

# A Comprehensive Field Study of End-User Programming on Mobile Devices

Sihan Li

North Carolina State University  
Raleigh NC, USA  
sli20@ncsu.edu

Tao Xie

North Carolina State University  
Raleigh NC, USA  
xie@csc.ncsu.edu

Nikolai Tillmann

Microsoft Research  
Redmond WA, USA  
nikolait@microsoft.com

**Abstract**—TouchDevelop represents a new programming environment that enables users to develop mobile applications directly on mobile devices. TouchDevelop has successfully drawn a huge number of end users, who have published thousands of TouchDevelop scripts online. To enhance end-user programming on mobile devices, we conduct a comprehensive field study of 17322 TouchDevelop scripts and 4275 users. Our study consists of an overall study on the characteristics of scripts (e.g., structural features, code reuse) and users (e.g., expertise), and a longitudinal study on how they evolve over time. Our study results show important characteristics of scripts such as dense external method calls, high code-reuse ratio, and also reveal interesting evolution patterns of users. The findings and implications in our study provide valuable guidelines for improving tool support or services for end users and increasing the popularity of end-user programming on mobile devices.

## I. INTRODUCTION

In recent years, mobile devices, especially smartphones, have become a prevalent computing platform for most people. A report [1] shows that there were 835 million smartphone users in 2011, and the number will keep increasing. As the usage of mobile devices grows, it becomes a very common activity to create mobile applications (also known as *apps*). Around one million apps have been offered by the Apple and Android stores for download [2].

Traditionally, developers use object-oriented programming languages (e.g., Java, Objective-C, C#) to write programs on their separate PCs, with an emulator to simulate the device environment, and then deploy the programs as apps to mobile devices. Such a manner of off-device development manner requires the installation of a Software Development Kit (SDK) on the PCs, advanced programming languages, and complex deployment of the apps to mobile devices. These requirements pose high barriers to end-user programming of apps.

TouchDevelop [3], [4] developed by Microsoft Research represents an emerging development model for mobile applications. It lowers the end-user programming barriers by enabling on-device app development and providing a fairly simple (e.g., allowing only built-in types) scripting language. With TouchDevelop, anyone can program their mobile device directly on this device; no PC is required for developing or deploying TouchDevelop scripts (i.e., apps). The target users of TouchDevelop are students, hobbyists, power users, and developers. The declared scope of TouchDevelop scripts is for fun (e.g., games), for personalizing mobile devices, and for creating productivity tools (e.g., arranging meetings).

Due to such a radically new programming environment, TouchDevelop brings up numerous important questions on end-user programming on mobile devices. For example, from the perspective of apps, what kinds of apps can be created with TouchDevelop? What are unique characteristics of the apps created with TouchDevelop, compared to those created with the traditional PC-based approach? From the perspective of end users, how do end users learn programming with TouchDevelop? What are their programming behaviors? Answers to these questions may not only provide important guidelines for enhancing end-user programming and improving tool support or services to end users, but also reveal valuable opportunities for drawing more end users and increasing the popularity of on-device app development. Although there have been some studies on the functionalities and code changes of TouchDevelop scripts [5] and the programming productivity with TouchDevelop [6], a lot of questions still remain unaddressed, calling for more investigation.

In this paper, we present a comprehensive field study of end-user programming on mobile devices. Our goal is to provide valuable implications for different stakeholders around mobile-device programming, e.g., the improvement of tool support and services for end users, the increase of usage popularity for service providers, and research opportunities for researchers. Our study is comprehensive in that it investigates all the TouchDevelop scripts and users in the cloud (in total, 17322 scripts and 4275 users). The study consists of two parts: an overall study on the characteristics of scripts and users, and a longitudinal study on their evolution over time. The first part analyzes different aspects of both scripts (e.g., structural features and code reuse) and users (e.g., expertise) to provide basic understanding of them. The second part studies how scripts and users evolve over time, and attempts to find trends or identify patterns of their evolution. Specifically, our study intends to answer the following research questions:

### RQ1. What do TouchDevelop scripts look like?

We are interested in the structural features of TouchDevelop scripts such as the number of lines of code, defined methods, and method calls per script.

### RQ2. What is the code-reuse ratio of TouchDevelop scripts?

It is important to understand code reuse of TouchDevelop scripts since code reuse becomes very helpful in on-device development for reducing programming burdens on end users due to the virtual keyboard and limited screen size.

