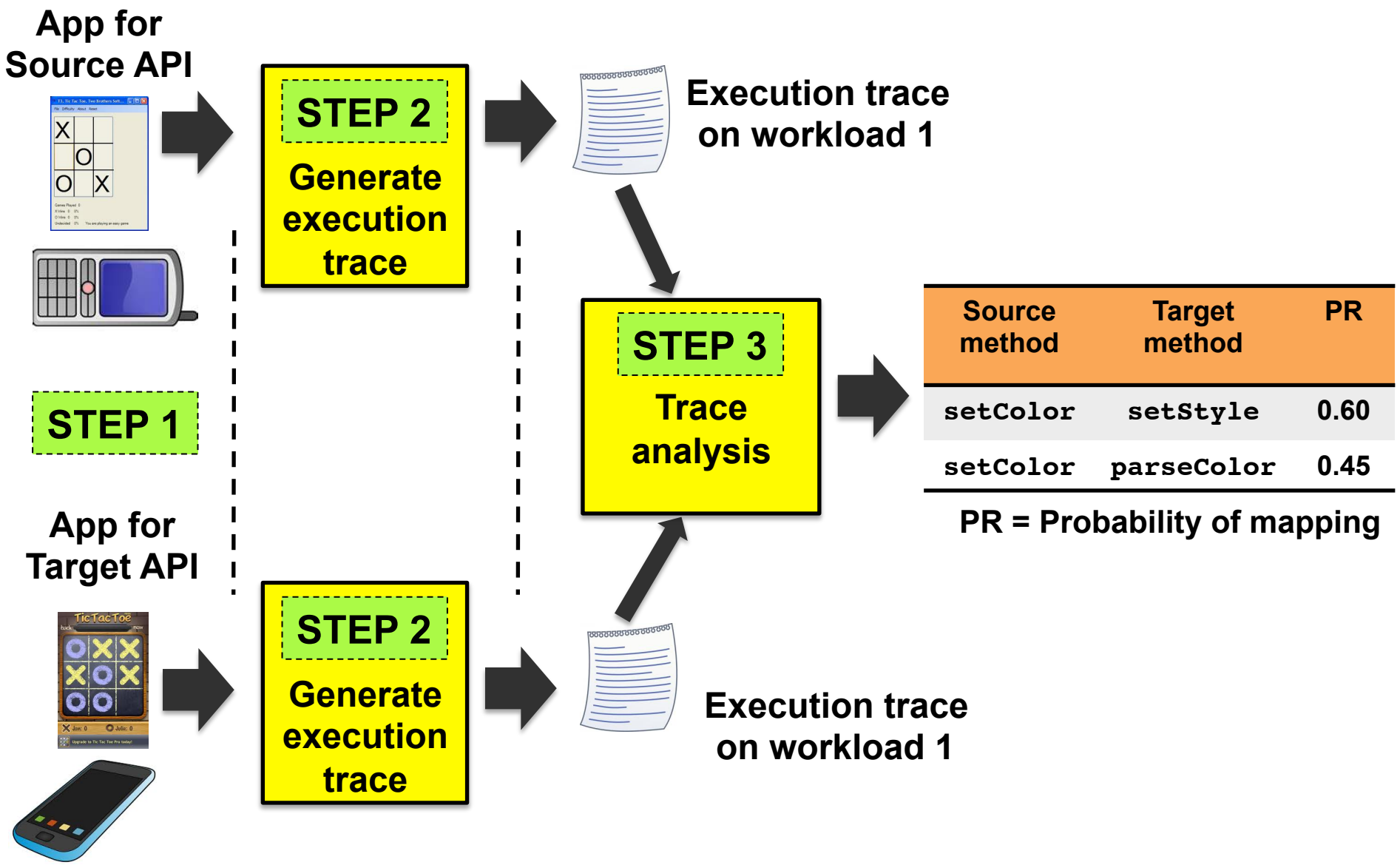

```
Graphics.setColor;
Graphics.fillRect;
...
```

```
Paint.setStyle;
Color.parseColor;
Canvas.drawLine;
...
```

Why ***likely*** API mappings?

