IS THE PROBLEMATIC IN CS1 A STUDENT'S PROBLEM SOLVING ABILITY?

Stanley T. Schuyler, Edinboro University, sschuyler@edinboro.edu
Robert Joseph Skovira, Ph. D., Robert Morris University, skovira@rmu.edu

ABSTRACT

A student's problem solving ability (PSA) is often considered a factor in determining their success in a first course in computer programming (CS1). Can a student's PSA be measured before taking a CS1 course? How? Would such an assessment effectively predict final exam scores? This paper presents the results of a pilot study guiding the development of a PSA assessment method. The approach attempts to use a student's words in response to an analysis task to extract quantifiers of PSA. This research is viewed as a compass to point out a direction (with much more road to travel). The tentative answer is "PSA may be detectable in student's expressions."
Keywords: problem solving, process, assessment, computer programming, semantics, linear and polynomial regression.

INTRODUCTION

Quantitative research studying factors that would predict student performance in their first course in computer programming (CS1) began in the early 1980's and continues to the present [2, 4, 14, 19, 20, 21]. A small subset of that body of research is cited here due to space limitations. An article published in IACIS'06 by Carl Farrell titled *Predicting (and Creating) Success in CS1* [4] helped catalyze this study. Farrell's research reported on the development of a computer programming placement test. This placement test [4], like others developed since the early 1960s [12], was designed to determine if a student is ready for a CS1 course or should be placed in a preparatory course (CS0). Farrell's results showed that two score boundaries separated students into three groups: top scorers almost always succeeded (22 of 23 participants), bottom scorers almost never succeeded (8 of 9 participants), and middle scorers were unpredictable (20 of 38 participants succeeded) [4]. Student scores on Farrell's test accounted for 26% of the variance in students' course grades [4].

A review of IBM's PAT [12], the sample questions in Farrell [4], and others [17, 18], reveal that the questions presented are similar to word problems used in mathematics and logic aptitude tests. There is an assumption that the problem solving skills used in such problems are required to learn computer programming [2, 4, 12, 17, 18]. However, much of the published research, regardless of focus, suggests that the root cause of poor student performance could be a student's problem solving ability (PSA) [2, 4, 6, 7, 8, 14, 16, 17, 18, 19]. If problem solving in mathematics were the same as in computer programming, one would expect mathematics-like assessments to be a more reliable predictor of student performance. What is reported is that many students with high math scores are unsuccessful in CS1 while other students with low math scores are successful [2, 7, 8, 9, 18]. These inconsistencies support the claim that many factors are interacting in unclear ways [16]; further, they could indicate that other, as yet unmeasured, aspects of problem solving skills are required for learning to program.

Several qualitative studies investigated other factors thought to be associated with PSA [6, 7, 8, 9, 11, 16, 17]. Examples include *learning style* [7], *verbal articulation ability* [13, 17], and *motivation* [5, 8, 12]. Some investigations assumed PSA was pre-existing in students who were successful [5, 17]; other studies postulated that students must be motivated by CS1 content before they will exercise or apply PSA that it is assumed they possess [16, 11, 21]. Still other studies indicated that students acquired problem solving skills as a byproduct of the course [8, 13]. A study by Lister et al [9] suggested another plausible reason for poor performance is that students lack the ability to read and trace through code. Three hypotheses in this study were [9]: if students demonstrate understanding and are able to complete existing code segments, then they possess basic programming knowledge and PSA; if students demonstrate understanding but cannot complete code segments, then students lack PSA; however, if students cannot demonstrate understanding existing code it cannot be concluded that the students lack PSA. Instead, the only conclusion that can be reached is they lack basic programming knowledge [9].

The Problem

This research attempts to reveal evidence of PSA in students prior to taking CS1. What form of assessment would cause PSA to show-up and in what form of expression would it show-up? Would the evidence detected provide the same or different information than current aptitude tests? In order to

answer these questions a definition of PSA is required.

The Problem Solving Process

A problematic situation is one that either exhibits symptoms of a problem or presents opportunities for improvement [3, 10, 15]. Problem solving occurs when an individual, faced with a problematic situation, uses a reflective thought process to identify causation and devise a solution that eliminates problems or changes the situation to an improved state [3]. George Pólya, in his book *How to Solve It,* describes general problem solving as a four-stage process [15]. Stephen Allen describes a five-stage process [1]. Lister reproduces a five-stage process from a multinational study conducted by McCracken in 2001 [9]. These processes are *sketched* (abbreviated) for comparison in Table 1 (below). Regardless of the author's terminology and level of detail, each process description semantically centers on the same shared concepts and stages.

