First Computer Programming Language Acquisition:

Applying Second Language Acquisition Theories to Computer Programming Curricula

Nathan Sidles

I. INTRODUCTION

Computer programming is the art of telling a computer how to do a thing. A computer programming language is a language designed to allow a human to communicate these instructions in a way that the computer can translate into action.

Learning computer programming languages at the postsecondary level is difficult and unpopular. In 2005, enrollment and interest in computer science programs had dropped more than 70% from its peak in 1983 – from nearly 5% of incoming freshmen to less than 1.5% in 2005 (Vegso, 2006). Though enrollment rates have increased in recent years, they have not increased enough to regain the popularity they once had. Beyond this, computer science programs have high rates of attrition. At some universities, the six-year graduation rate for computer science programs has been as low as 25% (Southeastern Louisiana University, 2010).

There are many explanations for the difficulty students encounter with computer science programs: poor student math skills, poor advising prior to entry to the program, and poorly designed introductory courses are ranked highest among them. (Beaubouef & Mason, 2005). Students may also be dissuaded from entering the field due to perceived lack of excitement, lack of human interaction, and lack of equal opportunities for women (Carter, 2006) (Cohoon, 2001).

Acquiring second language competency or fluency is both more commonly attempted and more commonly achieved. 9.1% of individuals over 5 in the United States reported both speaking English “very well” and speaking a foreign language as a first language (U.S. Census Bureau, 2003). Second languages are taught

according to many methods. Prominent among these are the interactive approaches: communicative language teaching, language immersion, and Teaching Proficiency through Reading and Storytelling (TPRS), among others.

These methods are more evaluated, more defined, and more successful than current methods of teaching computer programming languages. Computer science pedagogy can be improved by incorporating the techniques and methods of second language acquisition. Drawing on these techniques has the potential to increase student retention, graduation rates, and satisfaction with computer science programs – improve "first computer programming language acquisition," as it were.

This paper acknowledges the differences between a programming language and an actual language. The three most obvious differences are that (1) programming languages are not and cannot be used in the everyday situations of living, (2) programming languages are not used to communicate between people, but to command computers, and (3) programming languages refer almost exclusively to abstract concepts, while languages refer to both abstract concepts and to concrete objects. The fourth major difference is subtler and more salient: any level of proficiency with a language provides some utility, especially if the user is in a social context where that language is dominant. A student of a computer programming language, however, does not typically derive any utility from that programming language unless they have achieved a minimum level of competency. Without that level of competency, they are not typically admitted to a social context that encourages them to use their knowledge.

Beyond these differences, however, there are three similarities between a programming language and an actual language that allow second language acquisition methods to be applied profitably to programming languages. These similarities are that (1) both languages and programming languages communicate meaning, (2) both use semantically stable symbols to communicate that meaning, and (3) both have syntax, grammar, and vocabulary. These similarities are foundational enough that methods from teaching one can be applied to teaching the other.

This paper reviews second language acquisition methods, and reviews the commonly seen hardships with learning a second language. It then reviews programming language acquisition methods, the structures of postsecondary computer science curricula, the methods used to communicate concepts, and the commonly seen hardships with learning a programming language. It then discusses commonalities in second language acquisition methods, shows how second language acquisition methods can be applied to methods of teaching computer programming languages, especially at the postsecondary level, and draws lessons from second language acquisitions methods for computer programming language acquisition going forward.

II. LITERATURE REVIEW

This section summarizes the history of second language teaching methods, the characteristics of the most prominent and effective modern second language teaching methods, and the most common difficulties students encounter when

acquiring a second language. It also summarizes current computer programming language curricula, the methods these programs use to communicate concepts, and the difficulties that students encounter when learning a programming language and the concepts of programming.

A. Second Language Acquisition Methods

Traditional second language teaching methods emphasized learning the vocabulary of a language, the formal rules of that language, and the exceptions to those rules. Modern methods emphasize communication, interaction between the student and users of that second language, and comfortableness with a language (Ellis, 2010). Three of the most prominent of these interactive methods are the communicative language teaching method, the language immersion method, and Teaching Proficiency through Reading and Storytelling (TPRS) (as an offshoot of a method called Total Physical Response).

