Web-scale Data Gathering with BlueJ

Ian Utting, Neil Brown, Michael Kölling, Davin McCall and Philip Stevens
University of Kent
Canterbury, UK
{iau,nccb,mik,D.McCall,plcs}@kent.ac.uk

ABSTRACT
Many investigations of students' initial learning of programming are based on small-scale studies of their interactions with a learning environment. Although this research has led to significant improvements in the understanding of student behaviour (and tool support), it has often been restricted to small numbers of students at single institutions. This paper describes an initiative to instrument the widely-used BlueJ environment to collect data on a much larger scale, and make that data available to Computing Education researchers. The availability of this data has the potential to enable research not previously possible. This paper discusses the type of data that will be gathered, the restrictions placed on identifying students, and mechanisms for associating the data with contextual data gathered outside the scope of the initiative.

Categories and Subject Descriptors
K.3.2 [**Computing Milieux**]: Computer and Information Science Education – *Computer science education.*

General Terms
Measurement, Experimentation.

Keywords
CS1, student behaviour, initial programming, BlueJ, data collection.

1. BLUEJ
BlueJ [6] is a Java IDE specifically designed for beginning programmers. It was originally released free-of-charge in 1999 and is now published under the GPL. The software itself is translated into 17 different languages and is used in introductory programming courses at secondary schools and Universities worldwide. The most common use of BlueJ is in initial programming courses, with students moving on to full-featured professional environments fairly soon thereafter. The BlueJ website lists almost 1000 Universities which have indicated that they are using the software.

Since its first release, BlueJ has been downloaded over 10 million times, with current downloads running at over 2.5 million per year. This number is influenced by the number of new major releases in any given year, as well as an indeterminate number of downloads where the software is never installed, or is tried once and thrown away, or is downloaded once and installed on many machines.

Since 2009, the standard distribution of BlueJ has contained a function that reports use of the software (including BlueJ version, Java version and operating system) for maintenance and planning purposes. For instance, abandoning support for older versions of Java would have an impact on users of older Apple computers, which makes it important to know what proportion of the BlueJ user base are using them. This data gives a finer-grained picture of the use of BlueJ than raw numbers of downloads (but it still does not allow institutions, or individuals, to be identified).

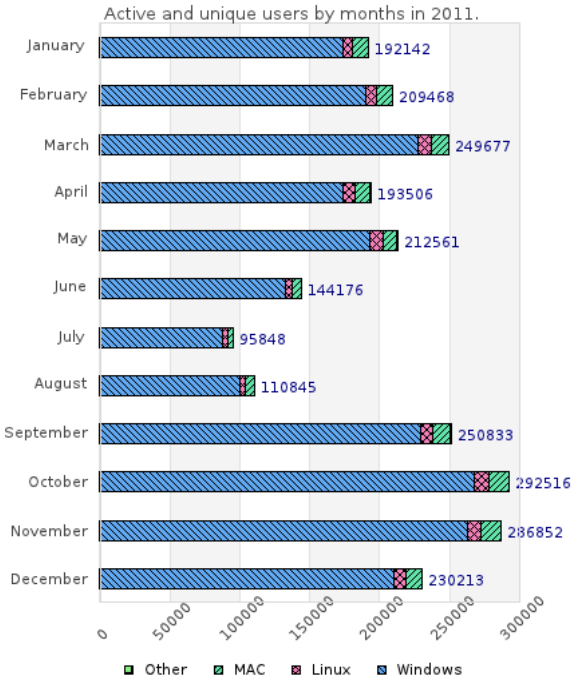

Figure 1: Unique BlueJ users per month in 2011

Data collected by this method is shown in the graph above for 2011 (Figure 1), which reports between 95,000 and 290,000 distinct BlueJ users depending on the month, with distinct peaks reflecting the typical start-date of school and college courses. Data also showed that the average duration of a user's contact with BlueJ (i.e. from first recorded use to last recorded use) was around 90 days, and that BlueJ had been installed in 202 countries around the world, with close to 1,000,000 unique users reported in the US. The 10 countries with the highest numbers of reported active BlueJ users are listed in Table 1.

ICER'12, September 9–11, 2012, Auckland, New Zealand.

