Test Transfer Across Mobile Apps Through Semantic Mapping

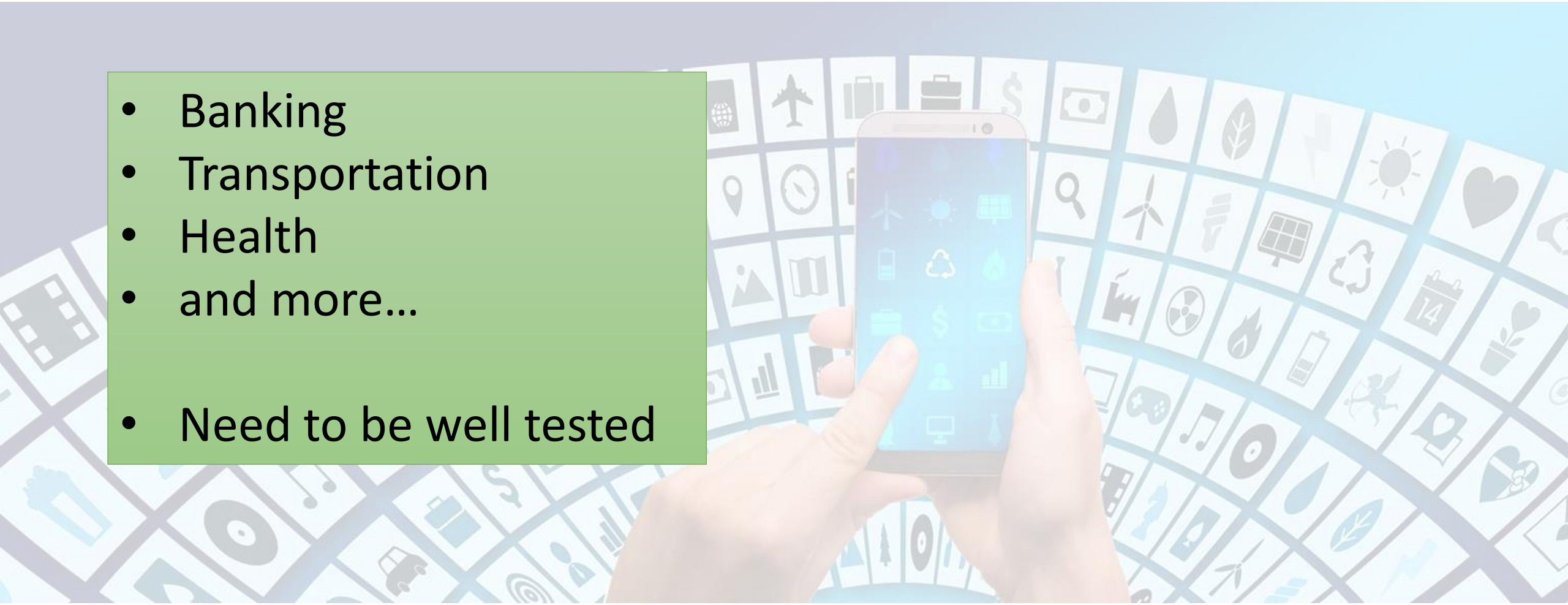
Jun-Wei Lin, Reyhaneh Jabbarvand, and Sam Malek
School of Information and Computer Sciences
University of California, Irvine



UCIrvine

Mobile Apps: Part of Our Lives

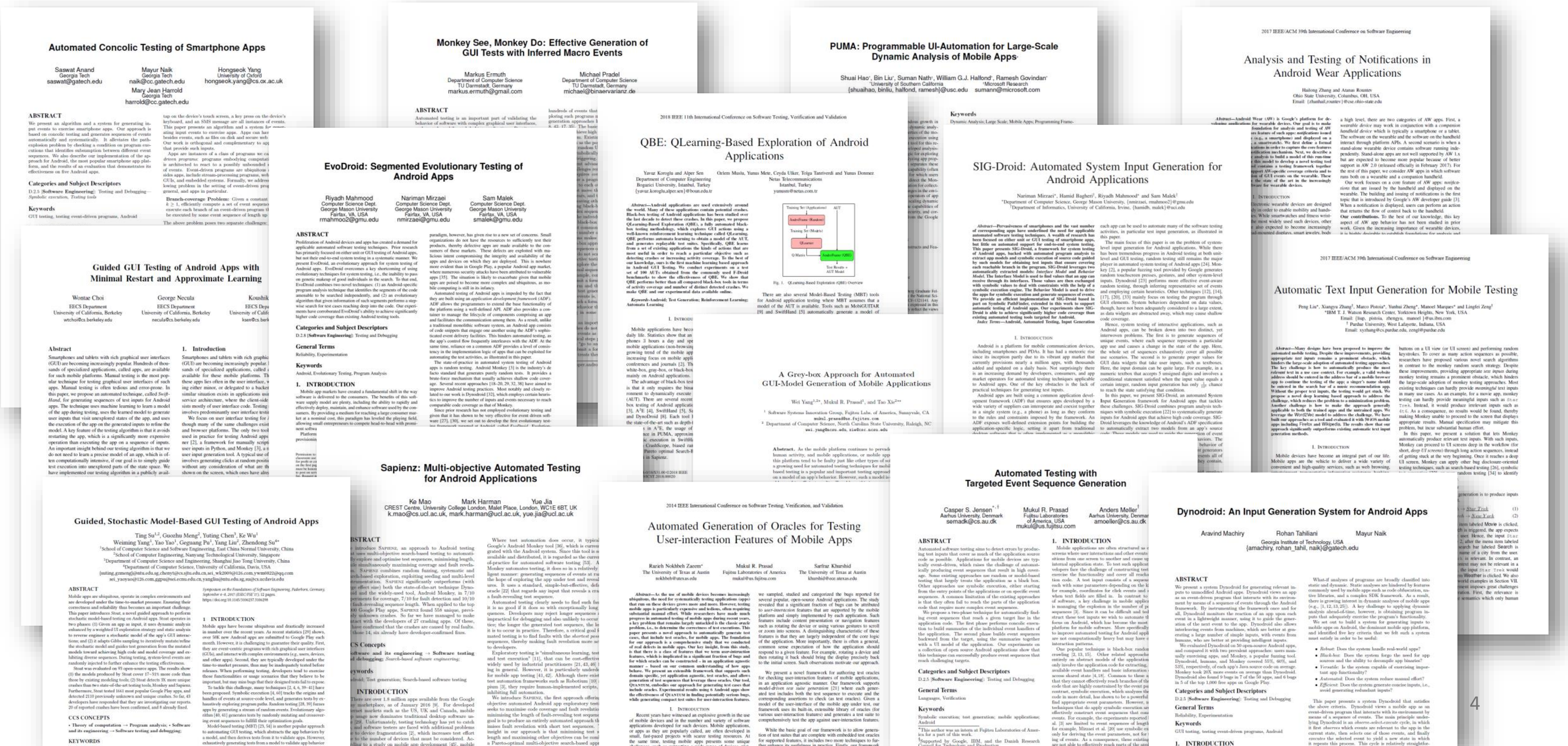
Mobile Apps: Part of Our Lives



- Banking
- Transportation
- Health
- and more…

- Need to be well tested

Automated Test Input Generation Techniques



Automated Test Input Generation Techniques

- Not widely adopted in practice
- Majority of the mobile app's testing is still performed manually

Reference:

- M. Linares-Vásquez, C. Bernal-Cardenas, K. Moran and D. Poshyvanyk, "How do Developers Test Android Applications," ICSME'17
- P. S. Kochhar, F. Thung, N. Nagappan, T. Zimmermann and D. Lo, "Understanding the Test Automation Culture of App Developers," ICST'15
- M. E. Joorabchi, A. Mesbah and P. Kruchten, "Real Challenges in Mobile App Development," ESEM'13

