A Second Language Acquisition Toolkit for Teaching Introduction to Computing

Michael R. Gardner, Nina K. Telang

Cockrell School of Engineering, The University of Texas at Austin
301 E Dean Keeton St, Austin, TX, 78712, USA
E-mail: mgardner@utexas.edu, telang@ece.utexas.edu

Abstract

Introduction to Computing and higher-level programming courses are common first-year engineering curricula at the university level and are key in developing logical thought processes in engineering students. Recent research has shown that employing second language acquisition (SLA) techniques to teach programming increases exam performance and student motivation compared to more classical approaches. However, the presentation of pedagogical techniques has been largely limited to higher-level languages with more intuitive linguistic analogs and has not been extended to lower-level computing course material. In this paper we present several SLA techniques and their analogs in a computing course setting and the results of implementing an SLA strategy in a first-year engineering course. Statistical analysis shows that students taught with SLA methods completed quizzes more quickly, enjoyed recitation more, and had a higher perceived value of the class when compared with students taught with non-SLA techniques.

1. Introduction

Introduction to Computing is a course offered at the University of Texas at Austin (UT) that lays a foundation of knowledge applicable to all engineering disciplines. The course is a bottom-up approach to computing, beginning with number systems, binary representation of signed and unsigned numbers, arithmetic and logical operations, basic digital circuit building blocks for combinational and sequential circuits. The content then progresses through design of finite state machines, a simple computer model, the computer instruction cycle, and programming in an assembly language. More broadly, this course trains students to think like engineers, adopting a new, logical "language."

Recent research has shown that SLA approaches to teaching programming can improve student outcomes.[1] However, the proposed SLA techniques have not been extended to bottom-up courses like Introduction to Computing.

The aims of this paper are threefold: (1) to make the case that SLA approaches can be extended to a computing course, (2) to offer several techniques that teachers might employ in a computing setting to enhance student engagement and understanding, and (3) to offer a rigorous statistical analysis on the results of implementing an SLA strategy.

2. Language and Computing Analogs

2.1 Language Skills

In classical SLA pedagogical approaches, language is split into two categories of skills.[2], [3] Receptive skills include listening and reading, whereas productive skills include writing and speaking. For the language learner, each of these skills is important for developing fluency. The learner will only be able to function well using the target language if she can receive language and produce language, both in written and oral forms.

Computing is similar to language in these ways. Like a language learner, a computing student should be able to recognize written forms of the target language (e.g. hexadecimal representation, logical operators, state graphs, etc.). Likewise, a computing student should become fluent in writing/drawing concepts presented in class. Furthermore, the computing student should also develop oral skills, able to communicate and receive computing concepts verbally.

2.2 Language Components

Lexis is one of a few major components in SLA. Lexis includes vocabulary (single words), collocations (e.g. *bus stop*), and chunks (e.g. *if you know what I mean*).[4] In computing, then, lexis can refer to fundamental ideas like data types (e.g. ASCII, floating point), without which the meaning cannot be conveyed. Scrivener points out that without a complete lexis, a language learner might be left to say, "I wonder if you could lend me your ____," but the meaning is lost without "calculator."[4] Similarly, a computing student who doesn't have a complete lexis cannot perform required operations (e.g. _____ OR _____).

Another major component in SLA is grammar. Grammar enables the formation of new sentences and structures. For example, a language learner might be taught how to conjugate verbs to talk about the past or how we might turn an adjective into an adverb. In the computing context, "grammar" can refer to how we combine inputs with operators (e.g. n-input AND gates output one value; decoders take n inputs and have 2^n outputs; etc.) to construct a system.

Of course, these analogs can be expanded as the course develops such that assembly instructions (operations, data movement, and control) take on "linguistic" structures of their

own, with particular "grammatical structures" and "lexical content."

Noticeably missing from this list of linguistic components is phonology, how language is expressed verbally. In the context of computing, verbal communication is a function primarily of understanding and employing computing terminology. Because the computing lexicon generally finds its basis in English, a student's "phonology" (ability to communicate computing ideas verbally) will naturally develop with understanding the course material.

With these established analogs between SLA and computing, we can explore particular SLA pedagogical techniques and how they can be effectively employed in a computing classroom context.

3. Toolkit for Teaching Computing

In this section, we offer tools, techniques and activities that have proven to be effective pedagogical methods in SLA settings and hold promise as effective methodologies for teaching computing.

3.1 Teaching Data Types: Semantic Mapping (Quiz 1)

One common SLA approach to reinforcing new vocabulary is semantic mapping, in which the teacher offers a single word and students build a network of related lexical items to fortify their expanding lexicon (e.g. given *festival*: *music, food, people, loud,* etc.).[5], [6] When reinforcing various binary representations, a teacher can use a comparable, more systematic technique for related lexical items.