- Rosetta analyzes runtime traces of similar apps built for the source and target APIs
- Trace analysis is heuristic
  - Considers various artifacts of trace structure
- The resulting API mappings are therefore not provably semantically equivalent
- Instead, we associate a ***probability*** with each inferred API mapping
  - Signifies likelihood of mapping being true

Workflow of Rosetta



Source method	Target method	PR
`setColor`	`setStyle`	0.60
`setColor`	`parseColor`	0.45

PR = Probability of mapping

Combining results from multiple apps

Source method	Target method	PR
setColor	setStyle	0.60
setColor	parseColor	0.45

Mappings from App/Workload 1

.....

.....

.....

Source method	Target method	PR
setColor	setStyle	0.90
setColor	parseColor	0.40

Mappings from App/Workload N

Source method	Target method	PR
setColor	setStyle	0.85
setColor	parseColor	0.42

Final Set of Mappings

Rosetta combines the mappings inferred using multiple apps and workloads pairs.

See paper for details of **STEP 4**

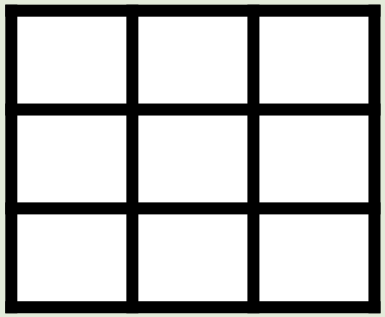
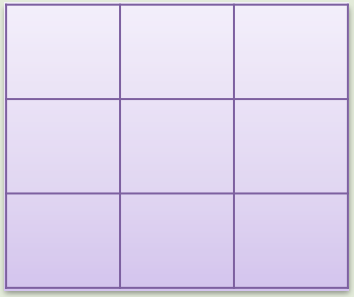
STEP 1 Collecting app pairs

Find functionally similar app pairs for source and target APIs (App_S, App_T)

App pair	Source Platform	Target Platform
($Chess_S$, $Chess_T$)		
($TicTacToe_S$, $TicTacToe_T$)		
($MSweeper_S$, $MSweeper_T$)		

STEP 2 Obtain trace pairs

Execute App_S and App_T on corresponding platforms with similar workloads

Workload	$TicTacToe_S$	$TicTacToe_T$
(1) Open game (2) Close game	 Source trace 1	 Target trace 1

STEP 2

Multiple traces for App_S, App_T

Controlled Execution

Step 1	…
Step 2	…

Execution trace

Workload name	Actions performed	
1. Basic	Step 1	Open TicTacToe game
	Step 2	Close TicTacToe game
2. Click on square	Step 1	Open TicTacToe game
	Step 2	Click on a square
	Step 3	Close TicTacToe game
3. Click on text menu	Step 1	Open TicTacToe game
	Step 2	Click on text menu option
	Step 3	Close TicTacToe game

STEP 3 Trace analysis

- **Intuition:** Traces under similar workloads will contain API methods that map to each other
- Analyze trace structure to infer these mappings
- Our algorithm uses four attributes of trace structure:

1. Relative frequency of method calls
2. Context of invoked methods
3. Relative positions of method calls
4. Method names

Used in the following example

STEP 3

Example of trace analysis

```
1. Graphics.setColor;
2. Graphics.fillRect;
3. Graphics.setColor;
4. Graphics.fillRect;
5. Graphics.fillRect;
6. Graphics.fillRect;
…
```

Source trace 1

```
1. Paint.setStyle;
2. Color.parseColor;
3. Canvas.drawLine;
4. Paint.setStyle;
5. Color.parseColor;
6. Canvas.drawLine;
7. Canvas.drawLine;
8. Canvas.drawLine;
…
```

Target trace 1

API methods or method sequences that map to each other must appear with similar relative frequency

STEP 3

Example of trace analysis

```
1. Graphics.setColor;
2. Graphics.fillRect;
3. Graphics.setColor;
4. Graphics.fillRect;
5. Graphics.fillRect;
6. Graphics.fillRect;
…
```

