Monitoring Novice Programmer Affect and Behaviors to Identify Learning Bottlenecks

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

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

Jessica O. Sugay[1,3]
jsugay@ateneo.edu

Emily Tabanao[1,4]
emilytabanao@yahoo.com

ABSTRACT
We analyze student affect data in order to locate bottlenecks in an introductory programming course. By tracking students' affective states and behaviors over five laboratory sessions distributed over nine weeks, we find that students exhibit a significantly greater amount of confusion when expected to implement object-oriented constructs such as constructors and object interaction. When asked to undertake an exercise similar in scope with a previous exercise, students expend less time and effort on the exercise. They exhibit less flow and frustration, and spend more time on off-task behaviors.

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

General Terms
Human Factors

Keywords
Novice Programmers, Affect, Achievement

1. MITIGATING CS STUDENT ATTRITION
Since 2000, enrollment in computer science courses has been on the decline [14]. In fall of 2007, the number of new CS majors in the United States was half of what it was in 2000. As a consequence, the number of CS undergraduate degrees granted fell by 43% from school year 2003/04 to 2006/07. A cause of concern for both academe and industry, the decline has prompted research into factors contributing to student retention or attrition as well as the formulation of student retention strategies that address the cognitive, affective, and social dimensions of learning computer science.

1.1 Student-centered learning
Universities have taken more student-centered teaching and learning approaches to computer science. For example, some universities have engaged undergraduate students in collaborative research programs early in their college life. Under the mentorship of a practicing scientist, early engagement has been shown to foster enthusiasm and interest among CS students [6, 13]. These collaborations improved performance, fostered active learning, and encouraged students to spend more time in laboratories, interacting with tutors and faculty [8]. The development of computer science-based games as capstone projects and the use of these games in the introductory computer science courses were inspiring and encouraging for both novice and advanced computer science students [2]. Barker et al [3] found that faculty can mitigate the attrition of women in particular by giving students opportunities to hear each other articulate what they are learning, by starting off theoretical lectures with practical applications of what students are about to learn, and by requiring in-class peer-reviewing of student work.

At the Rochester Institute of Technology, one student-centered approach used to address retention is the deceleration of required courses [15]. The faculty split the IT2 course, which is the second information technology course taken by computer science students, into a two-course sequence. The split improved overall student performance and enhanced student satisfaction. Partitioning the curriculum, as one would partition databases, optimizes performance. A key step in the partitioning process is the identification of bottlenecks in the curriculum, i.e. the courses, concepts and topics that impede students' learning.

1.2 Computer science student affect and its relationship to retention
Computer science education researchers are becoming increasingly interested in CS student affect—their feelings, interests, attitudes, values, and practices—and how this is related to the internalization of cognitive content [11]. Bergin [4] found a strong relationship between a student's perception of their understanding of a learning module and programming performance. The group of Lewis [10] found that students' intent to stay in the computer science course is determined in part by emotional intelligence, trust in one's instincts, and positive acceptance of change. Ethnic identity, on the other hand, defined as a part of the individual's self-concept that is derived from membership in a social group and the value attached to membership in that group, helps students maintain happiness when faced with stress but might actually lead to greater resistance to assimilation and therefore may not help retention [10].

The group of McKinney [11] found that student interest and enjoyment, perceived competence, effort, value-usefulness, and felt pressure or tension were correlated with CS1 course grades. They also found that all these factors shifted negatively from the start of the course to the end of the course, implying that, after the course, the CS1 students felt less competent and less interested. Furthermore, they found that students in CS1 sections employing affective teaching and learning strategies such as active and cooperative learning activities, team work, discussion approaches, and self-reflection had significantly higher scores for effort and

student-peer belonging than those sections that did not use the same techniques.