Limitations of Current Techniques

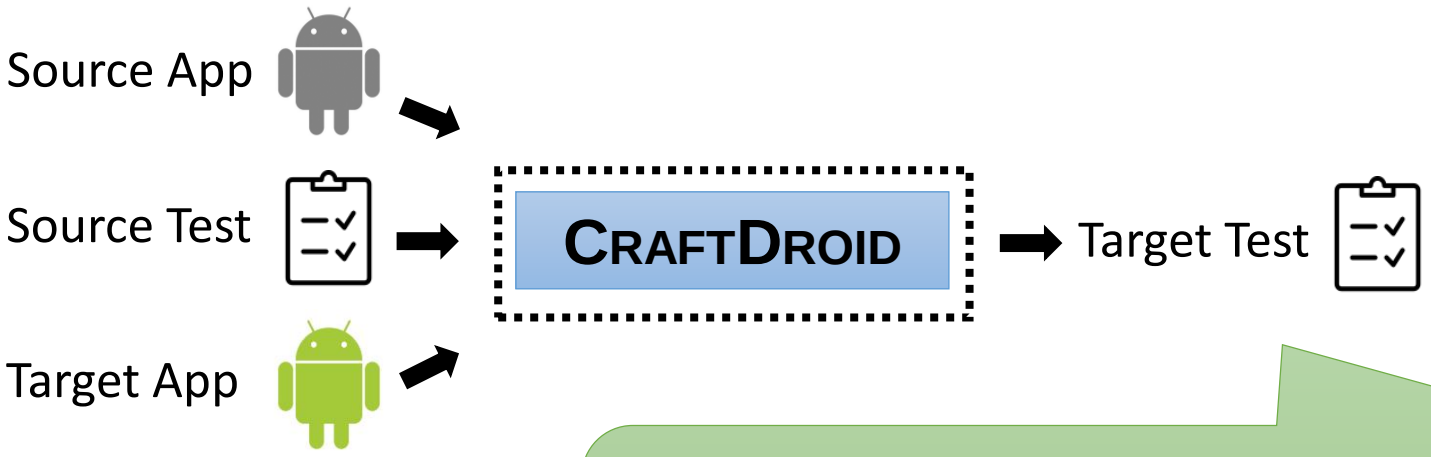
1. Lack of **context-aware text inputs**
   - e.g., city names for navigation apps; correct URLs for browser apps
   - Exploration may get stuck at the very beginning
2. Failing to generate **expressive tests**
   - Try to maximize code coverage or # of crashes
   - Tests are feature-irrelevant, not reflecting common usage scenarios
3. Absence of **test oracles**
   - Only focus on generating input events alone
   - Unable to identify failures other than crashes

CRAFTDROID: Test Transfer Across Mobile Apps

- Reuse existing tests, including oracles, for one app to test other similar apps

CRAFTDROID: Test Transfer Across Mobile Apps

- Reuse existing tests, including oracles, for one app to test other similar apps



1. Use **context-aware inputs**
2. Meaningful and **feature-relevant**
3. Contain suitable **oracles**

Insights Behind CRAFTDROID (1)

- Apps within the same category share similar functionalities
  - Email clients, web browsers, to-do lists, banking apps…
  - Exist across differ types of apps: registration, authentication, …

Reference:
- F. Behrang and A. Orso, "Test migration for efficient large-scale assessment of mobile app coding assignments," ISSTA 2018
- A. Rau, J. Hotzkow, and A. Zeller, "Transferring tests across web applications," ICWE 2018
- L. Mariani, M. Pezz`e, and D. Zuddas, "Augusto: Exploiting popular functionalities for the generation of semantic gui tests with oracles," ICSE 2018

Insights Behind CRAFTDROID (2)

- GUI for the same functionality are usually **semantically similar**, even if they belong to different apps with different looks and styles
- Semantic similarity: the conceptual relation between GUI elements and their **textual properties**, e.g., text, labels, variable names

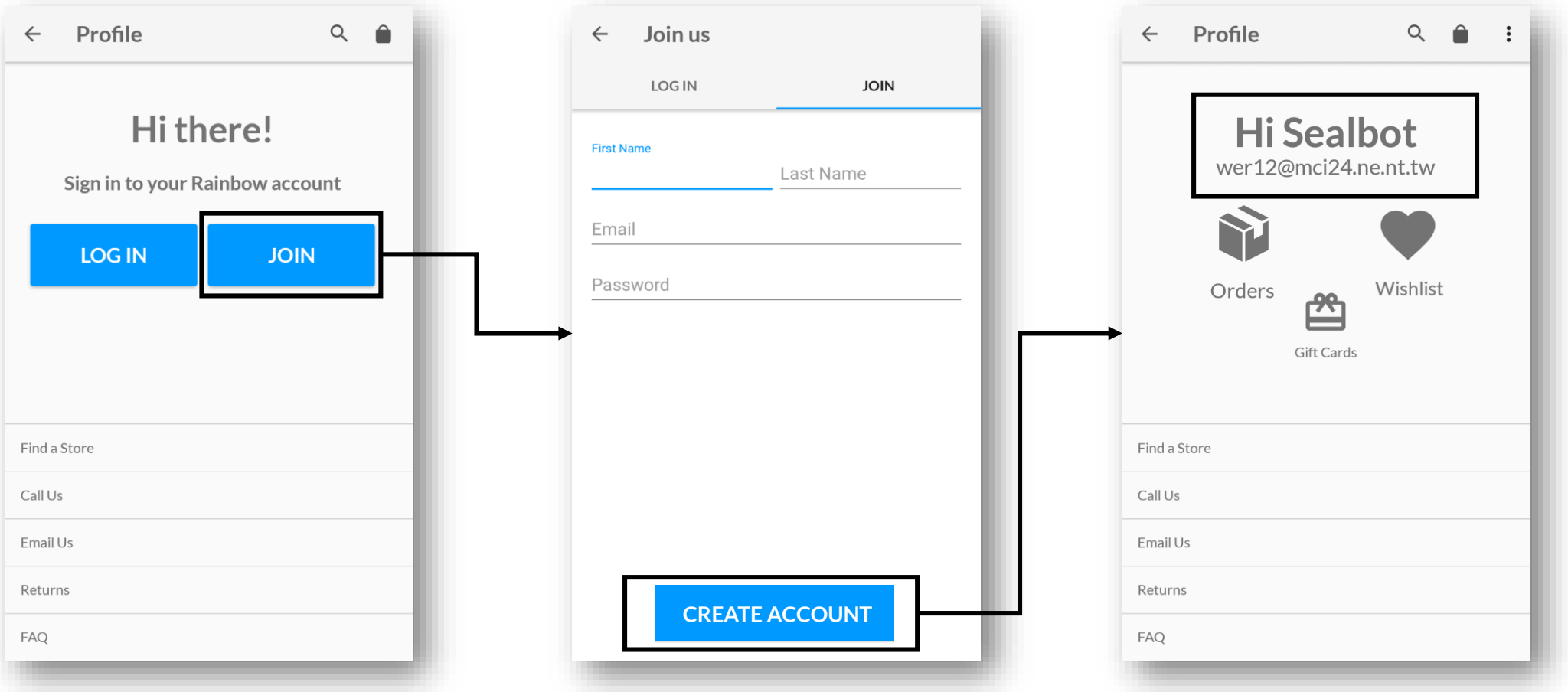
Reference:

- F. Behrang and A. Orso, "Test migration for efficient large-scale assessment of mobile app coding assignments," ISSTA 2018
- A. Rau, J. Hotzkow, and A. Zeller, "Transferring tests across web applications," ICWE 2018
- L. Mariani, M. Pezz`e, and D. Zuddas, "Augusto: Exploiting popular functionalities for the generation of semantic gui tests with oracles," ICSE 2018