Source trace 1

```
1. Paint.setStyle;
2. Color.parseColor;
3. Canvas.drawLine;
4. Paint.setStyle;
5. Color.parseColor;
6. Canvas.drawLine;
7. Canvas.drawLine;
8. Canvas.drawLine;
…
```

Target trace 1

Source API method	Raw Count	Relative frequency
`setColor`	2	0.33
`fillRect`	4	0.67

Target API method	Raw Count	Relative frequency
`setStyle`	2	0.25
`parseColor`	2	0.25
`drawLine`	4	0.50

STEP 3

Example of trace analysis

Source API method	Raw Count	Relative frequency
`setColor`	2	0.33
`fillRect`	4	0.67

Target API method	Raw Count	Relative frequency
`setStyle`	2	0.25
`parseColor`	2	0.25
`drawLine`	4	0.50

Can infer the following

Source method	Target method	Probability of mapping
`fillRect`	`setStyle`	
`fillRect`	`parseColor`	
`fillRect`	`drawLine`	?
`setColor`	`setStyle`	?
`setColor`	`parseColor`	?
`setColor`	`drawLine`	?

Large difference in relative frequencies

Likely mappings, but need more evidence

STEP 3

Example of trace analysis

```
1. Graphics.setColor;
2. Graphics.fillRect;
…
```

Source trace 2

```
1. Canvas.drawLine;
2. Paint.setStyle;
3. Color.parseColor;
…
```

Target trace 2

Results from first trace

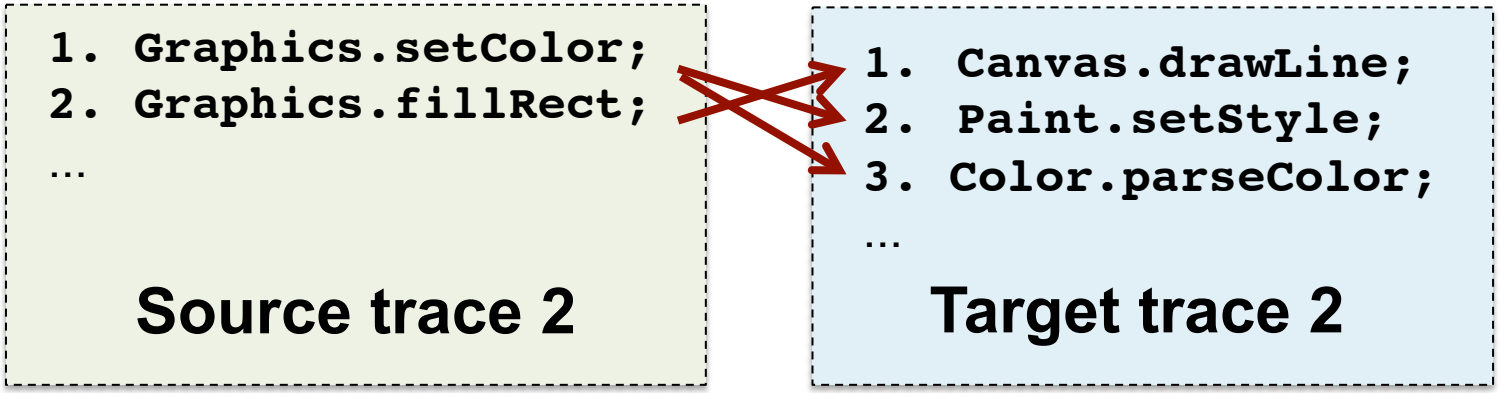
Source method	Target method	Probability of mapping
fillRect	setStyle	
fillRect	parseColor	
fillRect	drawLine	?
setColor	setStyle	?
setColor	parseColor	?
setColor	drawLine	?