Rank	Country	Users	% of all users
1	Germany	354123	23.1%
2	United States	327481	21.4%
3	India	136592	8.9%
4	United Kingdom	78005	5.1%
5	Brazil	60978	4.0%
6	Chile	47576	3.1%
7	Spain	38699	2.5%
8	Philippines	36383	2.4%
9	Canada	33417	2.2%
10	Mexico	29717	1.9%

Table 1: Top ten countries for BlueJ use (2011). Includes only users seen more than once.

The number of users is under-reported, as only users whose installation can send an HTTP request to an external website are recorded. This will have a significant impact on reported uses in regions where internet access is not universal (e.g. India, where BlueJ has 200,000 reported installations, but is used in some very large-scale "Standard 10" qualifications), or where local security policies bar internet access (e.g. secondary schools in the UK and USA; and China).

Conversely, the total numbers of users may be inflated by use of BlueJ at sites that routinely re-image PC's between uses, causing users to lose their profile data and BlueJ to consider them as new users on every invocation.

2. BEGINNING STUDENTS' INTERACTIONS WITH IDES

There have been many studies, using many approaches, of students' behaviour in introductory programming classes. Here we will restrict ourselves to discussing those that focused on their interactions with their programming environment, and the resulting program texts.

Thomas et al [7] recorded 4.7 million actions over a six-week period from 141 students using an Ada IDE, but these were very low-level events (e.g. captured from the GUI framework) and data-cleaning proved to be a significant problem, although they did also establish the viability of using such data to answer questions formulated after the data was gathered.

Ahmadzadeh et al [1] collected much coarser-grained data (including source code) from 192 students in the School of CS & IT at the University of Nottingham, mostly focused on compilation errors encountered by students whilst using the JCreator environment.

Edwards et al [3] collected result-focused data from 1101 students over a five-year period. The data gathered was students' work-in-progress as they submitted their evolving programs for testing by, and feedback from, the Web-CAT tool. They eventually captured 89,879 submissions from two courses using Java and one using C++.

Jadud [5] and Fenwick et al [4] have recently used bespoke extensions to the BlueJ Java IDE to capture student interactions at an intermediate granularity, capturing the input to and output from the Java compiler at every invocation. Jadud captured 42,000 events from 186 students over two years, and Fenwick et al captured 55,000 from 110 students in a single year.

The questions addressed by these studies, in their published output, has been quite diverse, from compilation and editing behaviour to time-on-task and start-time-to-deadline differences. In all cases except Thomas et al, the particular research questions to be addressed were part of the design of the data-gathering apparatus, but in many cases the data proved amenable for use in answering other questions, which became apparent only after the data had been gathered. Some of the studies have gathered very large numbers of event records, and some have involved large numbers of students, but in the published cases, all the students were studying at a single institution, and only in no cases was the tool used to gather the data re-used. We speculate that this was because the tool in question was implemented as an extension to BlueJ, talking to a Web Service fronting a standard database. This allowed not only for simple installation of the back-end, but also (and probably more crucially) simple installation of the front-end in a tool (BlueJ) which the students were already using, and without needing negotiation with those setting up student Labs.

These latter two issues seem to pose a significant barrier-to-entry to those trying to initiate or participate in these sorts of studies. If the instrumented tool is not part of the students' normal working environment, then their use of it will be, at best, artificial and potentially confounded by unfamiliarity with the tool itself. If a tool (or variant of a tool) needs to be specially installed by systems administrators (e.g. for use in a Lab), then that often requires significant lead-time, and it may be difficult to restrict use of the tool to the target population and context.

A further significant barrier to the implementation of these studies across multiple institutions has been the need to undertake paper-based informed consent protocols, requiring significant commitment from staff at all participating institutions, not just the initial investigators'.

3. DATA GATHERING

We propose a data-gathering project that will leverage the widespread use of the BlueJ system to gather more data than the Thomas et al study, from more students than the Edwards et al study, at a similar granularity to the Jadud and Fenwick et al studies. More importantly, we will gather data from students at many institutions around the world, but at a relatively consistent point in their formation as programmers.

3.1 What will be gathered?

One of the important decisions in the design of this work is the selection of the data to be captured by our mechanism. The exact detail of the data will determine the nature of the research questions that may later be investigated using this data. The challenge, therefore, is to capture data that allows as wide a selection of investigations later on, and not restricting its use to a set of previously determined studies.