Working in pairs, students write any four-letter ASCII character combination on a loose piece of paper (e.g. *AsEe*, *Dog*!, *pw12*). Students then pass their paper to a nearby group such that there is one classroom loop. After that, the following actions are performed by each pair on the new piece of paper before passing it to the next group:

1. Convert to hexadecimal; pass to the next group.
2. Convert to binary; pass.
3. Add the first two letters to the second two letters; pass.
4. Swap the most significant bit with the least significant bit; pass.
5. Add the first 8 bits to the last 8 bits; pass.
6. Consider as 2s complement representation. What is largest and smallest value in the class?

At various time points in the exercise, the teacher should check for understanding (e.g. "How many bits should you have now?" after step 3; "What is the largest possible value someone could have?" after step 6).

Similar to the analog SLA technique, this exercise minimizes teacher talk time and maximizes student engagement in the "target language."

3.2 Teaching Logic Operations: From Restricted Exposure to Authentic Output (Quiz 2)

When teaching logical operations (AND, OR, NOT), teachers might be tempted to settle for a student successfully calculating the output of an operator given any input. With an SLA approach, however, students should move beyond recognition to active use. This popular SLA lesson plan begins with restricted exposure, continues with a clarification stage and concludes with authentic output[4]—proposed here with Boolean operators.

Students begin with a think-pair-share exercise, converting several teacher-provided logical statements in English to Boolean representations (e.g. *If I've completed my homework (h) and my favorite show (s) is not on, I will go to the gym (g).* ⇒ g = h·NOT(s)). This kind of restricted exposure should be simplified with "high quantities of target-language items".[4]

The teacher uses the final "share" segment of the exercise as time for explanation (i.e. guided discovery) in which the class works together to identify the correct answers, under the guidance of the teacher who corrects and clarifies.

After this, students try using what they have learned in an open-ended way. Students should develop their own logical statements in English and then pair up with a new student. The students take turns reading their statements as the other converts them to Boolean representation. This final step of moving towards authentic output allows students to use whatever tools at their disposal in the target "language" with the aim of reinforcing logical thought patterns necessary for fluency in computing. Furthermore, students practice verbally communicating using the rules of their new "language."

3.3 Teaching Finite State Machines: Cloze (No Quiz)

When teaching finite state machines (FSMs), students often struggle with the format of each stage of development: (1) state graph, (2) transition table, (3) logical expressions, and (4) gate-level circuitry. This parallelism closely resembles writing structures wherein the author employs patterns to advance his thesis. Language teachers have adopted the "cloze" technique (first published by Taylor[7] in 1953 as a gap fill exercise for assessing readability) to reinforce common patterns and linguistic formulae.

In this FSM cloze exercise, students are given a drawing of a state graph, transition table, logical expressions, and gate-level circuitry. Each of these elements has strategic elements missing with a gap in its place. As if they were completing a Sudoku puzzle, students extend the provided data to the other parallel structures until all the gaps are completed.

3.4 Teaching Assembly Language: Cloze (Quiz 3)

A cloze exercise can be used in a different way for moving students from writing code in binary to writing code in the Assembly language. For this task, students are given a piece of paper with a grid having three columns: (1) binary, (2) hexadecimal, and (3) assembly language. Like the cloze technique described before, some locations in the grid are blank or only partially completed. The task of the student is the complete the table by making each blank match the other instruction(s) on the same row. With this implementation students fortify their understanding that hexadecimal is a friendlier way of representing a binary value, and they recognize that writing in assembly language is a particular way

of making binary values (instructions) even friendlier, or more intuitive.

3.5 Teaching Sorting Algorithms: Total Physical Response and Jigsaw (Quiz 4)

Understanding and implementing sorting algorithms is a common learning objective for programming courses. To teach several algorithms using an SLA approach, a teacher might consider a "jigsaw" technique[8] with "total physical response."[9], [10]

The class is split into groups of three, and students move to sit with their group. Half of the groups are given a written description of the "selection sort" method, and the other half of the groups are given a written description of "bubble sort." In the small groups, the three students use cards numbered 1-9 to practice their sorting algorithm; the students physically move the cards around on the table until they each understand the sorting technique.

Then, the students are paired with students from other groups and are asked to explain their sorting algorithm to their new partner and illustrate using cards. Finally, to check for understanding, one student who was originally given "selection sort" volunteers to show the entire class how "bubble sort" is implemented, and vice-versa.