Using ordering of method calls:

implies

STEP 3

Example of trace analysis



In fact, we can deduce that **one** of these holds:

Source method	Target method	Probability of mapping
fillRect	drawLine	👍
setColor	setStyle	👎
setColor	parseColor	👎

Source method	Target method	Probability of mapping
fillRect	drawLine	👎
setColor	setStyle	👍
setColor	parseColor	👍

STEP 3

Using belief propagation

- This is an example of **belief propagation**
  - *If* **`setColor`** likely maps to **`setStyle`**, *then* **`fillRect`** likely does **not** map to **`drawLine`**
  - More traces will lead to more such deductions
- We use *factor graphs* for belief propagation

Source method	Target method	Probability of mapping
`fillRect`	`drawLine`	
`setColor`	`setStyle`	
`setColor`	`parseColor`	

Source method	Target method	Probability of mapping
`fillRect`	`drawLine`	
`setColor`	`setStyle`	
`setColor`	`parseColor`	

STEP 3

Full trace analysis algorithm

- Combines multiple trace attributes using factor graphs to infer likely mappings
- Infers likely mappings between API method sequences as well
  - Does **`setColor`** map to **`setStyle`→`parseColor`**?
- Associates numerical values with  and  based upon likelihood of mapping
- Detailed algorithm presented in the paper

Output of Rosetta

Source API method	Target API method	Probability
`Graphics.setColor`	`Color.parseColor`	0.8
`Graphics.setColor`	`Paint.setStyle`	0.75
`Graphics.setColor`	`Canvas.drawLine`	0.43
`Graphics.fillRect`	`Canvas.drawLine`	0.87
`Graphics.fillRect`	`Color.parseColor`	0.62
`Graphics.fillRect`	`Paint.setStyle`	0.45

- For each source API method, a ranked list of target API methods that are likely mappings
- Akin to a search engine for API mappings

Rosetta

The Rosetta prototype

Source platform	Target platform
 JavaME Graphics API	 Android Graphics API
281 methods	**3837** methods

- Why JavaME and Android?
  - Both platforms use the same source language for app development
  - Eased development of initial prototype
  - But having the same source language is not a fundamental requirement for Rosetta
- Uses instrumentation to enable API tracing
- Uses Bayes Net Toolbox for factor graphs

Evaluation

- Dataset consists of 21 app pairs for JavaME and Android, mainly board games
  - Controlled experiments easy with board games
- Traced apps manually in similar ways, and analyzed traces to infer API mappings
- Evaluated validity of resulting API mappings by consulting API documentation

