AN EXPERIMENT IN DEVELOPING SMALL MOBILE PHONE APPLICATIONS COMPARING ON-PHONE TO OFF-PHONE DEVELOPMENT

by

TUAN ANH NGUYEN

Presented to the Faculty of the Graduate School of
The University of Texas at Arlington in Partial Fulfillment
of the Requirements
for the Degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

THE UNIVERSITY OF TEXAS AT ARLINGTON
August 2012

ACKNOWLEDGEMENTS

This thesis is inspired by the following people, who share my dream of developing programming language environments that are more convenient and accessible for everyone.

I want to express my sincere appreciation to my supervising professor, Dr. Christoph Csallner for his support in my imagination and trust in my abilities. This final thesis paper would not have been possible without his advice and his unending patience. He has been a teacher, a colleague, and a friend helping me navigate around the many obstacles I encountered while attending UT Arlington.

I am grateful to Dr. Manfrer Huber and Mr. David Levine for serving as members of my thesis committee. Especially, Dr. Huber was an invaluable source of knowledge and advice to help me improve my statistical analysis in this paper.

I thank Microsoft Research Connections for providing Windows Phones for the duration of the semester. A large thank you specifically to Nikolai Tillmann for supporting the study and reviewing our work.

This material is based upon work supported by the National Science Foundation under Grants No. 1017305 and 1117369.

Finally, I would like to thank my father, mother, sister, and my girlfriend for their unbounded support. They are the pillars that give me strength and encouragement to help me through my time in school.

July 9, 2012

ABSTRACT

AN EXPERIMENT IN DEVELOPING SMALL MOBILE PHONE APPLICATIONS COMPARING ON-PHONE TO OFF-PHONE DEVELOPMENT

TUAN ANH NGUYEN, M.S.

The University of Texas at Arlington, 2012

Supervising Professor: Christoph Csallner

TouchDevelop represents a radically new mobile application development model, as TouchDevelop enables mobile application development on a mobile device. I.e., with TouchDevelop, the task of programming say a Windows Phone is shifted from the desktop computer to the mobile phone itself. We describe a first experiment on independent, non-expert subjects to compare programmer productivity using TouchDevelop vs. using a more traditional approach to mobile application development.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

CHAPTER 1

INTRODUCTION

Programming applications for mobile phones used to be a niche task. However the recent wide adoption of mobile devices and especially smartphones, together with the app store mobile application distribution model, has elevated mobile application programming to a common activity. To support mobile application development, each producer of a major mobile application platform (iOS, Android, Windows Phone, etc.) provides a convenient software development kit (SDK). For example Google's Android SDK integrates nicely with the popular Java IDE Eclipse. This makes it easy for programmers to start programming mobile phone applications. I.e., programmers install an SDK on their desktop computer and develop a mobile application there as they would develop any other application. The SDK typically contains a powerful emulator or virtual device that allows programmers to simulate how their mobile applications will behave on an actual mobile device.

TouchDevelop represents a radically new mobile application development model [1], as TouchDevelop enables mobile application development on a mobile device. I.e., with TouchDevelop, the task of programming say a Windows Phone is shifted from the desktop computer to the mobile phone itself. While shifting the device on which to program, TouchDevelop promises to maintain much of the traditional, expressive programming style. I.e., TouchDevelop is still a traditional, expressive, Turing-complete, Pascal-like programming language.

The declared scope of TouchDevelop is small programming tasks that can be achieved in tens or hundreds of lines of code. Clearly, the small screen makes it a lot harder to manage medium or large-scale programs. This limited program size rules out large-scale professional development. So the declared circle of programmers centers on students and hobbyists [1]. Programmers, people that have programming experience, is different from non-programmers, people do not have programming experience. Non-programmers are sometimes also referred to as end-users.

While developing small programs in a traditional programming style on a phone is an interesting concept, it is not clear if it can be done in a way that allows average programmers to solve programming tasks efficiently. For example, the lack of a traditional keyboard may make coding tedious and slow. In prior work the TouchDevelop authors performed an informal self-experiment from which they concluded that the time taken to code in TouchDevelop is in the same order of magnitude as coding on a PC [1]. In this study, we perform a first experiment on independent, non-expert subjects to compare programmer productivity using TouchDevelop vs. using a more traditional approach to mobile application development [2].

In summary our study makes the following contributions.

- We provide an overview of the size of all TouchDevelop apps that have been written on a phone as of 17 February 2012.
- We perform a first experiment on independent, non-expert subjects to compare program size and programmer productivity using on-phone software development vs. using the more traditional off-phone approach to mobile application development.

CHAPTER 2

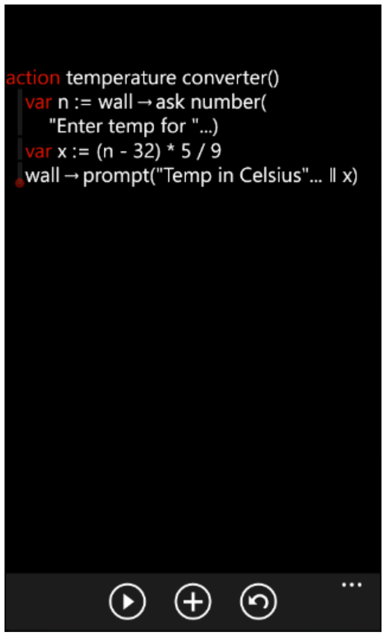
BACKGROUND

2.1 TouchDevelop

TouchDevelop includes the Turing-complete, Pascal-like TouchDevelop programming language, the TouchDevelop IDE, as well as a cloud service. The TouchDevelop IDE runs on Windows Phones and enables programmers to write and run TouchDevelop programs ("scripts" or "apps") on their phone. To be usable on a small touchscreen, the IDE employs a semi-structured code editor that treats all program tokens atomically and thereby minimizes the possibility of syntax or compile errors. I.e., the user either creates a new token or selects existing tokens from adaptive lists that reflect the current program context.

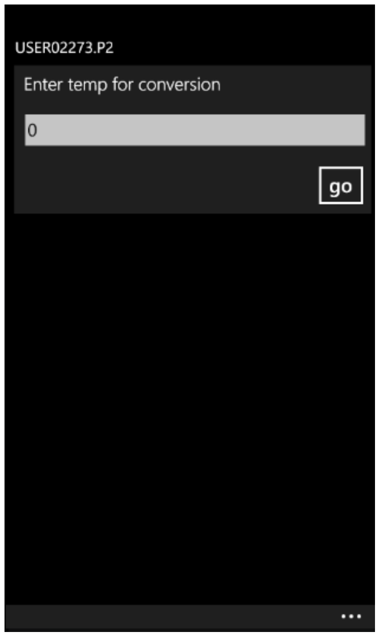
The TouchDevelop IDE is connected to the cloud, which allows developers to publish their TouchDevelop apps. By now users can upload TouchDevelop apps through the regular Windows Phone store, which contains all regular apps which runs on Windows Mobile phone. All published apps are made available with source code. Programmers are encouraged to download and extend apps written by others. The TouchDevelop cloud logs if a new app by programmer A is such a derivative of another app by programmer B. The cloud also logs if an app is simply a new version of an earlier app by the same author.

The TouchDevelop language is kept simple by focusing on tasks that can be accomplished on a phone. This allows TouchDevelop to implicitly assume many facts that have to be stated explicitly in other languages, which leads to simpler, more compact programs. For example, for printing some text a Java programmer has to specify a particular output stream. TouchDevelop only has one output stream, so a TouchDevelop app does not have to specify this detail and can be more compact. Figure 2.1 shows an example of an app in TouchDevelop which converts degrees Fahrenheit to degrees Celsius.



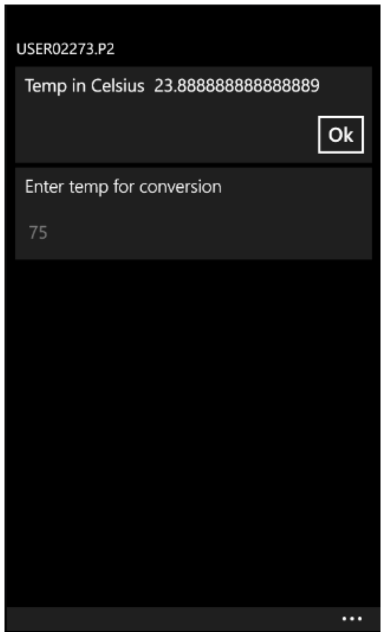
```
action temperature converter()
  var n := wall→ask number(
     "Enter temp for "...)
  var x := (n - 32) * 5 / 9
  wall→prompt("Temp in Celsius"... ∥ x)
```

(a) Source code



```
USER02273.P2
Enter temp for conversion
0
go
```

(b) Startup screen



```
USER02273.P2
Temp in Celsius  23.8888888888889
Ok
Enter temp for conversion
75
```

(c) Result screen

Figure 2.1. Example of an app in TouchDevelop, ”Convert degrees Fahrenheit to degrees Celsius”: The three screenshots all belong to the same app. The left screenshot is a view of the TouchDevelop IDE and red words are keywords of the language. The user can execute the app directly in the TouchDevelop IDE and the result is shown in the other two screenshots.

2.2 Android

The Android SDK provides tools and APIs for programmers to develop applications for Android mobile devices. In order to develop an Android application, programmers can follow four development phases [3]:

2.2.1 Setup

In this phase programmers install the development environment, which includes the Android SDK, Android Development Tools, and Android platforms. These tools and platforms are managed by using the Android SDK manager. Figure 2.3 shows the screenshot of Android SDK manager. Then programmers also create Android Virtual Devices (emulators) or connect their actual Android devices. They will later install and test their apps on these emulators and these devices. Figure 2.4 shows the screenshot of the Android virtual Device Manager tool, which helps the programmer to manage and configure emulators. Programmers can model many different actual devices by tailoring hardware and software options of emulators.

2.2.2 Development

In this phase programmers create and develop their Android app projects which contains all source code, resource files and manifest (configuration) files. Programmers are recommended to use Eclipse [1], an popular java IDE, with the Android Development Tools (ADT) plugin. Figure 2.2(a) shows source code of an example of an Android app (”Convert degrees Fahrenheit to degrees Celsius”).