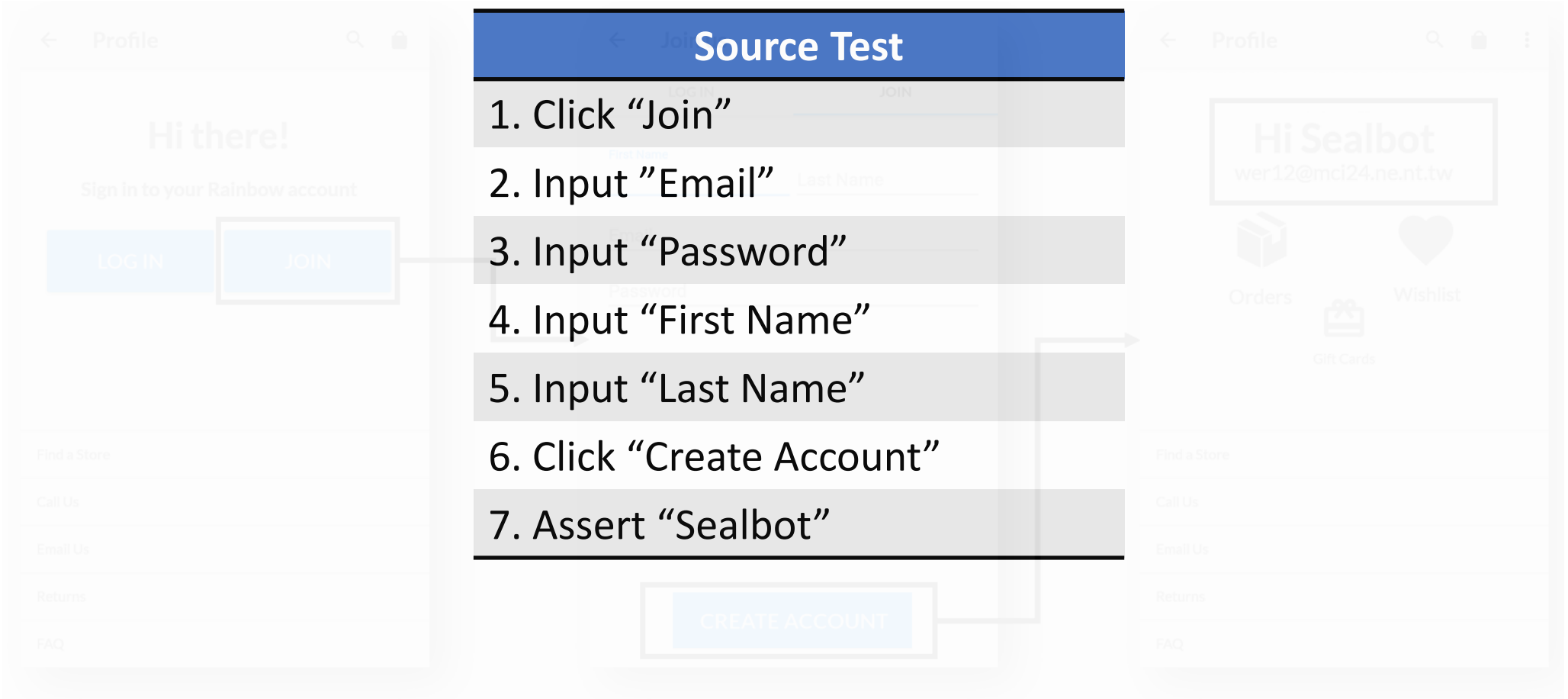
CRAFTDROID: Test Transfer Across Mobile Apps

- Introduction
- **Challenges and Motivating Example**
- Overview of CRAFTDROID
- Evaluation
- Conclusion

Source App: Rainbow Shops

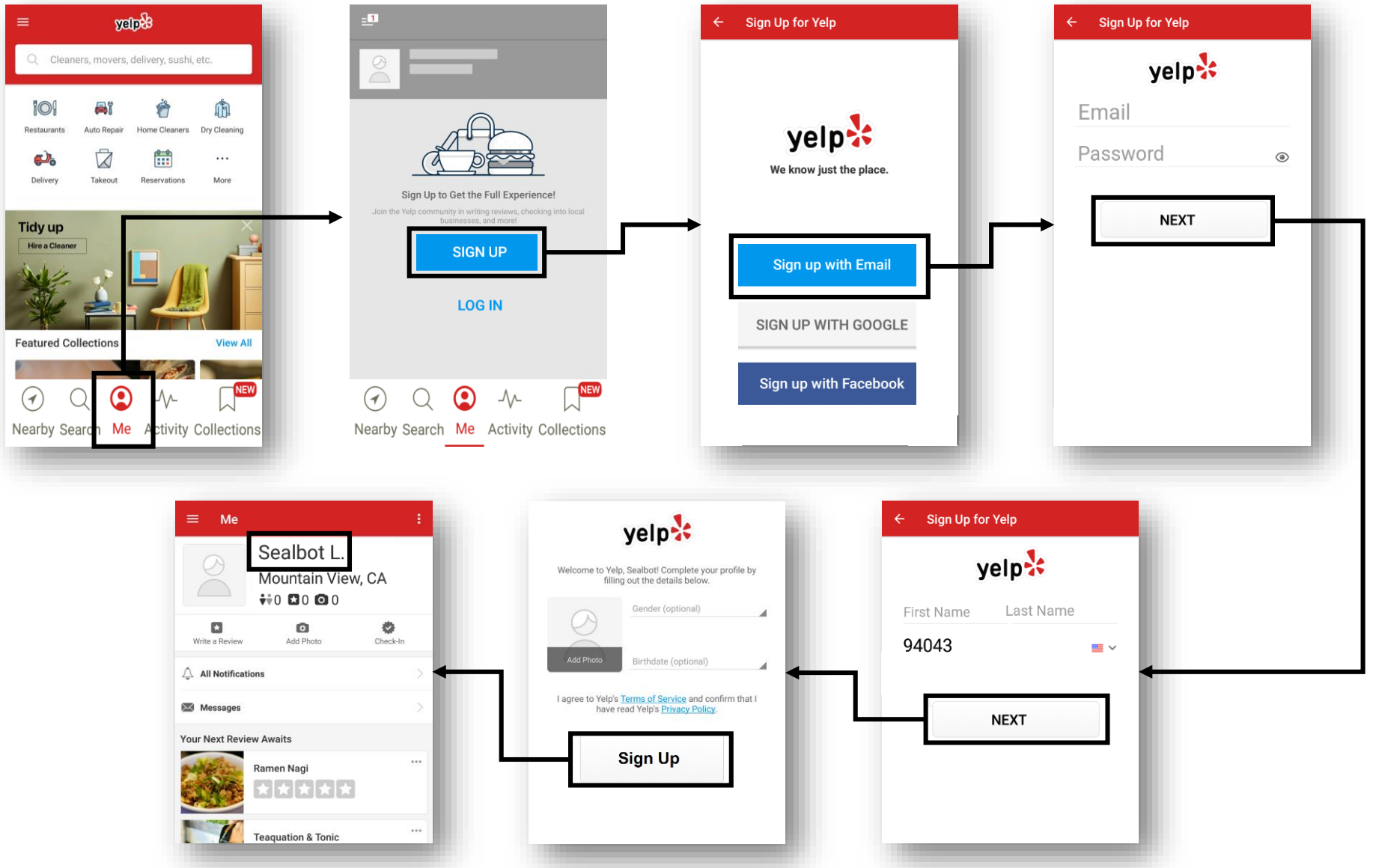


Source App: Rainbow Shops



Source Test
1. Click “Join”
2. Input ”Email”
3. Input “Password”
4. Input “First Name”
5. Input “Last Name”
6. Click “Create Account”
7. Assert “Sealbot”

Target App: Yelp



Target App: Yelp

Target Test
1. Click “Me”
2. Click “Sign Up”
3. Click “Sign up with Email”
4. Input ”Email”
5. Input “Password”
6. Click “Next”
7. Input “First Name”
8. Input “Last Name”
9. Click “Next”
10. Click “Sign Up”
11. Assert “Sealbot”

