Programming Perceptions: Differences among CS and non-CS majors

By James Penton

Advisor: Dr. Clark Turner

June 8, 2012

1 Related Work

Of particular interest to this study are surveys performed to measure the end users grasp of programming difficulty, as that is the area of this paper. These surveys would give empirical data that could be used to compare with the data gained from this study. The mashup paper by Zang & Rossum[1] is similar in this regard, although not an exact match.

1.1 Zang & Rossum

Perhaps the survey most similar to this one was done by Zang & Rossum in 2008. They performed a survey very similar to the one in this paper. However, instead of measuring programming task significance, Zang & Rossum studied end-user perceptions regarding mashups, a web application that integrates multiple data feeds into one interface. Their study group consisted entirely of students from non-programming disciplines.

First, user data was collected from the students in the forms of question such as “what is your major”, “what OS do you use”, and “do you own a PDA”. The participants were then shown a screenshot of a program and asked to gauge the difficulty in creating the mashup. They were then asked about the usefulness of mashups. Finally, they were then asked to write down the steps required to create the application. Zang & Rossum used, as an example application, a mashup of integrating CNN and Wikipedia.

Zang & Rossum found that most people gauged difficulty slightly lower than usefulness. They also concluded that most internet users do not understand enough about mashups to correctly identify the difficulty in creating one. Additionally, they also found that females tended to regard mashups as more difficult to create than males.

1.2 Pioro

In 2006, Pioro performed an analysis comparing the grades of students taking their first programming class at a university. It was found that people who took mathematics courses, specifically calculus and discrete math, received higher grades in their first programming course than those without. They also found out that the grades in the math courses didn't affect their programming grades, just whether or not the courses were completed.

1.3 Bevan & Azuma

In 1997, Bevan & Azuma measured end user program usability. They stated that in order to correctly gauge the effectiveness of software, you must specify the context of its use. You need to know the characteristics of the intended users, the tasks to which it will perform, and then environment in which the users are to use the

product. In their research, Bevan & Azuma also defined various terms that are very useful.

- Effectiveness: accuracy and completeness that a particular goal is met.
- Efficiency: relationship between level of effectiveness achieved and the resources spent.
- Satisfaction: both the physiological and emotional responses to using the system, as well as the acceptability of using the system.

1.4 Lurain & Weinshank

In an older paper(published in 2000), Lurain & Weinshank discuss the need for non-CS majors to learn programming. They concluded that programming is not and should not be required for non-computer majors, as the skillsets to use software today is far removed from actual programming. The trend to teach programming to non-majors has declined as people are now more tech-savvy than before. Additionally, in the past you needed to be able to program in order to work with data, you can now purchase cheaply or use other software (such as Excel) to do the number crunching jobs that 15 years ago you needed a programmer to do.

Additionally, Lurain & Weinshank performed a study over 5000 students from multiple universities who were enrolled in a “CS-0” course, or an introductory computer competence course. They were given various computer literacy tasks ranging from file edit permissions to creating web pages to working with Excel. It was concluded that these students are gaining conceptual understanding without learning to program, which is a discrete skill set not applicable to most non-CS majors.

1.5 Maiden & Sutcliffe

Maiden & Sutcliffe talk about editing existing software to fulfill a new role, and the complications that arise from having novice software engineers reuse code. However, the interesting parts come in their related works and background section regarding program understanding. Understanding unfamiliar software is complex and error-prone; this goes for most all people, including all but the most expert of software engineers. Novice and intermediate programmers still think superficially, and their views of programs differ little from other people. They still tend to clump together sub-programs and code into chunks that “look similar”, unlike expert programmers who clump together code by functionality. Expert programmers can think more abstractly and possess deeper and more principled representations of the flow of the program.

According to the research pointed to by Maiden & Sutcliffe, I should see very little change between undergrad CSC and non-CSC majors. The change in programing perceptions should not occur until the study is performed on the workforce with experienced software engineers.

1.6 McCracken et. al

216 students from different universities, taking either their 1st or 2nd programming course, were evaluated to measure their programming competence. The students were given multiple programming assignments to complete regarding programming an advanced calculator, and graded on how well the program turned out. Out of a possible score of 110, the average student score was 22. The conclusion was that new CS students were not performing up to expectations regarding programming skill levels.

1.7 Ramalingam and Weidenbeck

In a study by Ramalingam and Widenbeck, 75 students who were enrolled in their first programming course, C++, took part in a survey to find correlations between programming styles and competence. The researchers found that program comprehension is greater in novice programmers who focus on an object-oriented language, than those who focus on an imperative programming style.

1.8 Scaffidi et. al

Scaffidi, Ko, Myers, and Shaw performed a study based on a survey given by *Information Week.* The purpose of this survey was to gather program knowledge from people who work in the IT fields. Data was gathered regarding respondants' program and application usage(such as how often their firm used various programs and functions of those programs), along with a questionnaire to obtain their competency in various fields. Correlations were drawn between certain areas, such as people who used macros in word processors were more likely to use macros in spreadsheets as well. Other similar correlations were found. They found the most popular feature was "linked structure" features, such as 2D grids or graphs that allow people to highlight data more easily. It was also found that overall, skilled information workers do not vary greatly in their feature usage.

References

1. Nan Zang; Rosson, M.B.;, "What's in a mashup? And why? Studying the perceptions of web-active end users," Visual Languages and Human-Centric Computing, 2008. VL/HCC 2008. IEEE Symposium, pp.31-38, 15-19 Sept. 2008.

2. B.T. Pioro. 2006. Introductory computer programming: gender, major, discrete mathematics, and calculus. J. Comput. Small Coll. 21, 5 (May 2006), 123-129.

3. N. Bevan and M. Azuma. 1997. Quality in Use: Incorporating Human Factors into the Software Engineering Lifecycle. In *Proceedings of the 3rd International Software Engineering Standards Symposium (ISESS '97)* (ISESS '97). IEEE Computer Society, Washington, DC, USA, 169-.

4. Urban-Lurain, M., Weinshank D.J., Do non-computer science students need to program?, Journal of Engineering Education, 90 (4), 535-541, 2001.

5. Maiden, N.A.M.; Sutcliffe, A.G.; , "People-oriented software reuse: the very thought," *Software Reusability, 1993. Proceedings Advances in Software Reuse., Selected Papers from the Second International Workshop on* , vol., no., pp.176-185, 24-26 Mar 1993.

6. Michael McCracken, Vicki Almstrum, Danny Diaz, Mark Guzdial, Dianne Hagan, Yifat Ben-David Kolikant, Cary Laxer, Lynda Thomas, Ian Utting, and Tadeusz Wilusz. 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* (ITiCSE-WGR '01). ACM, New York, NY, USA, 125-180.

7. Vennila Ramalingam and Susan Wiedenbeck. 1997. An empirical study of novice program comprehension in the imperative and object-oriented styles. In *Papers presented at the seventh workshop on Empirical studies of programmers* (ESP '97), Susan Wiedenbeck and Jean Scholtz (Eds.). ACM, New York, NY, USA, 124-139.

8. Scaffidi, C.; Ko, A.; Myers, B.; Shaw, M.;, "Dimensions Characterizing Programming Feature Usage by Information Workers," *Visual Languages and Human-Centric Computing, 2006. VL/HCC 2006. IEEE Symposium on* , vol., no., pp.59-64, 4-8 Sept. 2006.