Affective and Behavioral Predictors of Novice Programmer Achievement

Ma. Mercedes T. Rodrigo[1]
mrodrigo@ateneo.edu

Ryan S. Baker[2]
rsbaker@cs.cmu.edu

Matthew C. Jadud[3]
matthew.c@jadud.com

Anna Christine M. Amarra[1]
camarra@ateneo.edu

Thomas Dy[1]
alpha8888_2@yahoo.com

Maria Beatriz V. Espejo-Lahoz[1]
blahoz@ateneo.edu

Sheryl Ann L. Lim[1]
lim.she@gmail.com

Sheila A. M. S. Pascua[4]
sspascua@gmail.com

Jessica O. Sugay[1,4]
jsugay@ateneo.edu

Emily S. Tabanao[1,5]
emilytabanao@yahoo.com

ABSTRACT
We study which observable affective states and behaviors relate to students' achievement within a CS1 programming course. To this end, we use a combination of human observation, midterm test scores, and logs of student interactions with the compiler within an Integrated Development Environment (IDE). We find that confusion, boredom and engagement in IDE-related on-task conversation are associated with lower achievement. We find that a student's midterm score can be tractably predicted with simple measures such as the student's average number of errors, number of pairs of compilations in error, number pairs of compilations with the same error, pairs of compilations with the same edit location and pairs of compilations with the same error location. This creates the potential to respond to evidence that a student is at-risk for poor performance before they have even completed a programming assignment.

Categories and Subject Descriptors
K.3.2 [**Computer and Information Science Education**]: Computer Science Education

General Terms
Human Factors

Keywords
Novice Programmers, Affect, Achievement

ITiCSE'09, July 6–9, 2009, Paris, France.

1. INTRODUCTION
Negative affect and behaviors have a powerful influence on novice programmers' learning [7, 11]. Students respond to bugs in a variety of ways, including non-constructive behaviors such as disengaging from the task by giving up or by attempting to fix bugs by guessing or by systematically applying code they have seen previously, regardless of is original context. When students perceive bugs as a valuation of their personal competence, rather than responding with the goal of mastering programming, their mistakes discourage them and can cause them to give up on programming [17].

Since at least the 1980s, computer science education researchers have searched for ways for computer science teachers can better support their students. Approaches include the study of common errors [cf. 14] and vignettes of student bugs to gain insight into student problem-solving processes [cf. 19]. Still, it is often a challenge for computer science educators to recognize indicators of diminishing performance and help students overcome learning problems. Given the spectrum of possible ways that a student can react to difficulties (both affective and behavioral), it is important for educators to know to which behaviors and emotions they should be most attuned. Which behaviors and emotions are most associated with novice programming students' eventual achievement?

There have been a variety of answers obtained through qualitative examination, including that programmer boredom and resentment should be mitigated [16], student motivation and initiative should be nurtured [9], and that negative emotions such as anger should be kept under control [12]. However, qualitative evidence of this nature only tells us that a problem exists – it does not tell us which problems are most important.

Recent studies have attempted to tease out these factors more quantitatively. Perfectionism and self-esteem have been found to be positive predictors of novice programmer learning while emotional health and social well-being have not [2]. Disliking programming appears to be associated with lower success in early programming courses [7]. High states of arousal such as delight or fear tend to improve programming performance [10].

Some studies suggest that it may be possible to automatically detect programming students' emotions, as well as other factors that lead to differences in achievement. [10] proposes that it may be possible to detect moods from programmer uses of the keyboard or mouse but has not published experimental results as of the time of this writing. [13, 15] have shown that there is a measurable relationship between students' compilation behaviors and their achievement levels, which are discussed in greater detail in the following section.

2. RESEARCH QUESTIONS

This study attempts to answer two main questions. First, which observable affective states and behaviors can predict each student's degree of achievement? Studies have shown that, in computer literacy learning, confusion significantly predicted post-test scores [8]. Learning outcomes in computer literacy also negatively correlates with boredom and positively with flow [8]. Behaviors such as gaming the system, e.g. systematic guessing or hint abuse, can result in poor learning in math [cf. 1]. We ask whether these and other such constructs can predict learning among novice programmers.

Second, which automatically distillable measures can predict achievement? Jadud defines a construct called an Error Quotient (EQ) [13]. EQ is a metric that determines how well or how poorly students cope with syntax errors. It is based on the number of pairs of compilations that end in errors, the number of pairs of compilations that end in the same error type, the number of pairs of compilations with the same error location, and the number of pairs of compilations with the same edit location.