Source Test	Target Test
	1. Click “Me”
1. Click “Join”	2. Click “Sign Up”
	3. Click “Sign up with Email”
2. Input ”Email”	4. Input ”Email”
3. Input “Password”	5. Input “Password”
	6. Click “Next”
4. Input “First Name”	7. Input “First Name”
5. Input “Last Name”	8. Input “Last Name”
	9. Click “Next”
6. Click “Create Account”	10. Click “Sign Up”
7. Assert “Sealbot”	11. Assert “Sealbot”

Challenge 1: The mapping of GUI widgets (esp. the syntactically different but semantically similar ones)

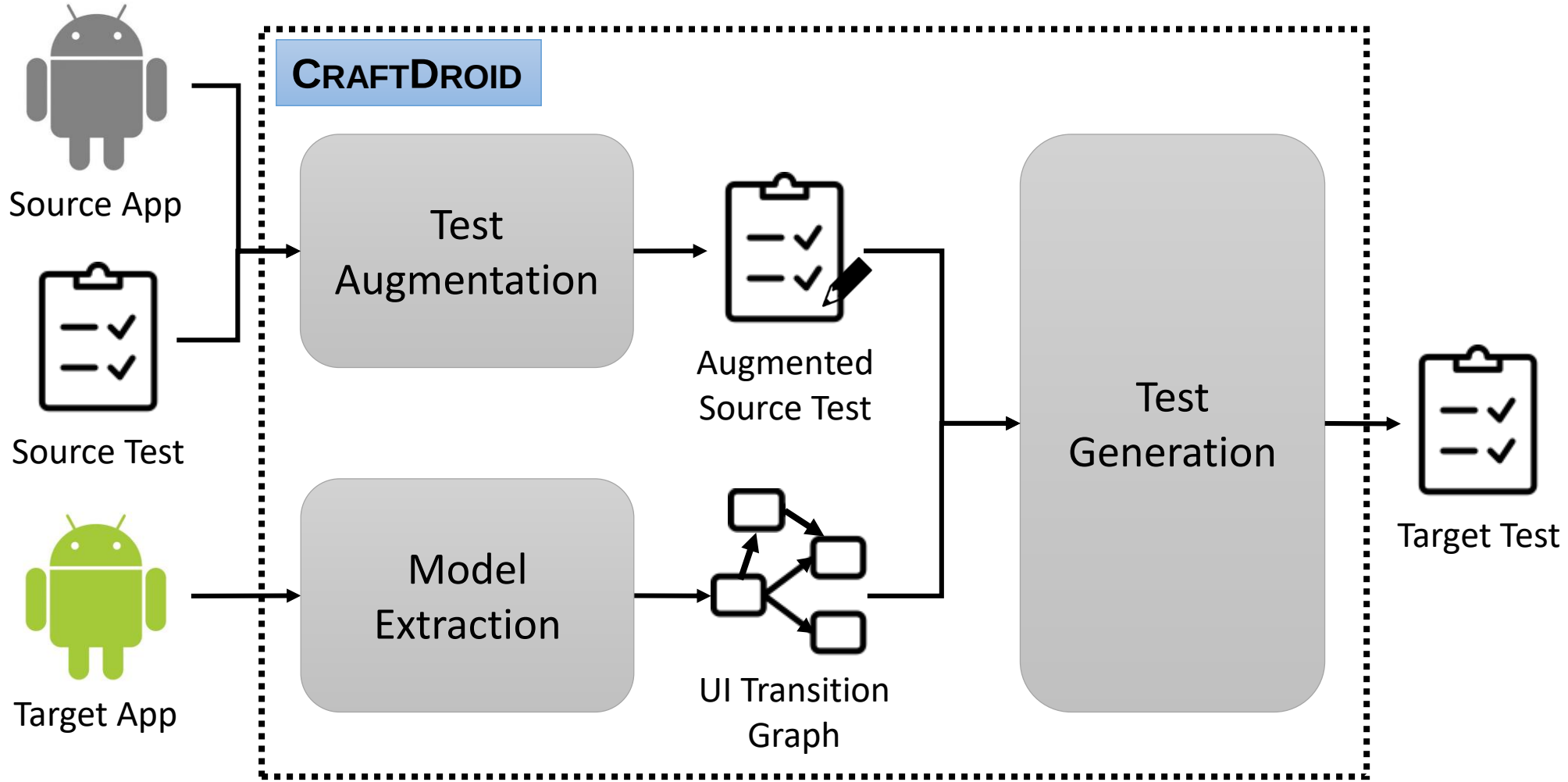
Source Test	Target Test
	1. Click “Me”
1. Click “Join”	2. Click “Sign Up”
	3. Click “Sign up with Email”
2. Input ”Email”	4. Input ”Email”
3. Input “Password”	5. Input “Password”
	6. Click “Next”
4. Input “First Name”	7. Input “First Name”
5. Input “Last Name”	8. Input “Last Name”
	9. Click “Next”
6. Click “Create Account”	10. Click “Sign Up”
7. Assert “Sealbot”	11. Assert “Sealbot”

Challenge 2: The mapping of test steps between two apps is not one-to-one

Source Test	Target Test
	1. Click “Me”
1. Click “Join”	2. Click “Sign Up”
	3. Click “Sign up with Email”
2. Input ”Email”	4. Input ”Email”
3. Input “Password”	5. Input “Password”
	6. Click “Next”
4. Input “First Name”	7. Input “First Name”
5. Input “Last Name”	8. Input “Last Name”
	9. Click “Next”
6. Click “Create Account”	10. Click “Sign Up”
7. Assert “Sealbot”	11. Assert “Sealbot”

CRAFTDROID: Test Transfer Across Mobile Apps

- Introduction
- Challenges and Motivating Example
- **Overview of CRAFTDROID**
- Evaluation
- Conclusion


CRAFTDROID
Source App
Source Test
Target App
Test Augmentation
Model Extraction
Augmented Source Test
UI Transition Graph
Test Generation
Target Test

Transfer Source Test to Target App

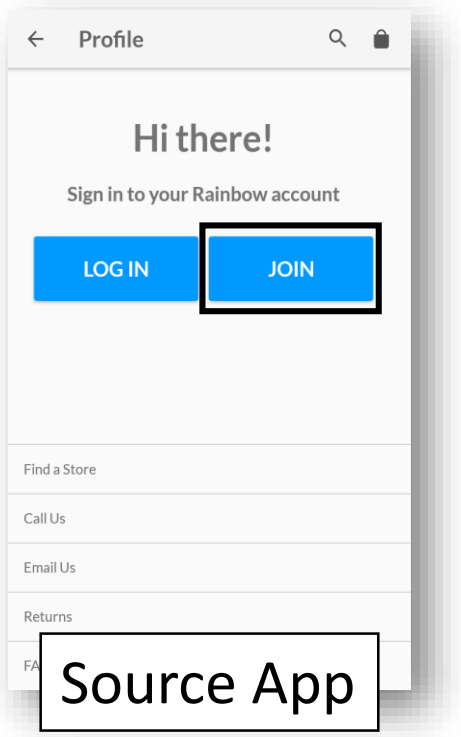
Iterate over every GUI or oracle event in the source test, trying to:

(1) Map the source widget to a target widget

(2) Identify the events leading to the target widget (if any)