2. TRACKING AFFECT TO IDENTIFY LEARNING BOTTLENECKS

In this paper, we are interested in using data about student affect to identify curricular bottlenecks. We ask the questions: Do novice programmer affective states and behaviors change over time? If so, at what points in the curriculum do these changes take place? The answers to these questions may have implications on how faculty can better manage subject matter and students' readiness—both cognitive and affective—to process the subject matter.

3. METHODS

This study was conducted with Computer Science freshmen and Management Information Systems sophomores of the Ateneo de Manila University during the first semester of school year 2007-2008. The students were taking their first collegiate programming course, CS21A Introduction to Computing, popularly called CS1 in the literature. There were five sections of CS21A during this semester, with a total of 146 students. Although the teachers for each section varied, the textbook, presentations 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 (17% female, 83% male). Of these 143, 10 were randomly selected from each section for behavior and affect observation.

The CS21A classes completed the lab 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. 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 Lab exercises

Over the first nine weeks during scheduled labs of the semester, the students were asked to write five small programs during their laboratory periods. These exercises helped students practice what had been discussed during earlier lectures:

3.1.1 Lab 1: Objects and output

Students were asked to implement a program that simulates the usage of a mobile phone. The `MobilePhone` class had attributes that tracked

- Credits left in pesos
- Total minutes called
- Rate per call in pesos

Students were expected to define the following methods:

- `void load( double pesos )`
- `void call( int minutes )`
- `double getLoadLeft()`
- `int getTotalMinutesCalled()`

3.1.2 Lab 2: Conditionals

The second lab exercise was an extension of the first exercise. Students were asked to modify the Lab 1 program to include the following methods:

- `void sendTextMessage()`
- `int getNumTextMessages()`
- `void changeRate( double newCallRate )`

Students were instructed to disallow calls and text messages that cost more than the remaining credits. Call rates could not be set to more than 10.00 per minute. There was a 25.00 peso minimum when loading credits. Negative amounts when calling, loading, changing call rates and so on should be forbidden.

3.1.3 Lab 3: Constructors

In this exercise, students had to simulate ticket sales at a movie house. They had to create two classes, `Ticket` and `Cinema`. The `Ticket` class had attributes that tracked

- Name of movie
- Ticket price
- Seat number

The `Ticket` class also had the following methods:

- `public Ticket( String movieName, double ticketPrice, String seatNumber )`
- `public Ticket( String movieName, String seatNumber )`
  `// default ticket price is P120.00`
- `public String getName()`
- `public double getPrice()`
- `public String getSeat()`

The `Cinema` class dispensed tickets based on the movie shown. It can dispense a ticket at a regular or premium price. It should keep track of the

- number of tickets sold
- total sales of the movie

Students were asked to implement the following methods in the `Cinema` class:

- `public Cinema( Sting movieName, double premiumPrice );`
- `public Cinema( String movieName )`
  `// default premium price is P130.00`
- `public Ticket printRegular( String seatNum )`
- `public Ticket printPremium( String seatNum )`

3.1.4 Lab 4: Object interaction

Students were asked to write a program that tracks gas trucks and stations. The `GasTruck` class had three attributes:

- Type of fuel

- Current amount of fuel in the truck
- Current `GasStation` where the truck is located

The `GasTruck`'s methods included

- `public GasTruck(String initGasType)`
- `public void goTo(GasStation station)`
- `public void loadGas(double liters)`
- `public void unloadGas(double liters)`
- `public String getGasType()`
- `public double getGasLevel()`
- `public GasStation getCurrStation()`

The `GasStation` class was assumed to carry two types of gas: diesel or unleaded. The `GasStation` class had three attributes

- the station's location
- the current levels of diesel fuel
- the current level so unleaded fuel

The `GasStation`'s methods included

- `public GasStation( String initLocation, double initDiesel, double initUnleaded)`
- `public void addGas( String gasType, double liters )`
- `public void dispenseGas( String gasType, double liters )`
- `public String getLocation()`
- `public double getGasLevel( String gasType )`

3.1.5 Lab 5: Object interaction and String equality

The last programming exercise for the first half of the semester was a simulation of the online public access catalog of a library. The program had two classes, `Book` and `Library`.

