End-user programmers on the loose: A study of programming on the phone for the phone

Balaji Athreya, Faezeh Bahmani, Alex Diede, Chris Scaffidi
School of Electrical Engineering and Computer Science
Oregon State University
Corvallis, OR 97331 USA
{ athreyab, bahmanif, diedea, scaffidc }@onid.orst.edu

Abstract—**Microsoft TouchDevelop is a programming environment enabling users use their phones to create scripts that run on the mobile phones. This is achieved via a semi-structured editor and a programming language with several distinctive features, such as support for using smartphone hardware. In order to uncover opportunities for future tool development aimed at facilitating end-user programming of phones on phones, we have investigated the kinds of scripts that people are creating with the current tool set as well as what problems they ask for help with solving. This paper is the first to study how end-user programmers "in the wild" are programming mobile phones. In particular, no previous study has investigated the ways in which end users programmatically use mobile phones' special hardware (e.g., GPS, accelerometer, gyroscope) for practical everyday purposes. We discovered that, in essence, people are using TouchDevelop to create apps: downloadable applications with small, fairly reliable feature sets that take advantage of mobile hardware. In addition, we identified several areas for further innovation aimed at enhancing the programming tool and the online repository where users share scripts with one another.**

Keywords—human-centric computing; end-user programming; empirical studies; mobile computing

I. Introduction

If you could pull out your phone and program it to do something, what would you program it to do?

Until the past year, this question would have been largely hypothetical because there was no practical way for end users to create programs on their phone. Instead, they generally had to obtain a personal computer such as a laptop, install a large suite of programming tools, write a program in a C-like programming language (such as Objective-C, Java, C#, JavaScript, etc.), configure the phone to work with the development computer, and deploy the program to the phone via USB or another cable. These tools, languages and configuration steps posed extremely high barriers to end-user programming of phones.

Microsoft TouchDevelop is a new programming environment intended to greatly reduce these barriers so that anyone can use a phone to program the phone [14]. For example, a user could program her phone to send a text message when she arrives at a certain destination (e.g., to notify friends automatically when she arrives at a party). Consequently, TouchDevelop is intended to let users customize their phone's behavior to provide real-time support for their personal lives. In addition, the programming environment is full-featured enough that users can create more sophisticated scripts, such as games, and the environment includes an online repository called the "bazaar" containing thousands of scripts that users have posted for one another to reuse.

Because it opens up these numerous new programming affordances in the rapidly-growing domain of mobile computing, and because of its online bazaar of existing scripts, TouchDevelop offers a valuable opportunity to investigate key research questions whose answers could shape development efforts aimed at providing phone users with even more refined or more powerful programming tools. In order to guide future work in this area, we have analyzed scripts in the bazaar, as well as user comments on related online forums, to answer three research questions:

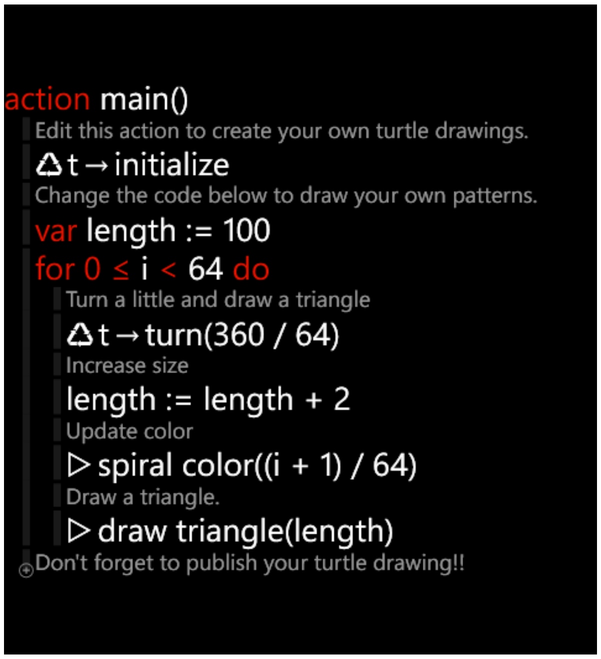
RQ1. What kinds of scripts have users posted? We are particularly interested in learning to what extent users are creating scripts that take advantage of the distinctive affordances of mobile phones, such as GPS and cameras. To answer this question, we analyzed the existing scripts that users have posted to the online TouchDevelop bazaar.

RQ2. How are TouchDevelop scripts changed over time? Since our overall goal is to guide tool development, we investigate how (and, implicitly, whether) people are using the existing TouchDevelop tools to modify scripts. We explored these issues by analyzing the modification logs of scripts in the online TouchDevelop bazaar.

