An Analysis of Novice Programmer Doodles and Student Achievement in an Introductory Programming Class

Marlene De Leon
Department of Information Systems and Computer Science
Ateneo de Manila University
Loyola Heights, Quezon, City, Philippines
+63 (2) 426 6071
mmana@ateneo.edu

Maria Beatriz Espejo-Lahoz
Department of Information Systems and Computer Science
Ateneo de Manila University
Loyola Heights, Quezon, City, Philippines
+63 (2) 426 6071
blahoz@ateneo.edu

Ma. Mercedes T. Rodrigo
Department of Information Systems and Computer Science
Ateneo de Manila University
Loyola Heights, Quezon, City, Philippines
+63 (2) 426 6071
mrodrigo@ateneo.edu

ABSTRACT
This study examines whether students' choice to annotate laboratory exercise sheets has any relationship with their academic achievement. Over the course of nine weeks, instructors of an introductory programming course gave students five programming exercises and allowed students to use the hard copies of the specifications as scratch paper. These papers were then collected. The authors of this paper classified the marks made by the students. They found that the majority of students, 62%, tended not to annotate their lab exercise sheets. However, those who opt to do so tend to be among the highest achievers of their class. The results support earlier studies that found that expert programmers find external memory aids helpful in managing information load.

Categories and Subject Descriptors
K.3.2 [**Computer and Information Science Education**]: Computer science education, literacy.

General Terms
Experimentation

Keywords
Java, doodles, novice programmer.

1. INTRODUCTION
Students of computer science and related disciplines are expected to be proficient computer programmers. Unfortunately, programming is difficult to learn. A seminal study by McCracken et al. [8] found that approximately 30% of computer science students in the United Kingdom and the United States did not understand programming basics even after completing their first programming course. Subsequent studies [cf. 3, 5] corroborated these results, noting that novice programmers were often unable to read, analyze, and trace through short fragments of code.

These findings are worrisome because these same students who cannot write correct programs today are expected to produce correct software tomorrow. Researchers are therefore motivated investigate what difficulties novice programmers have and how these can be addressed. Such analyses are valuable because they can provide educators with bases for adjusting instruction or instructional tools [2]. They may also enable educators to identify at-risk students early so that they can provide these students with timely and effective interventions [Garner].

1.1 The Relationship between Doodling and Achievement
A number of studies have been conducted on student doodles and their relationship with achievement. A "doodle" is defined as any kind of marking made by a student [7]. These may include numbers written over variables, numbers written under array elements, assignment statements, diagrams, and so on. Doodles are graphical representations of attempts to trace through code. When faced with a piece of code to read or understand, experienced programmers frequently doodle to determine the function of the code [5].

Some research findings bear out the hypothesis that doodling is helpful to students taking examinations in which they must read and trace through code. A study by Lister et al. [5] grouped student examination doodles into 12 categories:

- Blanks page – No annotations on the question
- Synchronized Trace – Shows values of multiple variables changing, generally in a table
- Trace – Shows values of a variable as it changes; more than one value is shown or a variable's value is overwritten with a new value
- Odd Trace – Something that appears to be a trace but neither a Synchronized Trace or a Trace, such as linking representations with arrows
- Alternate Answer – Students changed their answers to a question

- Ruled out – One or more alternative answers crossed out so it the answer appeared to be selected by elimination
- Computation – An arithmetic or boolean computation (not rewrite of comparison)
- Keeping Tally – Some value counted multiple times, variable not identified
- Number – Shows a single variable value, most often in comparison
- Position – Picture of correspondence between array indices and values
- Underlined – Part of the question is underlined for emphasis
- Extraneous Marks – Markings that appear meaningless or ambiguous and could not be characterized

The group found that if a student carefully traced through the code and documented the changes in the variables, the student is likely to arrive at the correct answer. Not doodling only leads to the correct answer one-half of the time. Findings from similar studies by McCartney et al's [6, 7] concur. Student performance improves when they annotate test questions [5]. Indeed, any form of annotation leads to better results that no annotation at all [6]!

These findings, though, must be qualified. Success seems to reside not in the use of the strategy per se, but the skill with which it was employed [3]. Because of an incomplete walkthrough, a careless mistake, or a semantic misunderstanding, the student may still end up with a wrong answer.

Also note that doodling requires discipline. When instructors answer students' questions by drawing a diagram, students express impatience, followed by amazement that the approach works (Thomas et al 2005 in [5]). Still, attempts to encourage students to draw diagrams were ineffective, even after the benefits of diagramming were explained and demonstrated.