The communicative language teaching method is a classroom-based approach that focuses on the concepts of meaning and "communicative competence." Communicative competence means the ability to "negotiate meaning – to successfully combine a knowledge of linguistic, sociolinguistic, and discourse rules" (Savignon, 1987). It encourages students to learn to communicate in a second language through examples, games, exchanges, peer exercises, and learning by teaching, presenting, and speaking. These activities take place in the classroom or in an academic setting.

The language immersion method is a holistic approach that places the student in a social and academic context where use of the second language is

required. The language is used for all aspects of teaching; it is not taught only as the subject (Cummins 1998). The goals of language immersion are to get the students comfortable using the language in everyday and academic situations. These activities take place in the second language classroom and in the student's larger academic setting. Language immersion methods have been shown to be effective in improving academic performance in students learning a second language (Cheng et al., 2010).

Teaching Proficiency through Reading and Storytelling (TPRS) is an approach that uses stories and reading to encourage students to use and to become comfortable using the target language (Cantoni, 1999). TPRS focuses on fluency and practice with the language above knowledge of the formal rules of that language. Students are introduced to the language through listening to simple stories. As the students gain experience and competency, these stories become more complex and the student begins to tell them, and finally to read and to write them. TPRS holds that students are more likely to retain knowledge if that knowledge is gained in the service of entertaining, memorable narratives. TPRS activities take place in the classroom or in an academic setting.

B. Computer Programming Language Curricula

Introductory courses to computer programming languages are most commonly divided cleanly by subject. A typical introductory computer programming syllabus is organized into seven sections: (1) basic concepts of computing, (2) variables and simple data types, (3) functions, (4) loops and operators, (5) parameters and pointers, (6) complex data types, and (7) classes and

objects (Ekbia) (Sherriff, 2011) (Mark, 2010). These sections may differ slightly in their number, their order, and their division, but the structure is the same. A student is taught one subject, a student masters one subject, and the course moves on, thereafter incorporating that subject into future subjects. For example, students manipulate simple data types when they learn about functions, and then use simple functions to learn about loops, and so on.

Teachers of programming languages commonly use metaphors to explain the concepts that students are learning (Carroll & Mack, 1999). For example, variables are like nouns (or collective nouns), functions are like sentences or paragraphs describing an action, and operators are like verbs (Sidles, 2011). A teacher can explain a function that uses variables and operators in real-world terms, as an object that gets added to, or multiplied, or deleted, or turned into something bizarre.

The most common difficulties with learning programming languages occur when these metaphors break down (Carroll & Mack, 1999). This breakdown typically occurs when students reach the subject of pointers (Milne & Rowe, 2002). Pointers do not have a simple analogy in the real world. The most common metaphor that teachers use to explain pointers is that of a catalogue entry: when one is in a library, one can reference a book by its title, its contents, or its location as recorded in the library's catalogue (Mark, 2010). This metaphor breaks down when one learns that one can use this catalogue entry to manipulate the object it references (or "points to") as easily as if one held the object in one's hand – and indeed, that it is preferable to manipulate the catalogue than to enter library. Explaining concepts like pointers is the most commonly reported barrier to success

in learning programming languages (Lahtinen & Järvinen, 2005). For many students, the difficulty they encounter with this and other concepts is insurmountable, and they drop out of their program.

III. RESULTS.

This section discusses the shared characteristics of the second language teaching methods discussed in this paper and the differences between second language teaching methods and programming language teaching methods.

A. Shared Characteristics of Second Language Teaching Methods

The most prominent second language acquisition methods discussed in this paper share a number of characteristics. These are: (1) a recognition of the importance of encouraging target language use to interact with peers and teachers, (2) both coercive and persuasive methods to encourage this use, (3) an emphasis on practice both in and out of the classroom, and (4) a de-emphasis on "correctness" in target language use in favor of general target language use.

Encouraging students to become comfortable using the target language is both a goal and a method of these interactive methods. Comfortableness with a language is an important aspect of language competency, but this comfortableness is also used as a means to encourage students to practice using the language, with the hope that practice will increase competency. Communicative language teaching encourages practice by playing classroom games, encouraging work with peers, and by holding conversations with the teacher. The immersion method encourages practice by requiring the student to perform many social functions in the target

language. TPRS encourages practice by using the target language to tell, to listen to, and to read interesting stories. All of these methods, however, encourage practice. Even if the student is using the language poorly, ignorantly, or "wrongly," they are still using the language.