In [20], it was shown that EQ correlates with achievement. Given, though, that EQ is a composite of several measures, and the weights on each measure in [13] are somewhat arbitrary, we disaggregate EQ to determine which of these metrics is most predictive of achievement.

Note that we do not attempt to use the automatic measures of student behavior to predict their affect. Instead, we use both automatic and observational measures when attempting to predict learning outcomes. Each type of measure is feasible for instructors to obtain. For example, teaching assistants could observe lab sessions to identify when students experience affective states found to be associated with poorer learning outcomes. The goal of this study is to determine which factors instructors should look for to inform early response to at-risk students.

3. METHODS

This study was conducted with Computer Science freshmen and Management Information Systems sophomores at the Ateneo de Manila University, during the first semester of school year 2007-2008. The students were taking their first collegiate programming course, CS 21 A, *Introduction to Computing*, generally called CS1 in the computer science education literature. There were five sections of CS 21 A during this semester, with a total of 146 students. Although the teachers for each section varied, the textbook, slides, examples, exercises, midterm exam, final exam, and programming projects were uniform.

The programming language used in the course was Java. During the first half of the semester, the students used the BlueJ Integrated Development Environment (IDE). During the second half of the semester, they shifted to JCreator.

At the beginning of the semester, all students were informed of the study and its purpose. They were then given a form in which they gave or denied consent to participate in the study. Of the 146 students, 143 agreed to participate. Of these 143, 10 were randomly selected from each section (50 total) for behavior and affect observation.

The CS 21 A classes completed hands-on programming exercises in computer laboratories with one-to-one student-to-computer ratios. The lab sessions were part of regularly scheduled class time and were graded, so all students were expected to be present. Each student was assigned a permanent seat and computer for the semester. Over the first nine weeks of the semester, the students were asked to write five small programs. These exercises helped students practice what had been discussed during earlier lectures, e.g. how to write conditional statements, how to create multiple constructors, how to create object associations or aggregations, and so on. All sections were give the same set of laboratory exercises. Each lab period lasted 50 minutes. During these lab periods, students were free to consult their books, notes, slides, classmates and the teacher.

3.1 Logging of online protocols

As the students completed these programming exercises, a BlueJ extension sent data about each student compilation to a SQLite database running in the background. The saved data included (but were not limited to) the computer number, time stamp, error message (if any), file name, and source code in compilation

A single student record for a particular lab was composed of many compilations. This collection of all programs submitted to the compiler is known as an online protocol [15]. Data was only retained for students who consented to participate in the study.

3.2 Observation method

During each lab period, two trained observers noted each student's affective state and behavior, based on previously validated methods for behavior and affect observation in classroom settings [e.g. 1, 18]. The observers were taken from a pool of five students currently taking their master's degrees in either Computer Science or Education. Each of these observers had teaching experience.

During each lab period, two observers studied the same 10 students per section, in a pre-chosen order. They did not tell the students who the 10 observed students were. During the lab period, the observers surreptitiously looked at one specific student at a time. They noted facial expressions, body language, utterances, and interactions with the computer, fellow students or teacher. They also wandered around the classroom, watching the current student from a distance. Since the entire class occupied the lab during the lab period, it was fairly easy to disguise who exactly the observers were watching and at what time. After 20 seconds, the observers shifted their attention to the next student. The observers recorded 15 observations per student per period.

The observers then coded one affective state and one behavior for that student for that time period. The affective states coded were taken from [5, 18]; examples of student behavior when experiencing each of these affective states are given below:

1. **Boredom** – slouching and resting the chin on his/her palm; statements such as "This is boring!"
2. **Confusion –**scratching his/her head, repeatedly looking at the same interface elements; consulting with a classmate or a teacher; flipping through lecture slides or notes; statements such as "Why didn't it work?"
3. **Delight** –clapping hands or laughing with pleasure; statements such as "Yes!" or "I got it!"
4. **Surprise** –jerking back suddenly or gasping; statements such as "Huh?" or "Oh, no!"
5. **Frustration –**banging on the keyboard or pulling at his/her hair; cursing; statements such as "What's going on?!?"
6. **Flow** – complete immersion and focus upon the system [4]; behaviors such as leaning towards the computer or mouthing solutions to him/herself while solving a problem.
7. The **Neutral** state, which was coded when the student did not appear to be displaying any of the affective states above, or the student's affect could not be determined for certain.