Table 1: Problem Solving Processes Compared

Table 1: Problem Solving Process Steps Defined			
Step	**Any Domain (Pólya, 1957)**	**Any Domain Allen (1995)**	**Programming Domain (Lister & Adams, 2004)**
1	Understand the Problem: -Know terms -Show what? -Restate -Diagram -Question?	Select the problem or situation or let it select you.	Abstract the problem from the description.
2	Devise a Plan -Imagine -guess, check -List possible -Patterns? -Exceptions? -Model/reason -Use/reuse … -Simplify -Go backwards	Observe, organize, and define the problem or situation.	Decompose the problem into sub-problems.
3	Carry out Plan -Iterate/persist -Trial/error -Change	Learn by questioning everything.	Transform sub-problems into sub-solutions.
4	Look back -Evaluate -Continuous improvement	Visualize solutions; select one, and refine it.	Synthesize sub-solutions to higher levels.
5		Employ the solution and monitor the results.	Evaluate the solution and iterate until verification.

Exploring Student Responses to Problematic Situation Descriptions

Having no clue as to how to articulate a path forward, it was decided that whatever the method was, it must evoke responses from students that could be analyzed algorithmically. The method chosen was to present a narrative describing a situation and ask students to explain in writing how they would analyze the situation to identify *problems*, *opportunities*, or develop *questions*. This approach was expected to yield expression related to a student's unique problem solving process, even if their problem solving process was tacit. The higher the volume of expression, the higher the probability student terms can be associated with terms in Table 1.

Proposed Questionnaire

Students in a CS1 course were presented with a description one of three situations and given the task to write down how they would analyze the situation described. They were told to imagine that their future role would involve developing a computer application to correct problems they perceived, or make improvements they could imagine. The three situation descriptions follow.

Situation 1 – Ambiguous (no explicit symptoms of problems or implied need). This narrative contains the verb "need to be" which is used to describe a generic data transformation process. The last sentence implies the transformation is operational and does not imply that there is, or is not, an unmet need.

"*Scientists, engineers, students, instructors, market analysts and others have data in the form of categories. Each category contains various sized lists of numeric data entries. These data need to be analyzed quantitatively by calculating totals, averages, and sometimes other statistical measures. The results of the calculations are used to summarize the category, make comparisons, draw conclusions, and communicate these to other interested parties.*"

Situation 2 – Ambiguous with implied need (no explicit symptoms of problems).This narrative contains the verb "need to be" which is used to describe a generic data transformation process used

for rendering presentations. The situation description ends abruptly and does not indicate whether the outputs are produced or if there are unmet needs.
"*Scientists, engineers, students, instructors, market analysts and others have conducted (and will periodically conduct) experiments. Each experiment produces a unique series of numeric values. These values need to be presented in various formats such as lists, tables and/or charts (for example xy scatter plots and/or histograms) on screens and in documents."*

Situation 3 – Unambiguous (explicit symptoms or problems are expressed). This narrative contains the verb phrase "is responsible to" which is used to describe a generic data transformation process used to produce paychecks. The last sentence clearly indicates the transformation process is operational and there are unmet needs.
"*A payroll administrator is responsible to issue paychecks for employees on Fridays based on the number of hours reported as worked per employee, hourly pay rates, overtime rates, deductions for various government mandated programs, insurances and other deductions. The administrator receives hand written time sheets and uses a PC to print the paychecks. Paychecks are often delayed to Mondays and sometimes contain errors requiring retraction and reissuing.*"

Student Tasks:

Each student was asked to perform three tasks that were described in detail. Students were required to read the task descriptions on their own. Briefly, in task 1, labeled *Analyze*, they were instructed to "describe how would you analyze the situation but not to analyze it;" in task 2, labeled *Problematize*, they were to "identify problems (needs), or use acts of imagination to suggest changes (opportunities), that would correct or improve the situation in some way;" and in task 3, *Question*, they were to write down as many questions as they could think to help them understand or respond to the situation described.

Methodology

The questionnaires with the three situations (labeled 1, 2, or 3) were administered in the second CS1 class session. Student responses were hand-written. All responses were transcribed into text files verbatim, except that misspellings were corrected and numbering schemes were standardized; e.g. if numbers like 1, 2, etc. represented steps in a method, they were augmented to "Step1," "Step2," etc. Participant's responses were reviewed sentence by sentence. If they related to a step in Table 1, the sentence was tagged with a meta-comment prior to the final exam. For example the meta-comment "<observes>" is added if a student describes an observation step or is clearly making observations. The final exam contained content similar to Lister's exam [9]. When exam grades were returned, each participant's responses were ready for analysis.