[1]There are other Thrid-Party Development Tools which are not preferred by Android team, see link: http://developer.android.com/tools/workflow/index.html

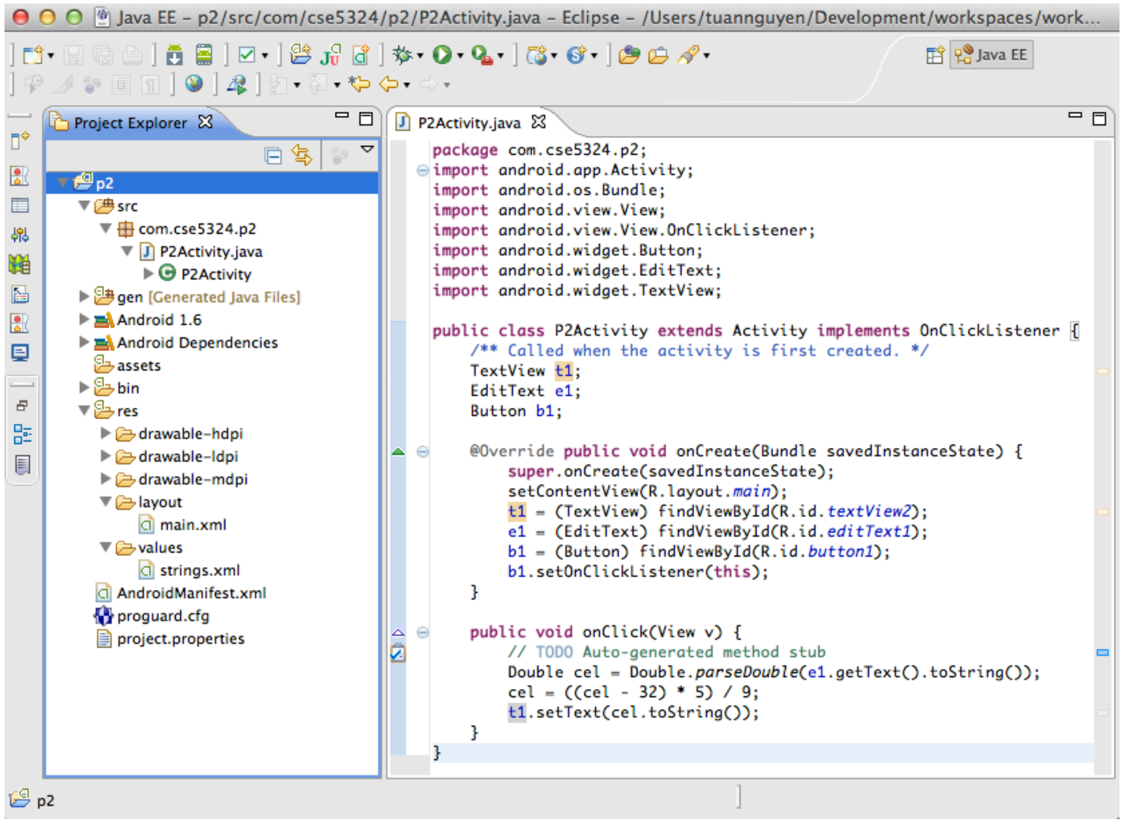
2.2.3 Debug and Testing

During this phase programmers build their project into an intermediate package used for debugging and testing their app. Then they install the package on one of the emulators configured in Section 2.2.1 or actual Android devices to run and test. Programmers can use command line tools provided by the Android SDK to build the package, or have it built automatically when using Eclipse. Figure 2.2(b) shows a screenshot of an Android app (Convert degrees Fahrenheit to degrees Celsius", when it runs on an emulator.

2.2.4 Publishing

In this phase, the programmer will either use a Android SDK command-line tool or Eclipse to build and sign the release package. Then programmers upload the package to Android Market (which is now called Google Play) and thereby release it to users.

∞



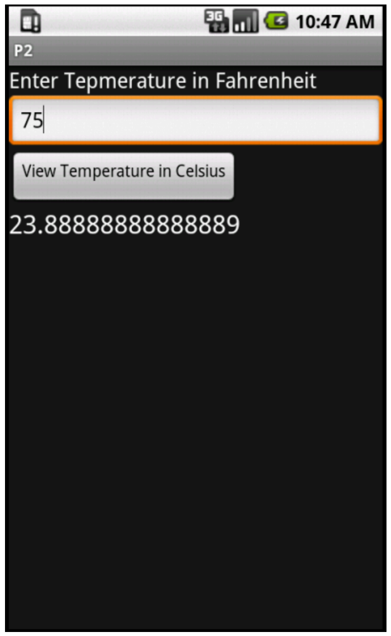
```
package com.cse5324.p2;
import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;

public class P2Activity extends Activity implements OnClickListener {
    /** Called when the activity is first created. */
    TextView t1;
    EditText e1;
    Button b1;

    @Override public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        t1 = (TextView) findViewById(R.id.textView2);
        e1 = (EditText) findViewById(R.id.editText1);
        b1 = (Button) findViewById(R.id.button1);
        b1.setOnClickListener(this);
    }

    public void onClick(View v) {
        // TODO Auto-generated method stub
        Double cel = Double.parseDouble(e1.getText().toString());
        cel = ((cel - 32) * 5) / 9;
        t1.setText(cel.toString());
    }
}
```

(a) Source code



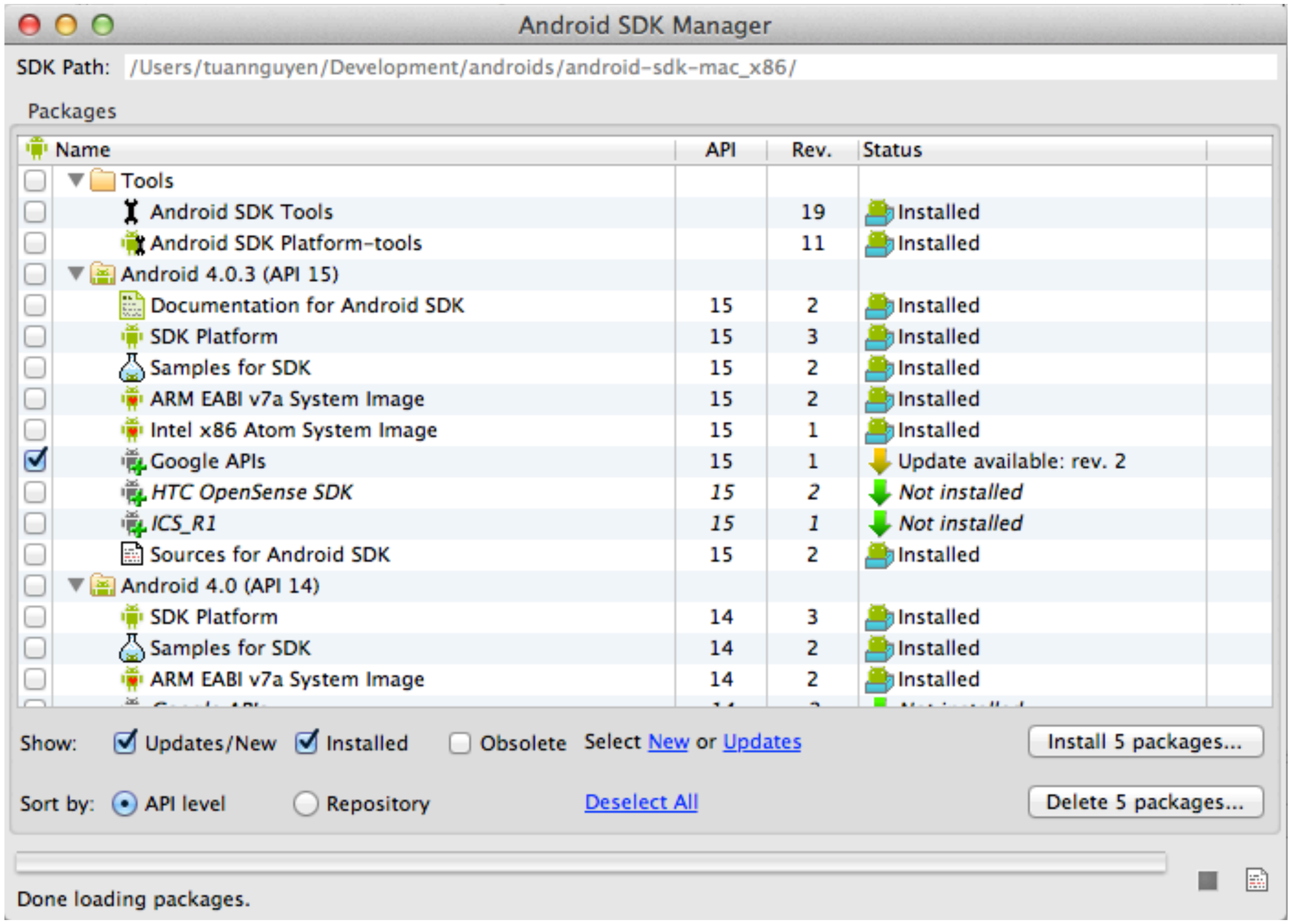
Enter Tepmerature in Fahrenheit

75

23.88888888888889

(b) Result screen

Figure 2.2. Example of an app in Android (”Convert degrees Fahrenheit to degrees Celsius”): Eclipse with Android Development Tools (ADT) is used to develop and publish Android apps. This also is an example of an Android app (”Convert degrees Fahrenheit to degrees Celsius”). Figure 2.2(b) is a screenshot of the execution of the Android app shown in Figure 2.2(a) when executed in the standard Android emulator. Figure 2.2(b) looks very similar to Figure 2.1(b) and Figure 2.1(c), which are the result when the same app is developed by using the TouchDevelop IDE.



Android SDK Manager

SDK Path: /Users/tuannguyen/Development/androids/android-sdk-mac_x86/

Packages

	Name	API	Rev.	Status
☐	▼ Tools			
☐	Android SDK Tools		19	Installed
☐	Android SDK Platform-tools		11	Installed
☐	▼ Android 4.0.3 (API 15)			
☐	Documentation for Android SDK	15	2	Installed
☐	SDK Platform	15	3	Installed
☐	Samples for SDK	15	2	Installed
☐	ARM EABI v7a System Image	15	2	Installed
☐	Intel x86 Atom System Image	15	1	Installed
✓	Google APIs	15	1	Update available: rev. 2
☐	*HTC OpenSense SDK*	*15*	*2*	*Not installed*
☐	*ICS_R1*	*15*	*1*	*Not installed*
☐	Sources for Android SDK	15	2	Installed
☐	▼ Android 4.0 (API 14)			
☐	SDK Platform	14	3	Installed
☐	Samples for SDK	14	2	Installed
☐	ARM EABI v7a System Image	14	2	Installed

Show: ✓ Updates/New ✓ Installed ☐ Obsolete Select New or Updates

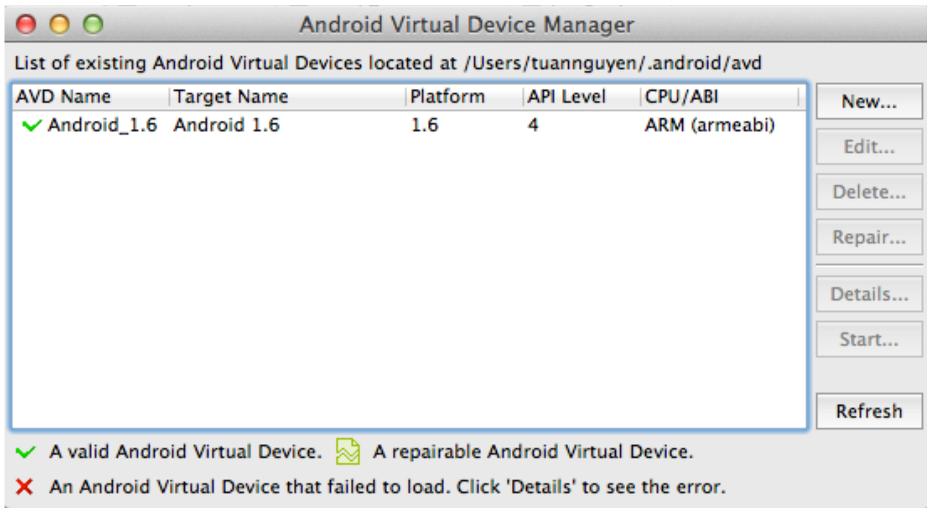
Install 5 packages...

Sort by: ● API level ○ Repository Deselect All

Delete 5 packages...