### RQ3. What kinds of TouchDevelop users are there?

It is useful to distinguish different kinds of users based on their

TABLE I. OUR MAJOR FINDINGS OF END-USER PROGRAMMING ON MOBILE DEVICES AND THEIR IMPLICATIONS

Characteristics of scripts and users	Implications
(1)-RQ1 TouchDevelop scripts are typically small; 72.6% of them are less than 100 LOC. However, there are still 5.4% relatively large scripts with more than 500 LOC. The largest script has 5030 LOC and is a rich-feature game.	Users usually write small scripts with TouchDevelop, but it is possible to use TouchDevelop to create large scripts with rich features.
(2)-RQ1 On average, a TouchDevelop script contains 106 external method calls (e.g., built-in TouchDevelop API method calls); every two lines of code contains at least one external method call.	TouchDevelop scripts heavily rely on external methods to achieve functionalities. Enriching the built-in APIs might help further increase the diversity of the scripts and the popularity of TouchDevelop.
(3)-RQ2 TouchDevelop scripts have a high code-reuse ratio: 57.9% of them have a parent <sup>1</sup> .	Users tend to modify existing scripts instead of creating scripts from scratch.
(4)-RQ2 In many cases (74.3%), users reuse their own code, i.e., having their script's parent as a script written by themselves. Among 50 sampled parent/child pairs, most child scripts (78.0%) are just updated versions of their parents with small modifications.	There is lack of code reuse across different users. A code-search engine of the code base may help users find others' scripts. Since there are many updates on scripts, a version-control system might be necessary for managing a series of scripts (different versions) for the same app.
(5)-RQ3 In terms of expertise, 74.2% of the users are novices, 21.6% of the users are ordinary users, and 4.2% of the users are experts.	Engaging these novices would help increase the popularity of TouchDevelop. Since each kind of users has a considerable population, it would be important to provide customized services to each kind of users, e.g., different UIs.
Evolution of scripts and users over time	Implications
(6)-RQ4 The number of scripts published per time period (nearly 3 months) increases with an average rate of 58.8%.	TouchDevelop is increasingly popular.
(7)-RQ4 The average size of the scripts in each time period stays small and stable, but the number of relatively large scripts increases with an average rate of 153.3%.	It is important to better support users to create large scripts. It might be helpful to make the TouchDevelop language more powerful and expressive or enrich the built-in APIs.
(8)-RQ4 On average, the code-reuse ratio increases by 3.6%; the number of scripts created as libraries <sup>2</sup> increases with an average rate of 63.0%.	Code reuse is becoming popular. Since more libraries are created by users, it is necessary to provide ways to discover them, document them, and detect duplicates.
(9)-RQ5 In terms of publishing scripts, 22.1% of the users are very active initially and become less active later. 9.6% of the users are not very active initially but become more active some time later. 68.3% of the users try publishing one or two scripts and then stop.	It would be useful to conduct user studies on why many users try TouchDevelop and leave, why those active users become less active, and what motivates those earlier less active users to become more active later. Such information could be used to design better strategies to retain users.
(10)-RQ5 16.3% of the users learn most language features initially and learn only a few later. 12.4% of the users learn some language features initially and also learn quite a lot at a certain point later. 71.3% of the users learn a few initially and then stop learning new language features.	Users have different learning behaviors. It would be important to provide an adaptive tutoring system that recommends tutorials to users based on their history of language-feature usage and the kind of scripts that they are writing.

expertise so that we can provide customized supports for each kind of users and better assess their scripts or comments.

#### RQ4. How are TouchDevelop scripts changing over time?

Finding the trends of how TouchDevelop scripts change in terms of their size and code reuse could guide future tool development to better accommodate these trends.

#### RQ5. What is the users' progress of developing TouchDevelop scripts?

We want to investigate users' programming behaviors and learning process by looking at their published scripts and language-feature usage in order to design strategies for engaging more users.

<sup>1</sup>A TouchDevelop script can inherit another script and directly modify its code. The script being inherited is called the parent.

<sup>2</sup>A TouchDevelop script can be created as a library for other scripts to reuse its defined methods. Without causing confusion, in this paper, we refer to such scripts as libraries.

Our major findings and implications are summarized in Table I. The details of our study are discussed in the later sections. Our paper makes the following main contributions,

- We discover special characteristics of TouchDevelop scripts: TouchDevelop scripts are typically small and contain a lot of external method calls; the code-reuse ratio of TouchDevelop scripts is high.
- We reveal important trends of the scripts including the increasing number of large scripts and code-reuse ratio.
- We find some evolution patterns of TouchDevelop users. There are consistently active users, but many initially active users become less active later.
- We provide valuable implications for improving future tool support or services for end users, such as building a code-search engine and a version-control system. We also suggest more user studies to figure out how to increase the popularity of end-user programming on mobile devices.

```

action main()
  // Finds songs not played yet.
  var found := 0
  var songs := media->songs
  for each song in songs
    where true
  do
    found := found + ▷display song(song)
    ("Songs played with this script: " || ↗played) → post to wall
    ("Songs never played: " || found) → post to wall

event active song changed
  // Increment the song played counter.
  ↗played := ↗played + 1

action display song(song : Song) returns result : Number
  // Post a song to the wall if not played yet and returns 1;
  // otherwise returns 0.
  if song->play count = 0 then
    song->post to wall
    result := 1
  else
    result := 0

```

Fig. 1. A "new song" example that finds songs not played yet and posts them to the wall.

## II. BACKGROUND: TOUCHDEVELOP

### A. The TouchDevelop Programming Environment

TouchDevelop [3], [4] developed by Microsoft Research, is a novel programming environment that enables users to program their mobile devices directly on the devices. TouchDevelop scripts are written and run in the TouchDevelop IDE, which can be installed on either a Windows Phone or a web browser for other platforms such as Android and iOS. To ease programming on mobile devices, the IDE provides a semi-structured code editor, which presents users with a small number of possible choices (e.g., different types of statement) at each step. In this manner, the users build the skeleton of the script via touching to select choices, leaving only variable names and expressions for typing.