The aim of capturing data widely (to allow the investigation of as many research questions as possible) has to be offset against issues of privacy and practicality. We have to ensure anonymisation of the data collected, and we have to ensure that the volume of data is manageable, both in the collection phase and the evaluation phase.

This latter aspect is closely related to the question of the abstraction level of the data being collected. On one end of the spectrum, low level events (such as key strokes and mouse events, potentially including every mouse movement) could be logged; at the other extreme, higher level events - such as logical interactions with system components - might be recorded. While low level data offers a more "complete" picture of activity, collection and evaluation is more complex, and often necessitates a transformation into higher-level events anyway, which can be confounded by variability and "noise" in the event traces. Higher level data is easier to interpret, but may exclude investigation of some research questions not previously anticipated.

For the BlueJ data-gathering project, we currently plan to collect the following data, tagged with the UID and timestamped:

- Compilation events – the result of every compilation ordered by the student, including details of any errors reported, and a snapshot of the code submitted to the compiler.
- Code edits, on a line-by-line basis. That is, when a line of code is edited, the modified line will be transmitted once the user moves the cursor to a different line. We will differentiate between single line and multi-line edits, the latter being likely copy-and-paste or reformatting operations, and separately identify edits caused by code-completion actions (i.e. auto-suggested methods names).
- Interactive invocations, including method calls and object creation, on the object bench and in the codepad (i.e. all user code invocation). The object bench is a unique feature of BlueJ which allows students to explore the behaviour of their programs interactively. The codepad is a more traditional direct code entry and execution mechanism which, in BlueJ, can interact with objects recorded in the object bench.
- Unit-testing – recording the execution and results of tests which use BlueJ's integrated JUnit support.
- Project-open and project-close events, giving a handle on time-on-task.
- Debugging – when the debugger is opened/closed, when breakpoints are set, when breakpoints are hit, when 'step'/'continue' are used.
- Use of version control – commit/update commands, through BlueJ's integrated SVN support.
- Location of the user to the "regional" level: the step between the national and the city. For some countries, this will not be available, so only the country will be recorded. Anonymity of institution (if not of individual student) requires this degree of imprecision.

3.2 How much data do we expect to gather?

The data gathering mechanism will use an explicit opt-in approach, with students having to consent to taking part via a pop-up dialogue in BlueJ when they first use it. Students will be able to rescind their consent at any time, and students under the age of 16 will be warned-off (they cannot give their consent under UK law). Therefore, the total amount of data will be determined by the number of overall BlueJ users and the proportion of users who agree to be involved. At current usage levels, derived from the usage monitoring mechanism described above, this will lead to a maximum of around 27,000 users per day, performing on average 3 sessions per day, and generating about 100 events per session over an average period of 90 days. This would mean overall a maximum case of 8 million events per day, or just under 100 events per second, and a total of 3 terabyte of data per year.

3.3 Who will we gather it from?

Students agreeing to contribute data to the repository will not be identified by name or institution. To help to ensure that student names are not accidentally gathered, "class comments" (which is where Javadoc @author tags appear) will be blanked out at source (blanked out rather than deleted to preserve line numbers, etc. for matching to compiler output). We will make no attempt to detect and blank identifying information placed elsewhere in the source.

For every participating user, on every BlueJ installation with which they work, a unique identifier (UID) will be generated and used to tag all the data generated from that place. Using this mechanism, multiple sessions by the same student (in the same place) over time can be linked, allowing longitudinal studies, without identifying the individual. However, an individual student may have more than one UID, either because they work on more than one machine (e.g. in a Lab and at home), or because a particular Lab setup does not preserve students' identity across login sessions. In this case, the user will appear as two users, with histories that are incomplete, but internally consistent.

4. WORKING WITH THE DATA

We intend this project to benefit the Computing Education research community, and as such we will provide access to all the collected data for researchers at bona fide Universities and research institutes. This access will be in the form of SQL-queries on a read-only mirror of the collected data, hosted by us.

We anticipate that researchers working on the data will benefit from collaborating in the production of common SQL-queries and tools and more general approaches to mining the data; and will likely be willing to share the resulting tools with other researchers. To support such collaboration and sharing of resources, we will host a community website for the researchers in the style of the existing Greenroom [2], which supports the sharing and collaborative development of resources through a Wiki-style interaction mechanism.