Done loading packages.

Figure 2.3. Screenshot of the Android SDK manager: The Android SDK manager is used to manage Android Development Tools and Android platforms.



List of existing Android Virtual Devices located at /Users/tuannguyen/.android/avd

AVD Name	Target Name	Platform	API Level	CPU/ABI
✓ Android_1.6	Android 1.6	1.6	4	ARM (armeabi)

✓ A valid Android Virtual Device. A repairable Android Virtual Device.

✗ An Android Virtual Device that failed to load. Click 'Details' to see the error.

Figure 2.4. Screenshot of Android Virtual Device Manager: The Android Virtual Device Manager is used to configure and manage emulators which are used to run, debug, and test Android applications.

CHAPTER 3

RESEARCH QUESTIONS (RQ), EXPECTATIONS (E), AND HYPOTHESES (H)

To evaluate this new TouchDevelop programming paradigm, we ask (a) how programmers have used TouchDevelop so far and (b) how their productivity using TouchDevelop compares with their productivity using a traditional mobile phone development approach. I.e., we investigate the following research questions and corresponding hypotheses.

- RQ1: How large are TouchDevelop applications?
  - E1: Given the limited size of a phone screen, we expect that most TouchDevelop apps are small.
  - H1: Programmers use TouchDevelop to write small applications, which have few low LOC.
- RQ2: For a given task, how do TouchDevelop solutions differ from solutions obtained with a traditional mobile phone development approach?
  - E2: By assuming and hiding many facts that have to be stated explicitly in other languages, we expect that a TouchDevelop app is typically smaller than a corresponding app in a traditional approach.
  - H2: For the same task, a TouchDevelop solution has fewer LOC than a corresponding solution using Android SDK.
- RQ3: For a given set of tasks, how does programmer productivity using TouchDevelop compare with using a traditional mobile phone application approach?
  - E3: The simplicity of the language may make TouchDevelop programmers more productive. But the difficulty of programming on a tiny screen without

a keyboard or mouse likely hampers productivity. So we expect TouchDevelop programmers to be less productive overall.

– H3: Given the same amount of time and the same set of small programming tasks, TouchDevelop programmers finish fewer tasks than programmers using Android SDK.

CHAPTER 4

DESIGN FOR ANSWERING RESEARCH QUESTIONS

hypotheses of Chapter 3.

4.1 Testing H1, Counting LOC

To test H1, we downloaded from the TouchDevelop cloud all apps that have ever been submitted. We removed all prior versions of an app published by the app's original author under the same app name. By doing that we only count the current version of each app. However we do not remove such apps if the prior and current app are by different authors or have different names.

H1 and H2 require us to count LOC. To make counting reproducible and comparable across techniques, we first normalized all TouchDevelop and Android apps and then counted their logical source statements (LSS) [4]. We did not count the content of configuration files such as the xml files each Android app uses to define layout, styling, etc.

4.2 Lab Experiment to Test H2, H3

To test H2 and H3, with IAB approval we recruited 27 graduate students of the CSE 5324 software engineering class taught by Dr. Csallner. CSE 5324 has a team project. Each team builds its own Android app. We designed a lab experiment that took place during one class period (80 minutes), towards the end of the course. At this point, most subjects had some experience developing for Android.

We stressed that experiment participation will not influence grades. To simulate individual development, we allowed subjects to consult web sources but forbid other communication during the experiment, except with the instructor and teaching assistants (the first two authors).

We defined a set of 11 simple programming tasks. For the experiment, Microsoft Research loaned us 10 mobile phones, which allowed us to have subjects develop TouchDevelop apps on a phone. We randomly assigned 10 subjects to one phone each and the control-group of the remaining 17 subjects to a lab Windows PC each that had installed an Android IDE (Eclipse) and SDK. The Android IDE contains an interactive wizard that generates a working hello-world Android app. The experiment used TouchDevelop v2.4.0.0 beta and Android v1.6.

The first 10 minutes were spent reading and signing the informed consent form and the Microsoft phone loan agreement. 60 minutes were available to subjects to configure their phone or PC (each taking about 5 minutes) and working on our programming tasks. Then we asked subjects to email us their solutions (PC) or return the phones. 10 minutes were then used for a questionnaire. 2 subjects did not email us solutions, leaving us with 15 Android subjects.

Some 10 weeks before the experiment we held an in-class exercise with the same subjects in the same lab with the same 10 phones, using different tasks and a different subject assignment, to gain experience for our IAB submission. Twice, before this trial and before the experiment, we emailed subjects a link to the TouchDevelop website and to a short introductory video[1]. Otherwise we did not train subjects in TouchDevelop.

4.3 Subjects' Prior Programming Experience for H2, H3

Figures 4.1 and 4.2 are box-and-whisker plots of our post-experiment questions that asked subjects to estimate their prior programming experience. Subjects were not given

[1]http://channel9.msdn.com/Blogs/Peli/TouchDevelop-Getting-Started

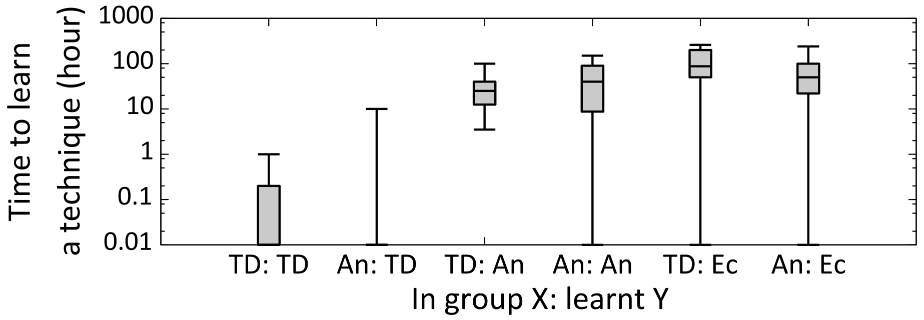


Figure 4.1. Subjects' prior experience - Time to learn techniques: Before the experiment, subjects had spent more time learning Android than learning TouchDevelop (Answers to question "How many of the following have you done before this exercise? Hours spent learning Y (watch video, read website, api, etc.)"). I.e., all TouchDevelop subjects reported having learned TouchDevelop for one hour or less (TD: TD). Ec is Eclipse, a popular Java and Android IDE.

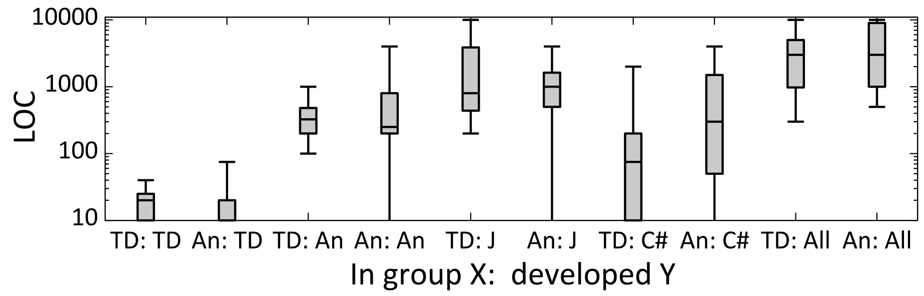


Figure 4.2. Subjects' prior experience - LOC of programming languages developed before: Before the experiment, subjects had written more Android than TouchDevelop LOC (Answers to question "How many of the following have you done before this exercise? Lines of Y code written"). I.e., all TouchDevelop subjects reported having written 40 or fewer TouchDevelop LOC (TD: TD). J is non-Android Java. All is code written in any language.

instructions on how to estimate.[2] In summary, Figures 4.1 and 4.2 show that our subjects had one or two orders of magnitude more experience with Android than with TouchDevelop, when measured as estimated time spent learning or as estimated LOC written for that approach.

[2]If a subject reported a range, we used the average of the lower and upper bounds, for a number with a plus or less-than ("100+", "<10") we ignored the operator, for an empty answer we used zero. A unit larger than hour we first converted to weeks (year = 52, semester = 12, month = 4) and then to hours (5), hours/week we assumed were given for one semester.

CHAPTER 5

RESULTS

5.1 Cloud: TouchDevelop Apps Are Small (H1)

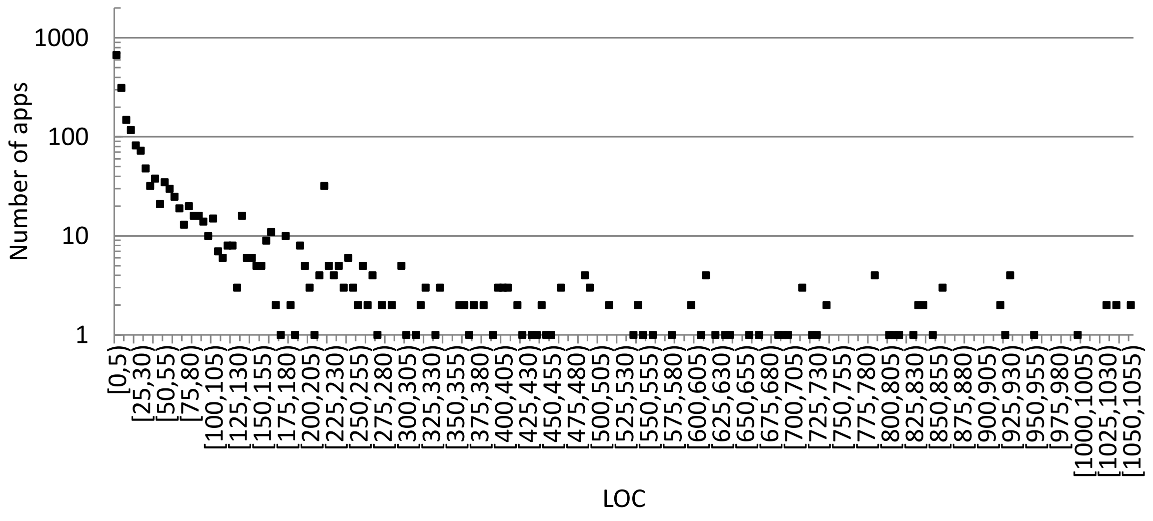


Figure 5.1. Size of all TouchDevelop apps published in the TouchDevelop cloud: Size in LOC of the 2,081 TouchDevelop apps published in the TouchDevelop cloud as of 17 February 2012. Not shown are the two largest apps of 1,742 and 1,675 LOC. Each dot represents all apps in one bin of 5 LOC. E.g., the left-most dot represents the 674 apps from 0 to 4 LOC.

Figure 5.1 shows the 2,081 TouchDevelop apps that have been published in the TouchDevelop cloud, grouped by their respective size in LOC. 47% (987) are 9 LOC or less and 60% are 19 LOC or less. This confirms that, at least so far, the majority of TouchDevelop apps are (very) small. Moreover, at the time of our experiment, on-phone development was the only way to get TD apps into the cloud, so we can reasonable assume that all TD apps in the TD cloud were indeed written on a phone [2].