Besides the IDE, TouchDevelop also provides a fairly simple scripting language that has both imperative and object-oriented characteristics. It allows users to not only define methods, modify local variables, but also access properties of objects. However, for simplicity, the language supports only built-in types: primitives and a set of types for objects. It does not allow users to define new types or properties. One important feature of the TouchDevelop language is that it provides rich interfaces to access a wide range of hardware and resources on the mobile device including various sensors, the camera, media, etc. Such feature facilitates the on-device programming.

A TouchDevelop script typically consists of *actions*, *events*, and *global data*. An action is just like a function in C or a method in Java or C#. It contains a sequence of statements, and can take parameters and return computation results. An event is a special action that is executed whenever the corresponding event occurs (e.g., phone shaking). The global data refer to global variables that are either persistent on a mobile device or in the cloud. Such variables can be shared by multiple

applications. Figure 1 shows a simple TouchDevelop script that finds songs not played yet and posts them to the screen, also known as the wall. The script contains two actions and one event. The *main* action is the entry of the script and calls the *display song* action to determine whether a song has been played. The *active song changed* event is executed whenever the player changes a song. This event increases the global variable *played*, the counter of songs played, by one.

TouchDevelop maintains a cloud that provides various services to users. First, users could publish/download their scripts to/from the cloud. All published scripts are available with source code, and users are encouraged to share their scripts or extend others' scripts. Each script/user is assigned a unique ID once published/registered. Moreover, the cloud stores many important statistics of every published script and its author, and provides public interfaces (cloud API) for retrieving the information of scripts and users. Anyone could write customized queries (e.g., retrieving IDs of all scripts published by a certain user) in a program or directly in the address bar of web browsers, and then the cloud responds to these queries with either JSON objects or plain texts.

### B. TouchDevelop Scripts and Users

From a database point of view, scripts and users are two critical types of entities in the TouchDevelop ecosystem. There is a one-to-many relation between users and scripts: one user can publish multiple scripts while one script can have only one user as its author. The script entity and the user entity also have a set of attributes. We next describe some key attributes that are investigated in our study of scripts and users, respectively.

For the script entity, we primarily investigate the attributes related to code reuse. TouchDevelop provides four ways for code reuse: (1) a script could inherit a parent script and directly modify the code of the parent script; (2) a script can be created as a library, and other scripts can reference the library script and use the actions and events defined in it; (3) a script can reuse the TouchDevelop built-in APIs, which are the properties and methods of object types provided by TouchDevelop; (4) a user can copy and paste code from one script to another. Since we do not have a good way to keep track of which portion of the code in the published scripts is via copy-and-paste, our study focus on only the first three code-reuse fashions. These three fashions respectively correspond to three attributes of a script: whether the script has a parent, whether the script is a library, and how many built-in API calls the script includes.

For a user, we investigate three important attributes that reflect the expertise of the user. First, the number of the published scripts indicates the script-publishing activity of the user. Second, users can give positive reviews to each other on their published scripts and posted comments. Hence, the number of the received positive reviews indicates the quality of the user's scripts and comments. Third, TouchDevelop keeps track of what TouchDevelop-language features are used by each user. The TouchDevelop-language features include TouchDevelop built-in APIs as well as basic programming-language concepts and keywords such as the concept of using parameters in an action and the *if* keyword for if statement. The number of language features used by a user indicates how much the user has learned about TouchDevelop programming.

TABLE II. METRICS ON THE SOURCE CODE OF A TOUCHDEVELOP SCRIPT

Metric	Description
#LOC	the number of lines of source code
#IM	the number of internal methods defined by users, i.e., actions and events
#EM	the number of external methods invoked by the script but defined outside the script
#IMC	the number of internal method call sites
#EMC	the number of external method call sites
EMC Density	the average number of external method calls per line of code, i.e., EMC/LOC

### III. METHODOLOGY

#### A. Subjects

The subjects in our study are all the TouchDevelop scripts published in the cloud and users who publish these scripts, starting from the publish time of the first script (late July 2011) to the time of our final experiment (early February 2013). In total, there are 17322 TouchDevelop scripts and 4275 users included in our study. For each script, we download the following resources: the source code, ID, and publish time of the script, the ID of the script author, the parent of the script (if any), and information on whether the script is a library. For each user, we download the ID of the user, the number of scripts published by the user, the number of TouchDevelop-language features used by the user, and the number of positive reviews given to the user's scripts and comments. We obtain all these data from the TouchDevelop cloud through the cloud APIs.

#### B. Metrics and Approaches

To answer RQ1 on the structural features of TouchDevelop scripts, we use a set of commonly-used metrics listed in Table II. All these metrics are calculated on every single TouchDevelop script. The *internal methods* refer to the actions and events defined in the script by the user, while the *external methods* refer to interfaces and actions that are invoked by the script but defined outside the script. Typically, external methods are TouchDevelop built-in APIs or actions defined in other library scripts. Note that the number of internal/external methods being measured here is actually the number of distinct internal/external methods defined/invoked in the script. While an internal/external method call is a call site of the method. There could be multiple call sites of a single method in the script. We construct the abstract syntax tree (AST) for each script using the TouchDevelop script parser, and traverse the AST to compute IM, EM, IMC, and EMC.