The behavioral states coded were taken from [1, 18] and described as follows:

1. **On-task** – working on the programming task
2. **On-task conversation: domain and problem-focused** – helping or asking for help from the instructor or another student about the program specifications or a Java construct
3. **IDE-related on-task conversation -** helping or asking for help from the instructor or another student about the IDE
4. **Off-task conversation** – talking about any other topic
5. **Off-task solitary behavior** – behavior that did not involve the programming task or another person (such as surfing the web, blogging or checking a cell phone)
6. **Inactivity –** the student stares into space or puts his/her head down on the desk.
7. **Gaming the System [cf. 1]** – sustained and/or systematic guessing, such as rapid-fire compiling without consulting the error messages; repeatedly requesting help in order to arrive at a solution

For tractability, observers coded only the first observed affective state and behavior per student per time slice. After 20 seconds, the observers moved on to the next student. After the 10^{th} student, the observers returned to the first. The observers gathered 15 affect and 15 behavior observations per student per lab period.

After observations for all five lab sessions were gathered, inter-rater reliability was computed. It was acceptably high with Cohen's [3] κ=0.65 for affect and κ=0.75 for behavior.

3.3 Data pre-processing

Once the data had been gathered, it was pre-processed for analysis. For each study participant, all compilation records not related to the lab exercise were deleted. Students whose observation or online protocol records were incomplete because of absences or technical problems were also deleted from the dataset. After all the deletions, the sample was reduced to 40 students. Twenty-seven were male and 13 were female.

For each of these students, we constructed a record with the student's section, computer number, midterm exam grade, an affective profile, an observed behavior profile, and a compilation profile. The students' midterm exam grades were used instead of the final exam grades because the labs and observations took place during the first half of the semester.

To arrive at the affective and observed behavior profiles of each student, we first computed for the percentage of observations in which each student exhibited an affective state or behavior for each lab period [cf. 1]. We then averaged the student's percentages per affective state or behavior across the five labs.

The student's compilation profile was drawn from his or her online protocols. We computed for the total number of errors, number of compilations, and average number of seconds between compilations per student per lab. As mentioned earlier, we disaggregated the EQ construct, so that for each student for each lab, we computed number of pairs of compilations in error, number of pairs of compilations with the same error, number of pairs of consecutive compilations with the same edit location, and number of pairs of consecutive compilations with the same edit location. All these measures were then averaged across all five labs.

4. RESULTS AND DISCUSSION

We performed a linear regression using each affective state, behavior, and automatically distillable measure independently, to determine whether any of these were predictors of achievement as represented by students' scores on the midterm exam.

From Table 1, Flow was positively related to achievement. Boredom and confusion are negatively related to achievement. Delight, surprise, frustration and neutrality were not predictors of achievement.

Two backwards elimination stepwise regressions [6] were performed on the significant observed factors, in order to develop the fullest possible model predicting student achievement. The first was performed on the observations. We began with the five factors: confusion, boredom, flow, on-task, and IDE-related on-task conversation. With each iteration, we then eliminated the individual factor that was least significant until all factors in the model were statistically significantly associated with achievement, even taking the other factors into account. Two factors were eliminated: flow and on-task. The remaining three factors—confusion, boredom, and IDE_related on-task conversation—resulted in a model which accounts for about one-third of the variance in midterm scores (R^2=0.347).

Table 1. Linear regression results; significant findings in dark grey, marginally significant findings in light grey

Factors	ß	R	Significance
Affective states			
Confusion	-82.452	0.432	$F(1,38)=8.755$ p=0.005
Boredom	-612.570	0.389	$F(1,38)=6.782$ p=0.013
Delight	-15.535	0.032	$F(1,38)=7.933$ p=0.846
Surprise	55.989	0.000	$F(1,38)=0.006$ p=0.939
Flow	49.892	0.346	$F(1,38)=0.120$ p=0.028
Frustration	-140.337	0.195	$F(1,38)=1.485$ p=0.231
Neutral	14.694	0.084	$F(1,38)=0.251$ p=0.619
Behaviors			
On-task	34.294	0.257	$F(1,38)=2.707$ p=0.108
Giving and receiving answers	-108.421	0.221	$F(1,38)=1.975$ p=0.168
IDE-related on-task conversation	-55.860	0.316	$F(1,38)=4.239$ p=0.046
Off-task conversation	101.338	0.130	$F(1,38)=0.651$ p=0.425
Off-task solitary	88.071	0.245	$F(1,38)=2.442$ p=0.126
Inactivity	-28.154	0.122	$F(1,38)=0.561$ p=0.458
Gaming the system	-481.503	0.130	$F(1,38)=0.655$ p=0.423
Automatically distillable measures			
Ave. number of errors	-0.255	0.277	$F(1,38)=3.148$ p=0.084
Average number of compilations	-0.175	0.245	$F(1,38)=2.437$ p=0.127
Ave. number of seconds between compilations	0.028	0.055	$F(1,38)=0.115$ p=0.736
Pairs of compilations in error	-0.378	0.326	$F(1,38)=4.529$ p=0.040
Pairs of compilations with same error	-0.637	0.303	$F(1,38)=3.859$ p=0.057
Pairs of compilations with the same error location	-0.562	0.298	$F(1,38)=3.700$ p=0.062
Pairs of compilations with the same edit location	-0.666	0.336	$F(1,38)=4.822$ p=0.034

