Exploring the Relationship Between Novice Programmer
Confusion and Achievement

Diane Marie C. Lee[1], Ma. Mercedes T. Rodrigo[1], Ryan S.J.d. Baker[2],
Jessica O. Sugay[1], and Andrei Coronel[1]

[1] Department of Information Systems and Computer Science
Ateneo de Manila University
Loyala Heights, Quezon City, Philippines
[2] Department of Social Science and Policy Studies
Worcester Polytechnic Institute
Worcester MA, USA
dianemarielee@gmail.com,mrodrigo@ateneo.edu,rsbaker@wpi.edu,
jsugay@ateneo.edu,acoronel@ateneo.edu

Abstract. Using a discovery-with-models approach, we study the relationships between novice Java programmers' experiences of confusion and their achievement, as measured through their midterm examination scores. Two coders manually labeled samples of student compilation logs with whether they represent a student who was confused. From the labeled data, we built a model that we used to label the entire data set. We then analysed the relationship between patterns of confusion and non-confusion over time, and students' midterm scores. We found that, in accordance with prior findings, prolonged confusion is associated with poorer student achievement. However, confusion which is resolved is associated with statistically significantly better midterm performance than never being confused at all.

Keywords: Confusion, novice programmers, student affect, affective sequences, student achievement, Java, BlueJ

1 Introduction

Learning to program is an emotional experience. According to interview-based reports in [13], when novice programmers begin reading problem specifications, a lack of comprehension of the instructions can lead to a feeling of disorientation. When they start to understand the problem, they can continue to feel lost, this time because they do not know how to begin the programming process. The first error they encounter can trigger strong negative emotions. Subsequent errors can lead to a sense of resignation and a reluctance to continue. Students also report different emotions based on their use of feedback while programming [13]. Students who effectively use feedback to guide their programming in some cases feel a constructive form of confusion that motivates them to systematically

experiment with their code and converge to a solution. In contrast, students who ignore or are unable to use feedback liken their experiences to running in a hamster wheel: They repeatedly try bug fixes with no reflection or understanding, eventually leading to a feeling of despair [13].

Recent studies illustrate the relationships between affective states and achievement in a variety of learning contexts. Craig et al. [6] found that, among students using an AutoTutor, an intelligent tutor for computer literacy, the affective states of confusion and flow [7] (called "engaged concentration" in [3]) were positively correlated with achievement. Boredom, regarded by Craig et al. [6] as the antithesis of flow, was negatively correlated with learning gains. Lagud and Rodrigo [15] found that high achieving students using Aplusix, an intelligent tutor for algebra, experienced significantly more flow than students with low achievement. Low-achieving students in the same study experienced the highest levels of boredom and, in contrast to [6], confusion.

In the context of learning to program, Rodrigo et al [19] found that boredom and confusion were again associated with lower achievement among students taking their first college-level programming course. Khan et al. [12] found that high levels of arousal regardless of valence – delight, rejoicing, even terror and restlessness – can lead to better debugging performance among students.

One of the limitations of these studies is that they examined these affective states in isolation, rather than as links in a cognitive-affective chain, cf. [10]. Researchers have therefore grown increasingly interested in studying affective dynamics, defined as the natural shifts in learners' affective states over time [8]. Studies by [3], [8], and [16], have found that certain affective states such as flow/engaged concentation, confusion, boredom, and frustration tend to persist over time. Confused students tend to stay confused; engaged students tend to stay engaged.

Affect and affective dynamics have also been shown to influence student achievement and key student behaviours that drive learning. Research by D'Mello et al [10] has found that students who are bored tend to get trapped in a "vicious cycle" of boredom, incorrect responses to problem solving questions, and negative feedback. Boredom has also been found to be an antecedent of gaming the system, a behaviour associated with significantly poorer learning, among students using simulation problem games and intelligent tutors [3]. By contrast, [10] found that the more positive affective state of curiosity leads to correct responses to problem solving questions, positive feedback, happiness, and continued curiosity. Confusion was found to have a dual nature. If a student is unable to work through confusion, the student is likely to give incorrect responses to questions, receive negative feedback and eventually experience frustration. If, on the other hand, confusion is resolved, the student's responses to questions will tend to be correct, the student will receive positive feedback and eventually transitions into a neutral affective state [10]. Some of D'Mello et al.'s [10] findings were corroborated by recent study by Rodrigo, Baker, and Nabos [20] that examined students using an intelligent tutor for scatterplots. They found that boredom was both persistent and detrimental to learning. Confusion could be positive if

punctuated with periods of engaged concentration. Prolonged confusion, on the other hand, had a negative impact on student achievement.