The `Book` class kept track of four attributes:

- Book title
- Book author
- Current borrower
- Number of times the book has been borrowed

The `Book`'s methods were as follows:

- `public Book( String initTitle, String initAuthor )`
- `public void setBorrower( String borrowerName )`
- `public void unmark()`
  `//when the book is returned, the borrower's name is set to '--'`
- `public String getTitle()`
- `public String getAuthor()`
- `public String getBorrower()`
- `public int getTimesBorrowed()`

The `Library` class only held two Book objects. The Library's methods were:

- `public Library()`
- `public void borrowBook( String studentName, String bookTitle )`
- `public void returnBook( String bookTitle )`

3.2 Observation method

During each lab period, two trained observers noted each student's affective state and behavior. 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.

The observers were synchronized by a timed PowerPoint presentation with slides numbered 1 to 150, each slide lasting 20 seconds. During each time slide, the observers surreptitiously looked at a student's facial expressions, body language, utterances, and interactions with the computer, fellow students or teacher. The observers studied the same 10 students per section, per lab period. The identities of the 10 students under observation were not revealed at any time Observers also made it a point to wander around the classroom and watch the subjects 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.

The observers then coded one affective state and one behavior for that student for that time period. The affective states coded were taken from [12] and are described as follows:

1. **Boredom** – behaviors such as slouching, and resting the chin on his/her palm; statements such as "This is boring!"
2. **Confusion –** behaviors such as 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** – behaviors such as clapping hands or laughing with pleasure; statements such as "Yes!" or "I got it!"
4. **Surprise** – behaviors such as jerking back suddenly or gasping; statements such as "Huh?" or "Oh, no!"
5. **Frustration –** behaviors such as 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 [5]; 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 [2, 12] and described as follows:

1. **On-task** – working on the programming task
2. **Giving or receiving answers** – helping or asking for help from the teacher or another student about the program specifications or a Java construct

3. **Other on-task conversation** – helping or asking for help from the teacher 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** – sustained and/or systematic guessing, such as rapid-fire compiling without consulting the error messages; repeatedly requesting for help in order to arrive at a solution [2].

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 10th student, the observers returned to the first. The observers gathered 15 affect and 15 behavior observations per student per lab session.

After observations for all five lab sessions were gathered, interrater reliability was computed. Cohen's kappa was acceptably high at κ=0.65 for affect and κ=0.75 for behavior.

4. RESULTS AND DISCUSSION

We used a repeated measures ANOVA to determine which of the affective states and behaviors varied over time. "Repeated measures" is a design in which the same data items are collected from the same subjects repeatedly over time [7]. In this particular case, the affect and behaviors of the same 10 students were taken during five lab sessions over nine weeks, making the repeated measures test appropriate.

One of the assumptions that we have to test when performing a repeated measures ANOVA is that of sphericity. Field [7] defines sphericity as the equality of variances of the differences between treatment levels. In our case, a treatment level is a single lab period. If you take all pairs of lab sessions and compute for the differences between each pair of affect or behavior scores, these differences should have equal variances. If this is not the case, the sphericity assumption has been violated and an error correction (**ε**) has to be imposed on the degrees of freedom.

Upon running the repeated measures ANOVA, it is first necessary to examine Mauchly's W to determine whether sphericity has been violated. If the test is significant, then we check the W value. If the W value is less than 0.75, we use the Greenhouse-Geisser correction. If the W value is greater than 0.75, we use of the Huynh-Feldt correction.

The corrected degrees of freedom are then used to compute for the F-value, and from there the p-value of the differences between groups.

Tables 1 and 2 show the results of the analysis. Note, that, in all cases, the assumption of sphericity was violated with all Mauchly's test p-values < 0.05. In all cases, either the Greenhouse-Geisser or Huynh-Feldt correction was imposed on the degrees of freedom. Despite the correction, we still arrived at results that were either significant or marginally significant.