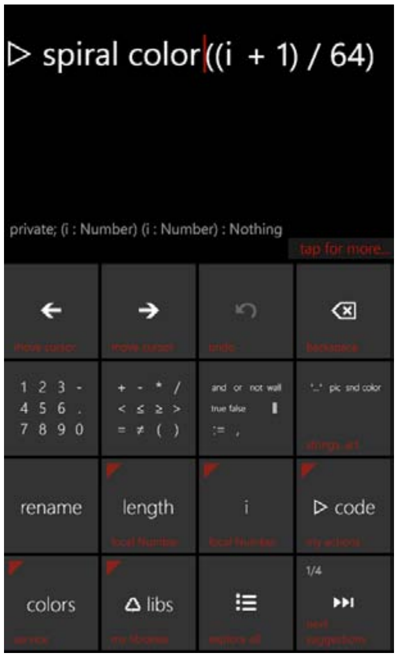
RQ3. What problems do users ask for help with solving? The complaints and questions of existing users can provide a valuable starting point for user-centered design of new and improved tools. We uncovered user problems with TouchDevelop by analyzing online forum discussions.

This paper is the first to study how end-user programmers "in the wild" are programming mobile phones. In particular, no previous study has investigated the ways in which end users programmatically use mobile phones' special hardware (e.g., GPS, accelerometer, gyroscope) for practical everyday purposes. Our investigation revealed interesting surprises. For example, the diversity of TouchDevelop programs was particularly noteworthy. In addition, we found TouchDevelop

We thank Microsoft Research for sponsoring this work by providing phones.



```
action main()
  Edit this action to create your own turtle drawings.
  ♻t → initialize
  Change the code below to draw your own patterns.
  var length := 100
  for 0 ≤ i < 64 do
    Turn a little and draw a triangle
    ♻t → turn(360 / 64)
    Increase size
    length := length + 2
    Update color
    ▷ spiral color((i + 1) / 64)
    Draw a triangle.
    ▷ draw triangle(length)
  Don't forget to publish your turtle drawing!!
```



```
▷ spiral color((i + 1) / 64)
private; (i : Number) (i : Number) : Nothing
```

Figure 1a (left): Sample TouchDevelop script, which draws a spiral with a turtle at the end of the spiral; the invoked actions (named "spiral color" and "draw triangle") are custom functions for drawing images onto the screen.
Figure 1b (right): User selected a line of code for editing, which brings up a menu of code for referencing common APIs.

has enabled users to create a wide diversity of programs that leverage mobile phones' unique affordances. These and other results are described below, after we first review the related work and summarize our research methods.

II. Background: Microsoft TouchDevelop

TouchDevelop is a new programming environment enabling end-user programmers to create scripts for their Windows-based smartphones [14]. Most existing tools such as Appcelerator Titanium [2] and Google/MIT App Inventor [7] require people to use another computer such as a desktop or laptop to create programs, which are then deployed to smartphones. In contrast, TouchDevelop allows users to create programs for the phone on the phone. A direct competitor for Android is the Google Scripting Layer [6], though this alternative does not include a large corpus of end-user programs for us to study.

Figure 1a shows an example of a TouchDevelop script provided by Microsoft as a training example. The code is organized into functions called actions. Prior to the `main()` action, a library for drawing a turtle is imported (not shown) and bound to the variable `t`. This library is initialized, and its `turn()` action is invoked within a loop. Custom actions, `spiral color()` and `draw triangle()`, are defined elsewhere (with a space in each action's name) and are invoked within the loop; these actions draw to the screen. The effect is ultimately to draw a turtle that traces a spiral path outward from the center of the screen.

The programming language has a few important features and constraints, some of which are demonstrated by this script.

- The language is primarily textual but, like a visual language, it uses a few non-ASCII graphical characters to represent some elements of the syntax. For example, an arrow → indicates a dereference of an object member, an arrow ▷ indicates a function/action invocation, and a recycling symbol ♻ precedes a reference to bound library object.
- Unlike most languages, the TouchDevelop language does not allow user-defined types or custom user interface controls (though these might become available in the future). The available types consist of the usual primitives (integers, string, etc) as well as a small set of types for objects, many of which are singletons (i.e., predefined instances of classes that cannot be further instantiated). The language supports events, event handlers, and other user-defined functions.
- It has native support for accessing all of the hardware widely available on mobile phones, some of which is not available on traditional desktop computers. For example, support is included for accessing GPS, gyroscope, accelerometer, camera, and microphone.
- The language includes support for storing truly global variables—such variables are stored on the cloud and accessible by programs on other phones.

Two significant challenges with programming on a phone are the limited screen space and cumbersome virtual keyboard. To overcome these challenges, the programming environment is designed to require little scrolling through programs and little textual input. This dual objective is accomplished by providing a semi-structured editor for creating scripts. Programmers select from a list of menu options, each of which corresponds to a kind of statement that can be added to the script. Once a statement is added to the script, then additional menus are available for filling in the pieces of the statement (Figure 1b). Thus, most of a script's abstract syntax tree is created through menu selections. The leaves and lower nodes of the syntax tree (such as variable names and expressions) can be edited using textual input. Programmers can edit a statement by touching it.