5.2 Experiment: TouchDevelop-LOC < Android-LOC (H2)

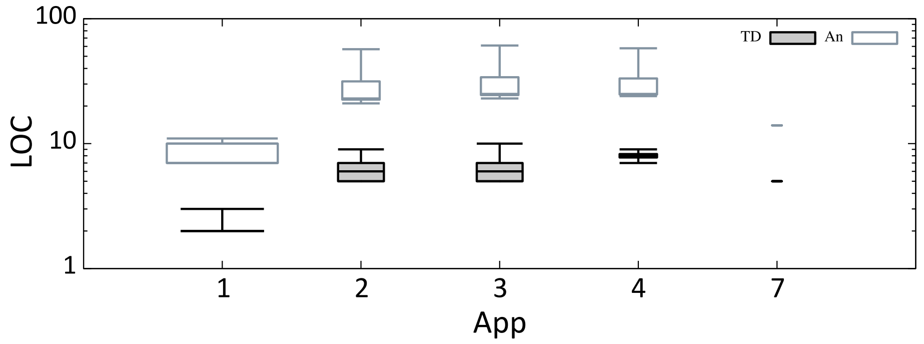


Figure 5.2. Size of the developed (correct) apps for each task: Size of the developed (correct) apps differs between Android and TouchDevelop. I.e., for each task, each correct Android app is larger than each of the correct TouchDevelop apps. The width of each box-and-whisker is proportional to the number of correct solutions we received. Not shown are tasks for which we did not receive any correct app.

Figure 5.2 is a box-and-whisker plot of the size in LOC of all correct solutions submitted by the subjects. For each task, the average correct Android solution was about four times larger in LOC than the average correct TouchDevelop solution. This confirms our expectation that TouchDevelop apps are smaller than corresponding Android apps [2].

Listing 5.1. Example task 1 solution in Android ("Hello World").

```
package edu.uta.cse.program1;
import android.app.Activity;
// .. two more import statements
public class P1Activity extends Activity
{
  @Override
  public void onCreate(Bundle bundle)
  {
    super.onCreate(bundle);
    TextView tv = new TextView(this);
    tv.setText("CSE␣5324");
    setContentView(tv);
  }
}
```

Listings 5.1 and 5.2 illustrate this difference in LOC. Both listings are example correct solutions of task 1 (“Any ‘Hello World’ program that prints ‘CSE 5324’ on the screen”). To work, the Android solution in Listing 5.1 must have a new class that (a) extends the Activity class and (b) overrides the onCreate method, which (c) must[1] call the overridden method. The onCreate method further should (d) create a text view, (e) set “CSE 5324” on the text view, and (f) add the text view to the main screen. In contrast, the corresponding TouchDevelop solution in Listing 5.2 consists of a single step, posting “CSE 5324” to the default output stream (the wall).

Listing 5.2. Example task 1 solution in TouchDevelop.

```
action main()
  'CSE 5324'→post to wall
```

Listing 5.3 Listing 5.4 also illustrate this different in LOC of task 2 (“a program that takes as input an integer number representing degrees Fahrenheit, converts it to degrees Celsius, and prints the resulting value.”). The Android solution, beside performs basic steps (a, b, c, d) which is similar to task 1, needs to perform addition steps: the onCreate method should (e) create a edit text to recieve user input (f) create a button to perform user action (g) calculate the result and (i) set the result to text view. On the other hand, the

[1]Documented as “Derived classes must call through to the super class’s implementation of this method. If they do not, an exception will be thrown.” http://developer.android.com/reference/android/app/Activity.html

corresponding TouchDevelop solution in Listing 5.4 consists of three simple steps: (a) get input from user (b) calculate the result and (c) output it.

Listing 5.3. Minimal task 2 (“Convert Fahrenheit to Celsius”) solution in Android.

```
package uta.edu;
import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;

public class p2 extends Activity
{
  /** Called when the activity is first created. */
  @Override public void onCreate(Bundle savedInstanceState)
  {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.p2);
    final EditText fahrenheit = (EditText) findViewById(R.id.entry);
    Button button = (Button) findViewById(R.id.ok);
    button.setOnClickListener(new OnClickListener()
    {
      public void onClick(View v)
      {
        // Perform action on clicks
        double celsius;
        int fah = Integer.parseInt(fahrenheit.getText().toString());
```

```
            celsius = ((fah − 32) ∗ 5) / 9;
            TextView tv = (TextView) findViewById(R.id.result);
            tv.setText(celsius + "");
        }
    });
  }
}
```

Listing 5.4. Minimal task 2 (“Convert Fahrenheit to Celsius”) solution in TouchDevelop.

```
action temperature converter()
  var n := wall→ask number("Enter temp for conversion ")
  var x := (n − 32) ∗ 5 / 9
  wall→prompt("Temp in Celsius " || x)
```

5.3 Experiment: TouchDevelop-Productivity > Android-Productivity (H3)

Table 5.1. Average of submitted code and correct code per group: TouchDevelop subjects on average submitted code for more tasks and completed more tasks than Android subjects.

	Subjects	Some code	Correct
TouchDevelop (TD)	10	3.7	2.4
Android (An)	15	3.2	1.7

Table 5.1 shows the number of tasks for which subjects submitted some code and correct code. TouchDevelop subjects on average finished more tasks (correctly) than Android subjects. Similarly, as shown by Figure 5.3, the likelihood that a task is finished was greater for TouchDevelop than for Android subjects. Given the low familiarity of our

subjects with TouchDevelop and their comparatively strong background in Android, this finding surprised us [2].

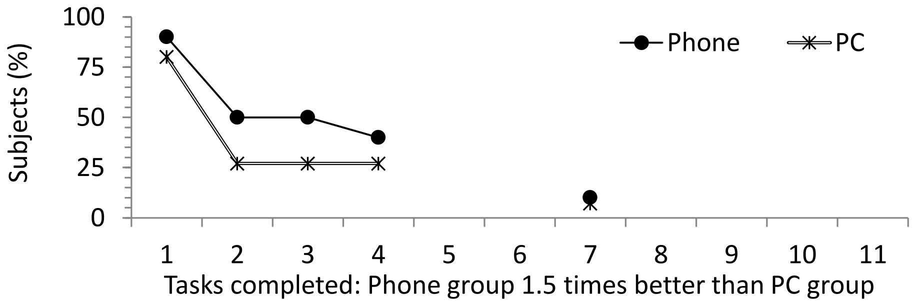


Figure 5.3. Tasks completed: Phone group 1.5 times more than PC group: Using TouchDevelop made it more likely that a task will be finished correctly. For example, the percentage of subjects finishing task 3 was 50% for TouchDevelop but less than 30% for Android.

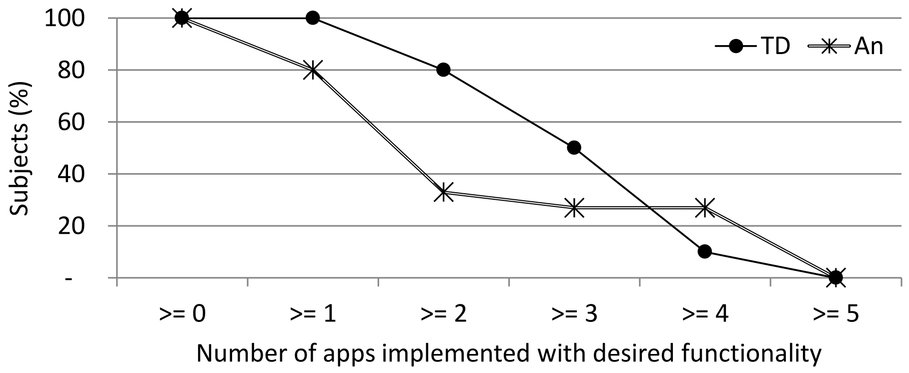


Figure 5.4. Number of apps implemented with desired functionality per subject: The percentage of TouchDevelop subjects, which finish 1,2,3 apps, is higher than Androids. However, there are more subjects that finish four applications than TouchDevelop subjects. From our observation, when the complexity of apps increases, the ability to reuse previous apps helps Android participants to achieve more correct apps.

CHAPTER 6

OBSERVATIONS AND OPEN QUESTIONS

One subject reported that the TouchDevelop IDE crashed ("When few lines of code were cut then TD crashed, had to retype"). From observing the subjects during the experiment we assume that this was not a problem for other subjects.

All Android subjects used the interactive Android IDE wizard to generate a simple hello-world app. This made it relatively easy to complete task 1. Other tasks however asked for interactive programs. Several subjects struggled to make the necessary modifications to the generated Android app.

We did not ask TouchDevelop subjects to publish apps during the experiment and nobody did. However, subjects also did not adapt any of the existing sample apps that are part of the TouchDevelop IDE. Maybe subjects did not judge it worthwhile to reuse an existing sample app because TouchDevelop apps are more compact, so there is less boilerplate code that can be reused?

More generally, we propose to investigate the following open questions, using appropriate user evaluation techniques.

- How do subjects reuse code between tasks?
- What code editing and other actions do subjects perform when working on their apps?
- How would our results change for tasks that require larger programs?
- How would our results change if subjects received more (or less) training in Android or TouchDevelop?

CHAPTER 7

THREATS TO VALIDITY

In this chapter is the explanation of our limits on internal validity that may bias our results and also the external validity to apply our study to other contexts. The following threats to validity are determined.

7.1 Our subjects are not a random sample

Our findings may not generalize well to the intended TouchDevelop audience, which extends beyond the UTA students taking CSE 5324 in Fall 2011 to any novice programmer world wide. I.e., we recruited subjects from a graduate course that clearly emphasized software engineering using Android and subjects self-selected this course. However, when selecting the course, subjects were not aware of this experiment.

7.2 Hands-off administration of tasks

We did not instruct subjects on how to work on the given tasks. Forcing subjects to work on certain tasks for a certain amount of time may produce results that are easier to compare. However, this level of control would prevent subjects from using a more natural programming style, in which subjects can switch between tasks and reuse solutions they found for a later task on an earlier task.

7.3 Result may not generalize to larger programs

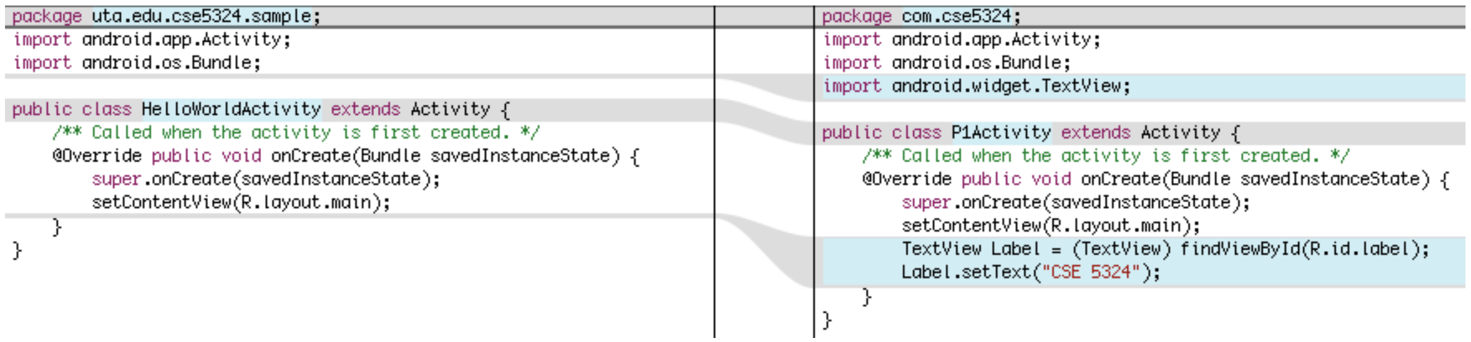
Due to the limited mobile phone screen size and the limited amount of time we had with the subjects, we designed all tasks to be simple and small. However, our study result may not be correct with large mobile tasks. With TouchDevelop, when program size increases, users may need more scrolling activities and navigation actions, which will likely slow down developers and lower programmer productivity.

