@InProceedings{karvelas2023ide,
author="Karvelas, Ioannis
and Dillane, Joe
and Becker, Brett A.",
editor="Keane, Therese
and Lewin, Cathy
and Brinda, Torsten
and Bottino, Rosa",
title="IDE Interactions of Novices Transitioning Between Programming Environments",
booktitle="Towards a Collaborative Society Through Creative Learning",
year="2023",
publisher="Springer Nature Switzerland",
address="Cham",
pages="603--614",
isbn="978-3-031-43393-1"}

Author's version of paper published in IFIP World Conference on Computers in Education, WCCE 2022: Towards a Collaborative Society Through Creative Learning pp 603–614
Version of record: https://dx.doi.org/10.1007/978-3-031-43393-1_54

IDE Interactions of Novices Transitioning Between Programming Environments

Ioannis Karvelas[1]0000-0003-2502-2903, Joe Dillane[1], and Brett Becker[1]0000-0003-1446-647X

University College Dublin, Dublin, Ireland

Abstract. Novices in introductory programming courses typically learn the fundamentals of programming using one of a wide of programming environments. These vary greatly in terms of the mechanisms they employ to assist programmers, including their approaches to compilation and error message presentation. It is yet to be established which, if any, of these mechanisms are more beneficial for learning. In this study, we utilize Java programming process data to investigate the interaction between novices and two different versions of the BlueJ pedagogical IDE, which differ substantially in terms of compilation mechanism and error message presentation. Specifically, we compare novices that used both BlueJ 3 and BlueJ 4 with those who exclusively used either and the effects of the order in which they transition between BlueJ versions. We find substantial differences between different cohorts in terms of error messages and compilation which provides evidence that programming environments play an important part in influencing the programming practices of novices. This work supports the hypothesis that the choice of programming environment significantly affects user behavior with respect to specific programming interactions and therefore it is reasonable to expect a difference in how these environments affect learning. . . .

Keywords: Blackbox, BlueJ, Compiler error messages, CS1, Editors, Educational data mining, IDE, Java, Novice programmers, Programming, Programming environments, Programming process data, Tools

1 Introduction

Novices enrolled in introductory programming courses - commonly called CS1 [10] - face substantial challenges [3] as they learn the theoretical aspects of programming and familiarize themselves with the software development process [6]. This is usually accomplished through programming assignments which aim not only to enable students to put theory into practice, but also to expose them to the often strenuous task of debugging and testing their code.

Students usually engage with programming while following small cycles of editing, compiling, and executing code [11] using a programming environment, ranging from simple text editors used to write source code files which they later compile manually at a command line, to complex industry-focused Integrated

Development Environments (IDEs). In the middle of this range are pedagogical environments designed for learning. All of these environments vary in terms of the tools they provide and mechanisms they employ. Two of the most commonly varied mechanisms are the compilation mechanism and feedback presentation, the latter of which normally involves error messages as a core facet.

Since environments act as a medium through which users create and interact with programs, it is important that this interaction is appropriate for student learning and efficient in assisting them in improving skills such as syntax mastery in order to overcome issues that may disrupt the core learning experience. Most instructors would agree that learning to program should not be complicated with learning the intricacies of an elaborate environment, or tools that otherwise hinder the learning of programming concepts. At a minimum, more advanced environments would likely impart a higher extraneous cognitive load on the student. For novices, there is also a particularly important feature that all environments should provide: constructive and informative (ideally formative) feedback on the code written by the student. This puts error messages, their mechanisms, and their presentation in the spotlight. However, little progress has been made on investigating the effectiveness of such mechanisms on achieving optimal outcomes [4]. Although some educators may encourage their students to use a pedagogical environment, this is not always the case and students may end up using IDEs designed for experienced programmers. The benefit of such environments for novices is not established [16].

In order for developers to include effective functionalities and feedback mechanisms when designing environments (pedagogical or professional), they should ideally be basing decisions on empirical evidence. To achieve this, more studies should focus on establishing evidence, frameworks and guidelines for designing these tools. This would ideally result in environments that benefit users in terms of programming patterns, compilation habits and the usefulness of information that users receive from the environment.

This research investigates student interaction with two versions of the BlueJ pedagogical environment [12], that differ in compilation and error message presentation. Through this, we aim to provide a more transparent view of how programming environments can influence novice programmer behavior and ultimately, learning.