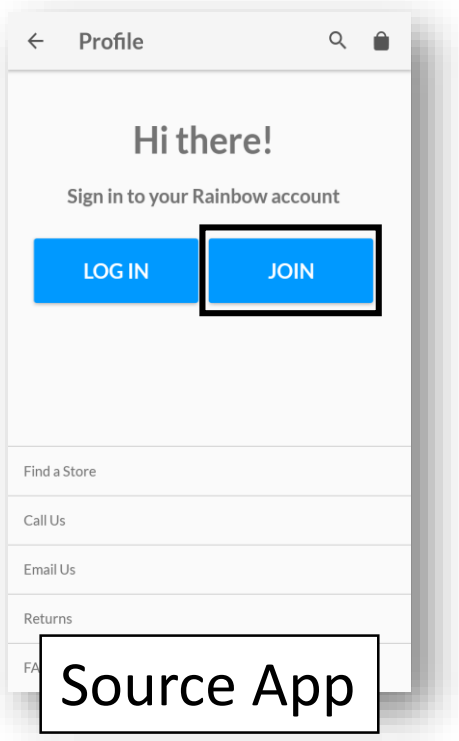
(3) Determine the action for the mapped target widget based on the source action

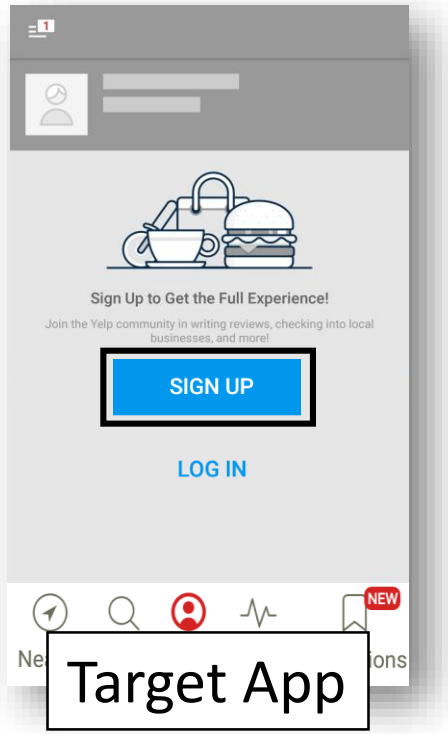
("Join", "Click")



1. Find the target widget which is most similar to the source widget

("Join", "Click")





Target widget w_t: "Sign Up"

Word2Vec

1. Find the target widget which is most similar to the source widget

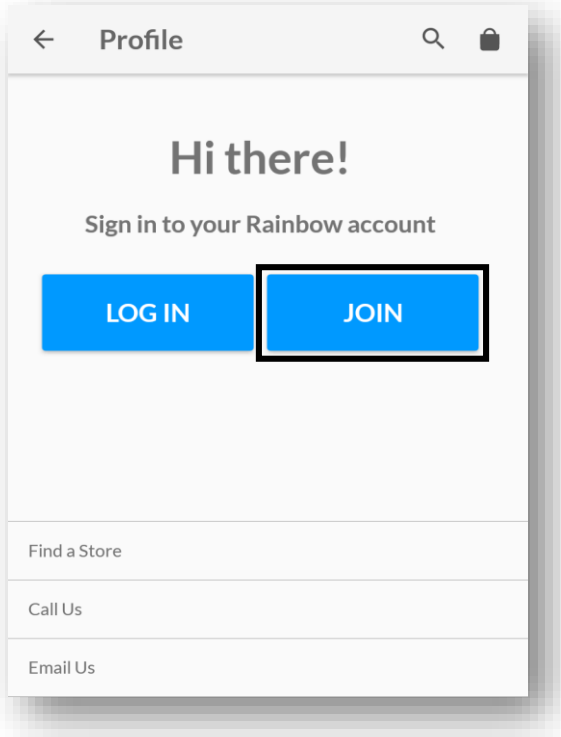
Word2Vec

- A neural network that is trained to reconstruct linguistic contexts of words
- Use a pre-trained model to get word embeddings (real-value vectors) for words
- Words more semantically related would be closer in terms of their cosine similarity

$$
\begin{aligned}
&Sim("Create","Sign") \\
&= \text{cosine}\big((0.204, 0.004, 0.073, 0.014, \dots), (0.012, 0.148, 0.102, 0.011, \dots)\big) \\
&= 0.405
\end{aligned}
$$

Computing Similarity Between Widgets

- Retrieve extra textual information from widgets
  - A widget has a set of word lists from multiple information sources



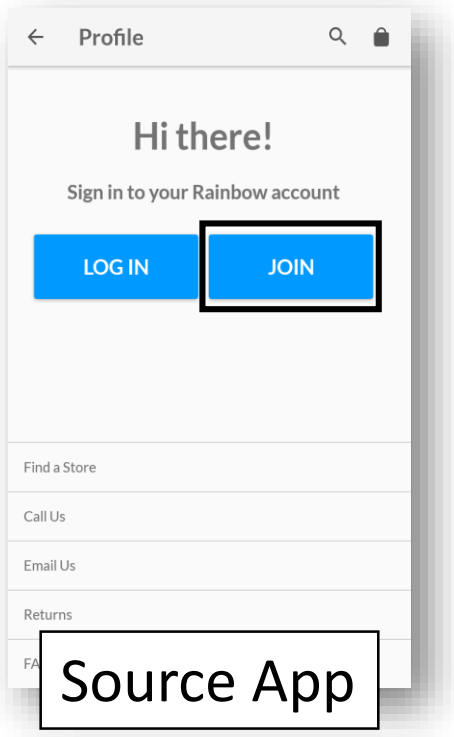

```
{
  "class": ["button"],
  "resource-id": ["button", "sign", "up"]
  "text": ["join"],
  "content-desc": "",
  "sibling_text": ["log", "in"],
  "activity": ["profile", "activity“],
   …
}
```

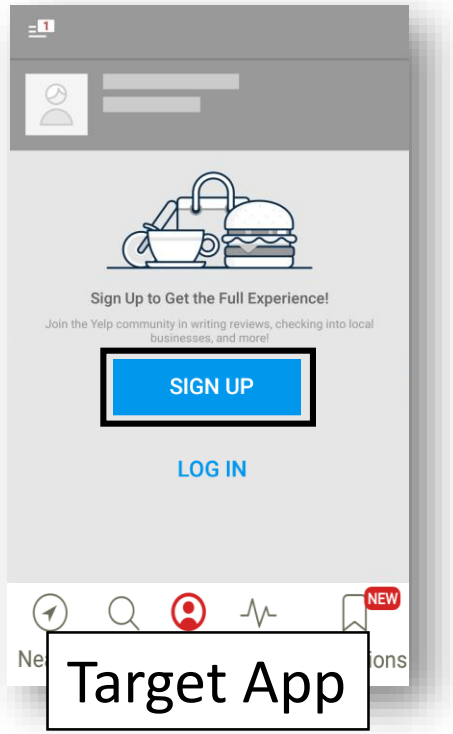
Computing Similarity Between Widgets

- Retrieve extra textual information from widgets
  - A widget has a set of word lists from multiple information sources
- Compute the textual similarity score for each source by leveraging Word2Vec
- Calculate a weighted sum of the scores of multiple sources