To answer RQ2 on the code reuse of TouchDevelop scripts, we adopt the same definition of code-reuse ratio from the previous work [5] for comparison purposes. The code-reuse ratio among scripts is defined as the ratio of the number of scripts that had a parent to the number of all scripts. To further study how users reuse code, we first investigate where the reused code comes from: the user herself or other users. This reused-code ownership is determined by checking whether the author of the parent script is the same as the author of the child script. We then calculate the percentage of newly

modified code in the child script. To do it, we obtain the LOC of modifications in the child script by using an internal cloud API, which compares the code differences between two TouchDevelop scripts, and compute the ratio of the LOC of the modifications to the LOC of the child script. Moreover, to see how users modify the inherited scripts, we randomly sample 50 parent/child script pairs, and manually go through the source code of each pair to understand the code modifications.

To answer RQ3 on characteristics of TouchDevelop users, we first classify users based on their expertise. We employ the clustering algorithm based on the Gaussian Mixture Model (GMM) [7] to group users with similar expertise, and choose three important indicators of the user expertise as the input attributes to the clustering algorithm, namely, the number of scripts published by the user, the number of TouchDevelop-language features used by the user, and the number of positive reviews given to the user's scripts and comments. We use Bayesian Information Criteria (BIC) to estimate the number of clusters in the input instead of arbitrarily choosing a number. BIC has been shown to be effective for Community Question Answering (CQA) datasets to determine how many users should be labeled as experts [8]. After clustering, we then analyze the characteristics of the users from each cluster and compare different clusters.

To answer RQ4 on the evolution of TouchDevelop scripts, we first equally divide the whole time period, ranging from the publish time of the first script to the publish time of the last script, into 6 smaller time periods (nearly 3 months per time period). Then we compare scripts from each time period in different aspects to find the trends of the script evolution. For example, we investigate how the number and the size of the scripts in each time period change over time, and whether the code-reuse ratio increases or not.

To answer RQ5 on the programming progress of TouchDevelop users, we investigate how many scripts or TouchDevelop-language features are published or used over time by each individual user. The number of published scripts could indicate the script-publishing activity of the user while the number of newly used TouchDevelop-language features could reflect how much they have learned about TouchDevelop. Particularly, we select users who started publishing scripts more than one year ago, and collect the published scripts and newly used language features in each of their first 12 months (starting from the time when each user published their first script). We use the number of published scripts or newly used language features in each month as 12 attributes and again employ the GMM clustering to group users with similar behaviors. After clustering, we analyze each cluster to identify common evolution patterns of users. We adopt this approach from previous work by Pal *et al.* [9], where they successfully used it to capture evolution patterns of experts in CQA.

## IV. CHARACTERISTICS OF SCRIPTS AND USERS

#### A. Structural Features of TouchDevelop Scripts

Since TouchDevelop provides its own scripting language, we want to see what TouchDevelop scripts look like and what features they have, compared to programs written in other languages from a structural perspective. We apply the set of software metrics listed in Table II to the source code of all

TABLE III. RESULTS OF STRUCTURAL METRICS FOR ALL TOUCHDEVELOP SCRIPTS

#LOC	Percent	#IM	Percent	#EM	Percent	#IMC	Percent	#EMC	Percent
(0, 100]	72.6%	1	23.1%	[0, 10]	44.6%	0	53.1%	[0, 50]	66.2%
(100, 500]	22.0%	(1, 10]	60.5%	(10, 50]	41.8%	(0, 10]	30.3%	(50, 100]	12.4%
(500, 1000]	2.8%	(10, 50]	14.8%	(50, 100]	10.0%	(10, 50]	13.1%	(100, 500]	16.6%
> 1000	2.6%	> 50	1.6%	> 100	3.6%	> 50	3.5%	> 500	4.8%
Avg.	133.1	Avg.	6.6	Avg.	23.2	Avg.	8.6	Avg.	105.5

TouchDevelop scripts, and find some special characteristics on their structural features.

Table III shows the results of each metric for all scripts. First, TouchDevelop scripts are typically small and simple. In the LOC column, the majority of the scripts (72.6%) have less than 100 LOC; the average number of LOC is only 133.1. In the IM and IMC columns, 23.1% of the scripts have only one internal method (i.e., each of these scripts has only the default main action); and 53.1% of the scripts have no internal method call (i.e., only the code in the main action is executed). These results indicate that users usually write small scripts with TouchDevelop, and TouchDevelop might be suitable for writing small-scale scripts. However, there are still 5.4% relatively large scripts with more than 500 LOC. The largest TouchDevelop script has 5030 LOC and 103 actions, and is a rich-feature game. This finding implies that it is entirely possible to use TouchDevelop to create large scripts with rich features.

From Table III, we can find that there are lots of external method calls (EMCs) in TouchDevelop scripts. The average number of EMCs in a script is 105.5, much larger than the average number of IMCs. These EMCs are primarily built-in API methods provided by TouchDevelop (e.g., the properties and methods of TouchDevelop built-in object types) and also a few actions defined in other library scripts. Furthermore, the average EMC density of all scripts is 0.6, which is very high, meaning that on average there is at least one EMC in every two lines of code. In addition, we also find a strong correlation between the number of lines of code and the number of EMCs. On the entire dataset, the correlation coefficient is 0.9.