There is some evidence that suggests that doodling is also beneficial when programming. Expert programmers tend to spend more time annotating their programs than novices (Davies, 1993 in [6]). Expert programmers make extensive use of external records to manage their information; they do not rely on working memory alone (Davies, 1996 in [5]). External annotation reduces the load on working memory by offloading the task of tracking details (Hegarty and Steinhoff 1997 in [7]).

1.2 Research Questions

It is this second situation—the use of doodles during programming—that we examine in this study. We are interested in determining whether our students doodle, what types of doodles they make and whether students who doodle differ in terms of academic achievement.

The findings from this study may direct students to problem-solving processes that may be boost achievement. They may also help educators spot students who are having difficulty and advise them on methods that can help them develop their programming comprehension.

2. METHODOLOGY

2.1 Population

The data for this study was gathered from the students of Ateneo de Manila University (ADMU). The subjects for the study were the ADMU Computer Science freshmen and Management Information Systems sophomores taking CS 21 A Introduction to Programming. This was the first programming course the students were taking for their degree programs. The language used for the course was Java. The majority of the students were taking studying programming for the first time.

2.2 Data Gathering

At the beginning of the semester, the students were informed of the objectives of this study and were invited to become participants. They indicated their willingness to participate by completing consent forms which they then returned to the researchers. The students who did not consent to be part of the study continued to be part of the class, receiving exactly the same treatment as those who consented. Any data from them, though, were excluded from the analysis. One hundred twenty (120) students participated in all.

Over the first nine weeks of the semester, the students were given five (5) graded lab exercises that put into practice recently-taught concepts. The labs are described as follows:

- Lab 1: Classes - Implementing a pre-paid mobile phone class that tracks number of minutes consumed and remaining load.
- Lab 2: Conditionals - Modifying the mobile phone class to include conditional statements, e.g. to forbid calls when the phone has insufficient load.
- Lab 3: Constructors - Implementing a Ticket class that sells tickets to a movie; the Ticket class has more than one constructor.
- Lab 4: Object interaction – Implementing GasTruck and GasStation classes where a GasTruck loads fuel from a GasStation.
- Lab 5: Object interaction with String equality – Implementation of Book and Library classes where Books may be borrowed from a Library.

During each lab session, the students were given hard copies of the exercise specifications. They were asked to write their names on the sheets and to use the sheets as scratch paper. They were free to write anything they wanted on the lab sheet. At the end of the period, the students returned the sheets to the instructors, who then turned over the sheets to the paper's authors.

2.3 Doodle Classification

When the data gathering was complete, the authors met as a group and reviewed the doodles cursorily to see whether the categories listed in [5] applied. They found that some doodle categories, e.g. Alternate Answer and Ruled Out applied to tests but not to laboratory exercises. Furthermore, they found that other there were other types of doodles, e.g. Clarification and Practice Code, that did not appear in [5] but appeared in the lab exercises.

The authors therefore adapted the list from [5], deleting categories that were not useful and adding new ones that they considered to be relevant to the context. The revised list of categories is defined and illustrated in Table 1.

Table 1. Doodle categories for lab sheets. Examples in `Courier` correspond to the text of the lab exercise instructions; Examples in Times New Roman correspond to student annotations.

Doodle Code	Doodle Name	Definition	Example
B	Blank Page	No doodle or any annotation in the hardcopy laboratory exercise specification.	
C	Computation	An arithmetic computation.	10 x 6.50 = 65
E	Extraneous Marks	Markings that appear meaningless or irrelevant to the laboratory exercise.	Drawings of cartoon characters
K	Keeping Tally	Some value counted multiple times, variable not identified	||||
N	Number	Shows a single variable value.	0 load 1 minute
S	Synchronized Trace	Shows values of multiple variables changing, generally in a table.	Call Load Total 0 300 0 10 235 10 5 200 15
T	Trace	Shows values of a variable as it changes; more than one value is shown or a variable's value is overwritten with a new value.	credits->load
O	Odd Trace	Something that appears to be a trace but neither a Synchronized Trace nor a Trace, such as linking representations with arrows.	0 1 2 / / 1 2
P	Position	Picture of correspondence between array indices and values	0 1 2 3 X = { 2 4 6 8 }
U	Underlined	Part of the question is underlined or encircled for emphasis.	`Premium price is P180`
PC	Practice Code	Shows a line or a snippet of a Java Code.	System.out.println("Credits left: ");
CL	Clarification	Indication that specifications were clarified or corrected during the lab exercise	`'Transformers' - seat A1` → one string
CM	Check Mark	A mark that indicates that a task is done	`public Cinema()` √

Working independently, two of this paper's co-authors classified each of the doodles made by the students on each lab sheet following the categories in Table 1. Inter-rater reliability of the classifications was acceptably high with Cohen's [1] κ=0.81.