Of the seven affective states studied, three varied significantly different over time: confusion, flow and neutrality. Variations in frustration were marginally significant.

Of the seven behaviors studied, four varied significantly over time: on-task behavior, other on-task conversation, off-task conversation, and solitary off-task behavior. Inactivity varied marginally significantly.

Given these results, we then ask the follow up question: how exactly do these affective states and behaviors vary over time? To arrive at the answers, we graphed the average incidence of the nine significant affective states and behaviors over the five labs (see Figures 1 and 2).

In Figure 1, notice that flow and frustration decrease over time. Confusion peaks at lab 3 then tapers off. Neutrality spikes in lab 5. In Figure 2, note that on-task behavior spikes in lab 4 then decreases in lab 5. Off-task conversation increases by lab 3 then decreases in labs 4 and 5. Other on-task conversation, solitary off-task behavior and inactivity increase in lab 5.

To determine the significance of these dips and spikes, we performed a pairwise comparison of the average incidence of each affective state or behavior per lab session. The results of the pairwise comparisons are summarized in tables 3 to 11.

Table 1. Repeated measures ANOVA results on affective states; Significant differences in dark gray; Marginally significant differences in light gray.

Affective State	Mauchly's Test							
	W	χ^2	df	p	ε	Error Correction	F-test	p
Boredom	0.146	91.213	9	0	0.721	Greenhouse-Geisser	F(2.884, 141.339) = .275	0.836
Confusion	0.699	16.958	9	0.049	0.897	Huynh-Feldt	F(3.588, 175.816) = 0.222	0
Delight	0.365	47.768	9	0	0.65	Greenhouse-Geisser	F(2.601, 127.458) = 1.547	0.21
Flow	0.656	19.963	9	0.018	0.931	Huynh-Feldt	F((3.726, 182.550) = 8.546	0
Frustration	0.302	56.719	9	0	0.654	Greenhouse-Geisser	F(2.614, 128.106) = 2.462	0.07
Neutrality	0.45	37.867	9	0	0.849	Huynh-Feldt	F(3.398, 166.478) = 7.898	0
Surprise	0.244	66.904	9	0	0.656	Greenhouse-Geisser	F(2.265, 128.619) = 0.570	0.613

Table 2. Repeated measures ANOVA results on behaviors; Significant differences in dark gray; Marginally significant differences in light gray.

	Mauchly's Test							
Behavior	**W**	**χ^2**	**df**	**p**	**ε**	**Error Correction**	**F-test**	**p**
On-task	0.674	18.702	9	0.028	0.919	Huynh-Feldt	F(3.675, 180.064) = 12.129	0
Giving and receiving answers	0.53	30.121	9	0	0.826	Huynh-Feldt	F(3.305, 161.958) = 1.566	0.199
Other on-task conversation	0.13	96.576	9	0	0.487	Greenhouse-Geisser	F(1.948, 95.433) = 10.482	0
Off-task conversation	0.604	23.936	9	0.004	0.882	Huynh-Feldt	F(3.530, 172.968) = 7.545	0
Solitary off-task	0.079	120.167	9	0	0.414	Greenhouse-Geisser	F(1.654, 81.055) = 8.310	0.001
Inactive	0.314	54.932	9	0	0.828	Huynh-Feldt	F(3.311, 162.229) = 2.406	0.068
Gaming the system	0.148	90.743	9	0	0.598	Greenhouse-Geisser	F(2.392, 117.210) = .729	0.507

Figure 1. Average incidence of affective states over time.