B. Differences between Current Second Language and Programming Language Teaching Methods

Second language teaching methods are largely focused on communicative competence and comfortableness with using a language, while programming language curricula are focused on sequential subject mastery.

Part of this difference comes from the differences between the two subjects. This common structure of programming language curricula, after all, is partly dictated by the nature of the subject: programming languages are not true languages, students cannot easily use them to communicate with each other or the teacher, and learning a programming language requires understanding concepts that are more alien to a first language than any aspect of a second language. Computer programming deals in abstract concepts that are, at best, only roughly analogous to real-world concepts, and students learning programming languages must learn both the syntax and vocabulary of these languages and the concepts behind them.

A large part of the difference in teaching methods comes from a difference in objectives. Programming language curricula focus on subject mastery, subject by subject, while second language teaching methods focus on communicating meaning as a means to encouraging practice and improving language competency.

IV. DISCUSSION

Both "real" languages and programming languages share several important characteristics: they both have vocabularies, they both have syntax, and they both were made to communicate meaning. Programming languages' meaning is primarily meant to be communicated to computers, but it can be used to communicate meaning to humans as well ("Oh, that's what he was trying to do."). These similarities allow three aspects of second language teaching methods to be applied to programming language teaching methods: (1) focusing on communicating meaning, rather than linguistic rules, (2) encouraging target language use through narratives and other interesting exercises, and (3) encouraging student interaction using the target language.

Programming languages are intended to communicate meaning – specifically, commands to computers to perform an action. Programming language teachers can adopt aspects of the communicative language theory by reminding students that aspects of programming languages are a means to an end, and providing exercises that reinforce this theory. Teachers can emphasize the ultimate goal of learning a programming language by providing exercises from the very beginning of the course that result in useful changes in a computer's behavior from simple student actions that they can practice on their own and with each other. These exercises should incorporate all aspects of the programming language, so students are introduced to and become comfortable with these aspects, even if that introduction is implicit.

TPRS promotes language acquisition by provoking interest through stories. Teachers of programming languages can adopt aspects of this method by providing exercises that require students to complete or to tweak programs in service of a larger story – for example, creating a course curriculum that will result in a functional, useful program by the end of the semester, made up of simple sub-programs that the students have designed or adapted as their competency improves.

The immersion method encourages target language use in a social setting outside the classroom. Because programming languages are not designed for human-to-human communication, this method is only indirectly applicable. Teachers of programming languages can still encourage interaction using the target language in a variety of ways, however, including individualized assignments that students are encouraged to discuss and to present in class, constant assignment of small, easy projects, and assignments that improve the management and function of the class – an automated grading system or document upload system, for example, or even something as creating a logo for the class or the student.

V. CONCLUSIONS

Computer programming languages are different than "real" languages, computer programming language acquisition is different than second language acquisition, and computer programming language teaching methods are different than second language teaching methods. These differences do not mean that

programming language teaching cannot be improved by drawing on interactive second language teaching methods.

Programming language teaching currently uses a sequential subject master approach: one subject is covered in detail, then incorporated into lessons on future subjects. This method is analogous to second language teaching methods largely abandoned decades ago: learning a vocabulary of "nouns" and the rules for using them, learning a vocabulary of "verbs" and the rules for using them, and so on. This method of teaching has two obvious effects: (1) it hampers programming language students from seeing useful results from their work until they achieve a relatively high level of competency, and (2) it makes failure more possible and more permanent.

Teaching programming languages in sequential, discrete subjects allows students to have a high degree of competence in the area they have just studied, but competence in this area does not translate into creating useful programs. Useful programs are made of all the subjects that introductory programming classes teach, and so students are denied the ability to create these programs until the end of the course. This is likely to be discouraging for programming language students. Researching the attitudes and motivations of students throughout traditional introductory computer science programs compared to the attitudes and motivations of students in second languages programs would likely be a fruitful field for future research.

The sequential organization of programming language curricula also means that stumbling in one subject virtually guarantees failure in future subject. Since

future lessons incorporate previous lessons, if a student fails to understand a concept at any point in the semester, their performance in the course is likely to fall progressively as the semester continues. For example, a student who does not understand the concept of pointers is unlikely to understand the concept of pointers that point to arrays.