Challenge 1: The mapping of semantically similar GUI widgets

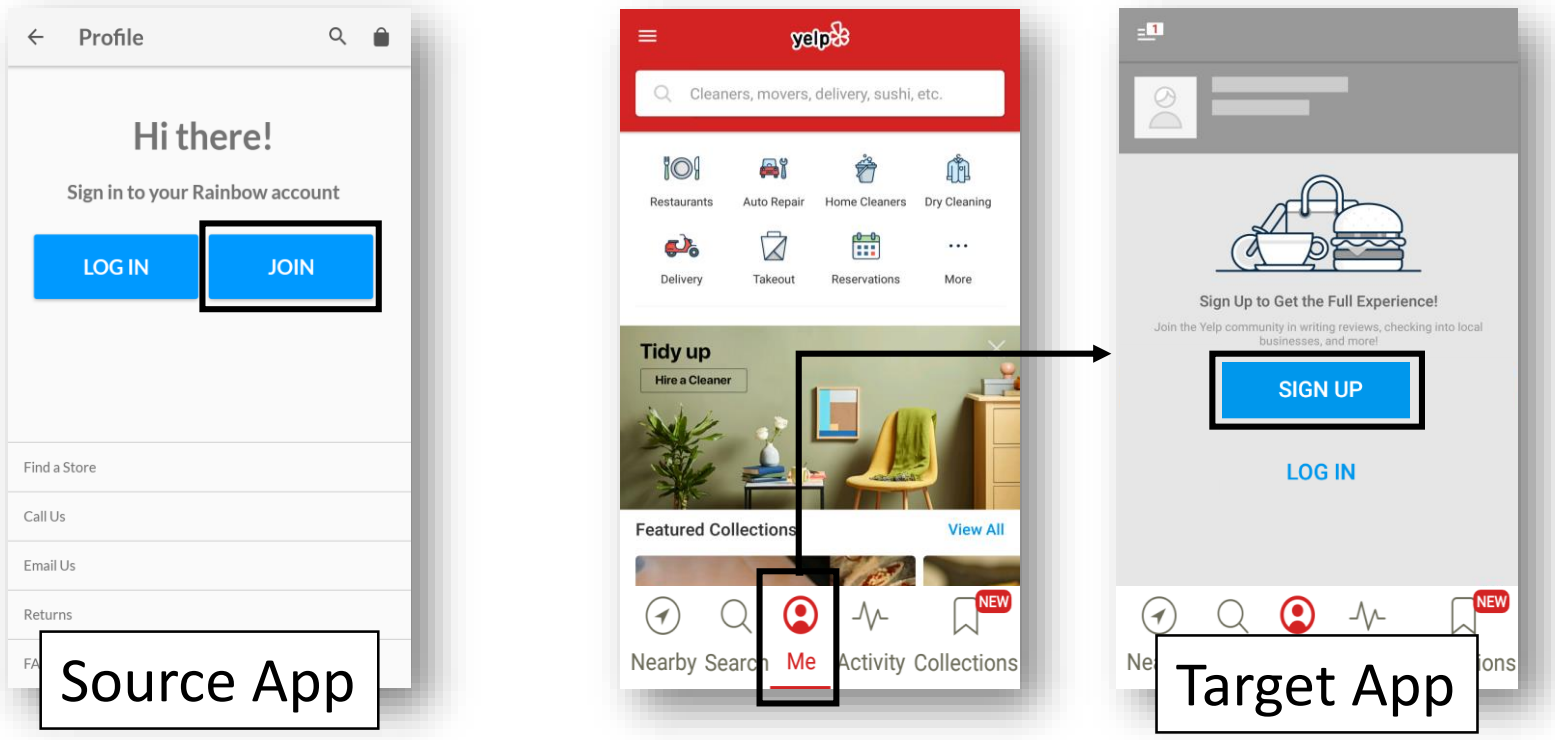
("Join", "Click")





1. Find the target widget which is most similar to the source widget
2. Find the events leading to the target widget (if any)

("Join", "Click")



Target widget w_t: "Sign Up"

$leadingEvents$ to w_t:
("Me", "Click")

1. Find the target widget which is most similar to the sour
2. Find the events leading to the target widget (if any)

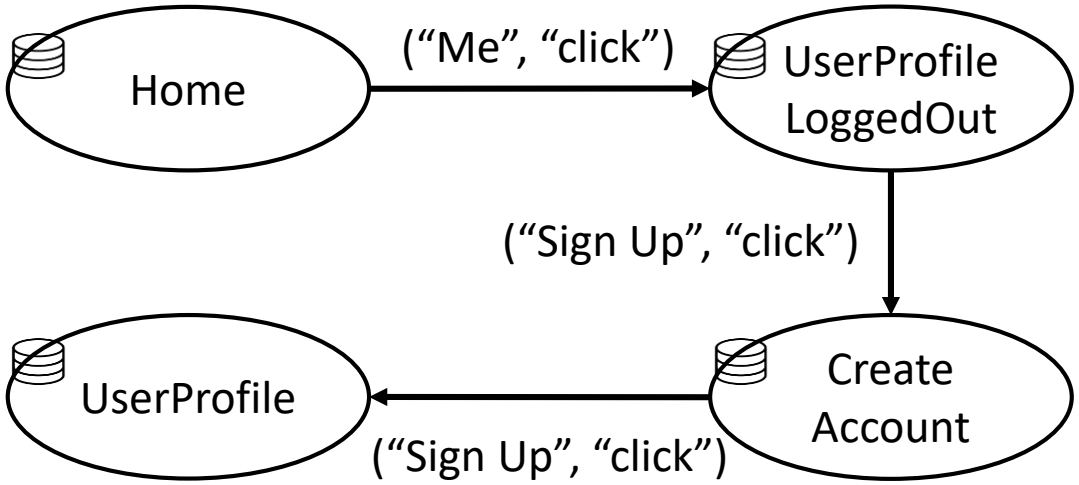
UI Transition Graph

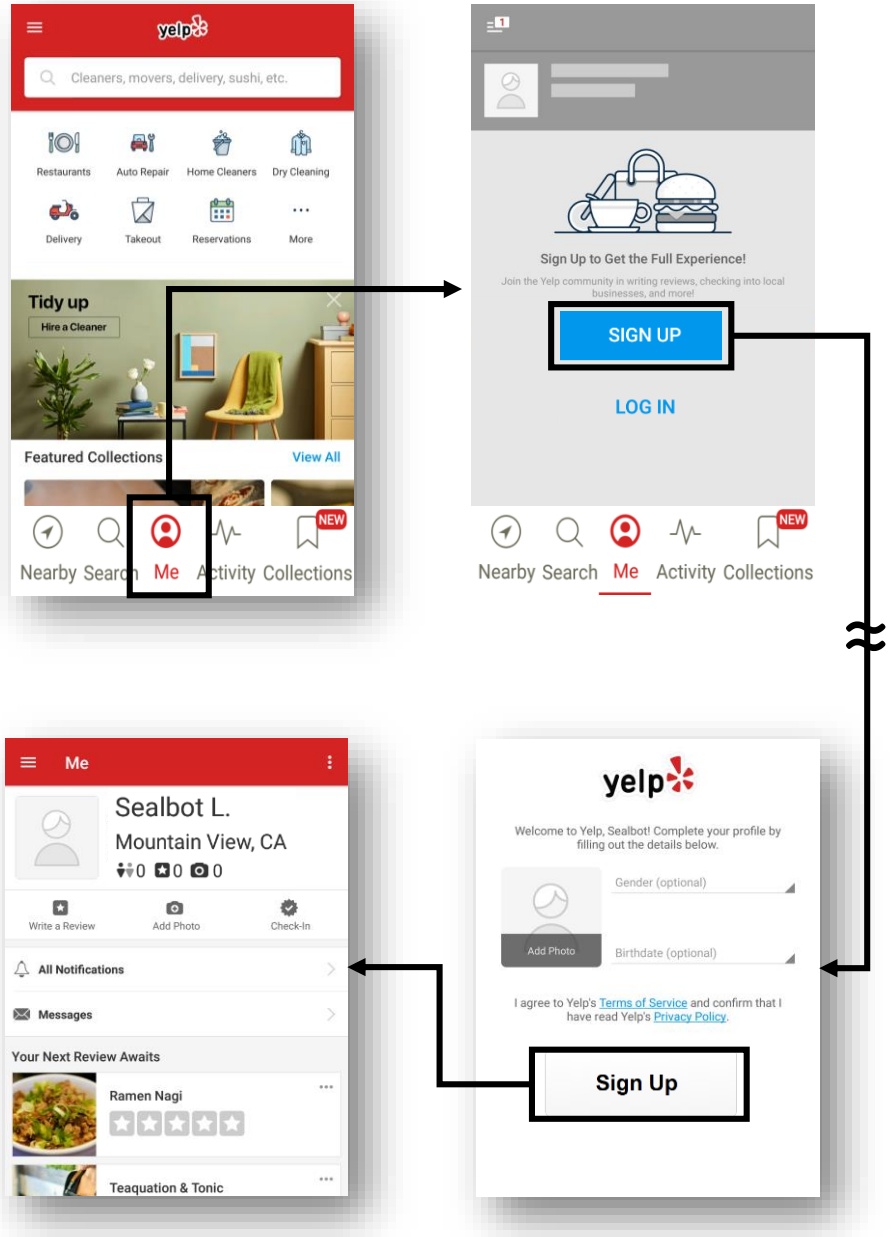
UI Transition Graph (UITG) for Target App