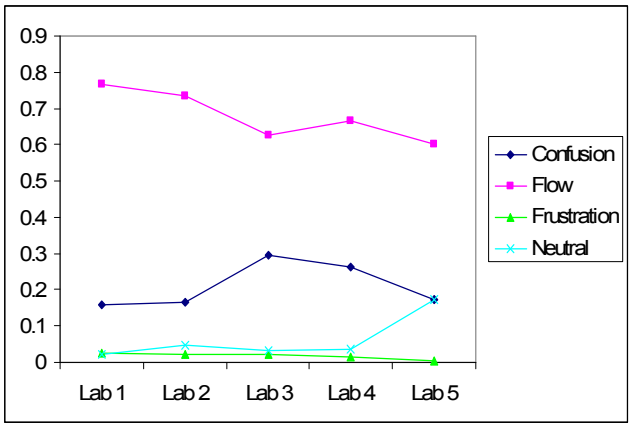
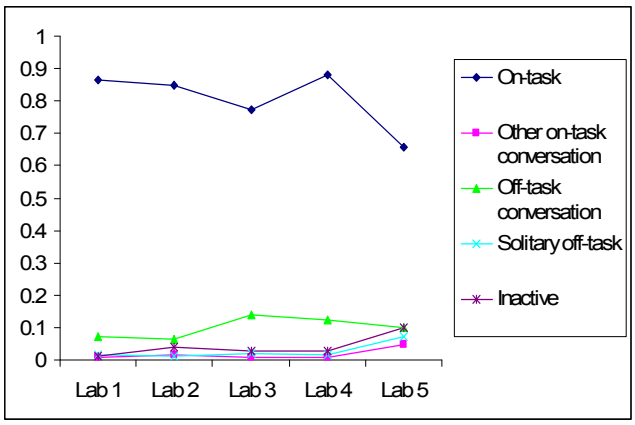


Figure 2. Average incidence of behaviors over time.



Note that the sign of the numerical result indicates directionality. Table 3 shows that students were less confused in Labs 1 (objects and output) and 2 (conditionals) than they were in Labs 3 (constructors) and 4 (object interaction). The inverse interpretation is also accurate: that students exhibited more confusion in Labs 3 and 4 than in Labs 1 and 2. Lab 3 was significantly more confusing than Lab 5 (object interaction with string equivalence). Table 4 shows that students experienced more flow in Labs 1 and 2 than they did in Labs 3, 4, and 5. They also experienced more flow in Lab 4 than they did in Lab 5. Lab 5 was least frustrating among all five lab exercises (Table 5). It was also in Lab 5 that we saw the most neutrality (Table 6).

In terms of behaviors, students worked most during Lab 1 (Table 7). It was in Lab 5 we saw the least amount of on-task behavior.

Lab 5 had the most on-task conversation (Table 8). Table 9 shows that Labs 1 and 2 had less off-task conversation than in Labs 3, 4, and 4. There was slightly more off-task conversation in Lab 3 than Lab 5. Lab 5 had less off-task solitary behavior than Lab 1, but had more off-task solitary behavior than Labs 2, 3, and 4 (Table 10). Finally, levels of inactivity were highest in Lab 5, especially when compared with Labs 1, 3, and 4 (Table 11).

What these findings imply is that introductory topics such as simple output and conditionals are relatively easy for students to understand. During Labs 1 and 2, students tended to be in flow, working or speaking to classmates or the teacher about topics other than the subject matter. The object-oriented constructs posed a greater challenge to students. Confusion significantly spiked in Lab 3 and continued to be high in Lab 4.

Lab 5 posed a special case because it was essentially a variation of Lab 4. The course instructors decided to design Lab 5 in this way because they perceived that the students were having difficulty with object interaction. They believed that an additional exercise would help students better understand the topic. This was a way of partitioning the topic.

Because they were already familiar with the topic, students exhibited less frustration overall. Possibly because the task was easier to complete, they also experienced less flow and exhibited less on-task behavior. Instead, they exhibited more off-task behavior, whether solitary or in conversation, and more neutrality. The implication is that they completed the task in less time and were therefore able to engage in non-programming tasks.