Within this paper, we present a study that uses a discovery-with-models approach [2] to investigate novice programmer confusion and its impact on student overall achievement in a course. Two human coders manually labeled text replays [4] of excerpts of student compilation behaviour (termed "clips") as to whether they represented student confusion. From the labeled data, we built a machine-learned model of student confusion and used this model to label the entire dataset. We then aggregated student affect over time into sequences of confusion and non-confusion, and correlated each student's frequency of demonstrating these sequences with the student's midterm exam score.

2 Data Collection

The population under study consisted of 149 freshman and sophomore college students aged 15 to 17 from the Department of Information Systems and Computer Science (DISCS) of the Ateneo de Manila University during the School Year 2009-2010. These students are younger than the typical college freshman in other countries because the Philippines only has 10 years of mandatory basic education, two years less than basic education in other countries. These students were enrolled in five sections of CS21 A-Introduction to Computing I and were divided into five sections.

We collected the students' compilation logs during four practical lab sessions spread over two months. Laboratory sessions were held in two classrooms. Each student was assigned to a computer installed with a specially instrumented version of the BlueJ Interactive Development Environment (IDE) for Java (Figure 1) [11]. This version of BlueJ was connected to a SQLite database server. Upon each compilation, BlueJ saved a copy of the students' program and any compile-time error messages, on the server.
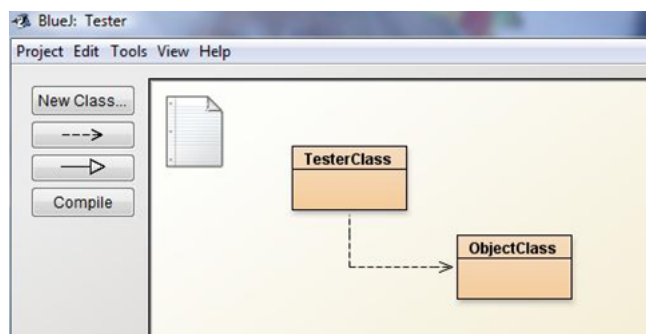

Fig. 1. Screenshot of the BlueJ IDE

There were some lab sessions during which the data collection server was not running correctly. We were therefore only able to collect logs from 340 student-lab sessions, giving a total of 13,528 compilations.

3   Data Labeling

The first step in the labeling process was deciding the level of granularity at which the compilations were going to be examined. It was not possible, for example, to tell whether a student was confused from a single compilation. Neither did we want to judge student confusion based on an entire session's worth of compilations. If, later on, we were to design interventions for student confusion, we would like these interventions to be sensitive enough to detect confusion while the student is still writing the program and not after he or she has finished. We needed a sequence of compilation events to conduct data labeling but had to decide on the sequence length. We decided to group the compilations into clips of 8 compilations each where the number of compilations, 8, was chosen arbitrarily.

There were cases in which a Java program consisted of two or more Java classes. When a student compiled, all Java classes within the program were compiled and logged with the same time stamp. To generate the clips, we sorted all compilations first by student identifier, then by Java class name, then by time stamp. We then grouped the compilations with same student identifier and Java class into sets of up to 8 compilations. This produced a total of 2,386 clips which could be coded.

We attempted to sample 2 random clips per student-lab for labeling, which would have given 680 clips. However, some students only had a single clip for a specific lab (for example, due to requiring fewer than 16 compilations to complete a lab). We therefore coded a total of 664 clips.

A small application was written to display the clips in the form of low-fidelity text replays (Figure 2) [4], where human judgment is applied to pretty-prints of sub-segments of log files. A previous study by Baker et al [4] showed that text replays enable coders to label student disengagement with greater efficiency than higher fidelity methods such as human observations and video replays, while maintaining good inter-rater reliability. This method of data labeling has also been used to study student meta-cognitive planning processes [17] and systematic experimentation behaviour patterns [21], again achieving good inter-rater reliability.