1.1 BlueJ and Blackbox

BlueJ [12] is a popular introductory programming environment for text-based Java programming used by millions of novices over more than two decades. BlueJ logs the programming process data of opted-in users in the Blackbox database [8,7]. BlueJ versions up to and including version 3 (2010-2017) featured only manual (click to compile) compilation. If errors were present in the code, only the line(s) corresponding to the first error were highlighted and only the first error message was displayed at the bottom of the window. BlueJ has a relatively long version cycle, and in 2017, BlueJ 4 was released and included substantial changes in the compilation mechanism and error message presentation. Specifically, automatic background compilation was added. This is triggered every time users change lines in the source code, load the source code, and create class instances. In addition, standard manual compilation was retained. By default, if errors are present in the code, no error messages are presented automatically, but all offending code is underlined in red. In order to see (truncated) error messages, users must hover over the specified area in the code with the keyboard or mouse, or click the compile button. If multiple errors are present, clicking the compile button more than once causes the full (non-truncated) error messages to appear one at a time, from first to last (and from there re-presents the error messages in round-robin fashion upon continued clicks).

Thus, these two BlueJ versions employ drastically different mechanisms for compilation and feedback presentation, while keeping the rest of the features in the environment largely unchanged. This enables us to infer that changes in programming behavior between the two versions are largely due to these feature changes.

1.2 Research Questions

Previous findings [1] showed that novice programming interaction with BlueJ shows substantial differences between BlueJ 3 and BlueJ 4. In BlueJ 4, users get exposed to more compiler error messages in the same amount of time, they compile manually less frequently and their manual compilations are more often successful. However, the cohort involved in that analysis consisted of users with substantial time spent using both BlueJ versions. Thus, any conclusions regarding the differences in the interaction of users with BlueJ 3 and 4 were not based on users who used exclusively one of the two versions, and did not take into account any effects on novice programming behavior imposed by the order of transitioning between versions. In this work, we investigate the differences between versions, and the effect of transitioning between versions by focusing on distinct user cohorts, compared to just one: users that used BlueJ 3 exclusively, users that used BlueJ 4 exclusively, and those that transitioned between BlueJ versions. For the latter, we study various subcohorts further, depending on the transition order ($3 \rightarrow 4$), ($3 \leftarrow 4$) and ($3 \leftrightarrow 4$). Our research questions are:
RQ1: How does transitioning between environments affect novice interaction regarding compilation and error messages as opposed to being exposed to a single environment?
RQ2: How does the order of transition between environments affect this interaction? Possibilities include: (BlueJ 3 $\rightarrow$ BlueJ 4), (BlueJ 3 $\leftarrow$ BlueJ 4) and (BlueJ 3 $\leftrightarrow$ BlueJ 4).

2 Methodology

We initially selected two cohorts from the Blackbox database:
1. Transition Users (TR): Users who switched between BlueJ 3 and BlueJ 4

versions between October 2017 and February 2018. We chose these dates as this coincides with the introduction of BlueJ 4. This cohort includes users regardless of their transition status (e.g. a user in this cohort could be switching from BlueJ 3 to BlueJ 4 or vice versa). All users in this cohort had programming activity in BlueJ 3 and BlueJ 4. We break these users down further later.
2. Exclusive Users (X): These users were selected randomly from users who had only a single BlueJ version (either BlueJ 3 or BlueJ 4) installed on their machine during the period their data were logged by Blackbox. All users in this cohort had programming activity only in one of these two versions. We study these separately later (we refer to users who only used BlueJ 3 as X3 and BlueJ 4 as X4).

Only programming events that were associated with Java version 8 were retrieved (which is also the most common version in Blackbox for the dates studied)[1]. In addition, the programming activity of both cohorts was expanded to the range of the 14th of January 2016 and the 24th of May 2019[2].

2.1 Compilation and Error Message Presentation Metrics

After retrieving the programming events of the users as described in Section 2, the programming time (H) in hours that every user spent programming in BlueJ was calculated. This was done by summing all time differences between the first and the last programming events of every session[3] for each user. This methodology presents a complication: sometimes, connection interruptions cause Blackbox to stop logging events requiring a manual means of calculating session duration. We discuss this further in Section 4.

We used the following metrics for describing the interaction regarding compilation and error message presentation for every user: (1) Displayed Compiler Error Messages per Hour (DCEMpH), (2) Manual Compilations per Hour (CpH), and (3) Percentage of Success of manual Compilations (PSC). This approach of quantifying the BlueJ users' interaction is consistent with previous work [1].