These results are within our expectation because in end-user programming, users often write very few algorithms but many high-level operations. The implementations of these operations are usually provided by TouchDevelop built-in APIs. These results imply that a lot of code logic lies outside the scripts, and the scripts heavily rely on built-in TouchDevelop APIs to achieve functionalities. Hence, enriching the built-in APIs might help further increase the diversity of the scripts and the popularity of TouchDevelop.

### B. Code Reuse in TouchDevelop Scripts

Due to the virtual keyboard and limited screen size, programming on a mobile device is more difficult than programming on a PC. Thus, code reuse becomes very helpful for reducing programming burdens on end users. Overall, the code-reuse ratio of TouchDevelop scripts is high: 57.9% of all the scripts have a parent script. By inheriting a parent script, the child script can be attained by directly modifying the code of the parent script instead of starting from scratch. Such high code-reuse ratio is reasonable in the case of on-device

programming where users intend to save the typing efforts. It may take less time for users to find the needed code and understand it than to implement it by their own.

To delve deeper into where the reused code comes from, we find that among all the parent/child pairs, there are 74.3% of them with the parent and the child sharing the same author. This result indicates that in many code-reuse scenarios, the user just reuses her own code. There is a lack of code reuse across different users. There are two inhibitors that might prevent users from reusing others' code: (1) there is no effective way for users to find out others' code that they need to reuse; (2) there are few descriptions for published code, and users may have a hard time in understanding others' code without documentation. To alleviate these problems, a code-search engine of the entire code base might help users to effectively find the code that they need. In addition, although typing is relatively difficult on mobile devices, users could still add some brief descriptions on their code to help other users to understand it, and thus increase the chance of the code being reused.

We then compute the percentage of the modified code in the child script in terms of LOC. We find that on average, for each child script, the modified code accounts for only 8.5% of the entire code. This result indicates that the modifications on the parent scripts are relatively small. Furthermore, from the 50 sampled parent/child pairs, there are 39 out of 50 (78.0%) child scripts that are just updated versions of their parents with small modifications. The average ratio of the modified code per script among the 39 child scripts is 7.9%. Many modifications do not implement substantial functionalities.

Since we observe that there are frequent updates on scripts, we suggest providing a version-control system for managing these updates. Currently, users update their scripts in a code-reuse manner by inheriting code from the previous script, modifying it, and publishing the updated script as an individual app. This way might not be good to maintain code because intuitively, these updated scripts should be treated as different versions of the same app rather than different individual apps. Users may get confused on whether the child script is an updated version of its parent or a completely new app. With a version-control system, these confusions can be alleviated because updated scripts are grouped together for one app. In addition, users can also have a clear overview of how this app evolves, and then easily pick the version that they want to modify.

### C. User Classification

Figure 2 shows the results of the GMM clustering on 4275 users, with the number of published scripts, the number of used TouchDevelop-language features, and the number of received

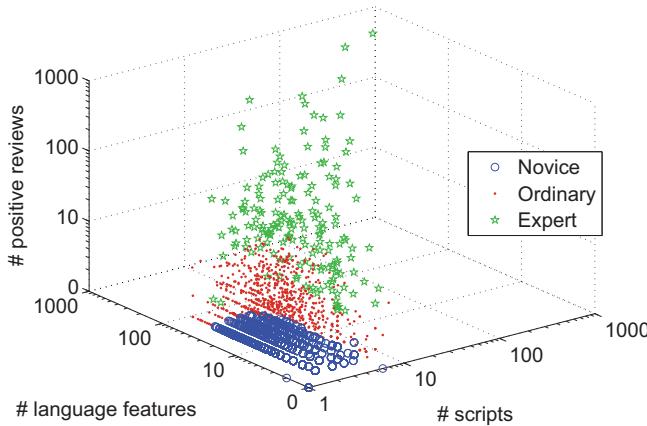


Fig. 2. The classification of users based on their expertise. The stars represent the experts, the dots represent the ordinary users, and the small circles represent the novices.

positive reviews as the input attributes. The figure is plotted on the log scale. Overall, the users are clustered into three groups. We regard the group of users with most published scripts, language features, and positive reviews as experts (represented by stars), the group with least of these metric values as novices (represented by small circles), and the remaining group as ordinary users (represented by dots).

In particular, there are 74.2% novices, 21.6% ordinary users, and 4.2% experts. Although the expert group is relatively small, each group of users has a considerable population. As shown in Figure 2, there are big differences between each kind of users on these three metrics. For example, some experts use hundreds of language features but many novices use only a few. Since we have three very different kinds of users each with a considerable population, it would be important to provide customized services rather than uniform services to each kind of users. For example, we may recommend tutorials very often to novices, but much less frequently to experts. We may also keep the UI for experts compact whereas provide a detailed UI with more descriptions for novices. Moreover, we can see that the novice group, which has the largest population, is rather inactive in general: many of them publish fewer than 10 scripts. It would be very useful to investigate these users and understand what keeps them as novices, e.g., lack of interest in TouchDevelop or difficulties in learning TouchDevelop. Engaging these novices would help TouchDevelop further increase its popularity.