The same process applied on the automatically distillable measures produced a model with two factors, Pairs of Compilations with the Same Edit Location and Pairs of Compilations with the Same Error Location. However, the predictive power of the model was relatively low (R^2=0.120), and The model as a whole was only marginally significant ($F(2,37)=2.524$; p=0.094).

In addition, a model combining all five factors is not significantly better than the model with just confusion, boredom, and IDE-related on-task conversation.

5. CONCLUSION

In this paper, we asked which observable affective states and behaviors and which automatically distillable measures can be used to predict student achievement in CS1 courses. Based on this study, we find that students who are confused, bored, or engaged in IDE-related on-task conversation will most probably do less well in the midterm exams. We also found that automatically distillable measures such as average number of errors, pairs of compilations in error, pairs of compilations with the same error, pairs of compilations with the same edit location and pairs of compilations with the same error location may also be useful in identifying students having academic difficulty. Future work may include a finer-grained analysis of the tractable behaviors associated with successful or unsuccessful students.

6. ACKNOWLEDGMENTS

The authors thank Ramil Bataller, Andrei Coronel, Darlene Daig, Jose Alfredo de Vera, Dr. Emmanuel Lagare, Ramon Francisco Mejia, Dr. John Paul Vergara, and the technical and secretarial staff of the Ateneo de Manila's Department of Information Systems and Computer Science for their assistance with this project. We thank the Ateneo de Manila's CS 21 A students, school year 2007-2008, for their participation. We thank the Department of Science and Technology's Philippine Council for Advanced Science and Technology Research and Development for making this study possible by providing the grants entitled *Modeling Novice Programmer Behaviors Through the Analysis of Logged Online Protocols* and *Observation and Diagnosis of Novice Programmer Skills and Behaviors Using Logged Online Protocols*. We would also like to credit support from the Pittsburgh Science of Learning Center, National Science Foundation award SBE-0354420. Finally, Dr. Rodrigo thanks the US Department of State, the Philippine American Educational Foundation and the Council for International Exchange of Scholars for her 2008-2009 Advanced Research and University Lecturing Fulbright Scholarship and Eduardo Sevilla for his helpful comments.

7. REFERENCES

[1] Baker, R.S., Corbett, A.T., Koedinger, K.R., and Wagner, A.Z. (2004) Off-task behavior in the Cognitive Tutor classroom: When students "Game The System". *ACM CHI 2004: Computer-Human Interaction*, 383-390.

[2] Bennedsen J. and Caspersen M. E. Optimists Have More Fun, But Do They Learn Better? - On the Influence of Emotional and Social Factors on Learning Introductory Computer Science. Computer Science Education, 18, 1, 2008, 1-16.

[3] Cohen, J. 1960. A Coefficient of Agreement for Nominal Scales. *Educational and Psychological Measurement,* 20, 37-46.

[4] Csikszentmihalyi, M. (1990). *Flow: The Psychology of Optimal Experience*. New York: Harper and Row.

[5] D'Mello, S. K., Craig, S. D., Witherspoon, A., McDaniel, B., Graesser, A. 2005. Integrating affect sensors in an intelligent tutoring system. In "Affective Interactions: The Computer in the Affective Loop Workshop" In conjunction with *International conference on Intelligent User Interfaces,* 7-13.

[6] Field, A. 2005. *Discovering Statistics Using SPSS*. London: Sage Publications. Psychology Press: Philadelphia, PA.

[7] Goold, A. and Rimmer, R. 2000. Factors affecting performance in first-year computing. *SIGCSE Bull.* 32, 2 (Jun. 2000), 39-43. DOI= http://doi.acm.org/10.1145/355354.355369

[8] Graesser, A. C. Chipman, P., King, B., McDaniel, B., and D'Mello, S (2007). Emotions and Learning with AutoTutor. *13th International Conference on Artificial Intelligence in Education (AIED 2007)*. R. Luckin et al. (Eds), (pp 569-571). IOS Press.