2.2 Removing Outliers

When dealing with large repositories of programming process data like that in Blackbox, it is expected that there will be many cases of irregular activity. In our case, there were users with unrealistically high programming time (for instance, tens of thousands of hours) or displayed compiler error messages over time (for instance, several hundred per hour). Extremely high programming times can be a result of idle activity in BlueJ, whereas many compiler error messages could be triggered by stuck keyboard keys or similar hardware failures or even a book

[1] This was done as compiler error messages are known to differ across Java versions [8].
[2] The range limits are equidistant from the first day that transition to BlueJ 4 was observed.
[3] A session is bounded by two distinct events sent from the user to the Blackbox database, indicating the launch and termination of BlueJ.

falling on a keyboard – with hundreds of thousands of users total and millions of events per day, strange things happen. Although these users were few in number, such extreme values could distort results. In order to mitigate against this, we excluded users in all TR and X cohorts independently, based on the following procedure as used in [1]: **Step 1:** Removal of users whose programming time in BlueJ 3 was greater than the maximum programming time in BlueJ 4. This was done to eliminate few cases where programming time was exceptionally high in BlueJ 3, something not observed for BlueJ 4. **Step 2:** Recalculation of the means and standard deviations after Step 1 and removal of users whose programming time (H) was greater than the mean increased by three standard deviations. **Step 3:** Removal of users whose DCEMpH was greater than the mean increased by three standard deviations.

2.3 Categorizing Transition Users

In this stage of analysis, we classified transition users (see Section 2) based on all three possible transition possibilities: transitions from BlueJ 3 to BlueJ 4 (we will use the acronym 3t4 later for these), transitions from BlueJ 4 to BlueJ 3 (4t3), and transitioning repeatedly between the two versions (Overlap).

2.4 Metric Restriction in BlueJ 3

In BlueJ 3, each compilation causes at most one error message to be displayed. In other words, users see a maximum number of error messages equal to the number of compilations they invoke (if all compilations involve an error). Based on this, we can define a relationship between the three metrics that are examined in this work (DCEMpH, CpH, PSC) using the formula described in Equation 1[4]. We will refer to this equation in later sections as the *BlueJ 3 equation.*

$$DCEMpH = (1 - PSC) \times CpH \tag{1}$$

2.5 Similarity Calculation

To gain a high-level view of differences between the BlueJ 3 and BlueJ 4 distributions, we quantified the differences between versions regarding our metrics for each cohort using three approaches:

Method 1 (M1) – Average minimum distance from BlueJ 3: Figure 1 represents the surface defined by Equation 1. This involves the calculation of the shortest Euclidean distance between the BlueJ 4 interaction of a user and this surface. Specifically, we defined a new function describing the distance between user interaction in BlueJ 4 and BlueJ 3. For every user, we used the Nelder-Mead downhill simplex algorithm [15] to obtain the minimized function value.

[4] This equation can be used to describe every user's interaction with BlueJ 3 (or similar environment). Values that do not satisfy the equation are a result of missing data in Blackbox.

This can be viewed as a process of answering the question: “What is the closest possible BlueJ 3 behavior to this particular BlueJ 4 user’s behavior?”.

Method 2 (M2) – Distance between BlueJ 3 and BlueJ 4 mean coordinate values: In this method, we created a hypothetical “average user” using the mean values of DCEMpH, CpH, and PSC of all users for each BlueJ version, and calculated the Euclidean distance between them.

Method 3 (M3) – Minimum distance between BlueJ 4 mean coordinate values and BlueJ 3: In this method, we used the mean coordinate values of BlueJ 4 (in the respective user cohort) to come up with one “average” BlueJ 4 user profile, and calculated its minimum Euclidean distance from the surface represented by the BlueJ 3 equation.

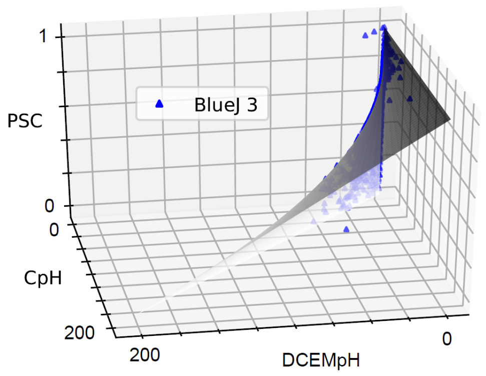

Fig. 1. Surface representing Equation 1 and each BlueJ 3 user mapped by their metric coordinates (blue triangles).