## V. EVOLUTION OF SCRIPTS AND USERS OVER TIME

### A. How Do Scripts Change Over Time?

As described in Section III, we divide the entire time period into 6 smaller time periods, with an equal length of nearly three months. The entire time period starts from the publish time of the very first TouchDevelop script and ends at the time we conducted our experiment. We apply the same set of previous metrics separately on the scripts from each of the 6 time periods and compare the results across time periods to see their changes. Table IV shows the results of metrics that have an increasing trend over time periods. The first column lists the 6 time periods. The other columns represent the number

TABLE IV. INCREASING TRENDS OF SCRIPTS OVER 6 TIME PERIODS

	#Script	#L. Script	#Library	Reuse R.
$T_1$	939	13	0	43.9%
$T_2$	1701	95	0	53.3%
$T_3$	2252	150	257	57.2%
$T_4$	3171	147	377	59.3%
$T_5$	2586	230	175	58.2%
$T_6$	6673	284	518	62.1%
Avg Inc.	58.8%	153.3%	63.0%	3.6%

of published scripts, the number of scripts with more than 500 LOC, the number of library scripts, and the code-reuse ratio in each time period, respectively. The last row represents the average increase rate of each metric over 6 periods. Note that the rate here is not the exponential growth rate. Instead, for each time period (except the first one), we calculate the increase or decrease rate on its previous period, and then use the mean value of all such rates as the average rate.

In general, the number of published scripts per time period increases with an average rate of 58.8%. Besides the number of published scripts, the number of new users per time period also increases with an average rate of 65.8%. The increasing numbers of both the scripts and the new users indicate the growing popularity of TouchDevelop.

From the structural perspective, the average size of the scripts in each time period stays small and does not change much. However, there are more and more relatively large scripts published in each time period, with an average increase rate of 153.3% as shown in the third column. This trend indicates the growing popularity of writing large scripts with TouchDevelop. We suggest providing better support for writing large scripts in the future. For example, we could make the TouchDevelop language more powerful and expressive so that it could allow users to create more sophisticated scripts. Besides, enriching the built-in APIs might also be helpful.

On the aspect of code reuse, the ratio of code reuse has an average increase of 3.6% per time period, implying the growth of popularity of code reuse. In addition, there is an increasing trend in the number of library scripts. In Table IV, the number of library keeps increasing generally. Note that there is no library script in the first two time periods because the feature of creating library scripts was not introduced at that time. As there are more and more library scripts being created, we need to find an effective way to manage them. For example, in order for users to find their desired libraries, it is necessary to provide search support of library scripts. We also need to provide documents for the libraries in order to help users to better reuse them. Moreover, since any user can create library scripts, it is important to ensure the quality of the libraries and also detect duplicated libraries.

### B. How Do Users Evolve?

To investigate how users evolve, we need users that have been using TouchDevelop for long enough time. We select those who started publishing scripts more than one year ago and find 816 such users. We apply GMM to cluster these 816 users based on the number of their published scripts and the number of their newly used language features, respectively.

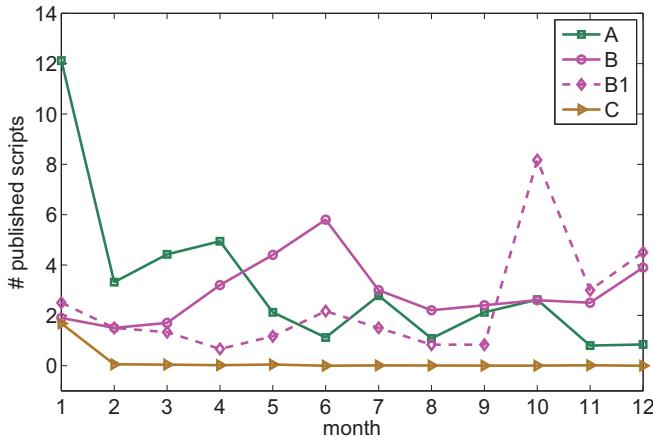


Fig. 3. Evolution patterns on the number of the published scripts of the 816 selected users.

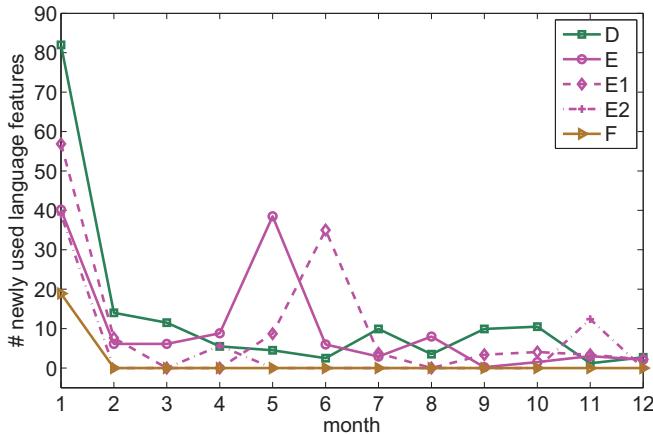


Fig. 4. Evolution patterns on the number of the newly used language features of the 816 selected users.

