Summer 2017

Investigating the Behavior of Novice Programmers in a Large Dataset

Ayse Hunt
University of Puget Sound, aysehunt@ups.edu

Investigating the Behavior of Novice Programmers in a Large Dataset

Ayse Hunt, Prof. Brad Richards
aysehunt@pugetsound.edu, brichards@pugetsound.edu

Background

Do novice Java programmers leave clues in their work habits that could predict their likelihood of success?

We aim to better understand how novice programmers approach problems by examining work habits of BlueJ users. In order to examine these patterns of work we must understand how a programmer arrives to a solution, we use the BlueJ state model.

A study from Stanford suggested that certain student behaviors exhibited as early as the first homework assignment can potentially indicate which students will struggle on the midterm

If work patterns were automatically detected by program-editing tools, they could adapt to better support novice programmers

What is BlueJ?

BlueJ is a integrated development environment (IDE). IDEs are to a programmer what Microsoft Word is to a writer—they allow us to write programs.

BlueJ is unique because it is geared towards beginners. Features including the Bench tool and the Codepad allow users to to test their code in a visually intuitive way.

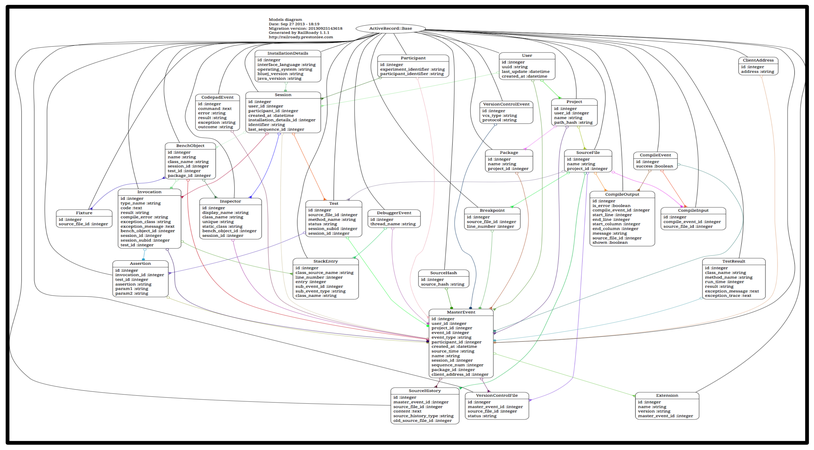
The team of researchers that run BlueJ also run Blackbox, a database of information sent by individual BlueJ users around the world.

The Blackbox Database

Blackbox is a database that contains traces collected from the IDE BlueJ. A trace is a sequence of actions such as a mouse click, a key stroke, the addition of a new file, or a compilation.

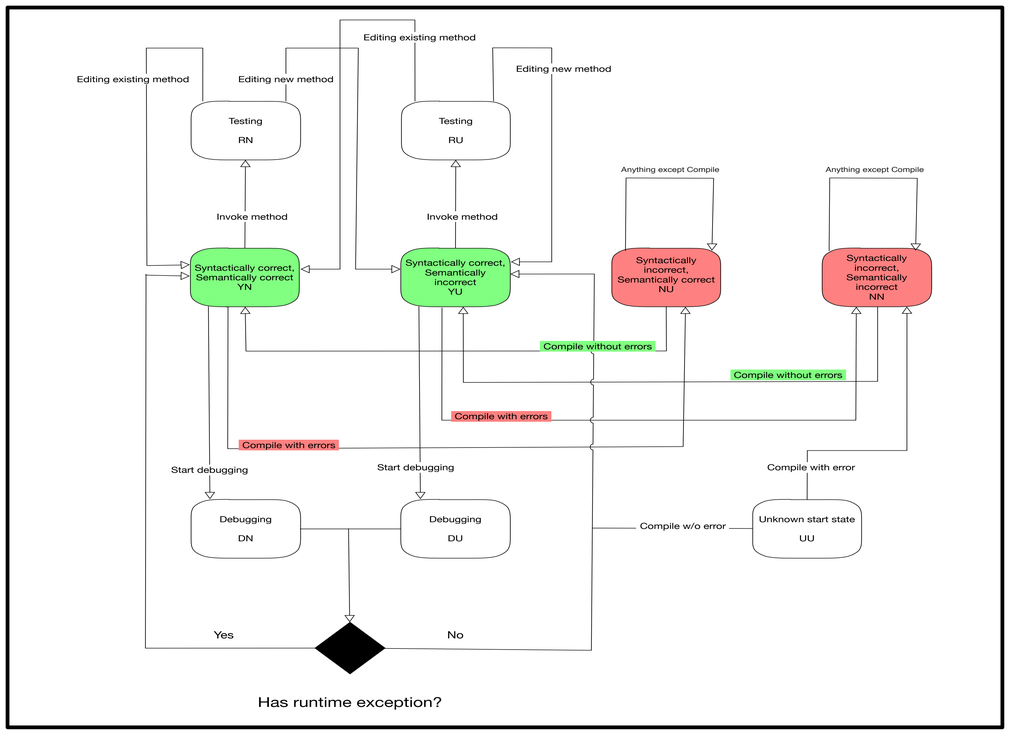
The fine grained nature of these traces are enough to reconstruct the user's code at different points in the writing process and records efforts to run the program.