Analytical Approach

Individual words and meta-tags were extracted and their frequency of occurrence tabulated across all students and situations as the rows of a matrix. Students were grouped by situation (1, 2, or 3) as the columns of the matrix. Words and meta-tags were analyzed separately, and both were sorted by frequency. Words were grouped and assigned to one of three expressive groupings: words with frequencies below five were categorized as "unique;" words used five or more times were considered "common." A uniqueness score was calculated using two criteria: the ratio of unique words to the median of all words in the same situation number; and the ratio of a participant's unique words to their total word count. These two ratios were combined heuristically for each student and ranked. The subtype of "unique" was assigned to a student if their score was ranked in the top quartile of uniqueness scores, "common" otherwise.

Students within each situation were then categorized as follows: "Not expressive" if in the bottom quartile of words produced; "Expressive Common" if in the top three quartiles and their subtype was "common;" and "Expressive Unique" if in the top three quartiles of word production and their subtype was "unique." These groupings formed the basis for comparing student performance and running multiple regression testing.

RESULTS

Participants, Word Frequency, and Exam Grades

Thirty Nine (39) of 42 students enrolled in two sections of a CS1 course agreed to participate in the study. Of the 39, 28 completed the course, and 23 of 28 performed at or above 70% on the exam. The ratio of 41% performing poorly or withdrawing is consistent with other studies cited. See Attachment A for a more complete description of the participants. There were no significant differences between males and females on exam or course grades (9 females in study). Further, there were no significant differences in self-reported math course grades or GPA.

Summary of One-way ANOVA Testing on Categories

Table 2 below presents the total number of commonly used words produced by each student based on the categories described above (Situation by Expressive category). Table 3 is similar but presents the number of qualitative meta-tag comments subjectively assigned to each student's expression. There was no expectation regarding the volume of expression across situations. The number of participants in each cell (Situation by Expressive Type) was too unbalanced in this pilot study to run a Two-way ANOVA. A one-way ANOVA indicated there were no significant differences in word volume between situations.

There was an expectation that students would vary widely in terms of word production.

Table 2: Frequency of Words by Situation and Expressive Type

All Words Shared

	Not Expressive	Expressive Unique	Expressive Common	n	Sit.Mean
Sit1	35	75	53		42
	6	50	53		
	3		52		
			53		
Exp.Mean	15	63	53	9	
Sit2	36	87	48		46
	25	56	51		
	11		73		
	31				
Exp.Mean	26	72	57	9	
Sit3	17	72	77		64
		67	53		
			86		
			82		
			61		
			82		
			42		
Exp.Mean	17	70	69	10	
m	8	6	14		
G.Mean	21	68	62		51

Table 3: Frequency of Qualitative Meta-comments

All Qualitative Comments Shared

	Not Expressive	Expressive Unique	Expressive Common	n	Sit.Mean
Sit1	11	15	12		8
	3	6	9		
	2		7		
			7		
Exp.Mean	5	11	9	9	
Sit2	4	10	8		7
	11	7	10		
	2		7		
	8				
Exp.Mean	6	9	8	9	
Sit3	5	8	9		8
		5	10		
			8		
			10		
			7		
			9		
			4		
Exp.Mean	5	7	8	10	
m	8	6	14		
G.Mean	6	9	8		8

A One-way ANOVA across expressive types was significant at the p.0001 level separating the *Not Expressive* category from *Unique* and *Common* (Table 2). There were no differences for qualified meta-tag counts in Table3.

Student Performance

The frequency of commonly used words was the surrogate measure of problem solving ability (PSA). A student's final exam grade was the dependent variable. Final exam grades are shown in Table 4 by situation and expressive type. There is some elevation in grades across expressive types. However, there were no significant differences across situations or expressive types.