3.6 Teaching Subroutines: From Restricted Exposure to Freer Output (Quiz 5)

One common SLA lesson plan structure for emphasizing writing takes the form of "restricted exposure" → "teach" → "freer output."[4] This approach typically exposes students to a specific grammatical structure in a reading text chosen because it uses that structure. Then, the students' attention is brought to that structure in the text, and the teacher elicits knowledge about that structure based on what the students read. Next, the teacher fills in the gaps before asking students to produce a writing that utilizes the grammatical structure.

A similar approach may be used in introduction to computing when teaching subroutines. Instead of a text, students are given an LC3 assembly code and asked to describe what it is doing. Specifically, students might be given a printed code that uses JSR before the students are taught how it work. Students attempt to describe what the code is doing, first individually, then in groups—adding good, written comments where appropriate.

Then, the instructor elicits knowledge about subroutines based on what students can observe in the program. "How does the JSR instruction seem to be working? What does RET do? How might RET use registers?" The teacher then fills in the gaps in understanding with a fuller explanation of subroutines before the students are asked to convert a previously written code (from a previous coding assignment) to a code that uses subroutines. The students are free to use subroutines in whatever way they want, but not free to create a new code for writing a subroutine (thus, "freer," not "free" or "authentic").

3.7 Teaching Stacks and Interrupts: From Restricted Exposure to Restricted Output (Quiz 6)

Another SLA lesson format is a slight modification on the aforementioned form. Here we have "restricted exposure" → "teach" → "restricted output."[4]

Very similar to teaching subroutines in assembly, here stacks and interrupts are presented via a written code distributed to students. The code uses stacks and interrupts before the students are presented with the details of how to write such code (albeit after the students have been presented with the concept of stacks). Then, the teacher elicits knowledge about the use of stacks and interrupts from the students. Finally, the students are asked to produce a code that "pops" a very specific sequence of numbers. The difference here is that students are intentionally more restricted in how they are asked to use stacks. Opting for restricted output over freer or authentic output is a good option if the concept being taught is more complicated or nuanced in its authentic setting than that for which the students are ready.

4. Methods

Introduction to Computing (BME 303) as taught in the Department of Biomedical Engineering at UT meets two times per week for a 75-minute lecture taught by the professor (Telang). In addition to these lectures, each student attends one of four recitation sections led by one teaching assistant (TA). During recitation, course content is reviewed and new material is presented. Two of the recitation sections were taught by a trained TA who presented the material using standard approaches (Group 1: Non-SLA). The other two recitation sections were taught by a different trained TA (the first author, Gardner) who presented the same material using SLA techniques (Group 2: SLA). Each TA taught both a morning and afternoon recitation so as to remove potential bias based on meeting time.[11]

Quizzes were administered in the first the lecture following the presentation of recitation material, and each quiz was designed to test student understanding of the material presented in recitation. The two groups (taught via traditional approach vs. SLA approach) are compared by a t-test, or a Mann-Whitney U-test in the case that a Levene's test for homoscedasticity (equal variances) fails to reject the null hypothesis. The null hypothesis for the t-tests and the non-parametric Mann-Whitney U-test are similar: the mean quiz grades between both groups is the same.

In addition to quiz scores, the amount of time to complete the quiz was also compared between the two groups. Because student-specific data and variance is inaccessible, means are reported and inferences made about the meaning. Here, it is desirable to know if SLA techniques may prove useful in helping students complete tasks more quickly.

A third analysis approach is comparable to that employed by Frederick et al.[1] in which an Intrinsic Motivation Inventory (IMI) survey evaluates student interest/enjoyment, perceived competence, effort, felt pressure and tension, and perceived choice.[12] Additionally, the NASA Task Load

Index (TLX) measures workload: mental demand, physical demand, temporal demand, performance, effort, and frustration.[13] Instead of conducting these surveys throughout the course, they were conducted once at the end. The results are compared using a t-test (or a Mann-Whitney U-test) to assess the null hypothesis that there is no significant difference between motivation or workload between the two groups.

5. Results

5.1 Checking for Bias

One potential for bias was identified as differences in student skill level being unevenly distributed amongst the four recitation sections. Standardized test scores such as SAT and ACT scores, and predicted GPAs are traditionally used as indicators of student preparation for college level coursework. Future studies would benefit from including ways to account for these factors.

5.2 Statistical Analysis

Altogether, three measurements were determined to be statistically different between the two groups (non-SLA vs. SLA). The students in recitations taught with SLA techniques had no statistical difference on quiz scores compared with the non-SLA sections, however, the time to complete the quizzes was shorter on average for the SLA students for every quiz. Interestingly, the temporal demand in the NASA TLX did not show that students were aware of a decrease in a demand on their time.