CHAPTER 8

ADDITIONAL RESULTS

8.1 Reusing Android apps between tasks

In this section we analyze how subjects can reuse Android source code segments between tasks. Figure 8.1, 8.2, 8.3, and 8.4 are source code solutions of one of the Android subjects who correctly finished 4 apps. Via the post-experiment questionnaire, the subject has self-reported to have written from 8,000 to 10,000 lines of code (our highest value is 10,000). In Figure 8.1, subjects can add one more Android TextView label to the HelloWorld template (left) to output "CSE 5324" in program 1 (right). In Figure 8.2, subjects can modify the HelloWorld template (left) in two steps to finish program 2 (right): (1) update layout: add one text view (t1), one edit text (e1), and one button (b1); (2) add a method (onClick) to calculate Celsius degree. In Figure 8.3 subjects can modify the program 1 (left) in two steps to finish program 2 (right): (1) update layout: delete command set value for the text view (Label), add one edit text (e1), and one button (b1); (2) add a method (onClick) to calculate Celsius degree. In Figure 8.4, subjects only have to make small changes in program 2 such as: (1) add one edit text box and (2) modify equation of onClick function to output the result to successfully create program 3. In conclusion, if a subject starts by reusing the HelloWorld template to create program 1 and 2, then reuses program 2 to create program 3, the subject may successfully finish first 3 programs within the time limit. However, if the subject cannot finish program 2, it is very difficult for the subject to develop program 3 correctly.



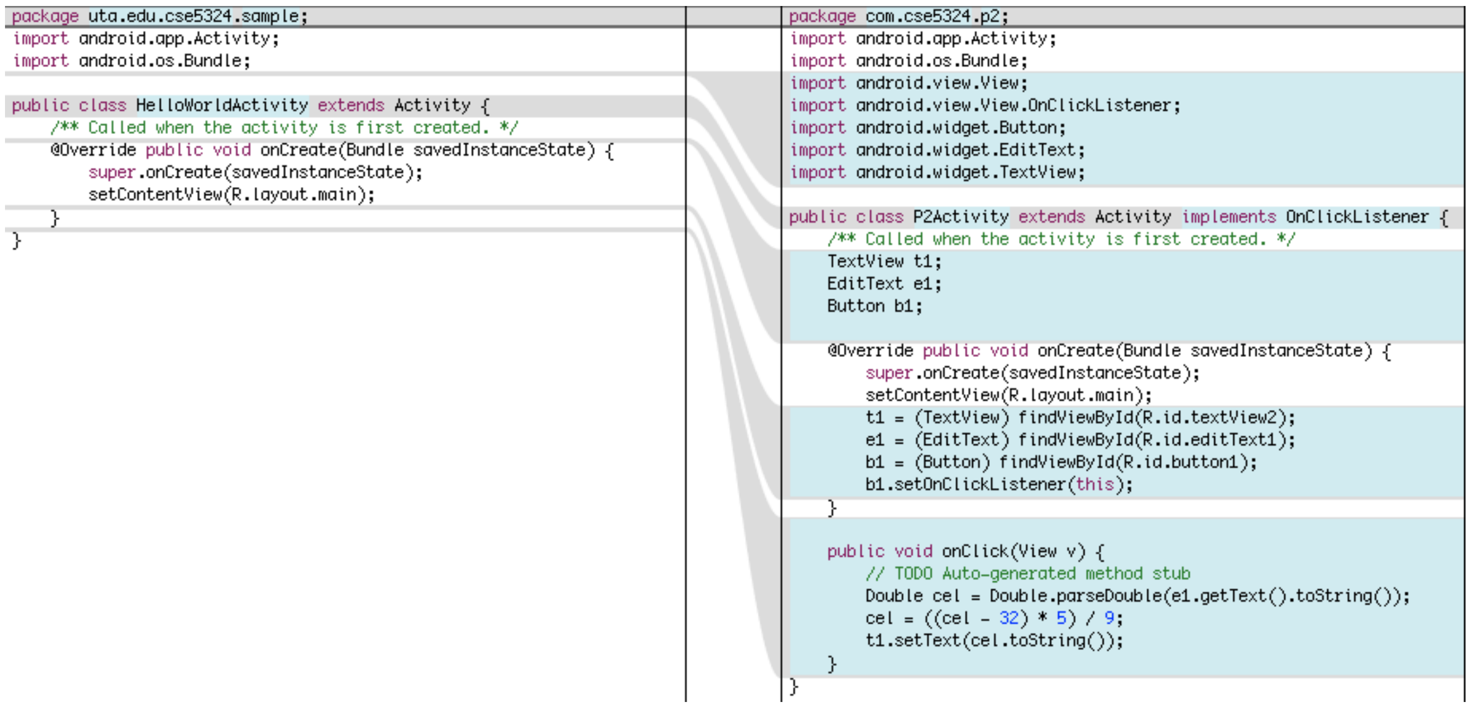
```
package uta.edu.cse5324.sample;
import android.app.Activity;
import android.os.Bundle;

public class HelloWorldActivity extends Activity {
    /** Called when the activity is first created. */
    @Override public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
}
```

```
package com.cse5324;
import android.app.Activity;
import android.os.Bundle;
import android.widget.TextView;

public class P1Activity extends Activity {
    /** Called when the activity is first created. */
    @Override public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        TextView Label = (TextView) findViewById(R.id.label);
        Label.setText("CSE 5324");
    }
}
```

Figure 8.1. Android apps - Comparing an example task P1 solution with the auto-generated hello-world app: One more label is added to HelloWorld template (left) to output "CSE 5324" in program 1 (right). The blue color indicates the strings, which are different, and the gray color indicates strings, which are the same.



```
package uta.edu.cse5324.sample;
import android.app.Activity;
import android.os.Bundle;

public class HelloWorldActivity extends Activity {
    /** Called when the activity is first created. */
    @Override public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
}
```

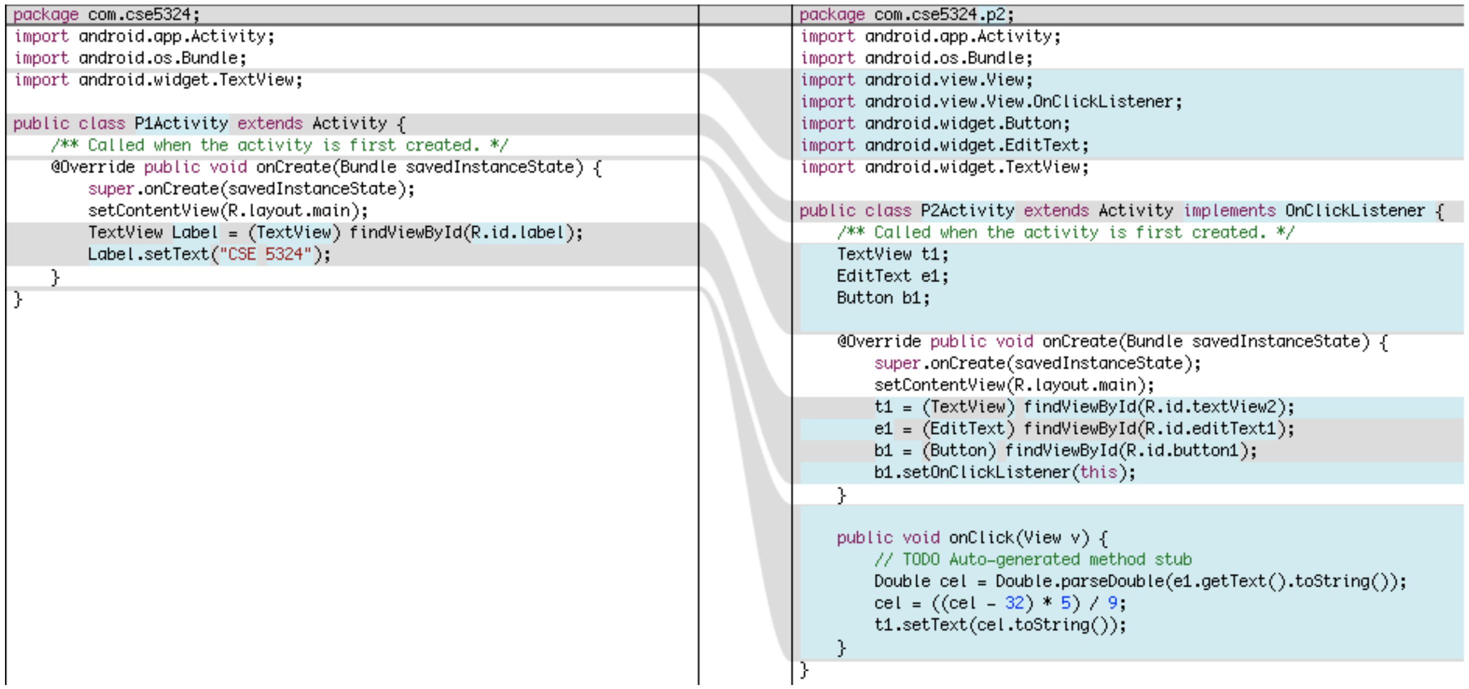
```
package com.cse5324.p2;
import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;

public class P2Activity extends Activity implements OnClickListener {
    /** Called when the activity is first created. */
    TextView t1;
    EditText e1;
    Button b1;

    @Override public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        t1 = (TextView) findViewById(R.id.textView2);
        e1 = (EditText) findViewById(R.id.editText1);
        b1 = (Button) findViewById(R.id.button1);
        b1.setOnClickListener(this);
    }

    public void onClick(View v) {
        // TODO Auto-generated method stub
        Double cel = Double.parseDouble(e1.getText().toString());
        cel = ((cel - 32) * 5) / 9;
        t1.setText(cel.toString());
    }
}
```

Figure 8.2. Android apps - Comparing an example task P2 solution with the auto-generated hello-world app: HelloWorld template (left) is modified in two steps to finish program 2 (right): (1) Layout is updated: One text view (t1), one edit text (e1), and one button (b1) are added; (2) a method (onClick) is added to calculate Celsius degree. The blue color indicates the strings, which are different, and the gray color indicates strings, which are the same.



```
package com.cse5324;
import android.app.Activity;
import android.os.Bundle;
import android.widget.TextView;

public class P1Activity extends Activity {
    /** Called when the activity is first created. */
    @Override public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        TextView Label = (TextView) findViewById(R.id.label);
        Label.setText("CSE 5324");
    }
}
```