Second language acquisition programs are both more popular and have a higher graduation rate than computer science programs. Adopting the holistic, interactive methods of modern second language teaching methods will improve programming language courses. A student who sees the immediate effects of their learning is likely to be more motivated than a student who learns a subject without any context in which it can be used to the extent that it is learned. A student who is able to fit what they learn into a compelling narrative is likely to be more motivated than a student who cannot see the forest for the trees. A student who uses and discusses a programming language with their peers and their instructor more often will be more engaged and practiced than a student who studies alone. Computer programming language teachers would do well to study the methods of second language acquisition.

BIBLIOGRAPHY

Beaubouef, T. & Mason, J. (2005). Why the High Attrition Rate for Computer Science Students: Some Thoughts and Observation. *The SIGCSE Bulletin*, 37. Retrieved from http://darkwynter.com/repo/Personal/Amanda/Thesis/papers/beaubouef_attrition.pdf

Cantoni, G. P. (1999). Using TPR-Storytelling to Develop Fluency and Literacy in Native American Languages. In J. Reyhner, G. Cantoni, R. N. St. Clair & E. P. Yazzie (Eds.) *Revitalizing Indigenous Languages*. Retrieved from http://jan.ucc.nau.edu/~jar/RIL_5.html

Carroll, John M. & Mack, R. L. (1999). Metaphor, computing systems, and active learning. *International Journal of Human-Computer Studies, 51*, 385-403.

Carter, L. (2006). Why Students with an Apparent Aptitude for Computer Science Don't Choose to Major in Computer Science. *SIGCSE*, 06. Retrieved from http://www.softwarebyrob.com/assets/whynotcs.pdf

Cheng, L., Li, M., Kirby, J., Qiang, H., & Wade-Woolley, L. (2010). English language immersion and students' academic achievement in English, Chinese and mathematics. *Evaluation & Research in Education*, 23. Retrieved from EBSCOhost database.

Cohoon, J. M. (2001) . Towards Improving Female Retention in the Computer Science Major. *Communications of the ACM*, 44. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.22.3568

Cummins, J. (1998). Immersion Education for the Millennium: What We Have Learned from 30 Years of Research on Second Language Immersion. In M. R. Childs & R. M. Bostwick (Eds.) *Learning through two languages: Research and practice. SecondKatoh Gakuen International Symposium on Immersion and Bilingual Education*. Retrieved from http://carla.acad.umn.edu/cobaltt/modules/strategies/immersion2000.pdf

Ekbia, H. CS110: Introduction to Programming. Syllabus. Retrieved from http://www.slis.indiana.edu/faculty/hekbia/web/other/CS110_Syllabus..pdf

Ellis, R. (2010). Second language acquisition, teacher education and language pedagogy. *Language Teaching, 43*, 182-201.

Lahtinen, E., Ala-Mutka, K., & Järvinen, H. M. (2005). A study of the difficulties of novice programmers. *Proceedings of the 10th annual SIGCSE conference on Innovation and technology in computer science education*. New York, NY: ACM.

Mark, D. (2010). *Learn C on the Mac*. New York, NY: Apress.

Milne I. & Rowe, G. (2002). Difficulties in learning and teaching programming – Views of students and tutors. *Education and Information Technologies, 7*(1), 55-66.

Savignon, S. J. (1987). Communicative Language Teaching. *Theory Into Practice*, 26. Retrieved from EBSCOhost database.

Sherriff, M. S. (2011, August). CS 1110/1111 - Course Schedule. Syllabus. Retrieved from http://www.cs.virginia.edu/~sherriff/cs1110/schedule.php

Sidles, N. (2011). *Beginning C Programming*. (Unpublished Technical Writing 307 project). Idaho State University, Pocatello, ID.

Southeastern Louisiana University. (2010). *Graduation Rates by Major: Fall 2003 Cohort* [Data file]. Retrieved from http://www.selu.edu/admin/ ir/rpg/ grad_rate_by_major/2003_cohort.pdf

Vegso, J. (2006). Interest in CS as a Major Drops Among Incoming Freshmen. *Computing Research News*, 17. Retreived from http://archive.cra.org/CRN/ articles/may05/vegso

U.S. Census Bureau. (2003). Language Use and English-Speaking Ability: 2000. Retrieved from http://www.census.gov/prod/2003pubs/c2kbr-29.pdf