In cases such as the records where the co-authors matched doodle classifications the duplicate record was removed.

3. FINDINGS AND DISCUSSION

3.1 What Types of Doodles, if any, do Students Make?

The authors found that the majority of students, 62%, left their exercise sheets blank. Only 37% tended to write on the sheets, in many cases combining different types of doodles while working on an exercise.

The authors tabulated each instance of a doodle category, including Blank Pages. A total of 1018 instances were recorded. Of these instances, the largest portion, 55%, were Blank Page. Clarifications accounted for 16%, followed by Check Marks with 9%, Underline with 6%, Extraneous Marks with 4%, both Computation and Practice Code with 3% each, Number with 2%,

Synchronized Trace, Odd Trace, and Trace with 1% each (Figure 1).

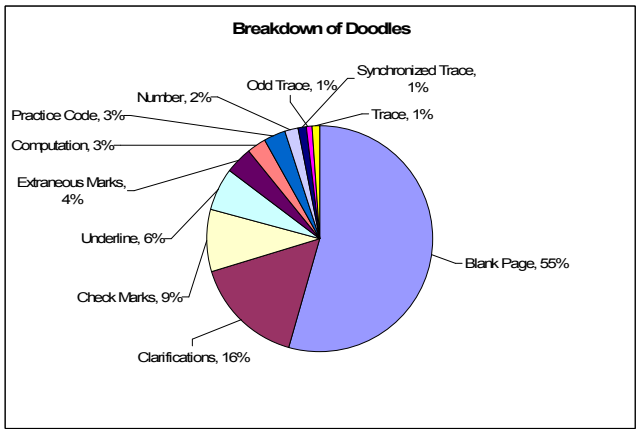

Figure 1. Overall Breakdown of Doodles

To facilitate the analysis of these multiple doodles, the authors grouped them into common categories patterned loosely on the groupings of McCartney, et al, 2004 [6]. The blank grouping comprised of all the recorded Blank Page exercise sheets. The Clarifications consisted of all corrections or additional instructions from the teacher. Meta-cognitive Processes refer to doodles that imply that students are tracking their own learning or progress. These include Check Marks and Underline. The Trace category was comprised of Computation, Practice Code, Number, Synchronized Trace, Odd Trace, and Trace. The final category consisted to Extraneous Marks.

Figure 2 illustrates the distribution of the doodles by these larger categories. The Blank grouping accounted for 55% of the pie, followed by Clarifications at 16%, Meta-cognitive processes at 15%, Tracing with 11% and Extraneous Marks at 4%.

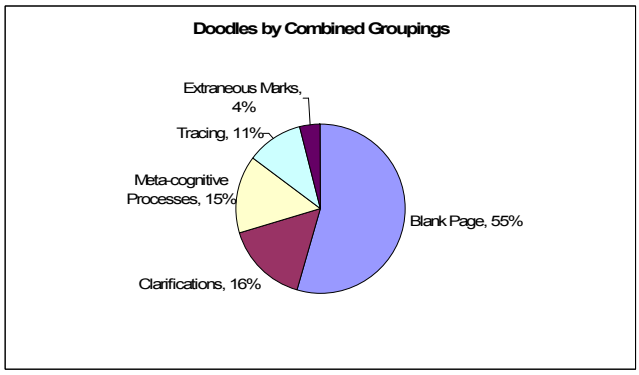

Figure 3. Doodles by Grouping

3.2 Do Students Who Doodle Differ From Those Who do not in Terms of Academic Achievement?

The authors examined the midterm scores of all the students in the population to determine whether the choice to doodle was related in any way to academic achievement. The authors found that all students who doodled, 37% of the population, had midterms scores between 91 and 100 out of a possible 100. Fifteen percent of the population also had midterm grades between 91 and 100 but opted not to doodle at all. Students with midterm scores of less 90 or less did not doodle on their lab sheets. The implication of this observation is that, while not doodling does not automatically compromise achievement, the act of doodling does seem to be helpful to most high achieving students. This observation is consistent with the findings mentioned in [5, 6, 7] that show that expert programmers tend to find annotation helpful. For our population, students with very high midterm scores tended to take note of clarifications from the instructor and perform some kind of tracing, and tend to keep track of their learning and progress. Other high achieving students who do not doodle might have had a very good understanding of the programming problem and did not have need for the additional mental aids.