Table 4: CS1 Exam Grade by Situation and Expressive Type

CS1 Final Exam Grade

	Not Expressive	Expressive Unique	Expressive Common	n	Sit.Mean
Sit1	84	86	89		73
	54	74	85		
	23		84		
			82		
Exp.Mean	54	80	85	9	
Sit2	91	97	89		78
	89	94	86		
	81		78		
	0				
Exp.Mean	65	96	84	9	
Sit3	67	53	98		65
		24	91		
			77		
			75		
			72		
			62		
			30		
Exp.Mean	67	39	72	10	
m	8	6	14		
G.Mean	61	71	78		72

Summary of Regression Testing by Situation Type

Each situation type was analyzed separately to identify whether there was a predictive relationship between the volume of words shared and final exam grades (by situation across expressive types). The volume of qualified meta-tags shared was similarly examined. Two types of regression were explored: linear and polynomial. Table 5 summarizes the findings.

Situation 1 was not expected to predict exam grades because it was ambiguous and did not evidence problems. Unexpectedly, the frequency of commonly used words in Situation 1 predicted exam scores and accounted for 69% (linear) and 81% (polynomial) of the variance (9 of 28 students). The polynomial fit is illustrated in Figure 1. Situation 2 was expected to provide some level of predictability. By eliminating two outliers 61% of the variation was accounted for in 7 of 28 students). Polynomial regression was not predictive for Situation 2. Situation 3 was expected to provide the most predictability but did not predict exam scores (10 of 28). In total, 16 of 28 (57%) of students scores were predictable with greater than 60% of the variance explained.

Table 5: Regression tests of Situation Type on Final Exam Grade using frequency of commonly used words

Dependent: Exam Grade	Regression Method	
Independent: Freq. of common words	Linear	Polynomial
Situation 1	Predictive: p.003 Adjusted R^2=69%	Predictive: p.003 Adjusted R^2: 81%
Situation 2	Predictive: p.02 Adjusted R^2=61% (when 2 outliers excluded: n=7)	Not Predictive
Situation 3	Not Predictive	Not Predictive

The subjective qualifying meta-tags inserted into participants responses unexpectedly showed significant predictability in Situation 1 and 3 (Table 6). In situation 1 commonly shared qualified meta-tags outperformed all words commonly shared (Table 5).

The most variance that common use word frequency explained was 81%, subjective human qualification improved this to 89%. For Situation 3, qualified meta-tags accounted for 46% of the variance and none for word frequency. This result enables predicting 7 additional students using Situation 3, although much less predictable. The total percent of students considered predictable is 23/28 or 82%.

Table 6: Regression Test of Situation Type on Final Exam Grade using frequency of qualified meta-tags

Dependent: Exam Grade	Regression Method	
Independent: Freq. of Meta-tags Assigned	Linear	Polynomial
Situation 1	Predictive: p.0005 Adjusted R^2=89%	Predictive: p.0005 Adjusted R^2: 89%
Situation 2	Not Predictive	Not Predictive
Situation 3	Not Predictive	Predictive: p.04 Adjusted R^2=46%

Figure 1: Situation 1-Polynomial Regression (n=9, R^2 = 81%)

Exam Grade = f (Frequency of Commonly Used Words)

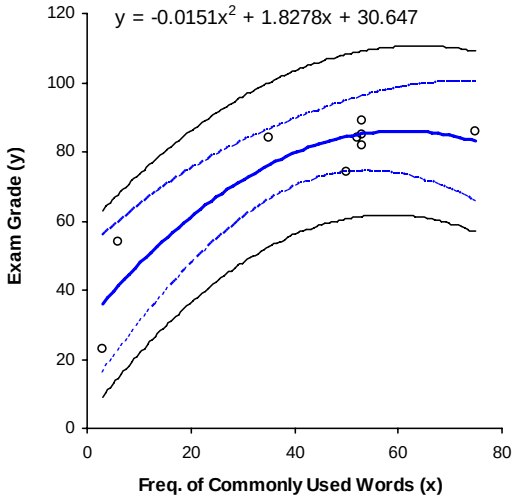
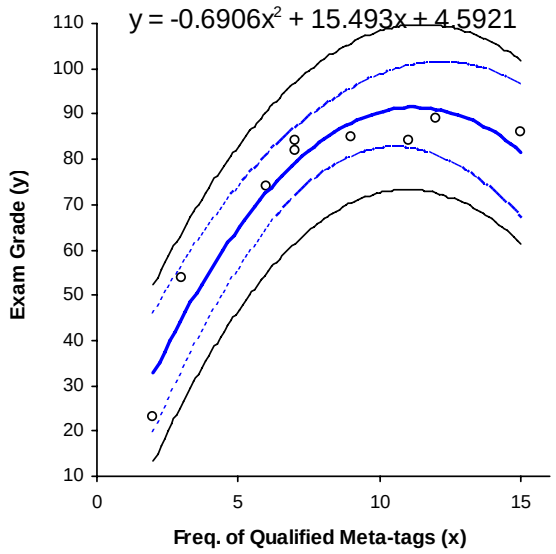