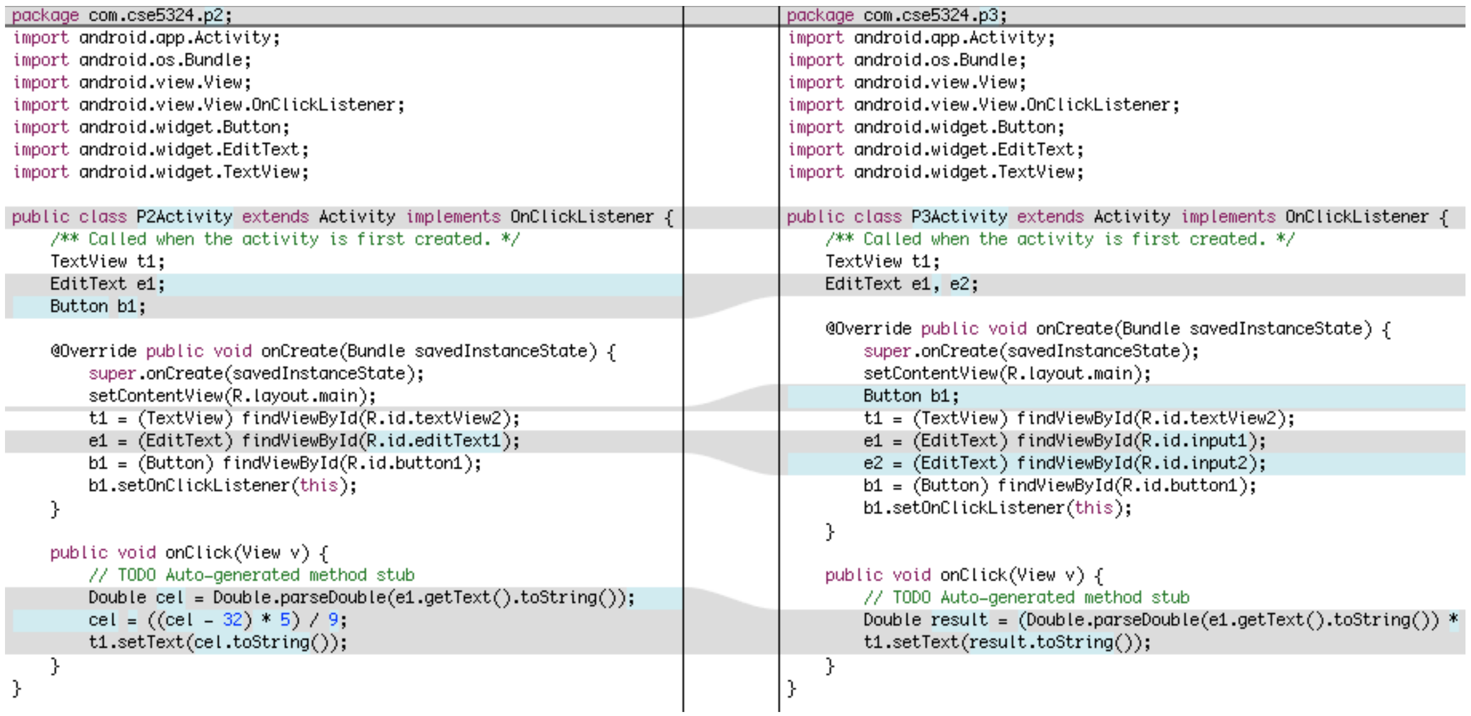
```
package com.cse5324.p2;
import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;

public class P2Activity extends Activity implements OnClickListener {
    /** Called when the activity is first created. */
    TextView t1;
    EditText e1;
    Button b1;

    @Override public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        t1 = (TextView) findViewById(R.id.textView2);
        e1 = (EditText) findViewById(R.id.editText1);
        b1 = (Button) findViewById(R.id.button1);
        b1.setOnClickListener(this);
    }

    public void onClick(View v) {
        // TODO Auto-generated method stub
        Double cel = Double.parseDouble(e1.getText().toString());
        cel = ((cel - 32) * 5) / 9;
        t1.setText(cel.toString());
    }
}
```

Figure 8.3. Android apps - Comparing an example task P1 solution with an example task P2 solution: Program 1 (left) is modified in two steps to finish program 2 (right): (1) update layout: delete command set value for the text view (Label), add one edit text (e1), and one button (b1); (2) add a method (onClick) to calculate Celsius degree. The blue color indicates the strings, which are different, and the gray color indicates strings, which are the same.



```java
package com.cse5324.p2;
import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;

public class P2Activity extends Activity implements OnClickListener {
    /** Called when the activity is first created. */
    TextView t1;
    EditText e1;
    Button b1;

    @Override public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        t1 = (TextView) findViewById(R.id.textView2);
        e1 = (EditText) findViewById(R.id.editText1);
        b1 = (Button) findViewById(R.id.button1);
        b1.setOnClickListener(this);
    }

    public void onClick(View v) {
        // TODO Auto-generated method stub
        Double cel = Double.parseDouble(e1.getText().toString());
        cel = ((cel - 32) * 5) / 9;
        t1.setText(cel.toString());
    }
}
```

```java
package com.cse5324.p3;
import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;

public class P3Activity extends Activity implements OnClickListener {
    /** Called when the activity is first created. */
    TextView t1;
    EditText e1, e2;

    @Override public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        Button b1;
        t1 = (TextView) findViewById(R.id.textView2);
        e1 = (EditText) findViewById(R.id.input1);
        e2 = (EditText) findViewById(R.id.input2);
        b1 = (Button) findViewById(R.id.button1);
        b1.setOnClickListener(this);
    }

    public void onClick(View v) {
        // TODO Auto-generated method stub
        Double result = (Double.parseDouble(e1.getText().toString()) *
        t1.setText(result.toString());
    }
}
```

Figure 8.4. Android apps - Comparing an example task P2 solution with an example task P3 solution: It is very simple to reuse source code of program 2 (left) to develop program 3 (right): add one edit text box and modify equation of onClick function to output the result. The blue color indicates the strings, which are different, and the gray color indicates strings, which are the same.

8.2 Success of Android SDK applications is coordinated with participants' prior experience but success of TouchDevelop applications is not coordinated with participants' prior experience

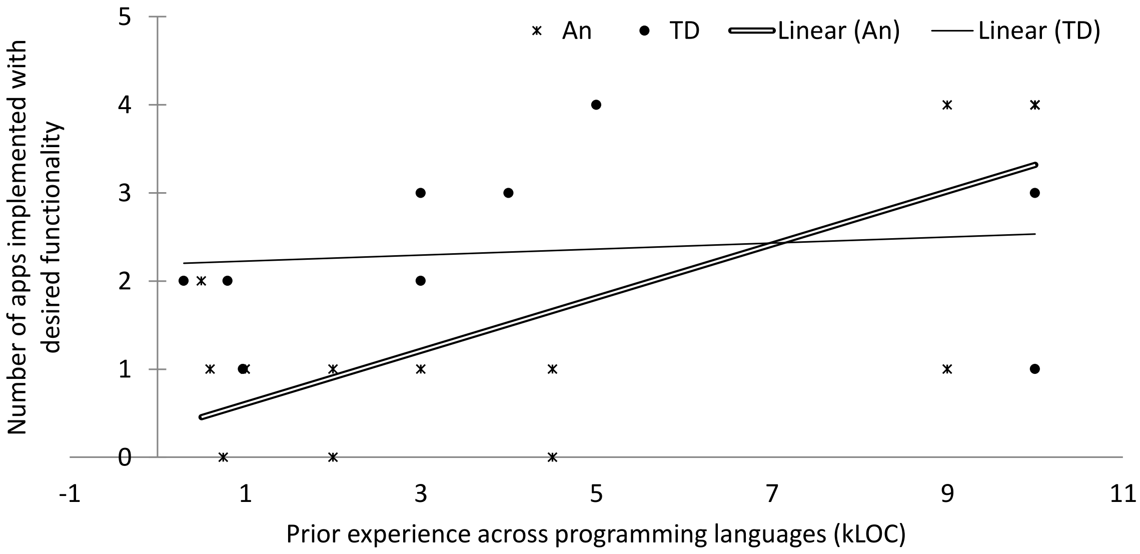


Figure 8.5. Links between the self-reported prior programming experience and the number of correctly implemented apps. For Android subjects, this link was stronger than for TouchDevelop subjects.

Four Android SDK subjects finished 4 applications. By analyzing the similarities of the subjects' app source code between tasks in Section 8.1, we notice that although Android SDK subjects write more LOC to finish an application, experienced subjects successfully reuse their previous correct applications as samples to develop new ones. This result helps to increase the application correctness of some Android subjects. In Figure 8.5, 3 out of these 4 subjects have the highest experience in the Android team. Moreover, the increasing "An" trend line of Android SDK team implies the correctness of application of Android team is directly proportional to their prior experience. On the other hand, the horizontal “TD” trend line of TouchDevelop team suggests that the correctness of application of TouchDevelop team does not originate from their prior experience.

8.3 Sources used by subjects and subjects’ comments

The tables 8.2 for Android and 8.3 for TouchDevelop are the questionnaire result in which subjects listed sources they used during the experiments and added their general comment about the experiments. The Table 8.1 shows the percentage of subjects that indicated in the questionnaire that they used the respective source. TouchDevelop subjects mainly used code samples. Android subjects mostly used API sources and web sources.

Table 8.1. The percentage of subjects using the respective source per group: The percentage of subjects that indicated in the questionnaire that they used the respective source. TouchDevelop subjects mainly used code samples. Android subjects mostly used API sources and web sources.

Source used	An(%)	TD(%)
Code samples	13	80
API	53	0
Web sources	53	20
Other sources	7	20

Table 8.2: Android: Sources used by subjects and subjects' comments

participant Id	Sources used: samples	Sources used: API doc	Sources used: examples from web	Sources used: other	Source was useful: samples	Source was useful: API doc	Source was useful examples from web	Source was useful: other	What did you enjoy in exercise?	Other comments
1		Yes					Yes			Could develop only the front end GUI in the allotted time
2		Yes				Yes			Developing Apps	
3		Yes					Yes		Coding	
4		Yes	Yes		Yes	Yes	Yes		Programming in Java	Eclipse is the best tool for Java Developement
5	developer .android .com								Everything	

6			Yes			pretty good but had to make changes		Developing the Apps		
7			Yes						Coding	
8			Yes			Yes			Developing GUI and working on new concept	
9		Yes				Yes				
10	Yes			Tutorials			Yes		Enjoyed removing errors from code	Lab session was fine, will be great if more time is allocated to lab session
11		Yes	Yes			Yes			Installing part and running app in emulator	More time was required to run the programs
12			Yes			Yes			Trying to code without taking help from internet	
13		Yes				Yes			Coding in Android	

14		Yes	Yes		Yes		Yes		Time limit	
15			Yes			Yes	Yes		Got interest in further developement in Android	Interesting exercise

Table 8.3: TouchDevelop: Sources used by subjects and subjects' comments

participant Id	Sources used: samples	Sources used: API doc	Sources used: examples from web	Sources used: other	Source was useful: samples	Source was useful: API doc	Source was useful examples from web	Source was useful: other	What did you enjoy in exercise?	Other comments
16	Yes						Yes		TouchDevelopement	Provide more time to write code
17	Yes				Yes				Programming	Nice Experiment
18	Yes				Yes				New Experience to develop in TD	

19	One sample video to get started				Easy to understand				It was a test of how quickly one can grasp the idea of new prgramming structure and I did well I thinng and it was nice	It was a good experience to work with Touch Develop
20									Programming, Exploring the IDE	
21	useful but difficult to interpret		Did not find easily	Will be useful if some help is provided about how to use the functions	Not very useful	Difficult to search on phone		Indication on how to develop in the touch develop interface will be helpful	Trying to figure out TD	Syntax is not very intuitive, may take time and patience to understand
22				took help from TA				help from TA	Learning TD	Enjoyed learning and working on TD

23	Yes				Did not have every-thing re-quired			More exam-ples in the web could help more	Developing apps in win-dows phone	
24	Yes								Learning basic syntax from samples and then applying them in code	Interesting exer-cise
25	Yes		Yes				Yes		First hands on experi-ence on TD	Hard to develop on TD as there was less info about the functions

CHAPTER 9

RELATED WORK

This chapter summaries studies, which are related to our study in different aspects. There is a survey of programming languages for novice programmers in [5] and TouchDevelop is a particular novice programming language. There evaluations of programming languages for non-programmers such as [6], [7], and [8]. Then there are empirical evaluations of other programming languages for novice programmers such as [9], [10], [11], and [12]. Moreover, there is a comparison between Android and iOS, an professional programming language which is similar to Android, in teaching environment [13]. There also is a study about problems when users use small display instead of large display screens as in [14]. However, TouchDevelop represents a new paradigm (coding on the phone) so there are no studies yet about TD-like environments.