- Nodes: Activities
- Transitions: GUI events (inter- and intra-activity transitions)
- Widgets attached to Activities

- Parse the Manifest and Resource files
- Perform static analysis on the source code and look for specific program constructs and methods
  - e.g., `setContentView()`, `findViewById()`, `setOnClickListener()`, …

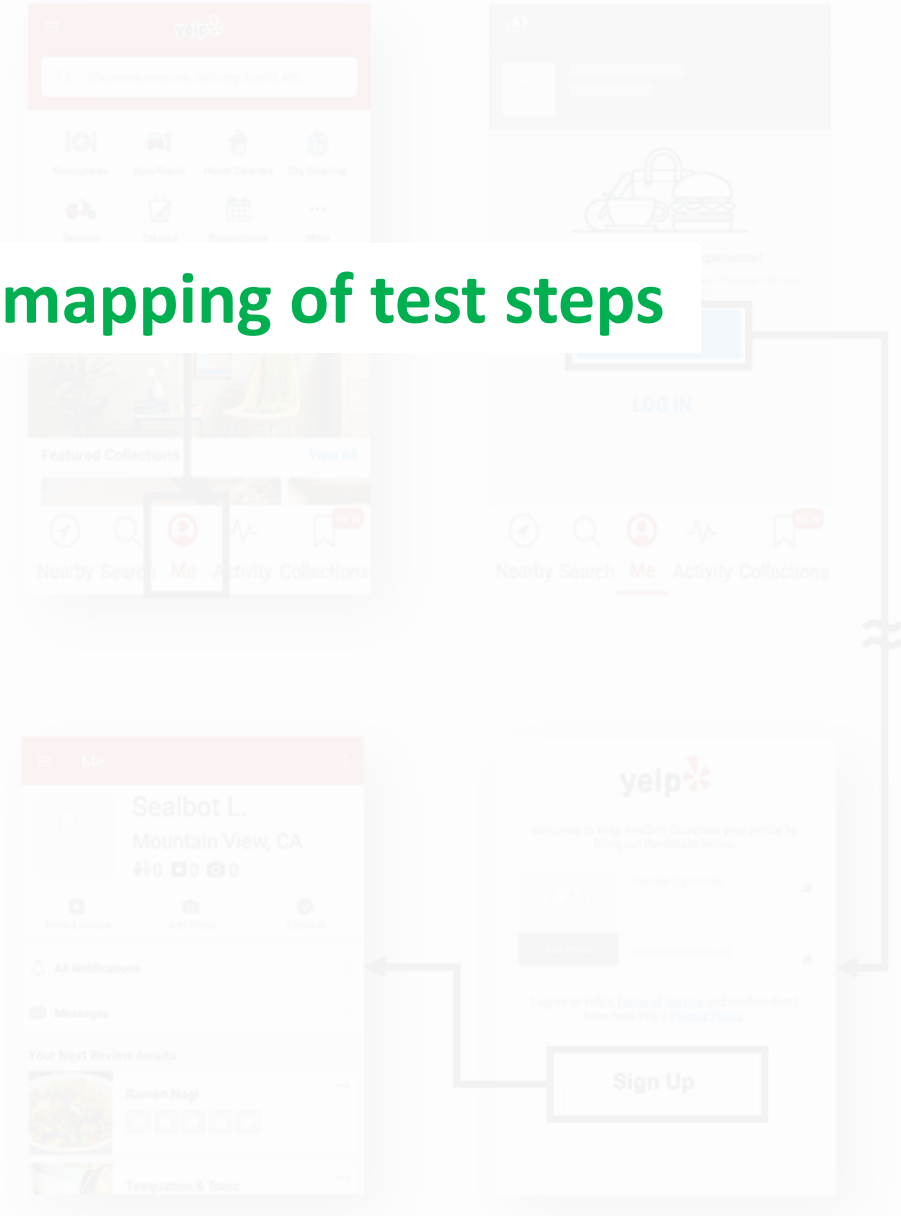
UI Transition Graph





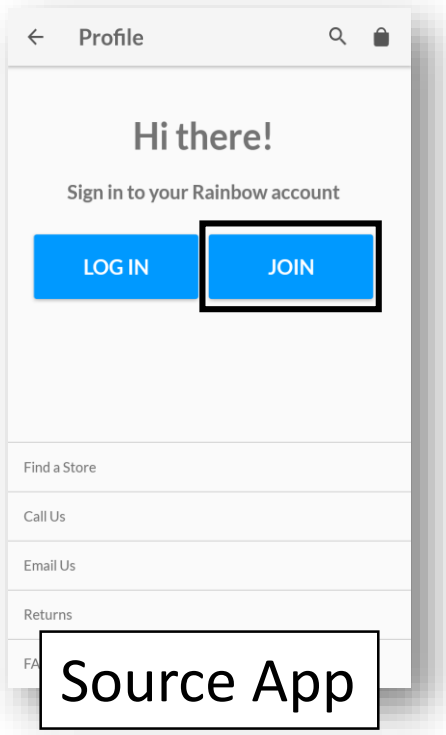
UI Transition Graph

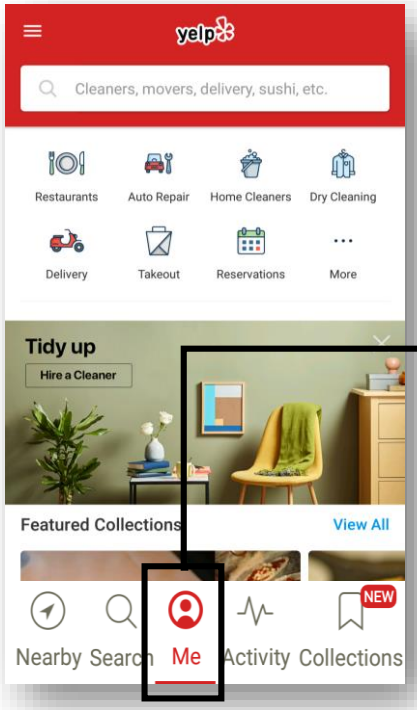
Challenge 2: Non one-to-one mapping of test steps

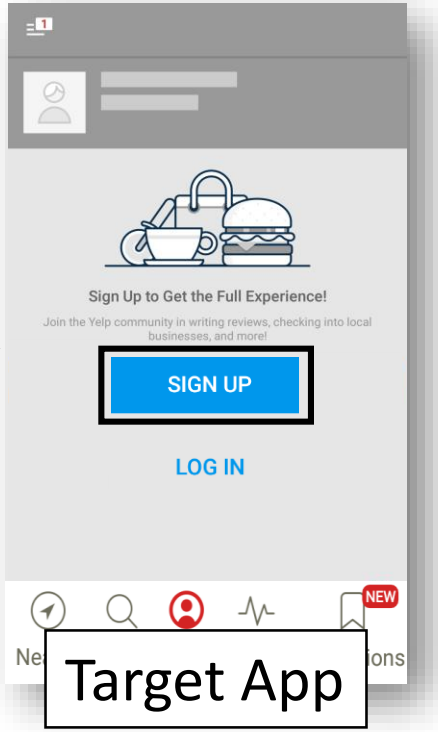


("Join", "Click")

("Me", "Click")
("Sign Up", "Click")







Target widget w_t: "Sign Up"

$leadingEvents$ to w_t:
("Me", "Click")