The IMI survey also revealed two student perceptions of the class. Students taught with SLA techniques reported a significantly higher level of enjoyment ($p<0.01$) and a higher perceived value for the course ($p<0.05$) when compared to the students taught with non-SLA techniques.

Tables 1-4 contains the complete results of the statistical analysis.

6. Discussion

The work in using SLA approaches published by Frederick et al. reports statistical differences in "effort" only in the end of course survey. That is, students taught with SLA methodology reported lower required effort when compared to those taught with non-SLA techniques. These results differ from the results reported here in that we found no statistical difference when considering required effort, but instead found differences concerning student enjoyment of recitation sections and the perceived value of the course. Both of the differences favored the SLA approach.

Also of interest is that the means for every exam score, for the final grade, and for all the quizzes (except for two) are higher for the students who sat under SLA teaching methodology, though not with p-values low enough to indicate statistical significance. However, the authors posit that given more students in subsequent semesters, the differences in exam scores between SLA and non-SLA groups would likely become statistically significant. ($t \propto n^{1/2}$.)

Finally, the difference in time required to complete quizzes was a surprising and interesting result. Students taught with SLA approaches were able to complete the quiz work more quickly, and anecdotal evidence suggests that the same held true for exams also. Students in the recitation sections utilizing SLA techniques seemed to finish more quickly that students in the non-SLA classroom. This strong difference between the groups is worth exploring more in future studies and could be worth considering as exams are prepared for students in similar classes taught by similar methods.

7. Summary

We have made a case for extending the application of SLA pedagogical methods to lower-level computing coursework and presented a toolkit for an introduction to computing class based on common SLA techniques. Our data shows that students taught with SLA techniques in recitation sections perform tasks more quickly, with more enjoyment, and with a greater appreciation for course content when compared with students taught with more traditional approaches.

Acknowledgement

Thank you to the students in BME 303, Fall 2017 for helping to create a positive and energetic learning environment. M.R.G. was supported by the NIH T32 training grant EB007507.

References

[1] C. Frederick and L. S. Ph.D., "Work in Progress: Using Second Language Acquisition Techniques to Teach Programming - Results from a Two-Year Project," in *ASEE Annual Conference & Exposition*, 2017.

[2] J. Harmer, "The practice of English language teaching," *London/New York*, 1991.

[3] N. F. Davies, "Receptive Versus Productive Skills in Foreign Language Learning," *Mod. Lang. J.*, vol. 60, no. 8, pp. 440–443, 1976.

[4] J. Scrivener, *Teaching Learning*. Macmillan: London, 2014.

[5] A. J. Sökmen, "Current trends in teaching second language vocabulary," *Readings Methodol.*, vol. 152, 1997.

[6] J. Aitchison, "Words in the mind: An introduction to the mental lexicon," *Cambridge, MasSachuSettS*, 1987.

[7] W. L. Taylor, "'Cloze Procedure': A New Tool for Measuring Readability," *Journal. Bull.*, vol. 30, no. 4, pp. 415–433, Sep. 1953.

[8] E. Aronson, *The jigsaw classroom.* Sage, 1978.

[9] J. J. Asher, "The total physical response approach to second language learning," *Mod. Lang. J.*, vol. 53, no. 1, pp. 3–17, 1969.

[10] J. J. Asher, "The Learning Strategy of The Total

Physical Response: A Review.," *Mod. Lang. J.*, vol. 50, no. 2, pp. 79–84, 1966.
[11] N. G. Pope, "How the Time of Day Affects Productivity: Evidence from School Schedules," *Rev. Econ. Stat.*, vol. 98, no. 1, pp. 1–11, Mar. 2015.
[12] E. McAuley, T. Duncan, and V. V Tammen, "Psychometric properties of the Intrinsic Motivation Inventory in a competitive sport setting: a confirmatory factor analysis.," *Res. Q. Exerc. Sport*, vol. 60, no. 1, pp. 48–58, Mar. 1989.
[13] S. G. Hart, "Nasa-Task Load Index (NASA-TLX); 20 Years Later," *Proc. Hum. Factors Ergon. Soc. Annu. Meet.*, vol. 50, no. 9, pp. 904–908, Oct. 2006.