Figure 3 shows the mean number of users' published scripts in their first 12 months for different clusters. Each line represents a cluster and each data point is the mean number of the published scripts in a certain month by each user from a cluster. Although there are 4 clusters found by GMM, they can be summarized into 3 patterns because some clusters are just variants of each other and they belong to the same pattern. In Figure 3, Line A represents a pattern that 22.1% of the users are very active initially, publishing many scripts, and become less active later, publishing only a few. Lines B and B1 represent another pattern that 9.6% of the users are not very active initially but become more active at some time later. Line C represents the third pattern that 68.3% of the users try publishing one or two scripts initially and then stop publishing. Since so many users belong to the third pattern, it is very important to know why these users stop publishing so that we could design better strategies or improve TouchDevelop to retain these users. In addition, knowing what motivates those earlier-less-active users to become more active later may also give hints to increase popularity.

Figure 4 shows the mean number of users' newly used language features for different clusters. Note that the clusters in Figures 3 and 4 are not the same because the clustering algorithm is applied to two independent data sets. There

are also 3 patterns of the users' learning behaviors. Line D represents the first pattern that 16.3% of the users learn most language features initially and learn only a few later. Lines E, E1, and E2 represent the second pattern that 12.4% of the users learn some language features initially and also learn quite a lot at a certain point later. Line F represents the third pattern that 71.3% of the users learn a few language features initially and then stop learning. We further sample 20 users from the second pattern and investigate why there is an increase later in their learning curve. We find the major reason to be that users switch their focus from writing one kind of scripts to another so that they use many new built-in APIs related to the new kind of scripts. Since the users have different learning behaviors, we suggest providing an adaptive tutoring system that recommends tutorials related to the kind of scripts that a user is writing and also avoids tutorials that the user already knows based on the language features used by the user.

## VI. THREATS TO VALIDITY

One threat to internal validity is that we do not investigate code reuse via copy-and-paste due to the difficulty of tracking users' edits. Copy-and-paste could be common in the scripts. Hence, the actual code-reuse ratio is underestimated in this study. Another threat is in our manual study on 50 sampled parent/child script pairs and scripts of 20 sampled users. Due to the inevitable human efforts, there might be subjectiveness in understanding the code modifications between the parent and the child. In addition, since TouchDevelop later released a web version, some of our subject scripts may be created with the browser on PCs. These scripts do not belong to the category of apps that are developed on mobile devices. Fortunately, we did our experiment shortly after the release of the web version. So there would not be many such scripts included in our study.

The major threat to external validity is that we conduct our study exclusively with TouchDevelop. Some of our findings and implications are specific to the TouchDevelop platform due to TouchDevelop's own scripting language, and might not be generalizable to other platforms for on-device app development [10]. To mitigate this threat, we plan to conduct studies on different platforms in our future work.

## VII. RELATED WORK

The most related work is an earlier study conducted by Athreya *et al.* [5] on TouchDevelop scripts. Both their work and our work studied the code-reuse ratio of TouchDevelop scripts and code modifications between parent scripts and child scripts. We had similar results on code modifications: most of the modifications were minor tweaks to existing functionalities. However, our results on code-reuse ratio were quite different. In their study, they inferred that the code-reuse ratio was only 5% because they had to randomly sample about 2000 scripts until they found 100 scripts with a parent. However, our results indicated that 57.9% of all scripts had a parent. We informed Athreya *et al.* of this difference and together figured out two possible reasons: the code-reuse ratio of scripts was increasing; their sampling might be an "unlucky" one, which sampled more scripts without a parent. Besides different results, their study focused on functionalities of scripts and problems posted by users, whereas our work mainly studied the structural features of scripts and the evolution of scripts

and users. Moreover, their entire study was done manually so that they investigated only a small portion of scripts. Our study was more comprehensive by including all scripts and users.

There are some other studies on end-user programming with TouchDevelop. Nguyen *et al.* [6] conducted a user study to compare programming productivity of TouchDevelop with the traditional off-device approach. They found that for small tasks, a programmer was more productive in writing TouchDevelop apps than writing Android apps. Tillmann *et al.* [11], [12] presented their successful experience on teaching middle and high school students programming using TouchDevelop, and proposed to conduct future programming teaching to mobile devices.

Besides the preceding studies, there is some other work related to TouchDevelop on security and programming language. Xiao *et al.* [13] used static-information-flow analysis to reveal how private information was used inside apps in order to assist users in granting permissions to apps. Burckhardt *et al.* [14] designed specialized cloud types at the programming-language level to achieve consistent data storage for mobile devices. Our study results might serve as motivations for future research on end-user programming on mobile devices.

A lot of research has been done by the data-mining community on the users in question-answering forums. Much such research focused on developing algorithms or approaches to either identify experts or analyze the behaviors of experts. Bouguessa *et al.* [8] developed an approach leveraging the Bayesian Information Criterion to determine experts in the Yahoo! Answers forum. Pal *et al.* [9] proposed an approach to identify the evolution patterns of experts in the Stack Overflow forum. We borrowed their approaches in our study to classify TouchDevelop users and analyze their evolution patterns.