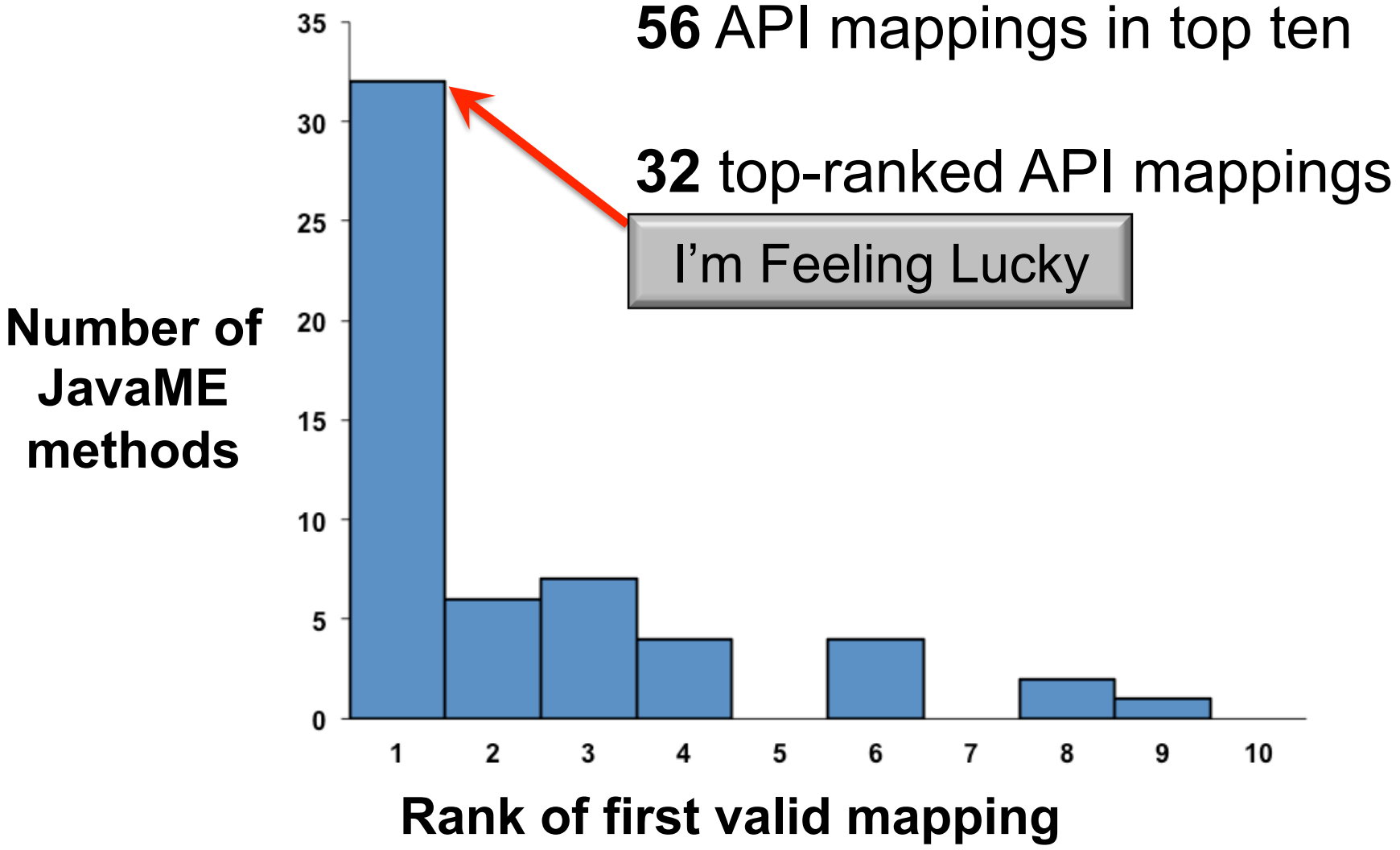
Examples of inferred API mappings

JavaME graphics API method		Android graphics API method	Rank
`Graphics.clipRect`	➡	`Canvas.clipRect`	1
`Graphics.drawRect`	➡	`Canvas.drawRect`	1
`Graphics.drawRect`	➡	`Canvas.drawLines`	7
`Graphics.drawChar`	➡	`Paint.setColor;` `Canvas.drawText`	2
`Alert.setString`	➡	`TextView.setText`	4

Highlights of results

Metric	Count
1. Distinct number of JavaME methods observed in traces	**80**
2. Total number of JavaME methods for which valid Android API mapping was found with **rank ≤ 10**	**56 (70%)**
3. Total number of JavaME methods for which valid Android API mapping was found with **rank = 1** I'm Feeling Lucky	**32 (40%)**

Distribution of first valid mapping



More results in the paper

- Rank distribution of **all** valid mappings, not just the first valid one
- Impact of various trace attributes to overall ranking of inferred mappings
- Cross-validation of the results of Rosetta against an off-the-shelf JavaME to Android translator
- Runtime performance evaluation

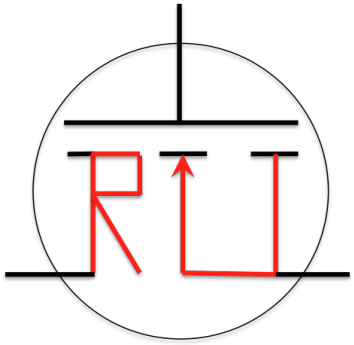
Conclusion

- It is becoming increasingly important to port apps to a variety of platforms
- **Key challenge**: Different platforms use different programming APIs
- API mapping databases help, but they are created manually by domain experts

We presented a methodology to automate the creation of API mapping databases

Thank You

Rosetta is part of the



MOSFET[RU] Project

Mobile Software Engineering Tools at Rutgers

Google **MOSFET Project Rutgers** for information