The TouchDevelop bazaar enables users to post scripts for one another to download. This repository appears to contain thousands of scripts and have thousands of users, but the exact size is not publicly disclosed. The bazaar also provides online textual areas where programs can display output; for example, the bazaar provides a "leaderboard" service so that games can post information about high scores.

Microsoft has used TouchDevelop as a basis for two research projects. One project was aimed at developing an algorithm that can detect when scripts are clones of one another so they can be clustered, in order to support a new search engine that allows users to browse through results by cluster [1]. In another research project, Microsoft has developed static analysis algorithms for determining whether a script "leaks" private information [16]. For example, this analysis can detect if a script might read a user's address book and post personal contact data to the web. Such an algorithm could be used to enhance the bazaar by analyzing uploaded scripts and issuing warnings to users before they download scripts that pose privacy risks.

Unlike in work, our focus is not on presenting particular new features for the TouchDevelop environment, but rather on exploring how TouchDevelop is used in practice: what kinds of scripts users are creating, how those scripts change over time, and what problems users ask for help with solving. This information can be used to uncover interesting use cases and opportunities for providing improved tools.

III. RELATED WORK

Ours is the first empirical investigation of scripts that end-user programmers are creating for mobile phones. However, there have been numerous prior empirical studies of the form, "What do end-user programmers seek to create in solution domain X, and what problems do they encounter?" These include studies in the context of spreadsheets [4], mashups [17], web and graphic design [10][12], web macros [3], and animations [5], as well as population-specific studies of teachers [15] and scientists [8][13]. Our study offers a chance to explore whether findings of prior studies generalize to the entirely new context of programming on the phone.

1) *Findings related to kinds of programs created*

Similar studies in other domains have uncovered numerous surprises and other interesting findings. For example:

- Among web macros in the CoScripter repository, some of most frequently-used were those for "mischievous" purposes such as *automatically playing online lotteries and games* [3].
- Users with high Technology Initiative (TI), as measured with questions about early technology adoption, wanted to create *different kinds of mashups* than users with low TI: those with high TI mostly wanted to mash up data and media, while those with low TI mostly wanted to create mashups to support socializing with other people [17].
- Web designers *create an huge variety* of different page behaviors: in one survey, 200 respondents reported needing to create 107 distinct behaviors, ranging from "an animated 'lens effect' list UI" to "a sliding dock" [10].
- A study of the EUSES Spreadsheet Corpus found that the *majority of spreadsheets were used to store information rather than to calculate* [4]—serving as small databases, rather than "programs" in the conventional sense.
- Most end-user animation programs in the repository for the Scratch tool *had no clear functional purpose at all* [5]. The majority of the others were games.

These and other studies illustrate that end-user programmers often use tools in ways that do not conform with our intentions or expectations. In our current study, we cannot guess what people are doing with TouchDevelop. Given the results of the studies above, TouchDevelop users might (for example) be mischievously writing scripts that place prank calls, creating scripts to socialize or to manipulate data, creating scripts with surprisingly complex behavior, using scripts to store and retrieve data without taking any real advantage of mobile computing affordances, or perhaps writing little of use at all.

2) *Findings related to reuse*

One finding that appears in several prior studies is the low level of code reuse among end-user programmers via repositories. One study found only 7% of web macros in a repository were ever run more than 6 times [3]. Another study found only 5% of Scratch animations were ever reused by another person to create a new animation [5]. A study of web page designers "found little evidence of code-scavenging" between people [12]. At best, as with teachers in another study [15], web page designers referred to other people's code in order to learn while creating new programs [12]. In contrast, reuse was higher of one's own code for graphic designers [10], web page designers [12], and scientists [8].

Several problems inhibit reuse of end-user programmers' code. First, compatibility problems plague some contexts, such as data formatting problems that require manual fixing during reuse [12]. Second, reuse may be inhibited when functions are not implemented in a generalized way [13], such as due to hardcoded values [12]. Third, reuse may be inhibited when few useful programs are available for reuse [5]. Fourth, the absence of commenting or other documentation impedes reuse, calling for reverse engineering of code's meaning (potentially with retrofitting of documentation) during reuse [8][13][15].

Given these results, we expect that reuse via the TouchDevelop bazaar might be low, but given the diversity of obstacles to reuse in other contexts, it is difficult in advance to know what problems might limit reuse of TouchDevelop scripts or, equally importantly, what actions might be required during reuse.

3) *Other challenges to end-user programming*

In addition to reuse-related problems, other challenges plague end-user and novice programming, though these problems varied in significance depending on context.

Web page designers often found it difficult to implement complex behaviors because a single small bug could cause the entire behavior to fail [10]. In contrast, an analysis of animations found that only 7% contained noticeable bugs [5]. Spreadsheets seemed moderately difficult to code correctly, with 25% or more of spreadsheets containing bugs in typical studies [11].

Team-related problems appeared in some contexts. Web page designers often worked in teams and depended on others to provide materials as well as periodic assistance [10][12]. In contrast, Scratch animation programmers showed little evidence of working in teams or of depending on the user community for help [5].