There are plenty of studies on end-user programming in other domains including web applications [15], [16], spreadsheets [17], animations [18], and other domain-specific visual languages [19]. Many of them [15], [17], [18], [19] studied what kinds of programs end users create and what challenges they face, and gave suggestions to tackle these challenges, while others [16] conducted user studies on the behaviors of end users during their programming process.

## VIII. CONCLUSION AND FUTURE WORK

In this paper, we have presented a comprehensive field study of end-user programming on mobile devices. We studied 17322 TouchDevelop scripts and 4275 TouchDevelop users, investigating not only the characteristics of scripts and users, but also their evolution. Our findings included (1) TouchDevelop scripts are small and contain lots of external method calls; (2) the code-reuse ratio of TouchDevelop scripts is high; (3) many TouchDevelop users are novices; (4) there are increasing trends of code-reuse ratio and the number of large scripts; (5) TouchDevelop users have some patterns in publishing scripts and learning language features. Based on these findings, we have provided a list of implications for improving tool support or services for end users and increasing the popularity of end-user programming on mobile devices.

In future work, we plan to build the proposed code-search engine and version-control system into the TouchDevelop

cloud. We also plan to interview some users of different kinds on their experiences with TouchDevelop and design strategies to engage more users. Finally, we plan to generalize our current study to other similar platforms.

## ACKNOWLEDGMENTS

This work is supported in part by NSF grants CCF-0845272, CCF-0915400, CNS-0958235, CNS-1160603, an NSA Science of Security Lablet grant, a NIST grant, a Microsoft Research Software Engineering Innovation Foundation Award, and NSF of China No. 61228203. We thank Christopher Scaffidi for discussing with us on code reuse and providing some of their experimental data. We also thank Yuan Yao for his valuable feedback and help.

## REFERENCES

- [1] “The future of mobile,” Business Insider. [Online]. Available: <http://www.businessinsider.com/the-future-of-mobile-deck-2012-3#-1>
- [2] R. Minelli and M. Lanza, “Software analytics for mobile applications - insights & lessons learned,” in *CSMR*, 2013, pp. 144–153.
- [3] N. Tillmann, M. Moskal, J. de Halleux, and M. Fahndrich, “TouchDevelop: programming cloud-connected mobile devices via touchscreen,” in *SIGPLAN, ONWARD*, 2011, pp. 49–60.
- [4] N. Tillmann, M. Moskal, J. de Halleux, M. Fahndrich, and S. Burckhardt, “TouchDevelop: app development on mobile devices,” in *FSE, Demo*, 2012, pp. 39:1–39:2.
- [5] B. Athreya, F. Bahmani, A. Dieder, and C. Scaffidi, “End-user programmers on the loose: A study of programming on the phone for the phone,” in *VL/HCC*, 2012, pp. 75–82.
- [6] 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 *USER*, 2012, pp. 9–12.
- [7] C. Stauffer and W. E. L. Grimson, “Adaptive background mixture models for real-time tracking,” in *CVPR*, 1999, pp. 246–252.
- [8] M. Bouguessa, B. Dumoulin, and S. Wang, “Identifying authoritative actors in question-answering forums: the case of Yahoo! answers,” in *KDD*, 2008, pp. 866–874.
- [9] A. Pal, S. Chang, and J. Konstan, “Evolution of experts in question answering communities,” in *AAAI*, 2012, pp. 274–281.
- [10] “Appcelerator,” Titanium Studio. [Online]. Available: <http://www.appcelerator.com/>
- [11] N. Tillmann, M. Moskal, J. de Halleux, M. Fahndrich, J. Bishop, A. Samuel, and T. Xie, “The future of teaching programming is on mobile devices,” in *ITICSE*, 2012, pp. 156–161.
- [12] N. Tillmann, M. Moskal, J. de Halleux, M. Fahndrich, and T. Xie, “Engage your students by teaching computer science using only mobile devices with touchDevelop,” in *CSEE&T*, 2012, pp. 87–89.
- [13] X. Xiao, N. Tillmann, M. Fahndrich, J. De Halleux, and M. Moskal, “User-aware privacy control via extended static-information-flow analysis,” in *ASE*, 2012, pp. 80–89.
- [14] S. Burckhardt, M. Fähndrich, D. Leijen, and B. P. Wood, “Cloud types for eventual consistency,” in *ECOOP*, 2012, pp. 283–307.
- [15] C. Bogart, M. Burnett, A. Cypher, and C. Scaffidi, “End-user programming in the wild: A field study of coscripter scripts,” in *VL/HCC*, 2008, pp. 39–46.
- [16] N. Zang and M. B. Rosson, “What’s in a mashup? and why? Studying the perceptions of web-active end users,” in *VL/HCC*, 2008, pp. 31–38.
- [17] C. Chambers and C. Scaffidi, “Struggling to Excel: A field study of challenges faced by spreadsheet users,” in *VL/HCC*, 2010, pp. 187–194.
- [18] A. Dahotre, Y. Zhang, and C. Scaffidi, “A qualitative study of animation programming in the wild,” in *ESEM*, 2010, pp. 29:1–29:10.
- [19] M. Jones and C. Scaffidi, “Obstacles and opportunities with using visual and domain-specific languages in scientific programming,” in *VL/HCC*, 2011, pp. 9–16.