Quizzes: Mean Scores						
score/10	**Quiz 1**	**Quiz 2**	**Quiz 3**	**Quiz 4**	**Quiz 5**	**Quiz 6**
Group 1: Non-SLA	9.43 (n=30)	8.13 (n=32)	8.33 (n=48)⁺	7.81 (n=32)	8.91 (n=32)	8.91 (n=32)
Group 2: SLA	9.83 (n=30)	9.06 (n=32)	9.67 (n=15)⁺	7.81 (n=32)	9.38 (n=32)	8.55 (n=31)
Levene's Test p-value:	*0.012**	*0.002**	*0.318*	*0.402*	*0.230*	*0.352*
t-test p-value:	-	-	*0.08*	*1.00*	*0.34*	*0.40*
Mann-Whitney U-test p-value:	*0.15*	*0.36*	-	-	-	-

Table 1. Levene's test was performed to check for homoscedasticity between the groups. Where variances were not the same, a Mann-Whitney U-test was performed instead of a t-test. Results for 6 quizzes show no statistical difference between means. The mean of each quiz, with the exception of quiz 4 and 6, is higher for group 2 (SLA). [⁺ For Quiz 3, only one of two sections was taught with the SLA technique.]

Quizzes: Mean Time to Completion						
Units: seconds	**Quiz 1**	**Quiz 2**	**Quiz 3**	**Quiz 4**	**Quiz 5**	**Quiz 6**
Group 1: Non-SLA	454	338	287	585	678	532
Group 2: SLA	410	237	204	492	505	386
Difference (SLA – Non-SLA):	*-44**	*-101**	*-83**	*-93**	*-173**	*-146**

Table 2. Mean times to qui completion show differences between the groups, though no statistical inference is made. Students taught with SLA techniques were faster at completing quizzes on average for every quiz, with no difference in quiz score outcome (see Table 1).

End of Course Survey: Mean Intrinsic Motivation Inventory Values					
Ratings: 1-7	**Enjoyment**	**Competence**	**Importance**	**Pressure**	**Value**
Group 1: Non-SLA (n=27)	3.963	3.683	5.381	2.704	4.958
Group 2: SLA (n=32)	4.407	3.817	6.181	2.638	5.061
Levene's Test p-value:	*0.0777*	*0.0177**	*0.0091**	*0.1089*	*0.0598*
t-test p-value:	*0.008**	-	-	*0.776*	*0.015**
Mann-Whitney U-test p-value:	*(0.004*)*	*0.194*	*0.660*	*(0.990)*	*(0.043*)*

Table 3. Results of an end of course survey measuring motivation (IMI) show a statistical difference between the reported enjoyment and the perceived value when comparing non-SLA and SLA groups. Students under SLA teaching reported enjoying recitation more and had a higher perceived value of the course. Levene's test was performed to check for homoscedasticity between the groups. Where variances were not the same, a Mann-Whitney U-test was performed instead of a t-test. Where there was homoscedasticity between the groups, a Mann-Whitney U-test is still included for added value, though the t-test is sufficient.

End of Course Survey: Mean NASA Task Load Index Values							
	Ratings: 1-7	**Mental Demand**	**Physical Demand**	**Temporal Demand**	**Performance Demand**	**Effort**	**Frustration**
Course	**Group 1: Non-SLA** (n=27)	6.111	4.333	5.111	3.815	5.852	4.885
	Group 2: SLA (n=32)	6.143	3.524	5.048	4.333	5.810	5.000
	Levene's Test p-value:	*0.1950*	*0.0690*	*0.5389*	*0.1295*	*0.8242*	*0.1457*
	t-test p-value:	*0.742*	*0.413*	*0.826*	*0.634*	*0.661*	*0.718*
	Mann-Whitney U-test p-value:	*(0.758)*	*(0.483)*	*(0.717)*	*(0.530)*	*(0.741)*	*(0.623)*
Recitation	**Group 1: Non-SLA** (n=27)	3.704	3.333	3.444	4.593	2.889	3.370
	Group 2: SLA (n=32)	4.429	2.500	4.048	5.143	2.571	3.048
	Levene's Test p-value	*0.9621*	*0.4743*	*0.3118*	*0.0110**	*0.6338*	*0.4196*
	t-test p-value:	*0.211*	*0.327*	*0.604*	-	*0.575*	*0.252*
	Mann-Whitney U-test p-value:	*(0.246)*	*(0.329)*	*(0.623)*	*0.790*	*(0.560)*	*(0.438)*

Table 4. Results of t-tests of an end of course survey measuring workload (NASA TLX) show no statistical differences between groups for questions focused on both recitation and the course as a whole. Levene's test was performed to check for homoscedasticity between the groups. Where variances were not the same, a Mann-Whitney U-test was performed instead of a t-test. Where there was homoscedasticity between the groups, a Mann-Whitney U-test is still included for added value, though the t-test is sufficient.