Finally, some programming environments appear to present challenges with using and combining APIs. For example, this was a problem with novice programmers learning to use Visual Basic APIs [9]. In contrast, Scratch animation programmers rarely asked for such help with APIs [5].

Our study offers an opportunity to see if TouchDevelop programmers struggle with problems similar to or different from these. Based on the results of our study, we will discuss general conclusions that can be drawn from the foregoing studies (above) and the present study.

IV. Methodological Overview

To obtain data for analysis, we downloaded randomly-selected sets of programs or user comments from the web, with one data set for each of our three research questions. In each section below, we describe how many programs and user comments we downloaded for answering the corresponding research question.

We applied qualitative analysis to develop a coding scheme based on the data rather than imposing an externally-determined coding scheme. Our procedure was as follows:

1. One researcher examined approximately 10% of the data and designed a categorization scheme, writing down coding rules for how to recognize members of each category.

2. Two researchers then independently applied that coding scheme to the dataset.

3. They compared their assignments and computed a reliability metric. When codes were mutually exclusive, they used Cohen's Kappa and also computed the percent of data items (scripts or user comments) where both researchers assigned the same code. In cases when multiple codes could be assigned, we used the Jaccard Coefficient.

4. Finally, the pair of researchers negotiated and resolved discrepancies. Based on a negotiated code assignment, we computed descriptive statistics to answer research questions.

The primary methodological limitation of this study approach is that it does not provide direct observations of people as they use the programming tool. Therefore, our study design is appropriate for characterizing the kinds of programs that people publish and the problems that they ask for help with solving, but we cannot speculate on what kinds of programs they are creating in private (i.e., not publishing to the bazaar), nor on the extent to which users' stated problems measurably slow down or otherwise curtail their creation of scripts.

V. What Kinds of Scripts Have Users Posted?

We analyzed 209 scripts from TouchDevelop's bazaar and discovered a total of 15 different program categories defined in terms of primary functional purpose. We agreed on 87% of assignments to these 15 categories (Cohen's Kappa 0.85) before negotiating and resolving discrepancies. Categories were mutually exclusive—to our surprise, very few scripts provided functionality corresponding to multiple categories, and even then the functionality clearly was predominantly of one specific category. For example, a few games provided a means for users to post scores to Facebook or other online sites, but the main functionality of the scripts were game-oriented, and the social functionality was completely subordinate.

Overall, there were three groups of script categories: fun/entertainment-related categories, non-entertaining utility categories, and a category of "no meaningful functionality" (NMF).

Approximately 38% of scripts fell into one of the fun/entertainment-related categories (Figure 1). Within this group of categories, the most commonly-occurring was *Games* (Table 1). Examples included rock-paper-scissor and tic-tac-toe. Such games were non-trivial but also not of the same complexity as games typically available for purchase from professional developers. Other scripts in the fun/entertainment-related categories provided convenient access to music, images, animation, or social activities.

In addition, we found 9 categories of non-entertainment utilities with a surprising amount of diversity. The most common of these were *Web Lookup* utilities, where the user entered a query or other information that was posted to the web to download other information that was then displayed. Examples included scripts that retrieved the latest panoramic image from the Bing.com homepage, and that retrieved the next San Francisco BART train arrival time. We also observed 2 interesting *Voice-Web Lookup* scripts that collected the initial query by voice and then looked up information on the web (e.g., by posting voice queries to a public Wolfram Alpha API).

As an indication of the diversity in the non-entertainment utilities, over 12% of all scripts clearly had some functionality but did not fall into one of the other categories. Examples included a program that could generate a random number when requested, and another that allowed the user to set a timer for an alarm.

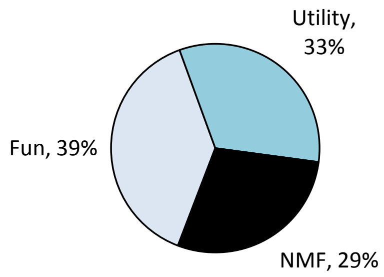


Figure 2. Distribution of scripts (NMF = "no meaningful functionality"). The Fun and Utility groups included several specific categories (Table 1, below).