[9] Halland, K. and Malan, K. 2003. Reflections by teachers learning to program. In *Proceedings of the 2003 Annual Research Conference of the South African institute of Computer Scientists and information Technologists on Enablement Through Technology* (September 17 - 19, 2003). J. Eloff, A. Engelbrecht, P. Kotzé, and M. Eloff, Eds. ACM International Conference Proceeding Series, vol. 47. South African Institute for Computer Scientists and Information Technologists, 165-172.

[10] Khan, I. A., Hierons, R. M., and Brinkman, W. P. 2007. Mood independent programming. In *Proceedings of the 14th European Conference on Cognitive Ergonomics: invent! Explore!* (London, United Kingdom, August 28 - 31, 2007). ECCE '07, vol. 250. ACM, New York, NY, 269-272. DOI= http://doi.acm.org/10.1145/1362550.1362606

[11] Kinnunen, P., McCartney, R., Murphy, L., and Thomas, L. 2007. Through the eyes of instructors: a phenomenographic investigation of student success. In *Proceedings of the Third international Workshop on Computing Education Research* (Atlanta, Georgia, USA, September 15 - 16, 2007). ICER '07. ACM, New York, NY, 61-72. DOI= http://doi.acm.org/10.1145/1288580.1288589

[12] Kumar, S. 2008. The rise and fall of a good programmer. *Ubiquity* 9, 14 (Apr. 2008), 1-5. DOI= http://doi.acm.org/10.1145/1366321.1366323

[13] Jadud, M. C. 2006. An Exploration of Novice Compilation Behavior in BlueJ. Doctoral thesis. University of Kent

[14] Joni, S., Soloway, E., Goldman, R., and Ehrlich, K. 1983. Just so stories: how the program got that bug. *SIGCUE Outlook* 17, 4 (Sep. 1983), 13-26. DOI= http://doi.acm.org/10.1145/1045083.1045086

[15] Lane, H. C. and VanLehn, K. 2005. Intention-based scoring: an approach to measuring success at solving the composition problem. In *Proceedings of the 36th SIGCSE Technical Symposium on Computer Science Education* (St. Louis, Missouri, USA, February 23 - 27, 2005). SIGCSE '05. ACM, New York, NY, 373-377. DOI= http://doi.acm.org/10.1145/1047344.1047471

[16] Ovans, R. 2004. The programmer life-cycle. *SIGSOFT Softw. Eng. Notes* 29, 3 (May. 2004), 25-26. DOI= http://doi.acm.org/10.1145/986710.986720

[17] Perkins, D. N., Hancock, C., Hobbs, R., Martin F., and Simmons, R. 1985. Conditions of Learning in Novice Programmers. Concept Paper. Educational Technology Center, Harvard Graduate School of Education..

[18] Rodrigo, M. M. T., Baker, R. S. J. d., Lagud, M. C. V., Lim, S. A. L., Macapanpan, A. F., Pascua, S. A. M. S., Santillano, J. Q., Sevilla, L. R. S., Sugay, J. O., Tep, S., & Viehland, N. J. B. (2007). Affect and usage choices in simulation problem-solving environments. In R. Luckin, K. R. Koedinger, J. Greer (Eds.), *13th International Conference on Artificial Intelligence in Education*, 145-152.

[19] Spohrer, J. C. and Soloway, E. 1986. Novice mistakes: are the folk wisdoms correct?. *Commun. ACM* 29, 7 (Jul. 1986), 624-632. DOI= http://doi.acm.org/10.1145/6138.6145

[20] Tabanao, E., Rodrigo, M. M. T., and Jadud M. 2008. Identifying at-risk novice programmers through the analysis of online protocols. Philippine Computing Society Congress 2008, (UP Diliman, Quezon City, February 23-24, 2008).

8. AUTHORS' INSTITUTIONAL AFFILIATIONS

[1]Department of Information Systems and Computer Science, Ateneo de Manila University, Loyola Heights, Quezon City, Philippines, +63 (2) 426-6071

[2]Human-Computer Interaction Institute, Carnegie Mellon University, Pittsburgh, PA, +1 (412) 268-9690

[3]Department of Computer Sciences, Allegheny College, Meadsville, PA, +1 (814) 332-2565

[4]Education Department, Ateneo de Manila University, Loyola Heights, Quezon City, Philippines, +63 (2) 426-6001 loc 5230

[5]School of Computer Studies, Mindanao State University-Iligan Institute of Technology, +63 (63) 221-4056