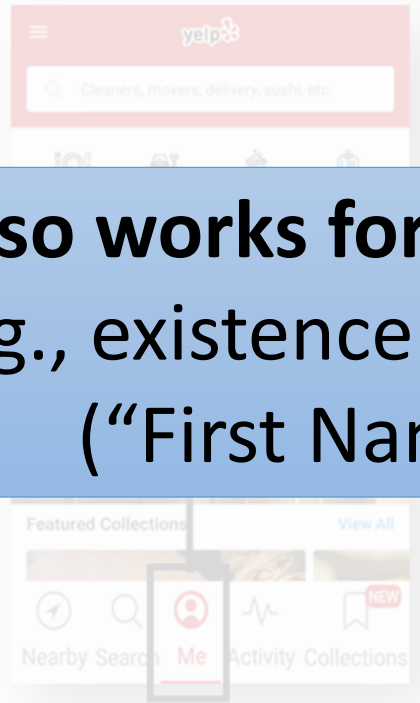
Action for w_t: "Click"
Target event:
("Sign Up", "Click")

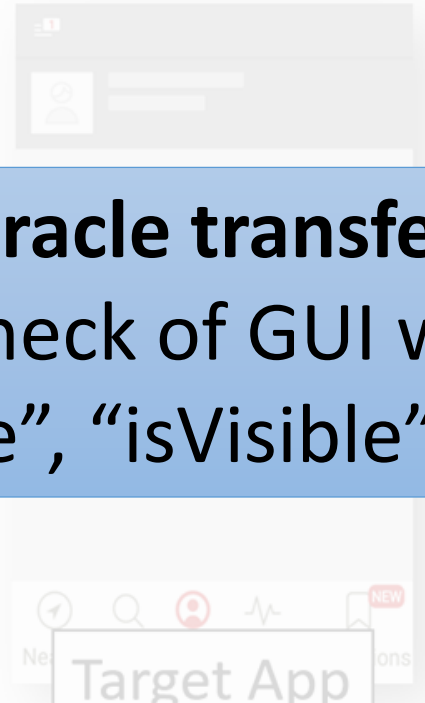
1. Find the target widget which is most similar to the source widget
2. Find the events leading to the target widget (if any)
3. Determine the action for the mapped target widget

("Join", "Click")

("Me", "Click")
("Sign Up", "Click")

Source App





Target App

Target widget w_t: "Sign Up"

$gEvents$ to w_t:
("Me", "Click")

for w_t: "Click"
event:
("Sign Up", "Click")

Also works for oracle transfer
e.g., existence check of GUI widget
("First Name", "isVisible")

1. Find the target widget which is most similar to the source widget
2. Find the events leading to the target widget (if any)
3. Determine the action for the mapped target widget

CRAFTDROID: Test Transfer Across Mobile Apps

- Introduction
- Challenges and Motivating Example
- Overview of CRAFTDROID
- **Evaluation**
- Conclusion

Evaluation

- 25 real-world subject apps
  - 5 categories
  - 5 apps for each category

Category	App (version)
a1-Browser	a11-Lightning (4.5.1) a12-Browser for Android (6.0) a13-Privacy Browser (2.10) a14-FOSS Browser (5.8) a15-Firefox Focus (6.0)
a2-To Do List	a21-Minimal (1.2) a22-Clear List (1.5.6) a23-To-Do List (2.1) a24-Simply Do (0.9.1) a25-Shopping List (0.10.1)
a3-Shopping	a31-Geek (2.3.7) a32-Wish (4.22.6) a33-Rainbow Shops (1.2.9) a34-Etsy (5.6.0) a35-Yelp (10.21.1)
a4-Mail Client	a41-K-9 (5.403) a42-Email mail box fast mail (1.12.20) a43-Mail.Ru (7.5.0) a44-myMail (7.5.0) a45-Email App for Any Mail (6.6.0)
a5-Tip Calculator	a51-Tip Calculator (1.1) a52-Tip Calc (1.11) a53-Simple Tip Calculator (1.2) a54-Tip Calculator Plus (2.0) a55-Free Tip Calculator (1.0.0.9)

Test cases for the identified functionalities

Category	Functionality	#Test Cases	Avg# Total Events	Avg# Oracle Events
a1-Browser	b11-Access website by URL	5	3.4	1
	b12-Back button	5	7.4	3
a2-To Do List	b21-Add task	5	4	1
	b22-Remove task	5	6.8	2
a3-Shopping	b31-Registration	5	14.2	5
	b32-Login with valid credentials	5	9	4
a4-Mail Client	b41-Search email by keywords	5	5	3
	b42-Send email with valid data	5	8	3
a5-Tip Calculator	b51-Calculate total bill with tip	5	3.8	1
	b52-Split bill	5	4.8	1
Total		50	6.6	2.4

Attempted transfers

- For each test case validating a functionality of an app, transfer it to the other four apps under the same category
- For each functionality
  - 5 (test cases) * 4 (transfers) = 20 attempted transfers
- 10 functionalities: 10 * 20 = **200 attempted transfers**

Evaluation Metrics

For each attempted transfer, check:

- **Whether the transfer is successful** (manually examined)
- Effectiveness of the widget mappings
  - **Precision**: how many generated target events are correct
  - **Recall**: how many source events are correctly transferred

Category	Functionality	GUI Event		Oracle Event		#Successful Transfer
		Precision	Recall	Precision	Recall	
Browser	b11	79%	100%	100%	100%	20/20 (100%)
	b12	85%	100%	100%	100%	20/20 (100%)
To Do List	b21	78%	100%	85%	100%	17/20 (85%)
	b22	69%	100%	85%	80%	11/20 (55%)
Shopping	b31	44%	90%	34%	67%	8/20 (40%)
	b32	53%	82%	56%	61%	10/20 (50%)
Mail Client	b41	100%	100%	100%	100%	20/20 (100%)
	b42	85%	80%	89%	89%	14/20 (70%)
Tip Calculator	b51	82%	100%	100%	80%	16/20 (80%)
	b52	80%	100%	100%	65%	13/20 (65%)
	Total	**70%**	**94%**	**79%**	**85%**	**149/200 (74.5%)**

Good successful transfer (75%), good precision (73%), excellent recall (90%)

Factors Impacting Effectiveness

- Length of test case (i.e., number of total events)

Pearson correlation coefficient between avg. test length and effectiveness

	GUI event		Oracle event		#Successful Transfer
	Precision	Recall	Precision	Recall	
Avg. Test length	-0.74	-0.60	-0.87	-0.51	-0.71

- Strongly negative correlations

Factors Impacting Effectiveness

- Complexity of app, in terms of interface and functionality
- Apps with (de-facto) design guidelines, e.g., Browser apps
  - Simple main screen with a search bar; fewer actionable GUI widgets
- Apps without uniform design guidelines, e.g., Shopping apps
  - Number of functionalities on a screen
  - Number of required steps for a functionality
  - Pop-up coupons, shopping preference configurations, …

Future Work:
Precise App Categorization

- Test transfer only makes sense when applied to apps sharing similar features
- Default categories (e.g., 33 on Google Play) are too coarse grained
  - Insufficient information about specific features contained in an app
- **Fine-grained and feature-based** app categorization **before** test transfer
  - Text clustering (i.e., unsupervised document classification)
  - Code clone detection
  - Repacked mobile apps detection

Future Work:
Better Semantic Analysis for Widget Mapping

- Non-native UI
  - Image-based widgets without text: computer vision, image classification
  - Dynamically-generated widgets: dynamic analysis
- Integrate other ways to compute similarity
  - Analyze the corresponding event-handler logic for widgets

Conclusion

- CRAFTDROID, a framework for transferring tests across mobile apps
  - Through semantic mapping of actionable GUI widgets
- Evaluation on 25 real-world apps from 5 categories
  - 75% success rate; 73% precision and 90% recall on widget mapping
- Practical test transfer is feasible but there is a long way to go
  - e.g., complex apps, long tests, precise app categorization, …

Thank you!