Table 1. Scripts of TouchDevelop users, based on primary area of functionality (mutually-exclusive categories)

Category	Definition *{Example}*	%
Categories related to fun/entertainment		**39**
Game	Pose challenge for user to finish *{Generate a random number that the user must guess}*	11
Music	Access/play/manipulate music files *{On phone face-down event, play the next song in a play list}*	10
Images	Capture/display/manipulate image *{Resize and display existing image}*	7
Animation	Create animation using text, shapes *{Rectangle animates on the screen as the user drags finger around}*	6
Social	Post via Facebook/Twitter/SMS/email *{Post name of song to FaceBook}*	6
Categories related to non-entertainment utilities		**33**
Web Lookup	Post-search-fetch content on the web *{Download timetable for local railway}*	6
Scientific	Calculation for physics/chemistry/math *{Calculate muzzle energy of projectile}*	5
Location	Search/locate a place on a map *{Display map based on an address in the contact list}*	3
Phone Call	Call a phone number *{Place call using a calling card}*	2
Network Config	View/manipulate network settings *{Display type of network to which the device is connected}*	2
Direction	Find direction (orientation) *{Buzz whenever user turns north}*	1
Voice-Web Lookup	Speech-driven information retrieval *{Siri-like app: user asks question, and program provides information in reply}*	1
Tutorial	Teaches about TouchDevelop/API *{Show videos from TouchDevelop site}*	1
Other Utilities	Automate other single tasks or multiple tasks (series of operations) *{Test if all hardware functions—camera, microphone, network—are fine}*	13
Category for other scripts with no clear purpose		**29**
NMF	No meaningful function *{Output "TouchDevelop is cool", then end}*	29

Many scripts in the non-entertainment utilities categories did make use of hardware and operating system features that are somewhat unique to mobile devices. For example, *Location* scripts typically used the GPS, *Phone Call* scripts used the microphone and the device's ability to place phone calls, *Direction* scripts used the built-in compass, *Voice-Web Lookup* scripts used the microphone, and some *Other Utilities* used the camera or microphone. In contrast, we did not notice many scripts in the fun/entertainment-related categories that used mobile functionality so extensively. Scripts in these categories used the graphics, music, and image functionality that also is available on traditional desktop computers. (The only exceptions were a few *Images* and *Social* scripts that allowed users to store a photograph with the camera.)

In addition to the fun/entertainment-related categories and the non-entertainment utility categories, we identified a category of scripts that had no meaningful functionality at all. Nearly 30% of scripts fell into this category (*NMF*) Most were of the "Hello World" variety. A handful were scripts that were not even correctly composed as executable code, though as a general rule, the scripts in this category actually ran but did not accomplish anything significant. We cannot speculate why users posted these scripts, whether intentionally (e.g., perhaps as a form of learning activity) or unintentionally (e.g., possibly due to confusion with the tool's user interface).

VI. How are TouchDevelop scripts changed over time?

We randomly downloaded scripts and, for each, checked to see whether it contained a reference to a "parent" script, indicating that it was an edited version of an earlier script. We continued downloading until 100 such scripts were found.

Overall, we found a low level of reuse: We found that we had to download 1965 scripts to obtain 100 that had a parent, indicating an approximate whitebox reuse rate of 5%.

We categorized edits based on their apparent effect, revealing 12 categories (Table 2). Categorizing was relatively easy to accomplish because the bazaar provides a feature for directly viewing which lines of code had been modified from one version to another. In some cases, an edit could be placed into more than one category because the edit had more than one effect. The Jaccard Coefficient of our coding was 0.93.

We found that most functionality-related edits did not contribute substantive new features. Most of these edits were minor tweaks to existing functionality that had the effect to *Format Output* differently (18%), to *Add Output* by generating more output information at a point in the code where output was already being generated (12%), to *Modify Functionality* by tweaking the behavior of an existing feature (11%), to *Remove Functionality* by deleting or commenting out code (7%), or to *Fix Functionality* (2%).

For example, many of the *Format Output* edits consisted of changes in the prompts shown to users (e.g., from "This is the message" to "Write message to speak"), changes to other output strings displayed to users, or changes to lines or other graphical shapes shown on screen. An example of a more substantive *Modify Functionality* edit was made to a script that plays a series of songs. This edit changed the order in which songs played and changed the specific user gesture event (a phone rotation) that could be used to advance to the next song.

In the end, we found that only 26% of edits actually had the effect to *Add Functionality* in the sense of contributing an identifiable new feature to the script. For example, two edits changed their respective scripts so that they posted game high scores to the leaderboard service of the bazaar. Another edit added a feature for retrieving a comic strip image and resizing it to fit properly on the screen. Another example enhanced an alarm script to add a feature so the user could select what song should play when the timer went off.

Table 2. Edits performed by users, categorized according to effect on the script (non-mutually exclusive categories)

Category	Definition *{Example}*	%
Categories related to functionality		
Add Functionality	Create new feature (other than just adding or formatting output) *{Music player script has code added to shuffle when the phone is shaken}*	26
Format Output	Only change is how output is formatted *{ print(x); becomes print("Number is: " + x);}*	18
Add Output	The only change is more data is output *{A script gets song title and posts to web. The script is changed to display an image as well as song title on wall}*	12
Modify Functionality	Change existing feature (other than just adding or formatting output, or adding exception handling) *{Feature for resuming a song on phone rotation, is changed to pausing the song}*	11
Remove Functionality	Eliminate feature *{A 'save' action and all references to it are removed}*	7
Fix Functionality	Correct malfunctioning feature (other than just adding exception handling) *{Turning phone to portrait does not pause song; code fixed so it pauses properly}*	2
Other categories		
Clean Code	Script changed to simplify code, often removing unnecessary code *{Lines that are repeated in separate sections of the script are put into a new function that is called instead}*	9
Add Documentation	Comments are added to make purpose of script more clear *{Comments are added to metadata saying how to use the script}*	9
Bug Insertion	User publishes script where a feature clearly no longer works properly *{Modified script has syntax errors when original does not}*	4
Credit Others	Script modified to give credit to others *{Insert comment in start of a function to indicate where code was copied from}*	3
Add Exception Handling	Code added to deal with runtime errors *{Insert a conditional to check if a variable is_invalid() before dereferencing properties}*	2
NMC	No meaningful change *{Function with no content is added}*	18