Figure 2: Situation 1-Polynomial Regression (n=9, R^2 = 89%)

Exam Grade = f (Frequency of Qualified Meta-Tags)



Summary of Regression Testing by Expression Category

Table 7 presents regression analysis performed across expression types. Both linear and polynomial regression indicate that meta-tags predicted exam grades for students categorized as *Expressive Common*, across Situations, accounting for 39% (linear) and 60% (polynomial) of the variance. The subjective tagging procedure was not useful for *Not Expressive* or *Expressive Unique* categories.

Table 7: Regression Test of Expression Type on Final Exam Grade using frequency of qualified meta-tags

Dependent: Exam Grade	Regression Method	
Independent: Freq. of Qualified Meta-tags	Linear	Polynomial
Not Expressive	Not Predictive	Not Predictive
Expressive Unique	Not Predictive	Not Predictive
Expressive Common	Predictive: p.01 Adjusted R^2: 39%	Predictive: p.002 Adjusted R^2: 60%

DISCUSSION

The different situation descriptions in this small sample indicate care must be taken in composing situation descriptions to evoke PSA. Students may have found it easier to describe how they would *analyze* in Situation 1 because there was no obvious problem to solve. When problems are evidenced, as in Situations 2 and 3, students may tend to describe solution approaches rather than stay focused on describing their analytic approach to the situation.

In Farrell's study students were segmented into groups based on pre-course aptitude scores. Course grades were predictable for about half of the students. In this study students were segmented by the situation description being evaluated. Within Situation 1 there *appeared* to be a high level of predictability (with the caveat that the sample size is too small). Another approach to segmentation was based on classifying the expressiveness of the students. If we ignore the issues of testing the situation descriptions, there *appeared* to be vocabulary dimensions that are important in understanding how to evoke PSA relevant responses.

CONCLUSIONS

The purpose of this study was to investigate directions for developing a PSA assessment and using it to predict student exam grades in CS1. The assessment used student expressions as the basis for PSA detection. What the results suggest is that the words students use to describe how they plan to analyze a problematic situation is worth a deeper investigation. The way a situation is described (with or without obvious symptoms of problems) affects the utility of using word frequencies as a predictor. Situation 1, the most ambiguous, yielded a measure useful for predicting student performance on a small sample of students. The exam grades of students categorized as "expressive common," that is those students that shared a common vocabulary, were predictable regardless of the situation type. Students categorized as "not expressive" or "expressive unique" were difficult to measure or predict regardless of category. Qualified meta-tags characterizing student expressions in situation 1 showed promise for predicting low and inconsistently expressive students. The direction in which the "compass points" is that we need to dig into the production of expression (limited, unique, or shared), and associated semantics (meta-tagging) in order to find a predictive measuring device for PSA. Whether this is truly a measure of PSA or not, it has predictive promise.