Once the application was completed, the two co-authors met to decide on criteria for labeling clips as "Confused","Not Confused" or "Bad Clip". The coders decided that the student's behavior implied confusion when

1. The same error appeared in the same general vicinity within the code for several consecutive compilations. The coders inferred that the student did not know what was causing the error and how to fix it.
2. An assortment of errors appeared in consecutive compilations and remained unresolved. The coders inferred that the student was experimenting with
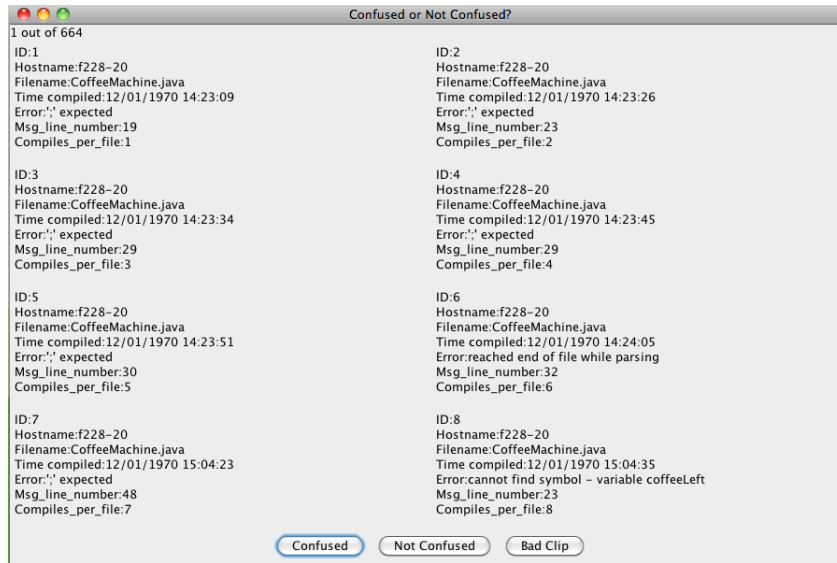
Fig. 2. Screen shot of the low-fidelity text replay playback program for a "confused" student

solutions, changing the actual error message but not addressing the real source of the error.

3. Code malformations that showed a poor understanding of Java constructs, e.g. "return outside method". The coders inferred that the student did not grasp even the basics of program construction, despite the availability of written aids such as Java code samples and explanatory slides.

When a clip showed a student successfully resolving an error or an assortment of errors, it was labeled as "not confused." If a clip had compilations of programs other than the assigned laboratory exercise, e.g. instructor-supplied sample code or tester programs, it was labeled "bad clip."

Interrater reliability was acceptably high with Cohen's [5] Kappa = .77.

Given these labels, our next step was to build a detector to label the rest of the data. In order to do so, we filtered out all "bad clips" and all clips with less than 8 compilations. It was also necessary to filter out data from one of the five class sections due to a logging error. Finally, we removed all clips in which the two raters disagreed, for the purposes of building a model. We were left with 418 clips with which to build the model.

4   Model Construction and Data Relabeling

RapidMiner version 5.1 [1] was used to build a decision tree, using the J48 implementation of the C4.5 algorithm [18] (Figure 3), using the following feature set:

1. Average time between compilations
2. Maximum time between compilations
3. Average time between compilations with errors
4. Maximum time between compilations with errors
5. Number of compilations with errors
6. Number of pairs of consecutive compilations with the same error

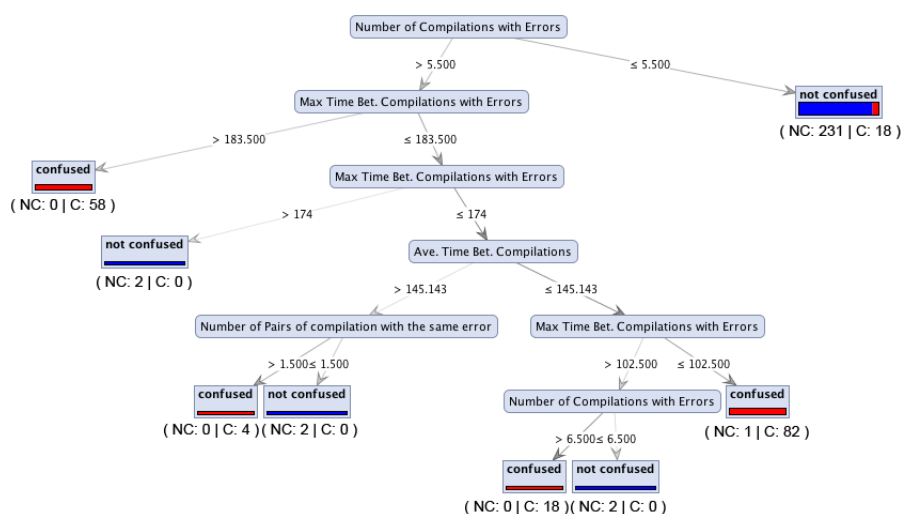Student-level 10-fold cross-validation of the model resulted in an excellent Kappa of .86.

Fig. 3. Decision tree depicting the criteria for labeling a clip as confused or not confused

The model was implemented as a Java program. The program was then used to label all of the clips in the data set. Removing all bad clips, we generated three sets of confused-not confused sequences. The first set consisted of single states. The second set consisted of sequences of two consecutive states (2-step). The third and final set consisted of sequences of three consecutive states (3-step).