On a positive note, across different categories, we noticed that very few edits were related to bugs. In particular, only 2% of the edits had the effect to *Fix Functionality*, and another 2% to *Add Exception Handler*. We found that 4% of the edits caused *Bug Insertion*. For example, one edit entirely deleted the actions in the script; other examples were edits that created scripts that were no longer syntactically valid.

The remaining edits that had any functional effect were to *Clean Code* (9%), to *Add Documentation* (9%) and/or to *Credit Others* (3%). For example, one *Clean Code* edit replaced statically hardcoded date values (that appeared to work properly) with API calls to compute date values dynamically, so that the script would continue working into the future. The *Add Documentation* edits generally added comments explaining the intent of the code. For example, one edit added a comment to explain the user interaction:

"// Turn your phone in landscape to the right --> next song"

In addition to all of the edits above that accomplished some effect in the code, we found that a non-trivial fraction of edits had no effect at all (18%). These included adding whitespace lines to the code, deleting comments, adding actions that had no body, and adding actions that were never invoked or otherwise referenced. It is impossible to speculate on the purpose of these edits or whether they were even intentional at all—as with NMF scripts, it is possible that users accidentally posted NMC edits to the bazaar without intending to do so.

VII. What problems do users ask for help with solving?

We looked for sources of TouchDevelop user comments by using Google to search for websites that contained the word "TouchDevelop." We painstakingly iterated through search results to find sites where TouchDevelop users appeared to be posting requests for help. We identified several sites: the TouchDevelop Facebook Wall, MSDN, the xda-developers forum, and the TouchDevelop Forum. Each of these sites has a threaded structure, where each web page contains multiple discussions, and each discussion has a first post that receives responses. We iterated through search results one page at a time until we had collected at least 100 (119, specifically). We only included discussions initiated by a post that presented a problem of some sort.

We then analyzed the posts that initiated these discussions to develop and apply a categorization scheme. Overall, we found 19 mutually exclusive categories, which fell into three groups: suggestions, questions, and bug reports (Table 3). After categorizing, we compared results. We found our results to be 86% consistent (Cohen's Kappa 0.85).

One common refrain of user comments was about APIs. Examples appeared in several categories, including *API Suggestions* (14%), *Documentation Suggestions* (3%), *Existence Questions* (8%), *How-To Questions* (7%), and *Documentation Questions* (7%) and *API Bug Reports* (8%). These included comments such as, "Any chance of adding An API to collect position with high accuracy in the future", "support of basic UI controls (buttons, sliders,toggle,color pick etc)", "expand API calls to settings like vibrate/ringer on/off", and "you can alter elements in a string collection. Any chance of the same for xml". No specific API suggestion, question, or bug report seemed to dominate or stand out as particularly common: requests were diverse and generally non-overlapping.

Although other studies have highlighted the difficulty that programmers encounter with finding and coordinating APIs when creating programs [9], an interesting issue in the TouchDevelop context was that users also seemed to struggle just as much with finding and combining *features*. Examples

Table 3. TouchDevelop problems that users ask for help with solving (mutually-exclusive categories)

Category	Definition *{Example}*	%
Categories related to suggestions		**39**
API Suggestions	Expand existing API or add new API *{"Access the file system in [version] 2.6"}*	14
Feature Suggestions	Add new tool feature *{"Run two scripts at the same time"}*	8
User Interface Suggestions	Suggestions for tool user interface *{"Remove TD logo from pinned script's tile"}*	4
Documentation Suggestions	For documentation on specific subjects *{"A tutorial on using the senses API"}*	3
Localization Suggestions	For other languages other than English *{"Wish there was a German language version"}*	1
Other Suggestions	Suggestions not in categories above *{"To see TouchDevelop be open-sourced"}*	8
Categories related to questions		**39**
Existence Questions	About if an API or feature exists *{"Is it possible to export my scripts to run as WP7 applications?"}*	8
How-To Questions	About how to do something with an API or feature *{"How do I access the script downloads?"}*	7
API Questions	About specific (known) APIs *{"About Picture Collection why can't I add or delete element?"}*	7
Documentation Questions	About the documentation *{"Where did the video tutorials go?"}*	7
Repository Questions	About repository or its scripts in general *{"Are there any known problems with downloading scripts?"}*	3
User Interface Questions	About specific (known) features in user interface *{"Problem with finding the 'rename' button … in touchdevelop 2.4"}*	2
Phone Questions	About phone (not TouchDevelop) *{"Change … phone lock screen"}*	2
Syntax Questions	About syntax of TouchDevelop scripts *{"Is code a reserved word now?"}*	1
Other Questions	Questions not in categories above *{"Are TouchDevelop scripts the same as WP7 applications?"}*	3
Categories for bug reports/complaints		**23**
Tool Bug Reports	About programming tool *{"I cannot download TouchDevelop."}*	12
API Bug Reports	About a specific API *{"The string trim function is cutting off characters that it shouldn't be"}*	8
Repository Bug Reports	About publishing a script *{"When I try to publish my script I get this error code..."}*	2
Other Complaint	Other complaint (not specific bug related to TouchDevelop) *{"There are not enough applications in the [Windows] Marketplace!"}*	1