9.1 Survey of end-user programming languages

Kelleher and Pausch [5] surveyed more than 80 articles about systems, which try to make programming language easier to learn for novice programmers of all ages. This survey describes two broad categories: (1) Teaching category designed to help people learn to program and (2) Empowering category designed to help people build programs as much as possible to tailor their own needs. They concluded their study with three main methods, which these systems used, such as (1) to simplifying the mechanics of programming (simplify entering code, finding a different way to input the program, replace coding by other activities, or improve programming languages), (2) to provide social supports for users (multiple students in informal groups to work together, or networked interaction), and (3) to inspire them to learn the language (entertainment or educational activity). In a similar way, TouchDevelop is designed with simple and compact programming language and its intelligent virtual input keyboard. TouchDevelop wants to target not only students but also hobbyists who write TouchDevelop apps for enjoyment and for personalizing their mobile phone [1].

9.2 Evaluations of programming languages for non-programmers

Pane and Ratanamahatana, Myers [6] studying solution which non-programmers create to solve the programing problems to create guideline for important features of future programming languages. In the experiment there are two groups of participants: ones is children, the other is a mix of adult and children, which have not programed before. Basing on common techniques and concepts of programming language, such as AND, OR, SUBSET, LOOP, or, SORT, the authors generated list of programming questions as movie scenarios. The participant can answers questions on papers by text or diagram. They claim that to solve a programming task, a programmer has to create a metal plan to solve the task before they apply this plan on particular programming language. They concluded in order to design a modern programming language, such as Java, with many features and requirements; there sometimes is a large distance between the mental plan and the programming language. Our study concludes the average correct Android solution was about four times larger in LOC than the average correct TouchDevelop solution. This seems that compact programming language environment, TouchDevelop, can help to reduce the distance between programmers' mental plan and the programming language (they coded less LOC). Moreover, The experiment also shows that TouchDevelop completed more correct tasks than Android subjects.

Myers, Pane, and Ko [7] propose a new programming language environment for children, which is close to natural language. The authors take advantage of prior results in human-computer interaction studied to create new programming language environment, which improves the usability of novice programmer fifth grade in creating programs and fixing bugs of programs. They conducted studies, which include developing video game, and simple business tasks, groups of non-programming and programming participant. Based on studies observations they design the new novice programming language environment, HANDS: is an event-based language, and includes natural queries, aggregate operators and animations. Both HANDS and TouchDevelop try to creates simple programming languages based on their observations on the way in which novice programmers executes programming tasks. However, HANDS uses animation approach, but TouchDevelop simplifies the syntax and structure of tradition language.

Resnick, Maloney, Monroy-Hernández, Rusk, Eastmond, Brennan, Millner, Rosenbaum, Silver, Silverman, Kafai [8] design a desktop tool, called Scratch, for children aged from 8 to 16 to learn programming. Its programming grammar is based on shaped blocks, and only blocks, which can fix with each other, make correct syntax. All source code of created applications is uploaded and shared with all users. After 27 months, the project is very successful with 500,000 projects (15% of projects are remixed of older ones) and about 1500 projects uploaded every day. TouchDevelop is designed in a similar way, and after 10 months, (from Apr-2011 to Feb-2012) it has 2,081 TouchDevelop apps published. Both of them have simple programming language environments, which help novice users efficiently understanding its syntax and structure. They design source statements using block, which eliminate many typing mistakes. Their online collaboration features (online discussion, public source code, original source code tracking, and derivative source code tracking) support and encourage participants developing new apps. However, because of the concept itself, TouchDevelop may require more mature users than Scratch. TouchDevelop also targets mobile platform, which is different from Scratch.

9.3 Empirical evaluations of programming languages for novice programmers

Boada, Soler, Prados, and Poch [9] develop web-based tool called ACMEp to help teachers teach introductory programming course to freshman programming students. Teachers can generate workbooks for students using ACMEp. Students can pseudo-code or many other language as solution to ACMEp and receive feedbacks. They had 120 students and each of these students uses this ACMEp tool. They are concerned about using a standard programming language vs. using pseudo-code. The two parts of their study are lecture (pseudo-code) and lab (standard programming language). Then they argue that using pseudo code first helps students use a standard programming language. Their main results come from questions / answers they use on students and teachers. On the other hand, our main results come from our controlled experiment. We also tried questions / answers but this did not give many interesting results.

DePasquale, Lee, and Pérez-Quiñones [10] evaluate the effects of applying programming language of novice programmers in three programming language environments: full programming language with complex user interface, full programming language with simple interface, and programming language subsets with simple interface. They had conducted an experiment with 165 students divided into three groups. Group 1 used a full IDE (.NET) throughout. Group 2 first used a simplified IDE (CS1 Sandbox) without language subsets and later a full IDE (.NET). Group 3 first used a simplified IDE (CS1 Sandbox) with language subsets and later a full IDE (.NET). Then they compared these three groups and found that grades were equivalent across the two groups. Our study compares student productivity and student solution correctness between Android and TouchDevelop. Android used Java, a popular and powerful programming language, and its desktop IDE is Eclipse with Android Development Tools plugin. On the hand, TouchDevelop use a simple and imperative programming language and its IDE also designed simple and compact to work with mobile phones [1].

Nanz, Torshizi, Pedroni, and Meyer [11] do experiment on 67 students of an architecture course to compare Java with SCOOP (Simple Concurrent Object-Oriented Programming) for concurrency. The experiment includes two phrases: (1) training phase, including self-study material and exercises, to help students be familiar with concurrent programming in both languages, (2) testing, including three tasks program comprehension, program debugging and program correctness, to evaluate the usability of Java and SCOOP. Although 67 students had multithread Java programming experience, they actually comprehend and debug programs better using SCOOP. However, The results were not statistically significant. Our research is similar to them in comparison between a well-known mobile programming language environment, Android, with new mobile programming language environment, TouchDevelop. In a specific kind of tasks, developing small mobile applications (theirs: developing object-oriented concurrent programming), our result shows that the new mobile programming language environment helps students improve their productivity. They assigned subjects randomly then used self-assessments and test questions to confirm split groups have similar backgrounds, which similar to what we did. We first random assignment and then use a questionnaire to figure out if the random assignment created groups of similar properties.

Browne, Werth, and Lee [12] conduct an experiment to evaluate CODE (Computation-Oriented Display Environment) with and without ROPE (Reusability-Oriented Parallel programming Environment) system based on two factors user productivity and software quality. After trained in use of CODE, 43 subjects who are graduated and senior students in science and computer engineering developed 25 parallel programs using CODE and 43 parallel programs using CODE with ROPE. Students, using CODE with ROPE, can access to a database with 70 software components to reuse. In conclusion, the authors argued that ROPE has signification effect on development time, and reduce error rates. In our experiment, all Android subjects reused the generated simple hello-world app to develop their apps. TouchDevelop did not reuse any apps still more productivity.

Hundhausen and Brown [15] developed a visual environment (ALVIS LIVE!), which helps novice programmers learn algorithm. Programmers can input simple pseudocode-like source code, and the tool can automatically evaluate the code and turn them into visual representation. They also conduct an experiment to validate the usability of "ALVIS LIVE!" on 21 novice programmers in one of their introductory programming computer science course. The experiment includes the following tasks: fixing a buggy algorithm (Find Max) and develop new one (Replace Zeroes). After the experiment, they reviewed the videotapes recorded during the experiment and the questionnaire shows that the tool got 7.3/10 points in addressing the participants learnability. They concluded "ALVIS LIVE!" is generally successful in empowering students to understanding their code and to develop correct algorithm solutions. However, they also point out some of its weaknesses. The concept of "ALVIS LIVE!" is similar to TouchDevelop concept. Both of them create new simple programming language environments to help novice programmer archive their own learning purposes. However, Alvis Live! does not work on phones. As authors pointed out they need to have a more rigorous experiment to prove the high usability of "ALVIS LIVE!". Their experiment lacks of comparisons with other similar language environment and statistical analysis on collected result.

9.4 Evaluations of programming languages for expert programmers

Goadrich and Rogers [13] compared advantages and disadvantages of between iOS and Android in teaching mobile development programming. They state iOS programmers can only develop applications by using Apple iFamily devices, but Android programmers can develop on any PC platforms such as: Mac OS, Windows and Linux. The authors mention programming using mobile virtual devices raises some difficulties in usability testing on both environments. They also suggest that these full environments more target upper-level computer science courses, and teachers should look for simpler alternative environments for programming introduction courses. TouchDevelop tried to shift the mobile development SDKs from PC to mobile device itself, which actually can increase usability testing. Moreover, its simple and compact programming language makes it easy to be used by novice programmers.

9.5 Small display

Jones, Marsden, Mohd-Nasir, Boone, and Buchanan [14] analyze how retrieving a task is affected when users shift from a big screen size (1024x768, 30 lines of content) to small screen size (640x480, 15 lines of content). They conducted an experiment with 20 subjects, assigned to two groups of 10 each. They discover that performance of simple tasks, such as reading or browsing, are not reduced when screen size is smaller. However, retrieving information task is 50% slower when users use a smaller screen. Users also make more incorrect data selection, and have more scrolling activities. The authors also suggest improvements such as: adding direct search features, improving navigation features, adding key information indexes, and reducing information on the page. TouchDevelop compacts source code to reduce displayed information. It replaces the virtual keyboard keys with appropriate identifiers (keywords, variables, function names) to improve searching and navigating performance. In addition, this experiment subjects used keyboard and mouse to interact, unlike multi-touch in TouchDevelop.

CHAPTER 10

CONCLUSIONS

Our findings indicate that (a) programmers so far have written TouchDevelop apps that are small and (b) a TouchDevelop app is smaller than a corresponding traditional mobile phone app (i.e., using Android). Surprisingly, (c), for small tasks, even with very little to no training in TouchDevelop, a student programmer who has prior Java and Android programming experience is still more productive in writing TouchDevelop apps on a phone than writing Android apps in a traditional PC-based fashion. Our web site contains additional details: `http://cseweb.uta.edu/~tuan/tdexp/`

APPENDIX A

POST-EXPERIMENT QUESTIONNAIRE

Following is the full questionnaire which we ask subjects to answer at the end of the experiment.