Currently, Blackbox holds data from almost 25 million work sessions from 2 million different users worldwide. This data set is unique in both its level of detail and large scale.



Figures:
To the left is a map of Blackbox—if you can't read the text, don't worry. This diagram is meant to illustrate the complexity of this database.

Below is the BlueJ State Model



BlueJ State Model

- Based on Carter and Hundhausen's Programming State Model, the BlueJ State Model describes the states that a program in development can be in
- At any given time while writing a program, a user will be in one of these states. User actions such as compiling, editing, or running the program cause them to move to a different state
- We can use the BlueJ State Model to analyze a user's work session by examining the order they move through states
- For example, if a user is having the same problem repeatedly, they would move between the same states over and over. We can use this information to learn more about where the user was struggling

Completed Work

- Requested and was granted access to Blackbox research resources, including the database of student traces
- Selected a common programming exercise to study, verified there were enough traces to analyze and that their complexity was appropriate
- Learned the database programming language SQL to interact with the UK-based Blackbox database
- Studied the structure of Blackbox and decided which pieces of information were relevant to our analysis
- Created a program to select desired traces, extract information about each trace event, and store it in local text files
- Ran the program to select and transfer the required data — a five-day process, retrieving 15,616 traces (3.5 million events)
- Developed BlueJ-specific State Model after reading relevant literature and studying previous models

Future Work

- Fine-tune BlueJ state model
- Develop metrics for measuring quality of final student programs
- Write program to apply state model to student traces
- Look for correlations between state model patterns and program quality

References

[1] Adam S. Carter, Christopher D. Hundhausen, and Olusola Adesope. 2015. The Normalized Programming State Model: Predicting Student Performance in Computing Courses Based on Programming Behavior. In *Proceedings of the eleventh annual International Conference on International Computing Education Research* (ICER '15). ACM, New York, NY, USA, 141-150. DOI: http://dx.doi.org/10.1145/2787622.2787710

[2] Adam Scott Carter and Christopher David Hundhausen. 2017. Using Programming Process Data to Detect Differences in Students' Patterns of Programming. In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education* (SIGCSE '17). ACM, New York, NY, USA, 105-110. DOI: https://doi.org/10.1145/3017680.3017785

[3] Chris Piech, Mehran Sahami, Daphne Koller, Steve Cooper, and Paulo Blikstein. 2012. Modeling how students learn to program. In *Proceedings of the 43rd ACM technical symposium on Computer Science Education* (SIGCSE '12). ACM, New York, NY, USA, 153-160. DOI=http://dx.doi.org/10.1145/2157136.2157182

I would like to thank the University of Puget Sound and the NASA Space Grant Consortium for funding this research project. I would also like to thank the Blackbox team for granting me access to the database.