3 Results

Table 1 summarizes results discussed in this section. Statistical tests were performed for each individual cohort to reveal the statistical significance in the difference between the metrics in BlueJ 3 and BlueJ 4. We carried out a Shapiro-Wilk test for normality [18] and after the null hypothesis for normality was rejected for all distributions, a Mann-Whitney U test for statistical significance [14] was performed along with a calculation of Cohen’s d as a measure of effect size (ES) [9]. Cohen’s d was calculated using BlueJ 4 as the experimental group and BlueJ 3 as the control group in all cases. Since the effect direction was consistent for every metric in all user cohorts (increase in DCEMpH, decrease in CpH, increase in PSC), only absolute values are displayed in Table 1. All tests revealed statistical significance with $p < 0.05$ and the effect sizes support our results in Table 1 and Sections 3.1 and 3.2. As there are two results (Mann-Whitney and Cohen’s d), three metrics (DCEMpH, CpH, PSC) and four data cohorts (3t4, 4t3, Overlap and X), there are 24 independent results. For space, we present the complete set of the statistical results along with the processed data in a Zenodo open source repository. [5]

Table 1. Mean values of programming time in Hours (H), Displayed Compiler Error Messages per Hour (DCEMpH), manual Compilations per Hour (CpH) and Percentage of Successful manual Compilations (PSC) for BlueJ 3 and BlueJ 4 of all user cohorts. Effect sizes (ES) using Cohen's d are displayed alongside each pair of metrics. The direction of effect is omitted in ES since it is consistent in all metrics across all cohorts. ES Sum refers to the cumulative effect size derived by summing all ES values in that row. The last three columns display the values of distance between BlueJ 3 and BlueJ 4 using the methods described in Section 2.5. In the first row, n_3 and n_4 refer to the number of users in cohorts X3 and X4 respectively.

User Cohort (n)	H		DCEMpH			CpH			PSC			ES	M1	M2	M3
	V3	V4	V3	V4	ES	V3	V4	ES	V3	V4	ES	Sum			
X ($n_3 = 727, n_4 = 536$)	19	30	11	17	.36	22	12	.59	.52	.76	1.2	2.15	1.39	12.11	1.22
3t4 ($n = 1008$)	101	41	7	9	.32	15	10	.34	.53	.74	1.2	1.86	.8	5.16	.66
4t3 ($n = 190$)	62	9	7	12	.55	17	15	.15	.58	.73	.74	1.44	.78	5.68	.54
Overlap ($n = 463$)	125	44	6	10	.48	14	11	.23	.54	.73	1.06	1.77	.76	4.92	.64

3.1 RQ1: Exclusive vs Transition Use

Our first research question was: *How does transitioning between environments affect novice interaction regarding compilation and error messages as opposed to being exposed to a single environment?*

Displayed Compiler Error Messages per Hour (DCEMpH) are greater in BlueJ 4 than in BlueJ 3 for all cohorts. The mean values in BlueJ 3 and BlueJ 4 for each cohort are: 11 and 17 for cohort X, 7 and 9 for cohort 3t4, 7 and 12 for cohort 4t3, and 6 and 10 for cohort Overlap. Manual Compilations per Hour (CpH) are lower in BlueJ 4 than in BlueJ 3 in all cohorts. The mean values in BlueJ 3 and BlueJ 4 for each cohort are: 22 and 12 for cohort X, 15 and 10 for cohort 3t4, 17 and 15 for cohort 4t3, and 14 and 11 for cohort Overlap. Regarding Percentage of Success of manual Compilations (PSC), the differences between BlueJ 3 and BlueJ 4 are very similar for all user cohorts, with more successful manual compilations in BlueJ 4. Manual compilations in BlueJ 4 are 73-76% successful compared to BlueJ 3, in which they are 52-58% successful.