There were students who did not have any 2-step or 3-step sequences for any specific lab. Hence, the number of students in each set varied. We had 113 students in the 1-step set, 95 students in the 2-step set and 71 students in the 3-step set.

We counted the number of occurrences of each state or sequence per student within each set. The total number of sequences per student varied. We therefore normalized the data by dividing the number of occurrences of each state or sequence per student by the total number of occurrences for that student. We then correlated these percentages with the students' midterm examination

scores. The midterm examination was composed of questions that tested students' programming comprehension. Students were given code fragments as well as whole programs. They were then asked questions about the code, including whether fragments would compile, what the output of a code fragment would be, or which part of the code performed a given action.

5   Results and Discussion

In terms of 1-step sequences, there was a marginally significant negative correlation between the incidence of confusion and student achievement (r= -.168; p=.075). This indicates, rather unsurprisingly, that students who are confused in general tend to receive lower scores on the midterm exam.

The 2-step sequences (Table 1) show that confusion sustained over two clips is also negatively correlated with midterm scores (r=- .229; p=.026). No other 2-step sequences were significantly different than chance, suggesting that only sustained confusion impacts learning negatively to a significant degree.

Table 1. r-values of the incidence of 2-step sequences with midterm scores. Numbers in parenthesis are the p-values. Statistically significant correlations are in dark grey

	Not Confused - Not Confused	Not Confused - Confused	Confused - Not Confused	Confused - Confused
Relationship with midterm	.064 (.539)	.139 (.180)	.144 (.163)	-.229 (.026)

Analysis of the 3-step sequences sheds further light on the relationship between confusion and learning. Confusion sustained over three consecutive clips, is again negatively correlated with midterm scores (CCC: r=-.337; p=.004). On the other hand, a student who over a 3-clip sequence starts out confused, resolves their confusion, and continues to be non-confused, then achieves higher midterm scores (CNN: r=.233, p=.05). This is consistent with D'Mello and Graesser's [9] cognitive disequilibrium model in which confusion has to be resolved through thought, reflection, and deep thinking to return the learner into a flow state. Unresolved confusion, on the other hand, leads to frustration and boredom.

Note that simply switching from confused to non-confused is not sufficient for positive learning; as shown in Table 2, other patterns of transition between confusion and non-confusion are not significantly associated with midterm performance; the key pattern for learning appears to be confusion which is resolved, and does not recur.

Also, in contrast to confusion which is resolved, never being confused at all is not significantly associated with midterm performance, r= -0.015, p=0.901. This finding is additional support for the hypothesis that some degree of confusion is

beneficial for learning. Deep processing of the subject matter appears to require being confused, but resolving that confusion.

Table 2. r-values of the incidence of 3-step sequences with midterm scores. Numbers in parenthesis are the p-values. Statistically significant correlations are in dark grey. N = Not Confused while C = Confused

	NNN	NNC	NCN	NCC	CNN	CNC	CCN	CCC
Rel.	-.015	.014	.062	-.046	.233	.163	.052	-.337
with midterm	(.901)	(.909)	(.610)	(.704)	(.05)	(.174)	(.665)	(.004)

These findings shed additional light on the dual nature of confusion. Students experience confusion when confronted with new material or new problems they cannot immediately solve. At this point, confusion may spur students to work through the problems and resolve them, returning eventually to a hopeful and enthusiastic state, as hypothesized in [14]. [9] consider this type of confusion to be productive. On the other hand students can become stuck in a state of hopeless – persistent – confusion, which does not lead to learning.

6   Conclusion

The purpose of this paper was to study the relationship between novice programmers' experience of confusion and their achievement as measured through their midterm examination scores. Using a discovery-with-models approach, we created a model of confusion using manually-labeled samples of student compilation logs. We then used this model to label the entire data set. From the labeled data set, we distilled students' patterns of transitions between being confused and not confused, and correlated these sequences' incidence with each student's midterm scores. We found that prolonged confusion has a negative impact on student achievement. Confusion which is resolved, however, is positively associated with midterm scores.

Overall, the findings of this study support D'Mello and Graesser's [9] model of cognitive disequilibrium, applicable to deep learning environments. The model proposes that confusion can be a useful affective state when it spurs learners to exert effort deliberately and purposefully to resolve cognitive conflict. If the learners are successful, they return to a state of flow [7]. If they are unsuccessful, though, they could become frustrated or bored, and may decide to disengage from the learning task altogether [9]. The model and these findings challenge educational designers to develop learning tasks that are complex enough to stimulate a constructive level of learner confusion while making scaffolding available to prevent hopelessness and disengagement.