appeared in *Feature Suggestions* (8%), *User Interface Suggestions* (4%), *Documentation Suggestions* (3%), *Existence Questions* (8%), *How-To Questions* (7%), *Documentation Questions* (7%), and *User Interface Questions* (2%).

Most feature-oriented comments touched on difficulties with finding existing features in menus or other parts of the programming tool. For example, these comments described a "problem with finding the 'rename' button to rename a variable in touchdevelop 2.4", a "problem with finding an option named 'Folders' in the menu at the bottom after 7.5/Mango update [to the Windows mobile operating system]", and a "problem finding available events and action types which control the media volume". Problems with finding TouchDevelop APIs are to a certain extent also feature-related problems in this environment (i.e., associated with the programming tool), since code to invoke many APIs is inserted into a script via menu commands (as shown in Figure 1b).

Other feature-oriented comments focused on features that users needed but that were missing. Specifically, three *Feature Suggestions* requested the ability to run multiple scripts simultaneously; one of these stated, "allowing the simultaneous execution of two TD apps(such as photoframe & timer) should be possible". Five users wanted the ability to compile for other platforms, including iOS, Android, Windows 7, and web browsers. Finally, two users wanted the ability to add custom types and user interface controls and widgets.

Overall, the API- and feature-related categories above accounted for approximately two-thirds of all comments.

After APIs and features, the next most common concern was compatibility. This concern appeared in two forms. First, almost all *Tool Bug Report* comments (12%) covered apparent tool-phone compatibility problems (or perhaps configuration problems) that occurred during installation. These bug reports generally provided detailed information about the problem encountered. Examples included "I can not install it on my HD7 in Greece with firmware 4.05.401.02" and "Ive got a problem download this app in Zune marketplace. whenever i want to download it Zune shows me this error: C00D11CD." Second, other comments discussed compatibility problems that users encountered when a script worked on one phone but not on the repository or another phone. Such comments appeared in the *Tool Bug Report* and the *Repository Bug Report* (3%) categories. For example, one user wrote, "Yesterday I uplouded my new digit clock script. For the time being nothing special but now the upload says it has errors! But when i look at my script, which is installed on my phone, to look after the error, there are none." Approximately 14% of comments were related to compatibility of one kind or the other.

VIII. Discussion

Now that users can pull out and program their phones, what programs are they creating?

In a word: apps. There are many noteworthy similarities between the TouchDevelop scripts that we observed and the apps that we have been using on our smartphones and other mobile devices for several years. Like many apps, scripts provided a small set of features, which were implemented fairly reliably, with few obvious bugs. Like many apps, they were free and available for download on demand. Like apps, many of them took advantage of mobile hardware. In short, Microsoft apparently has succeeded in providing an environment where users have created simple apps.

While this overall result is encouraging, our results highlight several areas for further innovation. Below, we describe three areas of opportunity informed by our empirical results interpreted in the light of related work.

A. One-third of scripts had no apparent functional purpose

As in a prior study of animation programs [5], a non-trivial proportion of TouchDevelop scripts had no apparent functional purpose. The key methodological limitation of the study is the lack of direct observations of people programming, so we cannot speculate on the motivations for why people created these scripts. Even though we found few bugs in any programs, it is hard to see why these scripts with no obvious function would be useful to other people, and Future work could include interviews of TouchDevelop users to investigate their motivations. From other users' standpoint, such scripts might just be distracting from valuable scripts on the repository, so future work might also enhance the repository in ways that help users to find and focus on what they consider truly useful.

B. The code reuse rate was extremely low (5%)

As in earlier studies [3][5][12], we found that very few TouchDevelop scripts were reused to create and publish a new script. Moreover, few edits actually added any new features. Research has suggested that code reuse may be inhibited when compatibility problems interfere [12], when functions are not implemented in a generalized way [12][13], when reusable code is mixed in a repository cluttered with less-useful code [5], or when code lacks comments and documentation [8][13] [15]. Each of these issues is consistent with our observations and could explain some of the low reuse that we observed and motivate new innovations, such as repository enhancements.