Table 3. Pairwise comparison of confusion; Significant differences in dark gray

	Lab 2	Lab 3	Lab 4	Lab 5
Lab 1	-0.008	-0.136	-0.106	-0.017
Lab 2		-0.128	-0.098	-0.009
Lab 3			0.030	0.120
Lab 4				0.089

Table 4. Pairwise comparison of flow; Significant differences in dark gray; Marginally significant differences in light gray

	Lab 2	Lab 3	Lab 4	Lab 5
Lab 1	0.035	0.140	0.101	0.167
Lab 2		0.106	0.066	0.132
Lab 3			-0.040	0.026
Lab 4				0.066

Table 5. Pairwise comparison of frustration; Significant differences in dark gray; Marginally significant differences in light gray

	Lab 2	Lab 3	Lab 4	Lab 5
Lab 1	0.004	0.007	0.013	0.021
Lab 2		0.003	0.009	0.017
Lab 3			0.007	0.015
Lab 4				0.008

Table 6. Pairwise comparison of neutral; Significant differences in dark gray

	Lab 2	Lab 3	Lab 4	Lab 5
Lab 1	-0.026	-0.014	-0.017	-0.154
Lab 2		0.012	0.009	-0.128
Lab 3			-0.003	-0.140
Lab 4				-0.137

Table 7. Pairwise comparison of on-task behavior; Significant differences in dark gray

	Lab 2	Lab 3	Lab 4	Lab 5
Lab 1	0.017	0.092	0.053	0.205
Lab 2		0.075	0.035	0.188
Lab 3			-0.039	0.113
Lab 4				0.153

Table 8. Pairwise comparison of other on-task conversation; Significant differences in dark gray

	Lab 2	Lab 3	Lab 4	Lab 5
Lab 1	-0.007	0.001	0.002	-0.041
Lab 2		0.008	0.009	-0.035
Lab 3			0.001	-0.043
Lab 4				-0.043

Table 9. Pairwise comparison of off-task conversation; Significant differences in dark gray; Marginally significant differences in light gray

	Lab 2	Lab 3	Lab 4	Lab 5
Lab 1	0.008	-0.067	-0.053	-0.026
Lab 2		-0.075	-0.061	-0.034
Lab 3			0.014	0.041
Lab 4				0.027

Table 10. Pairwise comparison of solitary off-task behavior; Significant differences in dark gray

	Lab 2	Lab 3	Lab 4	Lab 5
Lab 1	0.006	-0.001	-2.000e-11	0.054
Lab 2		-0.007	-0.006	-0.060
Lab 3			0.001	-0.053
Lab 4				-0.054

Table 11. Pairwise comparison of inactive; Significant differences in dark gray; Marginally significant differences in light gray

	Lab 2	Lab 3	Lab 4	Lab 5
Lab 1	-0.029	-0.015	-0.015	-0.087
Lab 2		0.015	0.014	-0.058
Lab 3			0.000	-0.073
Lab 4				-0.072

5. CONCLUSION AND FUTURE WORK

Based on the changes in students' affective states and behaviors, we observe that students seem to adapt easily enough to introductory topics such as output and conditions. However, constructs that are specific to object-oriented programming—constructors and object interaction—are confusing and may indicate an opportunity for partitioning.

In this case, the topic of object interaction was partitioned. An additional lab exercise, Lab 5, was introduced. Because students were already familiar with the topic, they did not need that much time or effort to complete the task, so they exhibited less flow, less frustration, less on-task behavior and more off-task behavior.

It would be interesting to examine the data in light of student achievement. For this current analysis we did not attempt to segregate the affect and behaviors of high performing students from low performing students. Determining whether student affect and behaviors are related to achievement may be a subject for future investigation.