As a response to these challenges, one important future use of this paper's model of confusion and the subsequent findings will be to support the incorporation of tools for automatically detecting novice programmer confusion, into

computer science education environments. As the detector is based solely upon log files, it can be deployed at scale quite easily. These tools may enable educators to identify novices who are at academic risk and provide these students with the support they need to learn the material and persist in their studies. In addition, the detector could eventually be the basis of automated response to student confusion.

6.1 Acknowledgements

References

1. http://www.rapid-i.com
2. Baker, R.: Data mining for education. International encyclopedia of education (3rd ed.). Oxford, UK: Elsevier (2010)
3. Baker, R., D'Mello, S., Rodrigo, M., Graesser, A.: Better to be frustrated than bored: The incidence and persistence of affect during interactions with three different computer-based learning environments. International Journal of human-computer studies 68(4), 223–241 (2010)
4. Baker, R., Corbett, A., Wagner, A.: Human classification of low-fidelity replays of student actions. In: Proceedings of the Educational Data Mining Workshop at the 8th International Conference on Intelligent Tutoring Systems. pp. 29–36. Citeseer (2006)
5. Cohen, J.: A coefficient of agreement for nominal scales. Educational and psychological measurement (1960)
6. Craig, S., Graesser, A., Sullins, J., Gholson, B.: Affect and learning: an exploratory look into the role of affect in learning with AutoTutor. Journal of Educational Media 29, 241–250 (2004)
7. Csikszentmihalyi, M.: Flow: The psychology of optimal experience: Steps toward enhancing the quality of life. Harper Collins Publishers (1991)
8. DMello, S., Taylor, R., Graesser, A.: Monitoring affective trajectories during complex learning. In: Proceedings of the 29th annual meeting of the cognitive science society. pp. 203–208 (2007)
9. DMello, S., Graesser, A.: Dynamics of Cognitive-Affective States During Deep Learning. Paper under review

10. DMello, S., Person, N., Lehman, B.: Antecedent-consequent relationships and cyclical patterns between affective states and problem solving outcomes. In: Proceedings of AIED (2009)
11. Jadud, M., Henriksen, P.: Flexible, reusable tools for studying novice programmers. In: Proceedings of the fifth international workshop on Computing education research workshop. pp. 37–42. ACM (2009)
12. Khan, I., Brinkman, W., Hierons, R.: Do moods affect programmers debug performance? Cognition, Technology & Work pp. 1–14
13. Kinnunen, P., Simon, B.: Experiencing programming assignments in CS1: the emotional toll. In: Proceedings of the Sixth international workshop on Computing education research. pp. 77–86. ACM (2010)
14. Kort, B., Reilly, R., Picard, R.: An affective model of interplay between emotions and learning. In: Proceedings of IEEE International Conference on Advanced Learning Technologies. pp. 6–8 (2001)
15. Lagud, M., Rodrigo, M.: The Affective and Learning Profiles of Students Using an Intelligent Tutoring System for Algebra. In: Intelligent Tutoring Systems. pp. 255–263. Springer (2010)
16. McQuiggan, S., Robison, J., Lester, J.: Affective Transitions in Narrative-Centered Learning Environments. Subscription Prices and Ordering Information 13(1), 40–53 (2010)
17. Montalvo, O., Baker, R., Sao Pedro, M., Nakama, A., Gobert, J.: Identifying Students Inquiry Planning Using Machine Learning. In: Proceedings of the 3rd International Conference on Educational Data Mining. pp. 141–150 (2010)
18. Quinlan, J.: C4. 5: programs for machine learning. Morgan Kaufmann (1993)
19. Rodrigo, M., Mercedes, T., Baker, R., Jadud, M., Amarra, A., Dy, T., Espejo-Lahoz, M., Lim, S., Pascua, S., Sugay, J., et al.: Affective and behavioral predictors of novice programmer achievement. ACM SIGCSE Bulletin 41(3), 156–160 (2009)
20. Rodrigo, M., Baker, R., Nabos, J.: The Relationships Between Sequences of Affective States and Learner Achievement. In: Proceedings of the 18th International Conference on Computers in Education. pp. 56–60 (2010)
21. Sao Pedro, M., Baker, R., Montalvo, O., Nakama, A., Gobert, J.: Using Text Replay Tagging to Produce Detectors of Systematic Experimentation Behavior Patterns. Proceedings of the 3rd International Conference on Educational Data Mining pp. 181–190 (2010)