The data that we collect directly will be anonymised. However, we anticipate that researchers will want to be able to run studies at their local institution where they collect additional data (age, gender, programming experience, etc) about the participants, and then be able to match the data with the participant's user in the database. To support this, users will be able to enter a code (provided by the researchers running the local study) that identifies them in the data. Data retrieved from the database can be restricted by tag, but tags will not be directly retrievable. Thus a researcher can restrict the data retrieved to only that with their tag (i.e. only the users participating in their experiment), but cannot see the tags applied to any other data, effectively denying them the ability to "see" other cohorts within the data.

Beyond this, it is clearly possible for researchers to get students to place identifying material (either at the student or intervention level) into their source code outside the class-comment. But such data will not be indexed in the repository, and so retrieving data on the basis of such material will be highly inefficient (unless pre-filtered on the basis of the indexed tags described above.

Of course, the per-student UIDs are stored locally, and so can be retrieved by a researcher (given the consent of the student) to tie-in with individually collected local data.

Any researcher who wished to identify their own students, or projects, will need to obtain local approval for their experiment (human subjects, or ethical, approval) as required. Approval for collection, use and sharing of the global, anonymised, data has already been obtained by the authors at the institution where the data will be kept.

5. SUMMARY

We propose to instrument the widely-used BlueJ beginners' programming environment to collect anonymised data about students' behaviour. This data will include code-edits, compilation events and other events such as tool invocations. The resulting data-set will be made available to other Computing Education researchers in order to support their research.

We believe that the scale of this data-set will enable not only quantitative differences in research (due to the large number of users likely to be involved in the collection), but also consequent qualitative differences. The large scale means that less-frequently used features (such as the debugger) or rarer error messages (such as private/public access problems) will have enough data that they can actually be studied, where previously this was not possible. The multi-institutional, even multi-national, scope of the data collection will allow comparisons between groups of students which have not been possible in smaller-scale research.

We also believe that this study has the potential to greatly enhance all of Computing Education research, by sharing the large body of data to enable collaborative research by researchers beyond those involved in collecting the data, and by researchers who do not necessarily have the opportunity (or time) to collect their own data at a teaching institution.

6. REFERENCES

[1] Ahmadzadeh, M., Elliman, D., and Higgins, C., An Analysis of Patterns of Debugging among Novice Computer Science Students. In *Proceedings of the 10th annual SIGCSE conference on Innovation and technology in computer science education (ITiCSE '05)* (Caparica, Portugal, 2005), pp. 84-88

[2] Brown, N., Stevens, P., and Kölling, M. 2010. Greenroom: a teacher community for collaborative resource development. In *Proceedings of the fifteenth annual conference on Innovation and technology in computer science education (ITiCSE '10)*. ACM, New York, NY, USA, 305-305.

[3] Stephen H. Edwards, Jason Snyder, Manuel A. Pérez-Quiñones, Anthony Allevato, Dongkwan Kim, and Betsy Tretola. 2009. Comparing effective and ineffective behaviors of student programmers. In *Proceedings of the fifth international workshop on Computing education research workshop (ICER '09)*. ACM, New York, NY, USA, 3-14.

[4] Fenwick, J. B., Norris, C., Barry, F. E., Rountree, J., Spicer, C. J., and Cheek, S. D. Another look at the behaviors of novice programmers. In *Proc. 40th ACM Tech. Symp. Computer Science Education*, ACM, New York, NY, 2009, pp. 296–300.

[5] Jadud, M. A first look at novice compilation behaviour using BlueJ. *Computer Science Education*, 15(1):25–40, March2005.

[6] Kölling, M., Quig, B., Patterson, A., and Rosenberg, J. The BlueJ system and its pedagogy. *Journal of Computer Science Education, Special issue on Learning and Teaching Object Technology*, 13(4), 2003.

[7] Thomas, R., Kennedy, G.E., Draper, S., Mancy, R., Crease, M., Evans, H., and Gray, P. Generic usage monitoring of programming students. In *Proceedings of the 20th Annual Conference of the Australasian Society for Computers in Learning in Tertiary Education (ASCILITE '03)* (Adelaide, Australia, Dec 7-10, 2003).