A Preliminary Analysis of App Inventor Blocks Programs

Johanna Okerlund and Franklyn Turbak
Computer Science Department, Wellesley College
Wellesley, Massachusetts, USA
Email:{johanna.okerlund, franklyn.turbak}@wellesley.edu

***Abstract*—App Inventor is a popular blocks programming environment for creating mobile apps for Android phones. In our ongoing project, we are analyzing App Inventor blocks programs to understand their effectiveness for creating apps and learning programming. Our ultimate goals are to give feedback to users about their programs and give guidance to the developers of App Inventor for improving its usability.**

I. Introduction

MIT App Inventor [1] is a visual environment for creating apps for Android mobile devices. Each app is specified by a project consisting of a set of user interface components and a program that describes the behavior of these components. The program is created by connecting blocks resembling jigsaw puzzle pieces. Blocks languages lower barriers for novices by eliminating or reducing many common programming errors and by providing visual guidance for choosing, assembling, and understanding program structures.

There are currently about 880,000 App Inventor users who have created nearly 2 million apps. App Inventor is used by students in many introductory computer science classes as well as by people with varying degrees of programming experience who want to create apps for themselves or for others. Little is known about how App Inventor programmers use blocks in their programs. The goal of this project is to study blocks usage in App Inventor programs in order to gain insight into the effectiveness of this visual language for creating apps and learning programming concepts.

II. Preliminary Data Collection and Analysis

Thus far we have collected and visualized data to understand simple aspects of App Inventor projects. App Inventor programs are stored in the cloud. We have access to the subset of these projects that have been updated since August 2012 and are stored in a backup repository. From these, we have extracted key features from the 270,000 App Inventor projects created by 40,000 randomly chosen users. Features of interest include counts of components, blocks, procedures declared, calls to each procedure, arguments per procedure, global variables, and projects per user. We are analyzing the results to understand how App Inventor is used in practice.

For example, a histogram of the number of blocks used in App Inventor programs (Fig. 1) shows that a surprisingly large percentage (31%) of programs have no blocks. This suggests that a significant number of users had trouble opening or using

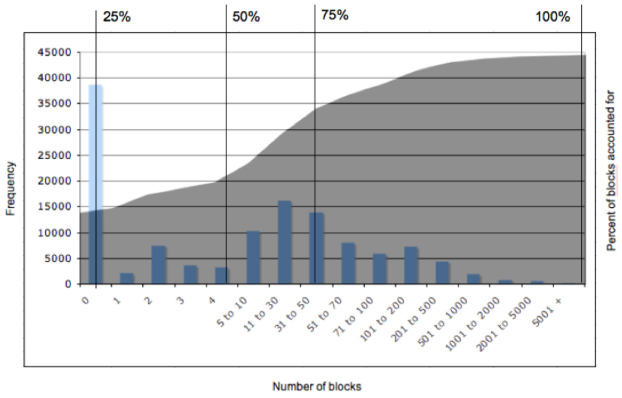

Fig. 1. Histogram of blocks per project

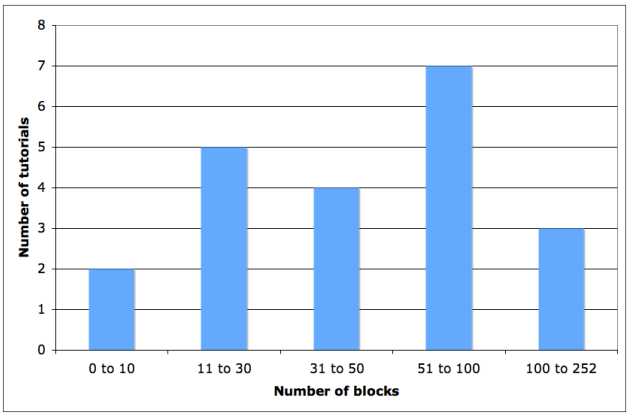

Fig. 2. Number of blocks in popular tutorials

the blocks editor. Another large percentage (13%) of programs have between 1 and 4 blocks, with which only a very simple program (such as the standard "Hello Purr" first program) can be written. Fig. 2 shows the frequency of the number of blocks in some popular tutorials [2] to give a sense of how many blocks are necessary to write an app with basic functionality.

As a second example, in a count of calls to each procedure (Fig. 3), 6% of procedures are declared but never called. Perhaps users creating these procedures don't know how to use them. On the other hand, App Inventor programs often include block assemblies that aren't currently used but might be useful later, and these might include uncalled procedures. Even more interesting is that the most common number of times to call a procedure is one. Rather than declaring a procedure to avoid repeating code, App Inventor programmers seem to be using procedures as way of organizing blocks on the screen. We