All three methods discussed in Section 2.5 show a larger difference between X4 and X3 interactions for users in cohort X than for the rest of the cohorts. Specifically: (1) the average minimum distance between cohorts and the BlueJ 3 equation is 1.39 for X4 and around 0.76-0.8 for the transition cohorts, (2) the

[5] Zenodo repositories cannot be anonymous. For review, this is omitted and presented in a Google Drive folder:
https://drive.google.com/drive/folders/1LZ8j40S_1N5iYVG5hMG0SdWN3SwH9ISw

distance between the means of the metrics for users in X3 and X4 is 12.11, while for the transition cohorts, the distances are between 4.92 and 5.68, and (3) the minimum distance between X4 mean coordinate values and the BlueJ 3 equation is 1.22, whereas for the rest of the cohorts it is between 0.54 and 0.66. The results of these three methods align with the cumulative ES reported in Table 1. This is expected since all discussed methods and the sum of the absolute values of the ES incorporate all three metrics in their calculation.

Based on these findings we conclude that the difference in programming behavior between using BlueJ 3 and using BlueJ 4 is substantially greater for users that used one of the two versions exclusively than for those who program using both versions. This is primarily the result of changes in numbers of displayed compiler error messages and manual compilations, as the difference in successful manual compilations between BlueJ versions remains relatively stable across different cohorts of users.

3.2 RQ2: Order of Transition

Our second research question was: *How does the order of transitioning between environments affect this interaction?*

Regarding DCEMpH, the difference between the means in BlueJ 3 and BlueJ 4 for the each of the transition cohorts from highest to lowest are: (1) 71% for cohort 4t3, (2) 67% for cohort Overlap, (3) 29% for cohort 3t4. Regarding CpH, the differences between the means in BlueJ 3 and BlueJ 4 for the each of transition cohorts from highest to lowest are: (1) 33% for cohort 3t4, (2) 21% for cohort Overlap, (3) 12% for cohort 4t3. Regarding PSC, the differences between the means in BlueJ 3 and BlueJ 4 are very similar for all user cohorts.

Methods 1-3 (discussed in Section 2.5) present similar numbers for the interaction difference between BlueJ 3 and BlueJ 4 for the users in cohorts 3t4, 4t3, and Overlap. Although there are minor differences, they are not on the same order of magnitude as those for the X cohort (which is around twice as high for all three methods). Cumulative effect sizes using Cohen's d also align with this as they range between 1.44 and 1.86 for all transition cohorts.

Based on these findings, we conclude that the order in which users transition from one BlueJ version to the other does not play a substantial role in the programming behavior change between BlueJ 3 and BlueJ 4 and the interaction regarding DCEMpH, CpH and PSC is not significantly altered.

4 Threats to Validity

Blackbox data are anonymous, and do not contain any information about the programming level of BlueJ users. Although BlueJ is not practical for experienced programmers, some users could be educators trying out the environment or making sure that the environment can execute certain exercises without issues. Additionally, one Blackbox user could in fact be several users working on the same machine, which is very common in institutional labs. This limitation is inherent to all studies that use these data [7].

The metrics involved in the current work incorporate the time spent on BlueJ. However, network interruptions can cause Blackbox to stop logging events for a session. In our analysis we treat the last event logged in the session as the true final event. This approach inevitably results in some missing data – if a user's connection is disrupted, the last logged activity for that session may not be complete. We regard this as a minor threat, since we are comparing two different BlueJ versions and the probability of an incomplete session should be the same for both versions, potentially mitigating this otherwise unavoidable issue to some degree.

Finally, one of the metrics we used (Displayed Compiler Error Messages per Hour) involves counting the numbers of shown compiler error messages that are logged in Blackbox. These logged events can sometimes be triggered by users inadvertently in BlueJ 4. Since these events are triggered by moving the cursor to the area of the code responsible for the error, if users accidentally move their cursor or if they click the offending code area to fix the error, this counts as a shown error message in the current analysis. We will work towards isolating these instances in the future.

5 Discussion

In this study, we explored the programming interaction between novices and two versions of the BlueJ pedagogical environment that differ fundamentally in compilation and error message presentation. BlueJ 3 features a click-to-compile mechanism and enforced first error message presentation. BlueJ 4 features automatic error checking and on-demand error message presentation. The aims of our research were: (1) to investigate how exposure to a single BlueJ version affects the interaction between novices and the environment regarding compilation and error messages compared to being exposed to multiple BlueJ versions, and (2) to what extent the order in which this exposure takes place affects the interaction.