6. ACKNOWLEDGMENTS

We thank Anna Christine Amarra, Ramil Bataller, Andrei Coronel, Darlene Daig, Jose Alfredo de Vera, Thomas Dy, Maria Beatriz Espejo-Lahoz, Dr. Matthew C. Jadud, Dr. Emmanuel Lagare, Sheryl Ann Lim, Ramon Francisco Mejia, Shiela Pascua, 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 CS21A 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**. Finally, Dr. Rodrigo thanks the Philippine American Educational Foundation and the Council for International Exchange of Scholars for her 2008-2009 Advanced Research and University Lecturing Fulbright Scholarship.

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] Barnes, T., Richter, H., Powell, E., Chaffin, A., and Godwin, A. 2007. Game2Learn: building CS1 learning games for retention. In *Proceedings of the 12th Annual SIGCSE Conference on innovation and Technology in Computer Science Education* (Dundee, Scotland, June 25 - 27, 2007). ITiCSE '07. ACM, New York, NY, 121-125. DOI= http://doi.acm.org/10.1145/1268784.1268821

[3] Barker, L. J., Garvin-Doxas, K., and Roberts, E. 2005. What can computer science learn from a fine arts approach to teaching? 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, 421-425. DOI= http://doi.acm.org/10.1145/1047344.1047482

[4] Bergin, S. and Reilly, R. 2005. Programming: factors that influence success. 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, 411-415. DOI= http://doi.acm.org/10.1145/1047344.1047480

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

[6] Dahlberg, T., Barnes, T., Rorrer, A., Powell, E., and Cairco, L. 2008. Improving retention and graduate recruitment through immersive research experiences for undergraduates. In *Proceedings of the 39th SIGCSE Technical Symposium on Computer Science Education* (Portland, OR, USA, March 12 - 15, 2008). SIGCSE '08. ACM, New York, NY, 466-470. DOI= http://doi.acm.org/10.1145/1352135.1352293

[7] Field, A. 2005. *Discovering Statistics Using SPSS*. London: Sage Publications.

[8] Guo, J. 2008. Using group-based projects to improve retention of students in computer science major. *J. Comput. Small Coll.* 23, 6 (Jun. 2008), 187-193.

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

[10] Lewis, T. L., Smith, W. J., Bélanger, F., and Harrington, K. V. 2008. Determining students' intent to stay in it programs: an empirical model. In *Proceedings of the 2008 ACM SIGMIS CPR Conference on Computer Personnel Doctoral Consortium and Research* (Charlottesville, VA, USA, April 03 - 05, 2008). SIGMIS-CPR '08. ACM, New York, NY, 5-11. DOI= http://doi.acm.org/10.1145/1355238.1355241

[11] McKinney, D. and Denton, L. F. 2004. Houston, we have a problem: there's a leak in the CS1 affective oxygen tank. In *Proceedings of the 35th SIGCSE Technical Symposium on Computer Science Education* (Norfolk, Virginia, USA, March 03 - 07, 2004). SIGCSE '04. ACM, New York, NY, 236-239. DOI= http://doi.acm.org/10.1145/971300.971386

[12] 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.

[13] Stephenson, P., Peckham, J., Hervé, J., Hutt, R., and Encarnação, L. M. 2006. Increasing student retention in computer science through research programs for undergraduates. In *ACM SIGGRAPH 2006 Educators Program* (Boston, Massachusetts, July 30 - August 03, 2006). SIGGRAPH '06. ACM, New York, NY, 10. DOI= http://doi.acm.org/10.1145/1179295.1179306

[14] Vegso, J. 2008. Enrollments and degree production at US CS departments drop further in 2006/2007. *CRA Bulletin*. Accessed 3 December 2008 from the CRA Bulletin web site: http://www.cra.org/wp/index.php?p=139.

[15] Whittington, K. J. and Bills, D. P. 2004. Alternative pacing in an introductory java sequence. In *Proceedings of the 5th Conference on information Technology Education* (Salt Lake City, UT, USA, October 28 - 30, 2004). CITC5 '04. ACM, New York, NY, 118-121. DOI= http://doi.acm.org/10.1145/1029533.1029563

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

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