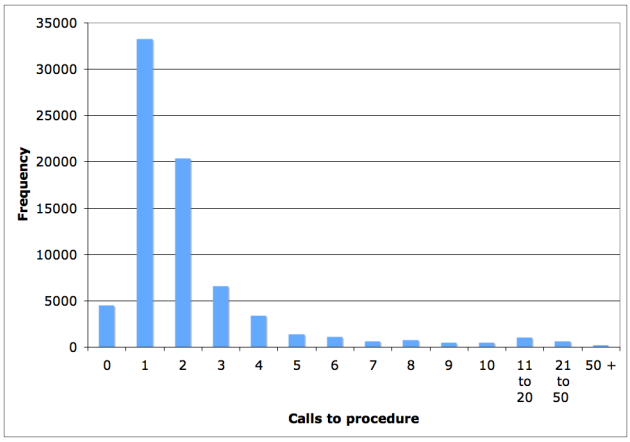

Fig. 3. Frequency of procedure calls

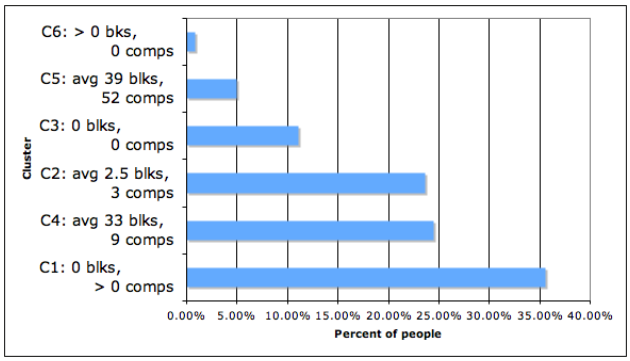

Fig. 4. Clusters resulting from k-means clustering on average number of components and blocks.

are also interested in these numbers because they suggest the complexity of the program and the proficiency of the user.

We have used simple statistics to cluster the users into groups whose members use App Inventor in similar ways. We used a k-means clustering algorithm with k=6 to cluster 10,000 randomly chosen users based on the average number of blocks and average number of components per project (Fig. 4). We can use these clusters to interpret the extent to which any given user has used App Inventor. Cluster C3 represents the people all of whose projects have zero components and zero blocks. C1 is the group of people all of whose projects have zero blocks, but some of which have at least one component. C6 represents people all of whose projects have zero components, but some of which have at least one block. These people don't have working programs. Clusters C2, C4, and C5 are groups of people some of whose projects have nonzero blocks and components.

III. Future Work

We are currently collecting more blocks usage statistics for a larger sample of App Inventor users. The summer of 2013 marks the release of App Inventor 2, a version of App Inventor in which the blocks editor is directly integrated into the web browser (as opposed to being a separate Java application). We are eager to see whether blocks usage statistics from App Inventor 2 indicate that the newer blocks editor is easier to use. App Inventor 2 also handles variables differently, and we can see if statistics involving global variables and procedure parameters differ between the versions.

Our next step (during the summer of 2013) is to develop simple notions of program sophistication in order (1) to determine the extent to which App Inventor users are learning programming concepts (do they write more sophisticated programs over time?) and (2) to provide feedback to users about their programs. For example, a program with multiple calls to a procedure is more sophisticated than a program with repeated copies of the code that would be in the procedure body. This notion can be used as the basis for an automatic tutor that suggests opportunities for proceduralization. Currently, we have only coarse-grained (roughly weekly) snapshots of user projects, which is not very helpful for tracking the evolution of a single program. But we can tell if later programs of users are more sophisticated than earlier ones.

A longer range goal is to instrument App Inventor to record the fine-grained steps of program construction. This wealth of information will give a more detailed narrative of how users write their programs. We expect that the high level nature of blocks programming edits will be more manageable and informative than the low-level nature of keystroke edits used in learning analytics systems for programming, such as [3]. We also plan to formulate a way to correlate program construction steps with compile time and runtime errors. This will help us understand how students build, debug, and test projects as they are learning to code and how best to improve their learning and the instruction by their teachers.

IV. Why is this Research of Interest to VL/HCC?

This research is of interest to the VL/HCC community because App Inventor is becoming an increasingly popular visual language for creating mobile apps and teaching programming. Our work is the first attempt to analyze a large percentage of all App Inventor programs and contributes to understanding the effectiveness of blocks languages for learning programming skills and computational thinking concepts.

V. Nature of this Showpiece

Our showpiece will consist of a poster that summarizes the key results from our ongoing data collection and analysis of blocks usage in App Inventor programs. If sufficient data are available about App Inventor 2, we will include a preliminary comparison of the two versions of App Inventor.

Acknowledgments

This material is based upon work supported by the National Science Foundation under Grant Number DUE-1226216. It was also supported by a Wellesley College Brachman Hoffman Fund Faculty Small Grants Summer Research Awards.

References

[1] MIT App Inventor home page, MIT Center for Mobile Learning, http://appinventor.mit.edu, accessed Jun. 7, 2013.

[2] MIT App Inventor tutorials page, MIT Center for Mobile Learning, http://appinventor.mit.edu/explore/tutorials.html, accessed Jun. 7, 2013.

[3] C. Piech, M. Sahami, D. Koller, S. Cooper, and P. Blikstein, "Modeling how students learn to program," in *43rd ACM technical symposium on Computer Science Education (SIGCSE'12)*, 2012, pp. 153–160.