A.1 Questionnaire

1. How many of the following have you done before this exercise?
    (a) Lines of code written, counting all programming languages (do not include plain html, but include JavaScript, C, C++, C#, Java, etc.)
    (b) Lines of (non-Android) Java code written
    (c) Lines of C# code written
    (d) Lines of Java for Android code written
    (e) Lines of TouchDevelop code written
    (f) Hours spent working with Eclipse (write Java code, etc..)
    (g) Hours spent learning TouchDevelop (watch video, read website, api, etc.)
    (h) Hours spent learning Android (watch video, read website, api, etc.)
2. In completing this exercise, which problems did you encounter?
    (a) Preparing the IDE, emulator, etc
    (b) Developing particular apps
    (c) Loading apps into the device
    (d) Other (please elaborat)
3. In completing this exercise, which sources did you use (web sites, etc.)?
    (a) Samples that were part of the tool
    (b) Official API documentation
    (c) Examples found on the web
    (d) Other (please elaborate)

4. Comparing these sources with other documentation you have used in the past, how useful were the sources you used in this experiment?
   (a) Samples that were part of the tool
   (b) Official API documentation
   (c) Examples found on the web
   (d) Other (please elaborate)
5. Which aspects of this exercise did you particularly enjoy?
6. Please let us know any additional comments you may have.

APPENDIX B

INSTRUCTION FOR SUBJECT AND APP DEVELOPMENT TASKS

Following is the full instructions, including the necessary setup tasks and the verbatim list of programming tasks we assigned to the subjects.

B.1 Android

1. What your instructor and TAs have done to the PC before class?
   (a) Install on the PC a JVM and Eclipse.
   (b) Install on the PC the Android SDK.
2. What you have to do in class?
   (a) During the class, do not communicate (talk, email, etc.) with anyone except the TAs.
   (b) Type the following address into the Firefox address bar http://dl.google.com/android/ADT-12.0.0.zip
   (c) Start Eclipse to install the Eclipse ADT plugin
      i. Help > Install New Software... > Add > Archive > Favorites > Downloads > ADT-12.0.0 > Ok
      ii. Select Developer Tools > next, next, accept terms, finish, ok, restart now
      iii. Window > Preferences...> Android > enter into "SDK Location":
         A. C:\Program Files\Android\android-sdk
   (d) Create and start an Android Virtual Device
      i. In Eclipse: Window > Android SDK and AVD Manager > Virtual Devices > New... > Name: device, Target: Android 1.6 > Create AVD
      ii. Start the Android emulator from the command line, in Windows: Click the Windows start button in the lower left corner > in the "search programs and files box" enter:

- cmd > cd “\Program Files\Android\android-sdk\tools” > emulator-avd device

(e) Use the Eclipse ADT plugin to develop the following apps and install them on the Android emulator

   i. Any “Hello World” program that prints “CSE 5324” on the screen. Name this program P1.

   ii. A program that takes as input an integer number representing degrees Fahrenheit, converts it to degrees Celsius (using the Fahrenheit to Celsius conversion rule: deduct 32, multiply by 5, then divide by 9), and prints the resulting value. Name this program P2.

   iii. A tip calculator that takes as input two integer numbers A, B from the user and prints the value of A*B/100. Name this program P3.

   iv. A program that takes as input an integer number and prints “even” if it is an even number and “odd” if it is an odd number. Name this program P4.

   v. A program that takes as input a string and a character, prints “contains” if the string contains the character or else prints “not in there”. Name this program P5.

   vi. A program that takes as input a string and prints out the string with first character in uppercase. Name this program P6.

   vii. A program that prints the system’s current time as text. Name this program P7.

   viii. A program that asks the user for a positive integer value n and prints odd numbers between 0 and n (including n if n is odd). Name this program P8.

   ix. A program that takes as input a string that consists of numbers separated by commas. The program should output the numbers in increasing order. Name this program P9.

x. A program that draws a circle on the screen. Name this program P10.

xi. A program that takes two strings as input and checks if they are equal. Name this program P11.

3. Notes
   (a) It is okay to look at the sample programs and use them as a basis for your apps.
   (b) It is also okay to surf web sites. The API documentation is at:
      i. http://developer.android.com/reference/packages.html.
   (c) During this exercise, do not talk with other students. The goal is to simulate individual development.
   (d) Do not log off from, shut down, or restart the PC! After you log off, the C: drive will be rewritten and all your programs will be lost. Instead, email the source code of the apps you developed to us (the whole project so we can re-compile it, do not just send the .java files): rumee.csedu@gmail.com; csallner@uta.edu

B.2 TouchDevelop

1. What your instructor and TAs have done to the phone before class?
   (a) Charge phone, so we do not need to distribute chargers in class.
   (b) Phone setup (language = English, accept phone terms, custom phone settings, next, Central Time zone, next, set current date and time, skip entry of Windows Live ID: not now as no Wifi connection
   (c) pin settings to start screen: top right arrow > bottom of list > press settings long > pin to start
   (d) settings > ringtones and sounds > ringer off, vibrate off
2. What you have to do in class?
   (a) Prepare:

    i. Read and sign Microsoft lease agreement paper form. Only after we receive the signed form we will hand out a phone.
    ii. During this exercise, do not use a PC, personal laptop, or other smart-phone that you may have brought to class. You may surf web pages, but only on the Windows Phone that was assigned to you.
    iii. During this exercise, do not communicate (talk, email, etc.) with anyone except the TAs.

(b) On the phone, you can always use the following two keys:
    i. The home key is at the bottom, center, directly under the Samsung sign. It takes you to the phones start screen.
    ii. The back key is to the left of the home key. The back key takes you to the previous screen.

(c) Wifi setup: bottom of page > settings > Wi-Fi > UTA Auto Login > enter you UTA netid + password (you may need to try this up to three times, depending on the quality of the network)

(d) Enter the below Live ID: settings > email and accounts > add an account > Windows Live > next:
    - Enter your passport id: <1 of our 10 passport ids>
    - And password: <corresponding password>

(e) Download TouchDevelop: Marketplace > bottom right search key: ”Touchdevelop” > install > allow > install

(f) TouchDevelop: top right arrow > bottom of list > TouchDevelop > I agree > start

(g) Use TouchDevelop to develop the following apps.
    i. Any “Hello World” program that prints “CSE 5324” on the screen. Name this program P1.

ii. A program that takes as input an integer number representing degrees Fahrenheit, converts it to degrees Celsius (using the Fahrenheit to Celsius conversion rule: deduct 32, multiply by 5, then divide by 9), and prints the resulting value. Name this program P2.

iii. A tip calculator that takes as input two integer numbers A, B from the user and prints the value of A*B/100. Name this program P3.

iv. A program that takes as input an integer number and prints “even” if it is an even number and “odd” if it is an odd number. Name this program P4.

v. A program that takes as input a string and a character, prints “contains” if the string contains the character or else prints “not in there”. Name this program P5.

vi. A program that takes as input a string and prints out the string with first character in uppercase. Name this program P6.

vii. A program that prints the system’s current time as text. Name this program P7.

viii. A program that asks the user for a positive integer value n and prints odd numbers between 0 and n (including n if n is odd). Name this program P8.

ix. A program that takes as input a string that consists of numbers separated by commas. The program should output the numbers in increasing order. Name this program P9.

x. A program that draws a circle on the screen. Name this program P10.

xi. A program that takes two strings as input and checks if they are equal. Name this program P11.

3. Notes

(a) It is okay to look at the sample programs in TouchDevelop and use them as a basis for your apps.

(b) It is also okay to surf web sites. The API documentation is at:

   i. http://www.touchdevelop.com/help/api

(c) During this exercise, do not talk with other students. The goal is to simulate individual development.

(d) The TA will collect the phones and remove all personal information, i.e., remove your WiFi setup. After studying your apps, we will reset the phone to the default factory setting.

REFERENCES

[1] N. Tillmann, M. Moskal, J. de Halleux, and M. Fähndrich, "Touchdevelop: Programming cloud-connected mobile devices via touchscreen," in *Proc. 10th SIGPLAN ONWARD*. ACM, 2011, pp. 49–60.

[2] T. A. Nguyen, S. T. Rumee, C. Csallner, and N. Tillmann, "An experiment in developing small mobile phone applications comparing on-phone to off-phone development," in *Proc. 1st International Workshop on User Evaluation for Software Engineering Researchers (USER)*. IEEE, 2012, pp. 9–12.

[3] Google Inc., "Android developers." http://developer.android.com/index.html, 2012, accessed Jan. 2012.

[4] R. E. Park, "Software Size Measurement: A Framework for Counting Source Statements," Software Engineering Institute, Tech. Rep., 1992.

[5] C. Kelleher and R. Pausch, "Lowering the barriers to programming: A taxonomy of programming environments and languages for novice programmers," *ACM Computing Surveys*, vol. 37, no. 2, pp. 83–137, 2005.

[6] J. F. PANE, C. . RATANAMAHATANA, and B. A. MYERS, "Studying the language and structure in non-programmers' solutions to programming problems," *International Journal of Human-Computer Studies*, vol. 54, pp. 237 – 264, 2001.

[7] B. A. Myers, J. F. Pane, and A. Ko, "Natural programming languages and environments," *Commun. ACM*, vol. 47, no. 9, pp. 47–52, 2004.

[8] M. Resnick, J. Maloney, A. Monroy-Hernández, N. Rusk, E. Eastmond, K. Brennan, A. Millner, E. Rosenbaum, J. Silver, B. Silverman, and Y. Kafai, "Scratch: programming for all," *Commun. ACM*, vol. 52, no. 11, pp. 60–67, 2009.

[9] I. Boada, J. Soler, F. Prados, and J. Poch, “A teaching/learning support tool for introductory programming courses,” in *Information Technology Based Higher Education and Training, 2004. ITHET 2004. Proc. of the FIfth International Conference on*, 2004, pp. 604 – 609.

[10] P. DePasquale, J. A. N. Lee, and M. A. Pérez-Quiñones, “Evaluation of subsetting programming language elements in a novice’s programming environment,” in *Proc. of the 35th SIGCSE technical symposium on Computer science education*. ACM, 2004, pp. 260–264.

[11] S. Nanz, F. Torshizi, M. Pedroni, and B. Meyer, “Design of an Empirical Study for Comparing the Usability of Concurrent Programming Languages,” *Empirical Software Engineering and Measurement (ESEM), 2011 International Symposium on*, pp. 325–334, 2011.

[12] J. C. Browne, J. Werth, and T. Lee, “Experimental evaluation of a reusability-oriented parallel programming environment,” *IEEE Trans. Softw. Eng.*, vol. 16, no. 2, pp. 111–120, 1990.

[13] M. H. Goadrich and M. P. Rogers, “Smart smartphone development: iOS versus android,” in *SIGCSE ’11: Proc. of the 42nd ACM technical symposium on Computer science education*, 2011.

[14] M. Jones, G. Marsden, N. Mohd-Nasir, K. Boone, and G. Buchanan, “Improving Web interaction on small displays,” in *WWW ’99: Proc. of the eighth international conference on World Wide Web*, 1999.

[15] C. D. Hundhausen and J. L. Brown, “What you see is what you code: A ”live” algorithm development and visualization environment for novice learners,” *J. Vis. Lang. Comput.*, vol. 18, no. 1, pp. 22–47, 2007.

BIOGRAPHICAL STATEMENT

Tuan A. Nguyen was born in Can Tho, Vietnam, in 1984. He received his B.S. degree from University of Science Ho Chi Minh City, The National University, Vietnam, in 2007. He works three years in FPT Software in six outsourcing projects with Hitachi and Nissan customers. He started his Masters degree in Computer Science in 2010 in University of Texas at Arlington. He will continue his PhD in Fall 2012. His current research interest is in mobile software engineering and software reverse engineering. His hobbies include programming, exercising, reading news, and contributing to open source communities.