The analysis was conducted using programming process data from four distinct user cohorts using two different BlueJ versions. The cohorts included users who exclusively used only one of the two BlueJ versions, users who transitioned from BlueJ 3 to BlueJ 4, users who transitioned from BlueJ 4 to BlueJ 3, and users who switched multiple times between the two. In order to answer our research questions, we utilized three metrics that describe user interaction with the environment regarding compilation and error message presentation: displayed compiler error messages per hour, manual compilations per hour, and percentage of successful manual compilations. Including these metrics in our study could allow researchers to generalize our findings outside of the BlueJ context, as the interaction they measure is common to almost all programming environments.

By quantifying the cumulative interaction comprised by the three metrics, we conclude that programming interaction difference between programming in BlueJ 3 and BlueJ 4 is higher for users who only used one of the two versions exclusively than for users who transitioned between versions, primarily due to the numbers of displayed compiler error messages and manual compilations. The order of transitioning between versions does not seem to play a role of similar magnitude in this difference however. It is reasonable for novices who learn programming while using a single environment to adapt their interaction habits according to how the environment operates while displaying substantially diverging behavior from those using another environment. In contrast, novices who transition between programming environments (regardless of the reason behind such transitions) could be influenced by mechanisms present in both versions and display a moderated behavior.

In terms of differences in displayed compiler error messages over time, the cohort of users who moved from BlueJ 4 to BlueJ 3 exhibited the largest variation, followed by those who kept switching repeatedly between versions, those who used exclusively one BlueJ version, and those who moved from BlueJ 3 to BlueJ 4. All cohorts showed changes situated between the small to large spectrum of effect size interpretations [17].

In terms of differences in manual compilations over time, the cohort of users who exclusively used only one BlueJ version exhibited the largest variation between versions, followed sequentially by those who moved from BlueJ 3 to BlueJ 4, those who kept switching between versions and those who moved from BlueJ 4 to BlueJ 3. All cohorts showed changes situated between the very small to large spectrum of effect size interpretations.

In terms of successful manual compilations, differences between versions are very high in all cohorts of users, and the magnitude of the differences is relatively stable. All cohorts showed changes situated between the medium (users who moved from BlueJ 4 to BlueJ 3) to very large spectrum (users who exclusively used one BlueJ version and users who moved from BlueJ 3 BlueJ 4) of effect size interpretations.

Regardless of whether users were exposed to a single BlueJ version or multiple versions, we made a common observation: in BlueJ 4, users see significantly more error messages, compile manually less frequently and have higher manual compilation success. This is in accord with previous findings that took a more holistic approach to analyzing programming activity in BlueJ 3 and BlueJ 4 [2], and without comparing transitioning users with those who programmed only in one of the two [1].

A further observation is that users exposed only to a single environment generated more error messages and compiled manually more frequently in both BlueJ versions than users exposed to both. An exception to this are novices that moved from BlueJ 4 to BlueJ 3, who seem to compile more frequently than users who programmed only in BlueJ 4. This requires further investigation.

Due to the nature of Blackbox data, we have no access to the reasons why users transition between environments. A survey targeting the reasons behind this could reveal important insights into the motivations for such behavior. We can speculate that some novices who used only one environment felt comfortable while programming in it, therefore not attempting to switch to another version.

Others may be constrained by institutional or other factors. Users who moved from one version to another may have had trouble with the manner in which the environment operated. For example, those who moved from version 3 to 4 could feel restricted by the need to compile manually or by only having access to the first error message. On the other hand, users who moved from 4 to 3 could feel overwhelmed by the constant red underlining of errors triggered by automatic compilations [5] and desired a more simple approach. Again, institutional and other factors beyond student control could also be at play.

Our research indicates that programming behavior is largely determined by the mechanisms of the programming environment, and that the transfer of behavior from one environment to another, if it occurs, is affected substantially by the restrictions of the environment. This is evidence that the choices of environment designers heavily affect novice programmer behavior and likely their learning opportunities. Programming educators should be aware of these effects on novices, but also of how the featured mechanisms of the chosen environments in their courses operate. It could be beneficial if educators are encouraged to facilitate short tutorials during which they explain the functionalities of the chosen programming environment and address students' questions on their use. These tutorials could be taking place in the beginning of the term, which is when students are just starting to get exposed to many new variables and concepts that introductory programming courses introduce, and serve as a proactive step against frustration and confusion, emotions that commonly emerge among CS1 students [13].