REFERENCES

1. Allen, Stephen D. (1995). *Winnie-the-Pooh on Problem Solving*. Dutton Penguin Books USA Inc., New York, NY.

2. Choi-man, Chung. (1988). *Correlates of Problem-solving in Programming*. Chinese University of Hong Kong Educational Journal, 16(2), 185-190.
3. Dewey, John. (1939). *Intelligence in the Modern World - John Dewey's Philosophy*. Ratner, Joseph (Ed.). Random House of Canada Limited, Toronto, Canada.
4. Farrell, Carl. (2006). *Predicting (and Creating) Success in CS1*. Pratt (Ed.), ISSUES IN INFORMATION SYSTEMS-IACIS 2006, Vol. VII, No. 1, pp. 259-263.
5. Fleming, Stewart T. (2003). *Talking past each other - Student and Staff Reflection in Undergraduate Software Projects.* Issues in Informing Science and Information Technology,Publisher@InformingScience.org (Ed.), http://www.cs.otago.ac.nz/stf (August 2006). pp. 93-102.
6. Holden, Edward, and Weeden, Elissa. (2006). *What makes Valuable Pre-experience for Students entering Programming Courses?* Issues in Informing Science and Information Technology, 3, pp. 279-293.
7. Ivins, Jim and Ong, Michele Poy-Suan. (2003). *Psychometric Assessment of Computing Undergraduates.* Department of Computing, Curtin University of Technology, Perth, Australia.
8. Jenkins, Tony. (2002). *On the Difficulty of Learning to Program*. Proceedings of the Centre for Information and Computer Sciences: Information and Computing Science reprint, University of Leeds, Leeds, UK. pp. 53-58.
9. Lister, Raymond, Adams, Elizabeth S., Fitzgerald, Sue, Fone, William, Hamer, John, Lindholm, Morten, McCartney, Robert, Mostrom, Jan Erik, Sanders, Kate, Seppala, Otto, Simon, Beth, and Thomas, Lynda. (2004). *A Multinational Study of Reading and Tracing Skills in Novice Programmers*. SIGCSE Bulletin, 36(4), pp. 119-150.
10. Marakas, George M. (2006). *Systems Analysis Design* (2nd ed.). McGraw Hill Irwin Companies Inc., New York, NY.
11. Martin, Fred. (2006). *Toy Projects Considered Harmful*. Communications of the ACM, 49(7), pp. 113-116.
12. McNamara, J.W. (1965). *The Selection of Computer Personnel - Past, Present, Future*. ACM On-line Archives. http://delivery.acm.org/10.1145/1150000/1142667/p52-mcnamara.pdf (September 2006), pp. 52-56.
13. Newstetter, Wendy, and Khan, Sabir. (1997). *A Developmental Approach to Assessing Design Skills and Knowledge*. Frontiers in Education Conference, 27th Annual Conference, Vol. 2, pp. 676-680.
14. Perrenet, Jacob, and Kaasenbrood, Eric. (2006). *Levels of Abstraction in Student's Understanding of the Concept of Algorithm: the Qualitative Perspective*. Implagliazzo, Goldweber and Paola (Eds.), SIGCSE Bulletin, 38(3), pp. 270-274.
15. Pólya, George. (1957/1945). *How to Solve It*. Doubleday and Co., Inc., Garden City, NY.
16. Rountree, Nathan, Rountree, Janet and Robins. (2004). *Interacting Factors that Predict Success and Failure in a CS1 Course*. SIGCSE Bulletin, 36(4), pp. 101-104.
17. Simon and Cutts, Quinton, Haden, Patricia, Sutton, Ken, Box, Ilona, Hamer, John, Lister, Raymond, Tolhurst, Denise, Fincher, Sally, Robins, Anthony, Baker, Bob, deRaadt, Michael, Hamilton, Margaret, Petre, Marian and Tutty, Jodi (2006). *The Ability to Articulate Strategy as a Predictor of Programming Skill*. Eighth Australian Computing Education Conference (ACE2006), 52, pp. 1-8.
18. Tukiainen, Markku, and Monkkonen, Eero. (2002). *Programming Aptitude Testing as a Prediction of Learning to Program.* PPIG 2002 - 14th Workshop of the Psychology of Programming Interest Group, 14, pp. 45-57.
19. Wiedenbeck, Susan, LaBelle, Deborah and Kain, Vennila, N.R. (2004). *Factors Affecting Course Outcomes In Introductory Programming*. PPIG 2004 – 16th Workshop of the Psychology of Programming Interest Group, Retrieved April 23, 2007 from www.ppig.org.
20. Wilson, Brenda Cantwell, Shrock, Sharon. (2001). *Contributing to Success in an Introductory Computer Science Course: A Study of Twelve Factors*. Technical Symposium on Computer Science Education. Proceedings of the 32nd SIGCSE technical symposium on Computer Science Education, pp. 184 - 188
21. Whittington, Kieth J. (2006). *Increasing Student Retention and Satisfaction in IT Introductory Programming Courses using Active Learning*. Proceedings of the 2006 Informing Science and IT Education Joint Conference, Salford, UK. pp. 307-312.

Attachment A: Participants in Pilot Study

CS1 (C++) Fall 2006							
39 Enrolled					Fail or Drop		Majors Legend:
	Expression Category			59%	41%		
Major	**Not**	**Unique**	**Common**	**Total**	**Withdrew**	**Failed**	• ARTS (Graphic Design and Communications) • CS (Computer Science) • ED (Education) • MA (Mathematics) • Psych (Psychology) • SCI (physics, biological science) • UNK (Unknown-missing data)
ARTS	1		1	**2**	3		
CS	1	1	5	**7**	4		
ED			1	**1**	1		
MA	2	3	4	**9**	1	2	
Psych	1	1	1	**3**		2	
SCI	2	1	2	**5**	2	1	
UNK	1			**1**	0		
Totals	**8**	**6**	**14**	**28**	**11**	**5**	