4. CONCLUSIONS

In this study, we examine whether students doodle, what types of doodles they make and whether the choice to doodle has a relationship with academic achievement. We found that the majority of our students, 62%, do not doodle. The ones who do opt to doodle, 37% of the population, had midterm scores between 91 and 100 out of a possible 100. Fifteen percent of students had similarly high midterm scores but did not opt to doodle. The remainder of the population who opted not to doodle had midterm scores of less than 90 points.

Not doodling at all does not necessariliy mean a student is not doing well. Students who do not doodle but who get high grades nonetheless might already have a good understanding of the excercises assigned to them, which meant that they have no need for the extra mental aids.

5. ACKNOWLEDGMENTS

The authors thank Christine Amarra, Andrei Coronel, Darlene Daig, Jose Alfredo de Vera, Dr. Matthew C. Jadud, Dr. Emmanuel Lagare, Sheryl Lim, Ramon Francisco Mejia, Jessica Sugay, Emily Tabanao, Dr. John Paul Vergara, Dr. Kardi Teknomo 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 2006-2007, for their participation. Finally, 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 grant entitled "Modeling Novice Programmer Behaviors Through the Analysis of Logged Online Protocols."

6. REFERENCES

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

[2] Ebrahimi, A. and Schweikert, C. 2006. Empirical study of novice programming with plans and objects. *SIGCSE Bull.* 38, 4 (Dec. 2006), 52-54. DOI= http://doi.acm.org/10.1145/1189136.1189169.

[3] Fitzgerald, S., Simon, B., and Thomas, L. 2005. Strategies that students use to trace code: an analysis based in grounded theory. In *Proceedings of the 2005 international Workshop*

on Computing Education Research (Seattle, WA, USA, October 01 - 02, 2005). ICER '05. ACM, New York, NY, 69-80. DOI= http://doi.acm.org/10.1145/1089786.1089793

[4] Garner, S., Haden, P., and Robins, A. 2005. My program is correct but it doesn't run: a preliminary investigation of novice programmers' problems. In *Proceedings of the 7th Australasian Conference on Computing Education - Volume 42* (Newcastle, New South Wales, Australia). A. Young and D. Tolhurst, Eds. ACM International Conference Proceeding Series, vol. 106. Australian Computer Society, Darlinghurst, Australia, 173-180.

[5] Lister, R., Adams, E. S., Fitzgerald, S., Fone, W., Hamer, J., Lindholm, M., McCartney, R., Moström, J. E., Sanders, K., Seppälä, O., Simon, B., and Thomas, L. 2004. A multi-national study of reading and tracing skills in novice programmers. *SIGCSE Bull.* 36, 4 (Dec. 2004), 119-150. DOI= http://doi.acm.org/10.1145/1041624.1041673

[6] McCartney, R., Mostrom, J. E., Sanders, K., and Seppala, O. 2004. Questions, annotations, and institutions: Observations from a study of novice programmers. *4th Finnish/Baltic Sea Conference on Computer Science Education* (Koli, Finland, October 1-3, 2004), 11-19.

[7] McCartney, R., Mostrom, J. E., Sanders, K., and Seppala, O. 2005. Take note: The effectiveness of novice programmers' annotations on examinations. *Informatics in Education, 3*(1), 69-86.

[8] McCracken, M., Almstrum, V., Diaz, D., Guzdial, M., Hagan, D., Kolikant, Y. B., Laxer, C., Thomas, L., Utting, I., and Wilusz, T. 2001. A multi-national, multi-institutional study of assessment of programming skills of first-year CS students. In *Working Group Reports From ITiCSE on innovation and Technology in Computer Science Education* (Canterbury, UK). ITiCSE-WGR '01. ACM, New York, NY, 125-180. DOI= http://doi.acm.org/10.1145/572133.572137 Whalley, J., Prasad, C., and Kumar, P. K. 2007. Decoding doodles: novice programmers and their annotations. In *Proceedings of the Ninth Australasian Conference on Computing Education - Volume 66* (Ballarat, Victoria, Australia, January 30 - February 02, 2007). S. Mann and Simon, Eds. ACM International Conference Proceeding Series, vol. 239. Australian Computer Society, Darlinghurst, Australia, 171-178.