References

1. Anonymous, A.: title removed for anonymous review. In: Proceedings of the 51st ACM Technical Symposium on Computer Science Education. p. xxxxx. SIGCSE '20, Association for Computing Machinery, New York, NY, USA (2020). https://doi.org/xxxxx
2. Anonymous, B.: title removed for anonymous review. In: United Kingdom & Ireland Computing Education Research Conference. p. xxxxx. UKICER '20, Association for Computing Machinery, New York, NY, USA (2020). https://doi.org/xxxxx
3. Anonymous, C.: title removed for anonymous review. Commun. ACM (jul 2021). https://doi.org/10.1145/xxxx
4. Anonymous, D.: title removed for anonymous review. In: Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education. ITiCSE '20, Association for Computing Machinery, New York, NY, USA (2020). https://doi.org/10.1145/xxxx
5. Anonymous, E.: title removed for anonymous review. In: Proceedings of the 53rd ACM Technical Symposium on Computer Science Education V. 1. SIGCSE 2022, Association for Computing Machinery, New York, NY, USA (2022). https://doi.org/10.1145/xxxx
6. Anonymous, F.: title removed for anonymous review. In: Proceedings Companion of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education. ITiCSE 2018 Companion, Association for Computing Machinery, New York, NY, USA (2018). https://doi.org/10.1145/xxxx

7. Brown, N.C.C., Altadmri, A., Sentance, S., Kölling, M.: Blackbox, five years on: An evaluation of a large-scale programming data collection project. In: Proceedings of the 2018 ACM Conference on International Computing Education Research. p. 196–204. ICER ’18, ACM, NY, NY, USA (2018). https://doi.org/10.1145/3230977.3230991
8. Brown, N.C.C., Kölling, M., McCall, D., Utting, I.: Blackbox: A large scale repository of novice programmers’ activity. In: Proceedings of the 45th ACM Technical Symposium on Computer Science Education. p. 223–228. SIGCSE ’14, ACM, NY, NY, USA (2014). https://doi.org/10.1145/2538862.2538924
9. Cohen, J.: Statistical power analysis for the behavioral sciences. Routledge (2013). https://doi.org/10.4324/9780203771587
10. Hertz, M.: What do ”CS1” and ”CS2” mean? investigating differences in the early courses. In: Proceedings of the 41st ACM Technical Symposium on Computer Science Education. p. 199–203. SIGCSE ’10, ACM, NY, NY, USA (2010). https://doi.org/10.1145/1734263.1734335
11. Jadud, M.C.: Methods and tools for exploring novice compilation behaviour. In: Proceedings of the 2nd International Workshop on Computing Education Research. p. 73–84. ICER ’06, ACM, NY, NY, USA (2006). https://doi.org/10.1145/1151588.1151600
12. Kölling, M., Quig, B., Patterson, A., Rosenberg, J.: The BlueJ system and its pedagogy. Computer Science Education **13**(4), 249–268 (2003). https://doi.org/10.1076/csed.13.4.249.17496
13. Lishinski, A., Rosenberg, J.: All the pieces matter: The relationship of momentary self-efficacy and affective experiences with cs1 achievement and interest in computing. In: Proceedings of the 17th ACM Conference on International Computing Education Research. p. 252–265. ICER 2021, Association for Computing Machinery, New York, NY, USA (2021). https://doi.org/10.1145/3446871.3469740, https://doi.org/10.1145/3446871.3469740
14. Mann, H.B., Whitney, D.R.: On a test of whether one of two random variables is stochastically larger than the other. The Annals of Mathematical Statistics **18**(1), 50–60 (1947), http://www.jstor.org/stable/2236101
15. Nelder, J.A., Mead, R.: A simplex method for function minimization. The Computer Journal **7**(4), 308–313 (01 1965). https://doi.org/10.1093/comjnl/7.4.308
16. Reis, C., Cartwright, R.: Taming a professional IDE for the classroom. In: Proceedings of the 35th SIGCSE Technical Symposium on Computer Science Education. pp. 156–160. SIGCSE ’04, ACM, NY, NY, USA (2004). https://doi.org/10.1145/971300.971357
17. Sawilowsky, S.S.: New effect size rules of thumb. Journal of modern applied statistical methods **8**(2), 26 (2009). https://doi.org/10.22237/jmasm/1257035100
18. Shapiro, S.S., Wilk, M.B.: An analysis of variance test for normality (complete samples). Biometrika **52**(3/4), 591–611 (1965), http://www.jstor.org/stable/2333709