A distinctive challenge in the TouchDevelop context, however, is the extremely broad variety of scripts that users have created, which were more diverse than what we observed in some other environments. In other studies, at least we could create categories for all of the programs we observed (e.g., [4][5]). In the current study, many categories had only a few scripts, and 12% of scripts essentially required categories of their own in terms of functional purpose.

Thus, few users might benefit from extending any given script—and, conversely, any given script in the bazaar might be of interest to a vanishingly small proportion of the overall community. Yet the current bazaar, as with most repositories, is designed around the assumption that if a script is posted, then a significant number of other people might benefit from reusing it. Our results lead us to suspect that this assumption is not true in the case of TouchDevelop. Therefore, new approaches might be needed besides the existing bazaar search engine for helping users to benefit from existing code, enabling them to take advantage of these existing resources (e.g., in ways *other* than wholesale reuse of entire scripts).

C. Users call for more APIs, features, and platform support

Users' comments indicate they want more APIs for programs that they seek to create. The most common feature requested was support for deploying scripts on other phone operating systems or even on Windows 7 desktop, indicating that TouchDevelop could potentially be valued as a general-purpose programming environment. Comparing TouchDevelop user comments to those in other tool forums would reveal whether this demand for more features is unique to TouchDevelop or instead typical of users at a certain point in a programming tool's adoption.

Meeting these API, feature and platform requests would be difficult. Compatibility issues are already a problem and would only worsen on multiple platforms. Moreover, given the variation in mobile hardware and operating system capabilities, new TouchDevelop APIs or API variants might need to be supported on different phones, on top of the diversity of APIs that users are already suggesting. Additionally, some users have asked for help with finding APIs and features in the tiny mobile programming tool, which might become even more cluttered. Other environments such as Excel illustrate the confusion that users can experience when a programming tool gradually accumulates features [4]. Avoiding such confusion while still presenting new features and APIs will be crucial for ensuring that users obtain maximal benefit from programming on the phone for the phone.

REFERENCES

[1] Akhin, M, Tillmann, N, Fahndrich, M, de Halleux, J, Moskal, M. (2011) *Code Similarity in TouchDevelop: Harnessing Clones*, Technical Report MSR-TR-2011-103, Microsoft Research.

[2] Appcelerator (2012). *Titanium Studio*. www.appcelerator.com

[3] Bogart, C, Burnett, M, Cypher, A, Scaffidi, C. (2008) End-user programming in the wild: A field study of CoScripter scripts. *VL/HCC*, 39-46.

[4] Chambers, C, Scaffidi, C. (2010) Struggling to excel: A field study of challenges faced by spreadsheet users. *VL/HCC*, 187-194.

[5] Dahotre, A, Zhang, Y, Scaffidi, C. (2010) A qualitative study of animation programming in the wild. *ACM-IEEE International Symposium on Empirical Software Engineering and Measurement*, 1-10.

[6] Google. (2012) *Scripting Layer for Android*. http://code.google.com/p/android-scripting/

[7] Google and MIT. (2012) *App Inventor for Android*. http://appinventor.mit.edu/

[8] Jones, M, Scaffidi, C. (2011) Obstacles and opportunities with using visual and domain-specific languages in scientific programming. *VL/HCC*, 9-16.

[9] Ko, A, Myers, B, Aung, H. (2004) Six learning barriers in end-user programming systems. *VL/HCC*, 199-206.

[10] Myers, B, Park, S, Nakano, Y, Mueller, G, Ko, A. (2008) How designers design and program interactive behaviors. *VL/HCC*, 177-184.

[11] Panko, R. (1998) What we know about spreadsheet errors. *Journal of Organizational and End User Computing (JOEUC)*. *10*, 2, 15-21.

[12] Rosson, M, Ballin, J, Nash, H. (2004) Everyday programming: Challenges and opportunities for informal web development. *VL/HCC*, 123-130.

[13] Segal, J. (2004) *Professional End User Developers and Software Development Knowledge,* Technical Report 2004 / 25, Department of Computing, Open University, Milton Keynes, UK.

[14] Tillmann, N, Moskal, M, de Halleux, J, Fahndrich, M. (2011) TouchDevelop: Programming cloud-connected mobile devices via touchscreen. *Symp on New Ideas, New Paradigms, Reflections on Programming and Software*, 49-60.

[15] Wiedenbeck, S. (2005) Facilitators and inhibitors of end-user development by teachers in a school. *VL/HCC*, 215-222.

[16] Xiao, X, Tillmann, N, Fahndrich, M, de Halleux, J, Moskal, M. (2011) *Transparent Privacy Control Via Static Information Flow Analysis*, Technical Report MSR-TR-2011-93, Microsoft Research.

[17] Zang, N, Rosson, M. (2008) What's in a mashup? And why?: Studying the perceptions of web-active end users. *VL/HCC*, 31-38.